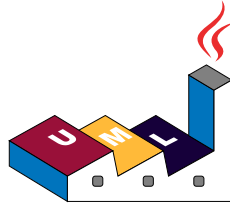


PlantUML を使った UML の描き方



PlantUML 言語リファレンスガイド

(Version 1.2023.11)

PlantUML は、以下のようなダイアグラムを素早く作成するためのコンポーネントです。

- シーケンス図
- ユースケース図
- クラス図
- オブジェクト図
- アクティビティ図
- コンポーネント図
- 配置図
- 状態遷移図 (ステートマシン図)
- タイミング図

以下のような、UML 以外の図もサポートしてます。

- JSON Data
- YAML Data
- Network diagram (nwdiag)
- ワイヤフレーム
- アーキテクチャ図
- 仕様及び記述言語 (SDL)
- Dita
- ガントチャート
- マインドマップ
- WBS 図 (作業分解図)
- AsciiMath や JLaTeXMath による、数学的記法
- ER 図

各ダイアグラムは、シンプルで直感的に書くことができます。

1 シーケンス図

PlantUML によるシーケンス図の作成は、驚くほど簡単です。この使いやすさは、直感的で覚えやすいように設計された、ユーザーフレンドリーな構文に大きく起因しています。

- 直感的な構文：

何よりもまず、PlantUML が採用している、わかりやすく直感的な構文を、ユーザは高く評価しています。このよく考え抜かれたデザインは、ダイアグラム作成が初めての人でも、基本を素早く簡単に理解できることを意味します。

- テキストとグラフィックの相関：

もう一つの際立った特徴は、テキスト表現とグラフィカルな出力の間の緊密な類似性です。この調和のとれた相関性により、テキスト原稿がグラフィカルなダイアグラムに正確に変換され、最終的なアウトプットに不快な驚きを与えることなく、まとまりのある予測可能なデザイン体験を提供します。

- 効率的な制作プロセス：

テキストとグラフィカルな結果との間に強い相関関係があるため、作成プロセスが単純化されるだけでなく、大幅にスピードアップします。ユーザーは、時間のかかる修正や調整の必要性が少なくなり、より合理的なプロセスの恩恵を受けることができます。

- 作図中の視覚化：

テキストを下書きしながら、最終的なグラフィカルな仕上がりをイメージできる機能は、多くの人にとって貴重なものです。最初の草稿から最終的なプレゼンテーションへのスムーズな移行を自然に促進し、生産性を高め、ミスの可能性を減らします。

- 簡単な編集と修正：

重要なのは、既存のダイアグラムの編集が手間のかからないプロセスであるということです。ダイアグラムはテキストから生成されるため、グラフィカルなツールを使って画像を変更するよりも、調整がかなり簡単で正確であることがわかります。

PlantUML は、シーケンス・ダイアグラムの作成と編集に、わかりやすくユーザフレンドリーなアプローチを提供し、初心者と熟練したデザイナーの両方のニーズを満たします。PlantUML は、視覚的に説明的で正確なダイアグラムを作成するために、テキスト入力のシンプルさを巧みに活用し、ダイアグラム作成ツールキットの必携ツールとしての地位を確立しています。

PlantUML の一般的なコマンドについては、ダイアグラム作成の経験を向上させるために学ぶことができます。

1.1 基本的な例

シーケンス図は、2 人の参加者の間のメッセージを描くために使用されます。参加者は明示的に宣言する必要はありません。

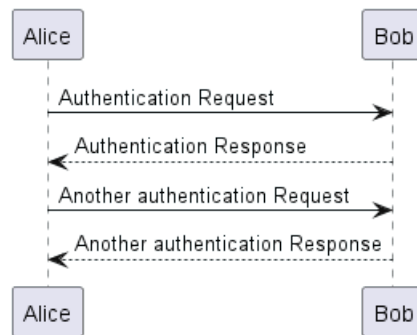
点線の矢印を持つためには、--> を使用します。

<- と <-- を使用することも可能です。描画は変わりませんが、読みやすさが向上する可能性があります。これはシーケンス図にのみ当てはまることで、他の図ではルールが異なることに注意してください。

```
@startuml
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

Alice -> Bob: Another authentication Request
Alice <-- Bob: Another authentication Response
@enduml
```





1.2 分類子の宣言

キーワード `participant` を使って分類子を宣言すると、分類子の表示を調整することができます。

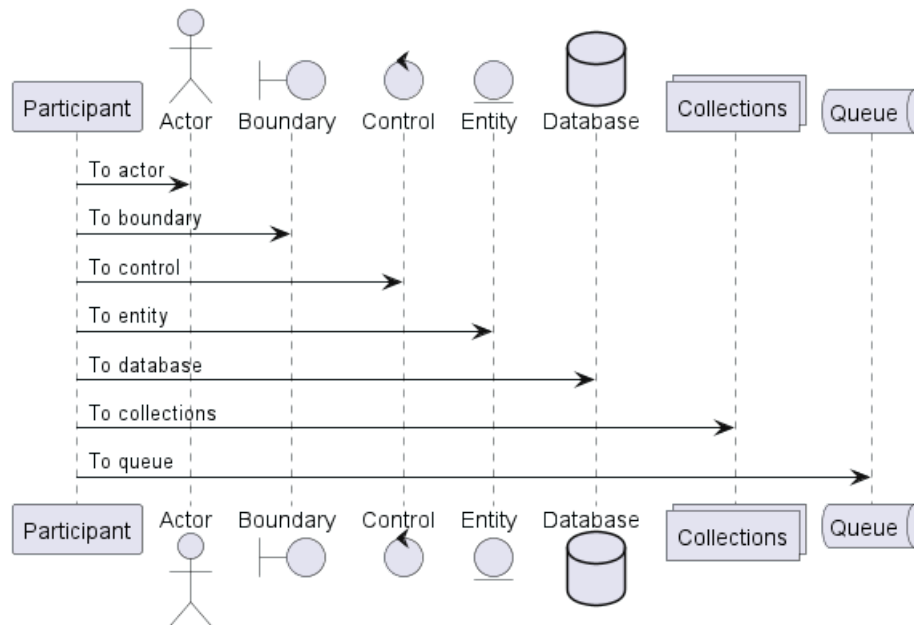
宣言した順序が、デフォルトの表示順になります。

分類子の宣言に別のキーワードを使用すると、分類子の形を変えることができます：

- `actor`
- `boundary`
- `control`
- `entity`
- `database`
- `collections`
- `queue`

```

@startuml
participant Participant as Foo
actor Actor as Foo1
boundary Boundary as Foo2
control Control as Foo3
entity Entity as Foo4
database Database as Foo5
collections Collections as Foo6
queue Queue as Foo7
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7 : To queue
@enduml
  
```

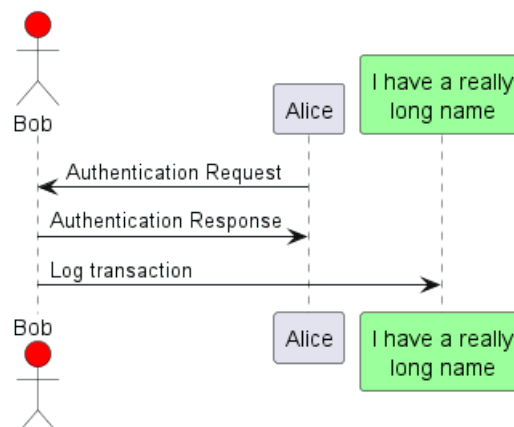


キーワード **as** を使って分類子の名前を変更することができます。

アクターや分類子の背景色を、HTML コードや色名を使って変更することもできます。

```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
    participant L as "I have a really\nlong name" #99FF99
  '/
```

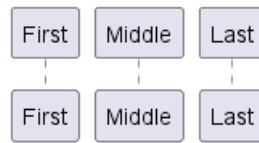
```
Alice->>Bob: Authentication Request
Bob->>Alice: Authentication Response
Bob->>L: Log transaction
@enduml
```



order キーワードを使って、分類子が表示される順序を変更することもできます。

```
@startuml
participant Last order 30
participant Middle order 20
participant First order 10
```

```
@enduml
```



1.3 複数の行を持つ分類子の宣言

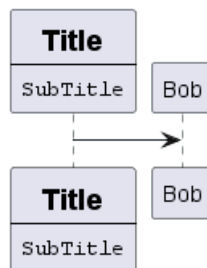
複数の行を持つ分類子を宣言できます。

```
@startuml
participant Participant [
    =Title
    ----
    ""SubTitle""
]
```

```
participant Bob
```

```
Participant -> Bob
```

```
@enduml
```

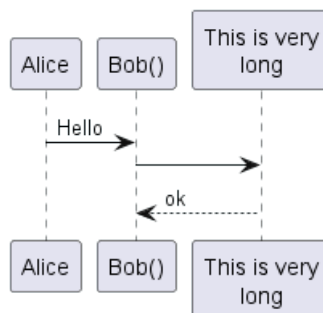


[Ref. QA-15232]

1.4 分類子名にアルファベット以外を使う

分類子を定義するときに引用符を使用することができます。そして、分類子にエイリアスを与えるためにキーワード `as` を使用することができます。

```
@startuml
Alice -> "Bob()" : Hello
"Bob()" -> "This is very\nlong" as Long
' You can also declare:
' "Bob()" -> Long as "This is very\nlong"
Long --> "Bob()" : ok
@enduml
```

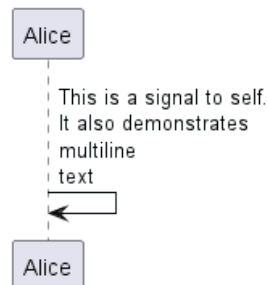


1.5 自分自身へのメッセージ

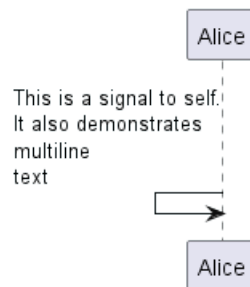
分類子は自分自身へメッセージを送信できます。

を使用して、複数行のテキストを扱えます。

```
@startuml
Alice -> Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```



```
@startuml
Alice <- Alice: This is a signal to self.\nIt also demonstrates\nmultiline \ntext
@enduml
```



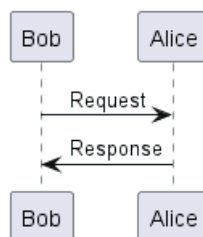
[Ref. QA-1361]

1.6 テキストの位置

`skinparam sequenceMessageAlign` を使用すると、矢印に表示するテキストの位置を `left`、`right`、`center` に設定することができます。

`direction` または `reverseDirection` を使用すると、矢印の方向に応じてテキストの位置が決定されます。この機能の詳細は `skinparam` のページを参照してください。

```
@startuml
skinparam sequenceMessageAlign right
Bob -> Alice : Request
Alice -> Bob : Response
@enduml
```



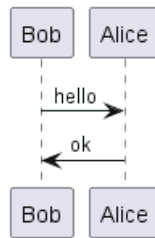
1.6.1 応答メッセージのテキストを矢印の下に表示する

`skinparam responseMessageBelowArrow true` コマンドを使用すると、応答メッセージのテキストを矢印の下に表示することができます。

```

@startuml
skinparam responseMessageBelowArrow true
Bob -> Alice : hello
Alice -> Bob : ok
@enduml

```



1.7 矢印の見た目を変える

矢印の見た目をいくつかの方法によって変更できます。

- メッセージの消失を示す最後の **x** を追加
- \ や / を < や > の代わりに使うと
- 矢印の先端が上側だけまたは下側だけになります。
- 矢印の先端を繰り返す (たとえば >> や //) と、矢印の先端が細くなります。
- -- を - の代わりに使うと、矢印が点線になります。
- 矢じりに最後の "O" を追加
- 双方向の矢印を使用する

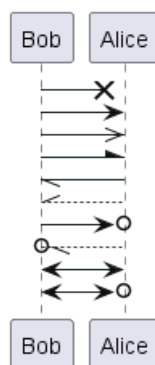
```

@startuml
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\\- Alice
Bob //-- Alice

Bob ->o Alice
Bob o\\-- Alice

Bob <-> Alice
Bob <->o Alice
@enduml

```



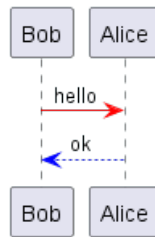
1.8 矢印の色を替える

以下の表記を使って、個々の矢印の色を変えることができます。

```

@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml

```



1.9 メッセージシーケンスの番号付け

メッセージへ自動で番号を振るために、キーワード **autonumber** を使います。

```

@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
@enduml

```



autonumber < 開始 > で開始番号を、また、**autonumber** < 開始 > < 増分 > で増分も指定することができます。

```

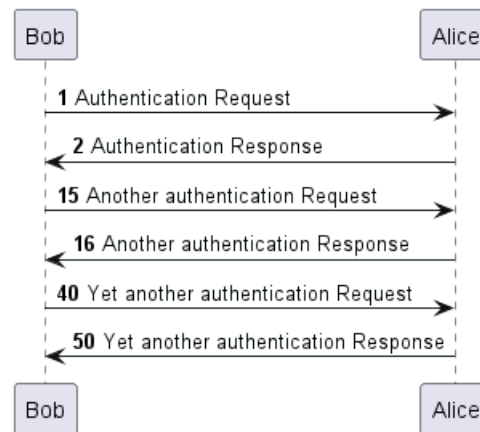
@startuml
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml

```



二重引用符で囲って番号の書式を指定することができます。

その書式指定は Java の `DecimalFormat` 方式で行います。(0 は桁を表し, # は存在しない場合は 0 で埋める桁を意味します)。

一部の HTML タグを書式に使うこともできます。

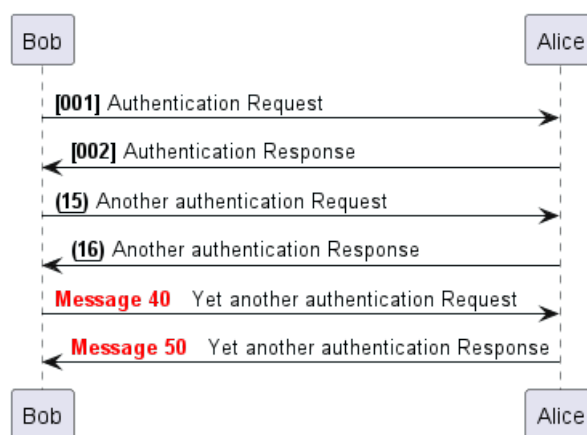
```

@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber 15 "<b>(<u>##</u>)"
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10 "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

@enduml
  
```



`autonumber stop` と `autonumber resume < 増分 > < 書式 >` を自動採番の一時停止と再開にそれぞれ使用することができます。

```

@startuml
autonumber 10 10 "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

autonumber stop
Bob -> Alice : dummy
  
```



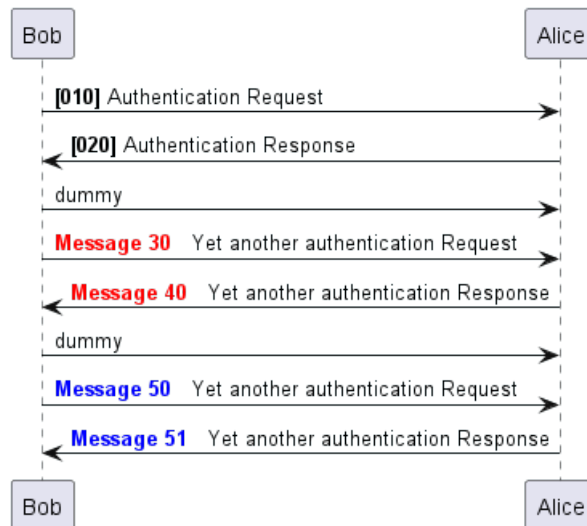
```

autonumber resume "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume 1 "<font color=blue><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml

```



開始番号は、2 つまたは 3 つの部分からなる数字の列を指定することもできます。各部分は、.、;、,、: またはこれらの組み合わせで区切ります。例えば、1.1.1 や 1.1:1 のようにします。

自動的に最後の数字が増加していきます。

最初の数字を増加させるには `autonumber inc A` を、2 番目の数字を増加させるには `autonumber inc B` を使用します。

```

@startuml
autonumber 1.1.1
Alice -> Bob: Authentication request
Bob --> Alice: Response

autonumber inc A
'Now we have 2.1.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc B
'Now we have 2.2.1
Alice -> Bob: Another authentication request
Bob --> Alice: Response

autonumber inc A
'Now we have 3.1.1
Alice -> Bob: Another authentication request
autonumber inc B
'Now we have 3.2.1
Bob --> Alice: Response
@enduml

```

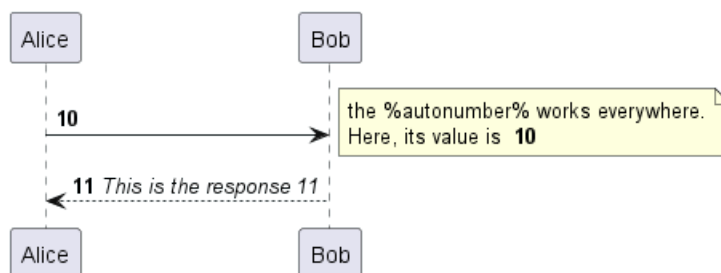




現在の `autonumber` の値は`%autonumber%` 変数で参照することができます:

```

@startuml
autonumber 10
Alice -> Bob
note right
    the <U+0025>autonumber<U+0025> works everywhere.
    Here, its value is ** %autonumber% **
end note
Bob --> Alice: //This is the response %autonumber%//
@enduml
  
```



[Ref. QA-7119]

1.10 タイトル、ヘッダー、フッター

`title` キーワードはページにタイトルをつけるのに使われます。

`header` や `footer` を使うことにより、ページにヘッダーやフッターをつけて表示することができます。

```

@startuml

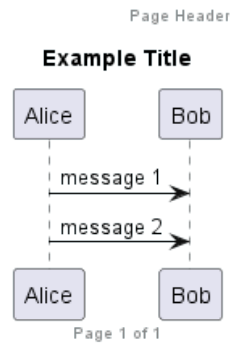
header Page Header
footer Page %page% of %lastpage%

title Example Title

Alice -> Bob : message 1
Alice -> Bob : message 2

@enduml
  
```





1.11 図の分割

図を複数の画像に分けるためにキーワード **newpage** を使います。

新しいページのタイトルをキーワード **newpage** の直後に書くことができます。

これは、複数ページにわたる長い図を書くときに便利な機能です。

```
@startuml
```

```
Alice -> Bob : message 1
```

```
Alice -> Bob : message 2
```

```
newpage
```

```
Alice -> Bob : message 3
```

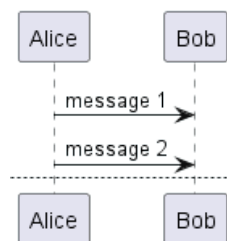
```
Alice -> Bob : message 4
```

```
newpage A title for the\nlast page
```

```
Alice -> Bob : message 5
```

```
Alice -> Bob : message 6
```

```
@enduml
```



1.12 メッセージのグループ化

次のキーワードを使えば、メッセージをまとめてグループ化できます。

- alt/else
- opt
- loop
- par
- break
- critical
- group 表示するテキスト

ヘッダ部分に文字列を追加することが可能です。(group については、後述の「group の 2 つ目のラベル」を参照)

グループを閉じるにはキーワード `end` を使用します。

注：グループはネスト可能です。

```
@startuml
Alice -> Bob: Authentication Request

alt successful case

    Bob -> Alice: Authentication Accepted

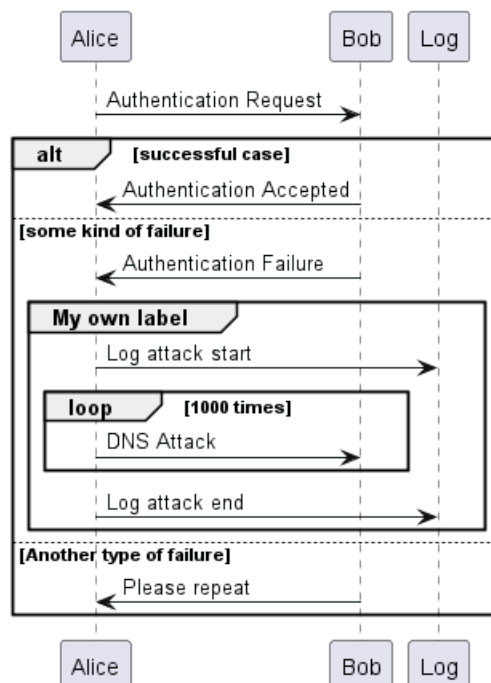
else some kind of failure

    Bob -> Alice: Authentication Failure
    group My own label
    Alice -> Log : Log attack start
        loop 1000 times
            Alice -> Bob: DNS Attack
        end
    Alice -> Log : Log attack end
    end

else Another type of failure

    Bob -> Alice: Please repeat

end
@enduml
```



1.13 group の 2 つ目のラベル

group では、[と] の間に 2 つ目のラベルを設定し、ヘッダに表示させることができます。

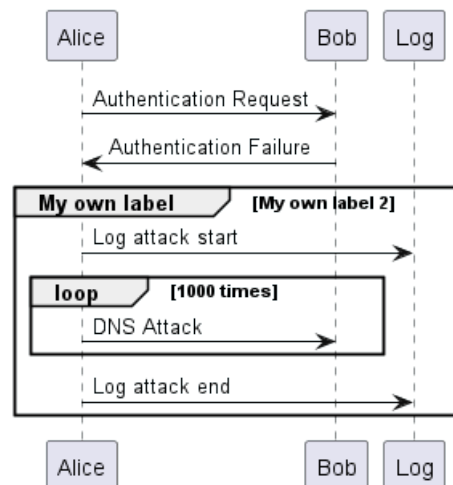
```
@startuml
Alice -> Bob: Authentication Request
```



```

Bob -> Alice: Authentication Failure
group My own label [My own label 2]
  Alice -> Log : Log attack start
  loop 1000 times
    Alice -> Bob: DNS Attack
  end
  Alice -> Log : Log attack end
end
@enduml

```



[Ref. QA-2503]

1.14 メッセージに付けるノート

メッセージのすぐ後ろにキーワード `note left` または `note right` を使用し、メッセージにノートを付けることが可能です。

`end note` キーワードを使って、複数行のノートを作ることができます。

```

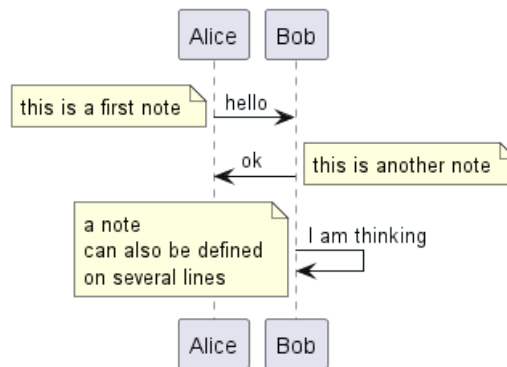
@startuml
Alice->>Bob : hello
note left: this is a first note

Bob->>Alice : ok
note right: this is another note

Bob->>Bob : I am thinking
note left
a note
can also be defined
on several lines
end note
@enduml

```





1.15 その他のノート

`note left of`、`note right of`、`note over` のキーワードを使って、分類子からの相対位置を指定してノートを配置することもできます。

ノートを目立たせるために、背景色を変えることができます。

また、キーワード `end note` を使って複数行のノートを作ることができます。

```

@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

```

```

note right of Alice: This is displayed right of Alice.

```

```

note over Alice: This is displayed over Alice.

```

```

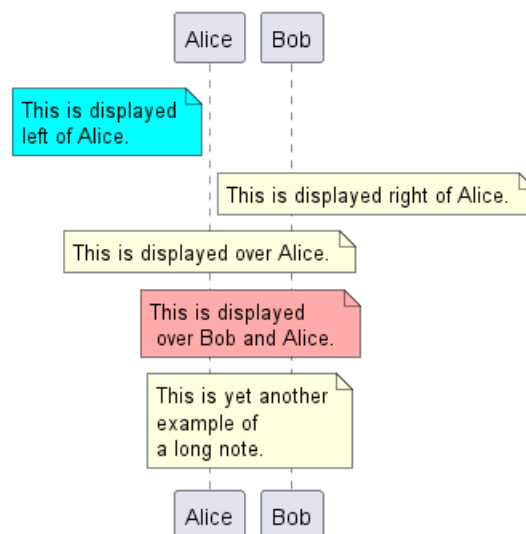
note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

```

```

note over Bob, Alice
This is yet another
example of
a long note.
end note
@enduml

```

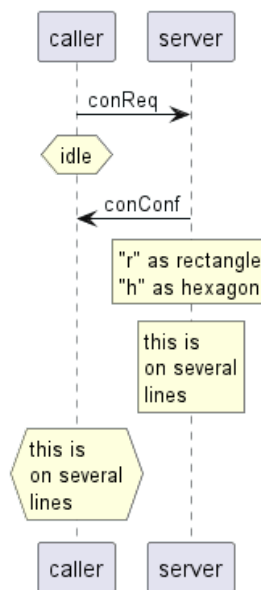


1.16 ノートの形を変える [hnote, rnote]

キーワード `hnote` と `rnote` を使ってノートの形を変更できます。

- `hnote` で六角形のノートになります
- `rnote` で四角形のノートになります

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
  "r" as rectangle
  "h" as hexagon
endrnote
rnote over server
  this is
  on several
  lines
endrnote
hnote over caller
  this is
  on several
  lines
endhnote
@enduml
```



[Ref. QA-1765]

1.17 すべての分類子にまたがるノート [across]

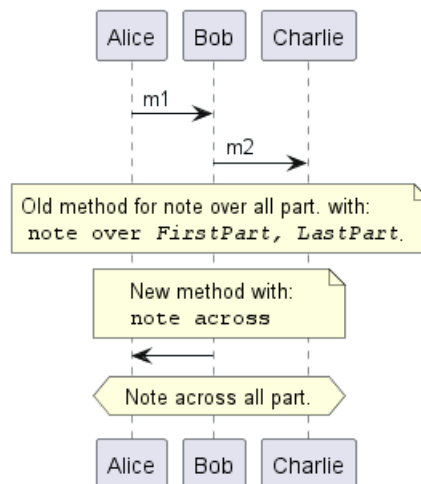
次の構文で、すべての分類子にまたがるノートを直接作ることができます：

- `note across`: ノートの記述

```
@startuml
Alice->Bob:m1
Bob->Charlie:m2
note over Alice, Charlie: Old method for note over all part. with:\n ""note over //FirstPart, LastPart
note across: New method with:\n""note across""
Bob->Alice
hnote across:Note across all part.
```



@enduml



[Ref. QA-9738]

1.18 複数のノートと同じレベルに並べる [/]

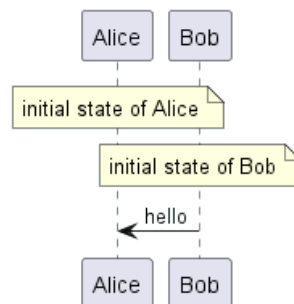
/を使って、複数のノートと同じレベルに並べることができます：

- /を使わない場合（デフォルトでは、ノートは整列されません）

```

@startuml
note over Alice : initial state of Alice
note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml

```

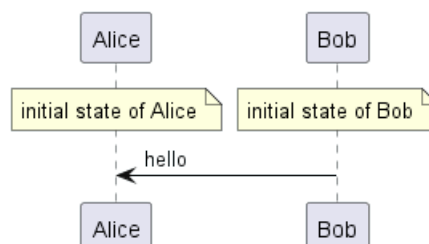


- /を使った場合（ノートが整列されます）

```

@startuml
note over Alice : initial state of Alice
/ note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml

```



[Ref. QA-354]



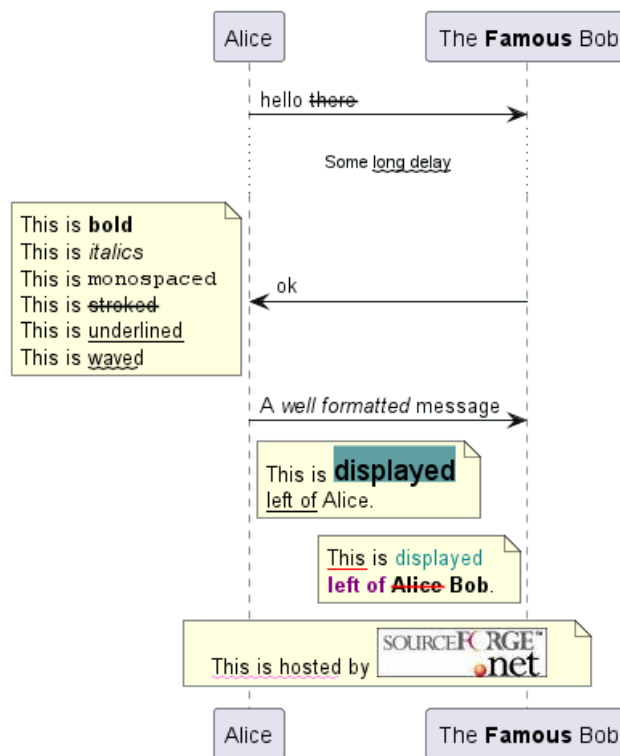
1.19 Creole と HTML

PlantUML では creole フォーマットを使うこともできます。

```
@startuml
participant Alice
participant "The **Famous** Bob" as Bob

Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
  This is bold
  This is italics
  This is "monospaced"
  This is --stroked--
  This is __underlined__
  This is ~~waved~~
end note

Alice -> Bob : A //well formatted// message
note right of Alice
  This is <back:cadetblue><size:18>displayed</size></back>
  __left of__ Alice.
end note
note left of Bob
  <u:red>This</u> is <color #118888>displayed</color>
  **<color purple>left of</color> <s:red>Alice</strike> Bob**.
end note
note over Alice, Bob
  <w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
@enduml
```



1.20 境界線（区切り線）

== を使って、図を論理的なステップに分けることも出来ます。

```
@startuml

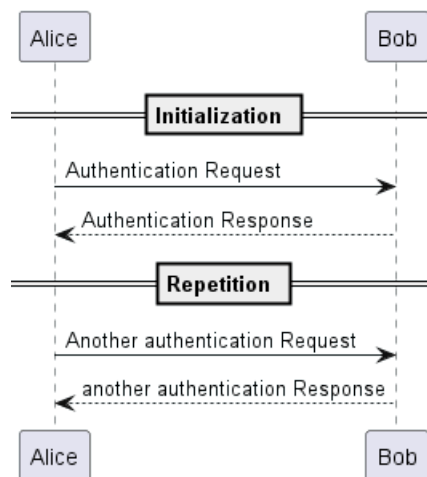
== Initialization ==

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

== Repetition ==

Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response

@enduml
```



1.21 リファレンス

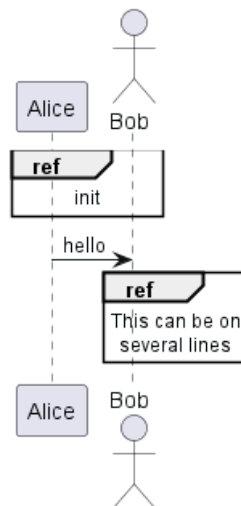
キーワード `ref over` を使用して、図中にリファレンスを挿入できます。

```
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

ref over Bob
    This can be on
    several lines
end ref
@enduml
```



1.22 遅延

処理の遅延を表すために ... が使えます。また、作成した遅延にコメントを付けることもできます。

```
@startuml
```

```
Alice -> Bob: Authentication Request
```

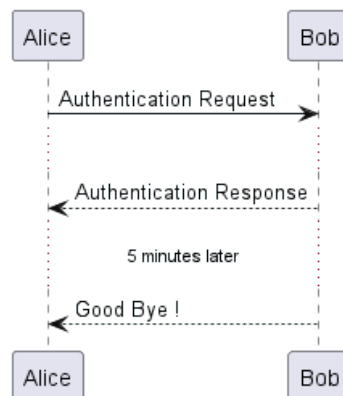
```
...
```

```
Bob --> Alice: Authentication Response
```

```
...5 minutes later...
```

```
Bob --> Alice: Good Bye !
```

```
@enduml
```



1.23 テキストの折り返し

を使って改行することで、長いメッセージを折り返すことができます。

また、`maxMessageSize` を設定するという方法もあります。

```
@startuml
```

```
skinparam maxMessageSize 50
```

```
participant a
```

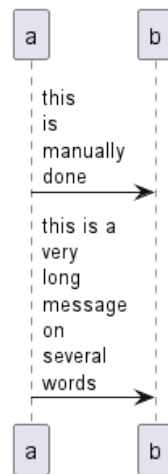
```
participant b
```

```
a -> b :this\nis\nmanually\ndone
```

```
a -> b :this is a very long message on several words
```

```
@enduml
```





1.24 間隔

図の間隔を調整するために、記号 `|||` を使用することができます。

さらにピクセル数を指定することもできます。

`@startuml`

Alice -> Bob: message 1

Bob --> Alice: ok

|||

Alice -> Bob: message 2

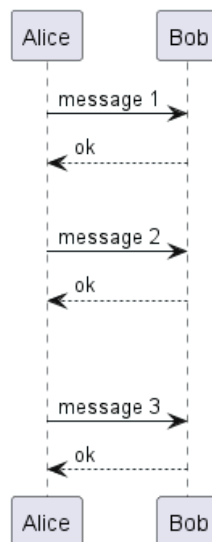
Bob --> Alice: ok

||45||

Alice -> Bob: message 3

Bob --> Alice: ok

`@enduml`



1.25 ライフラインの活性化と破棄

`activate` と `deactivate` を使って分類子の活性化を表します。

分類子の活性化はライフラインで表されます。

`activate` と `deactivate` は直前のメッセージに適用されます。

destroy は分類子のライフラインが終わったことを表します。

```
@startuml
participant User

User -> A: DoWork
activate A

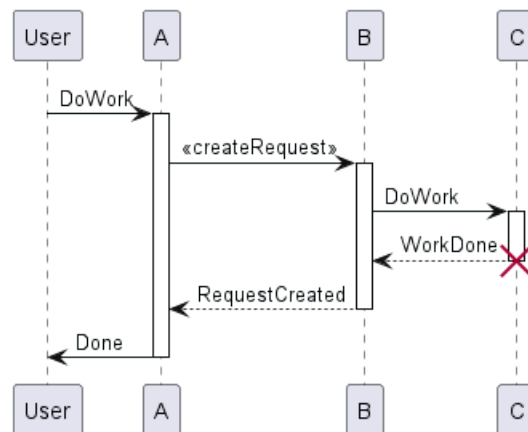
A -> B: << createRequest >>
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: RequestCreated
deactivate B

A -> User: Done
deactivate A

@enduml
```



ライフラインはネスト (入れ子に) することができ、色をつけることもできます。

```
@startuml
participant User

User -> A: DoWork
activate A #FFBBBB

A -> A: Internal call
activate A #DarkSalmon

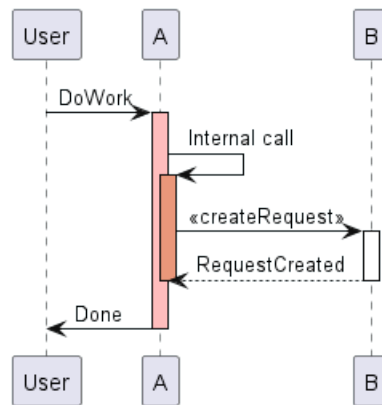
A -> B: << createRequest >>
activate B

B --> A: RequestCreated
deactivate B
deactivate A

A -> User: Done
deactivate A

@enduml
```

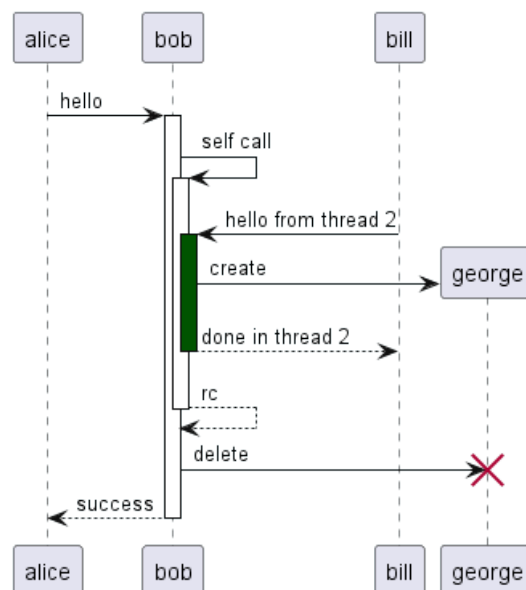




自動的に活性化 (autoactivate) することもできます。この場合は return キーワードを使用します。

```

@startuml
autoactivate on
alice -> bob : hello
bob -> bob : self call
bill -> bob #005500 : hello from thread 2
bob -> george ** : create
return done in thread 2
return rc
bob -> george !! : delete
return success
@enduml
  
```



1.26 Return

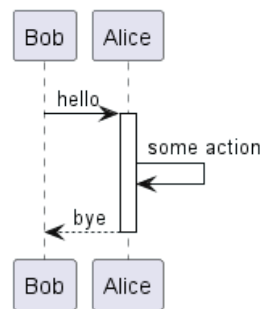
新しいコマンド **return** は、リターンメッセージを生成し、オプションでテキストラベルをつけることができます。リターンする先は最も最近活性化したライフラインです。構文は単純に **return** ラベルです。ラベルを与える場合には、通常のメッセージに与えることが可能な文字列を何でも与えることができます。

```

@startuml
Bob -> Alice : hello
activate Alice
Alice -> Alice : some action
  
```



```
return bye
@enduml
```



1.27 分類子の生成

キーワード `create` を、オブジェクトが最初のメッセージを受信する直前に置くことにより、このメッセージがオブジェクトを新しく生成していることを強調して表現できます。

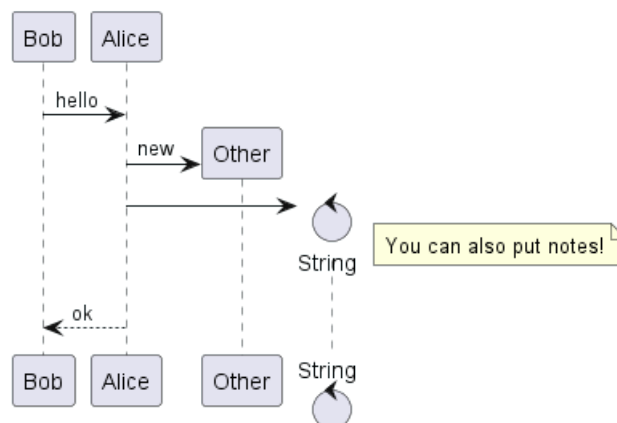
```
@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

create control String
Alice -> String
note right : You can also put notes!

Alice --> Bob : ok

@enduml
```



1.28 活性化、非活性化、生成のショートカット記法

対象の分類子を記述した直後に、次の記法を使うことができます。

- ++ 対象を活性化する (続けて色を記述することもできます)
- -- 起点側を非活性化する
- ** 対象のインスタンスを生成する
- !! 対象のインスタンスを破棄する

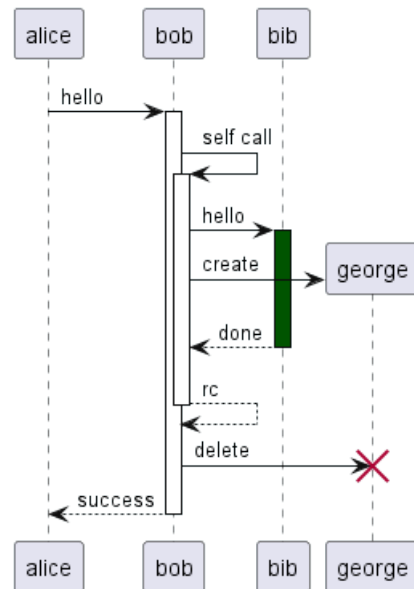
```
@startuml
alice -> bob ++ : hello
```



```

bob -> bob ++ : self call
bob -> bib ++ #005500 : hello
bob -> george ** : create
return done
return rc
bob -> george !! : delete
return success
@enduml

```

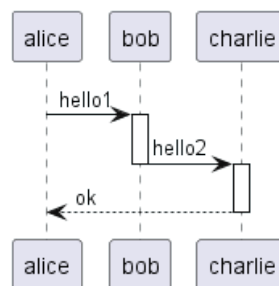


同じ行に活性化と非活性化を混ぜることもできます:

```

@startuml
alice -> bob ++ : hello1
bob -> charlie ---+ : hello2
charlie --> alice -- : ok
@enduml

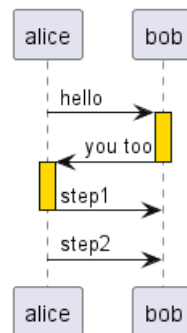
```



```

@startuml
@startuml
alice -> bob ---+ #gold: hello
bob -> alice ---+ #gold: you too
alice -> bob --: step1
alice -> bob : step2
@enduml
@enduml

```



[Ref. QA-4834, QA-9573 and QA-13234]

1.29 インとアウトのメッセージ

図の一部だけにフォーカスを当てたい場合には、「外から入ってくる」または「外に出ていく」メッセージを使えます。

左角括弧”[”を使って図の左端、右角括弧”]”を使って図の右側を表せます。

```

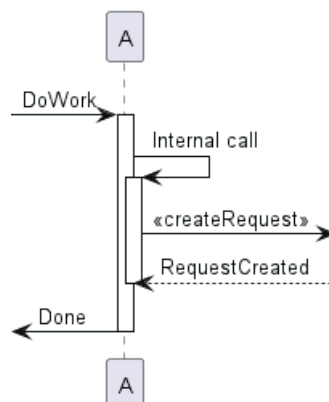
@startuml
[-> A: DoWork

activate A

A -> A: Internal call
activate A

A ->] : << createRequest >>

A<--] : RequestCreated
deactivate A
[<- A: Done
deactivate A
@enduml
  
```



また、次の書き方も使えます：

```

@startuml
participant Alice
participant Bob #lightblue
Alice -> Bob
Bob -> Carol
...
[-> Bob
[o-> Bob
[o->> Bob
  
```

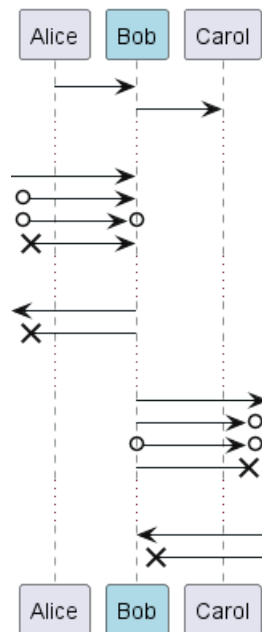


```

[x-> Bob
...
[<- Bob
[x<- Bob
...
Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]
...
Bob <-]
Bob x<-]

@enduml

```



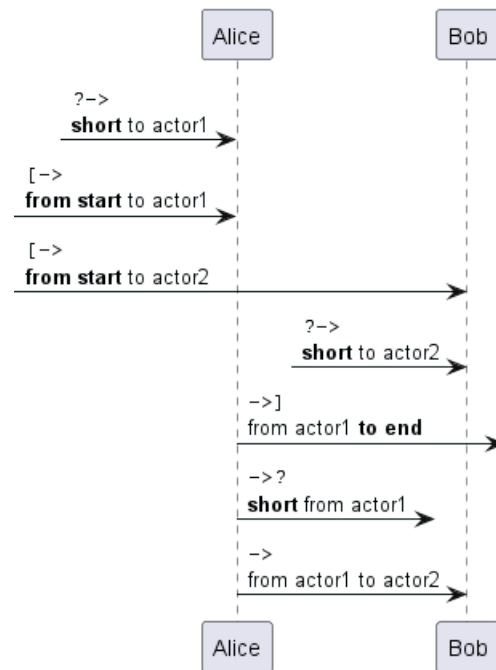
1.30 インとアウトのメッセージに短い矢印を使う

? で短い矢印を使用することができます。

```

@startuml
?-> Alice : ""?->""\n**short** to actor1
[-> Alice : ""[->""\n**from start** to actor1
[-> Bob : ""[->""\n**from start** to actor2
?-> Bob : ""?->""\n**short** to actor2
Alice ->] : ""->""\nfrom actor1 **to end**
Alice ->? : ""->?""\n**short** from actor1
Alice -> Bob : ""->"" \nfrom actor1 to actor2
@enduml

```



[Ref. QA-310]

1.31 アンカーと持続時間

teoz を使用するとダイアグラムにアンカーを追加することができ、それによって持続時間を表現することができます。

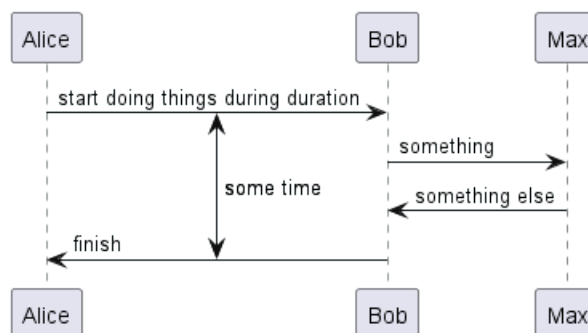
```

@startuml
!pragma teoz true

{start} Alice -> Bob : start doing things during duration
Bob -> Max : something
Max -> Bob : something else
{end} Bob -> Alice : finish

{start} <-> {end} : some time

@enduml
  
```



コマンドラインで-P オプションを使用して pragma を指定することもできます:

```
java -jar plantuml.jar -Pteoz=true
```

[Ref. issue-582]



1.32 ステレオタイプとスポット

<< と >> を使い分類子にステレオタイプをつけることができます。

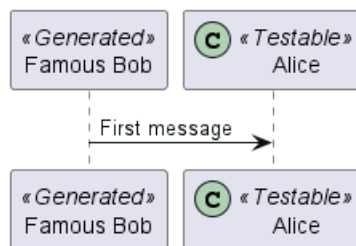
(X,color) と記述することによりステレオタイプに色付きの文字と円のアイコンをつけることができます。

```
@startuml
```

```
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>
```

```
Bob->Alice: First message
```

```
@enduml
```



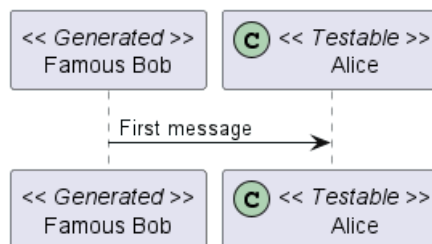
デフォルトでは *guillemet* キャラクターがステレオタイプを表示するために使用されます。スキンパラメータ *guillemet* を使用してこの動作を変更することができます：

```
@startuml
```

```
skinparam guillemet false
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>
```

```
Bob->Alice: First message
```

```
@enduml
```

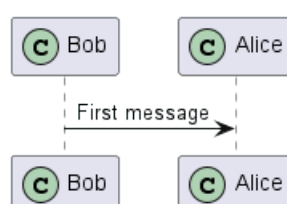


```
@startuml
```

```
participant Bob << (C,#ADD1B2) >>
participant Alice << (C,#ADD1B2) >>
```

```
Bob->Alice: First message
```

```
@enduml
```



1.33 タイトルについての詳細

タイトルには creole フォーマットが使用できます。

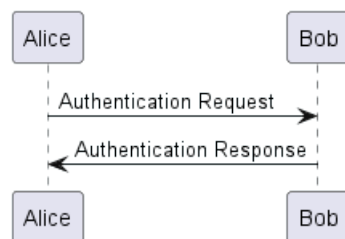
```
@startuml

title __Simple__ **communication** example

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```

Simple communication example



タイトルの記述では `\\n` を使用して新しい行を追加することができます。

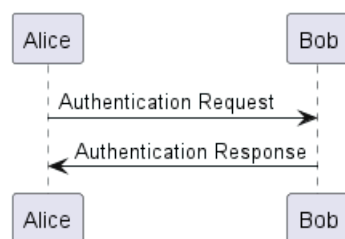
```
@startuml

title __Simple__ communication example\\non several lines

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```

**Simple communication example
on several lines**



また、キーワード `title` と `end title` を使うことにより、タイトルを複数行にわたって記述できます。

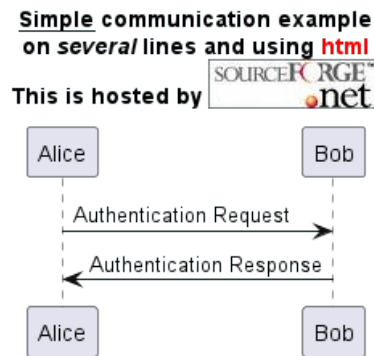
```
@startuml

title
  <u>Simple</u> communication example
  on <i>several</i> lines and using <font color=red>html</font>
  This is hosted by <img:sourceforge.jpg>
end title

Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

@enduml
```





1.34 分類子の囲み

キーワード `box` と `end box` を使い、分類子のまわりにボックスを描くことができます。

タイトルや背景色をキーワード `box` に続けて任意で追加できます。

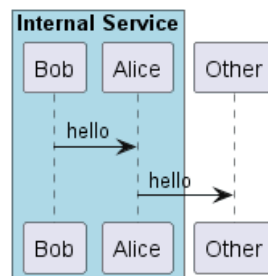
```

@startuml

box "Internal Service" #LightBlue
    participant Bob
    participant Alice
end box
participant Other

Bob -> Alice : hello
Alice -> Other : hello

@enduml
  
```



1.35 フッターの除去

図からフッターを削除するにはキーワード `hide footbox` を使います。

```

@startuml

hide footbox
title Foot Box removed

Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response

@enduml
  
```



1.36 スキンパラメータ

ダイアグラムの色やフォントを変更するには `skinparam` コマンドを使用します。

このコマンドは以下の場面で使用できます。

- ダイアグラム定義内で他のコマンドを同様に。
- インクルードされたファイル内。
- 設定ファイルのコマンドライン内や ANT タスク内。

次の例のように他のパラメータを変えることもできます。

```

@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20
skinparam maxmessagesize 60
skinparam sequenceParticipant underline
  
```

```

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C
  
```

```

User -> A: DoWork
activate A
  
```

```

A -> B: Create Request
activate B
  
```

```

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C
  
```

```

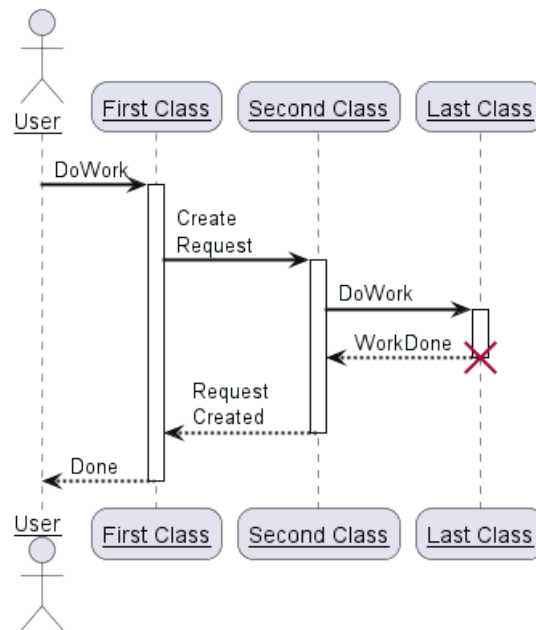
B --> A: Request Created
deactivate B
  
```

```

A --> User: Done
deactivate A
  
```

```

@enduml
  
```



```

@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

skinparam sequence {
  ArrowColor DeepSkyBlue
  ActorBorderColor DeepSkyBlue
  LifeLineBorderColor blue
  LifeLineBackgroundColor #A9DCDF

  ParticipantBorderColor DeepSkyBlue
  ParticipantBackgroundColor DodgerBlue
  ParticipantFontName Impact
  ParticipantFontSize 17
  ParticipantFontColor #A9DCDF

  ActorBackgroundColor aqua
  ActorFontColor DeepSkyBlue
  ActorFontSize 17
  ActorFontName Apex
}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
deactivate C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
  
```

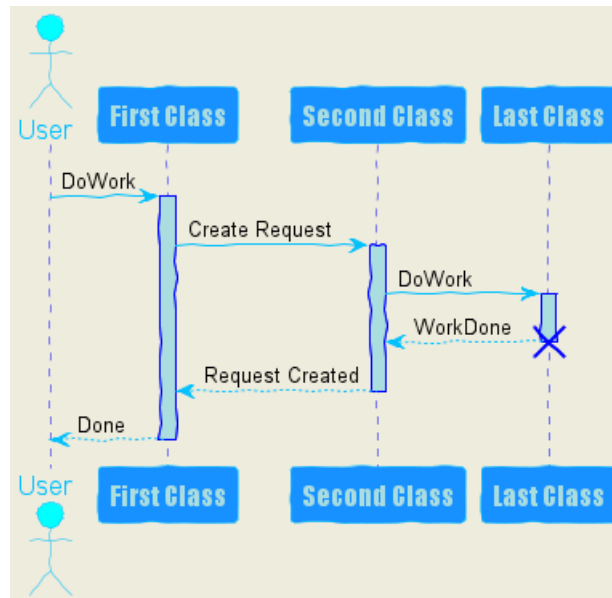
```

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



1.37 パディングの変更

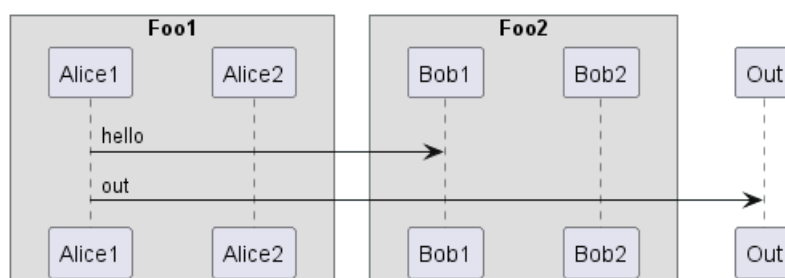
パディングの設定を変更することができます。

```

@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1
participant Alice2
end box
box "Foo2"
participant Bob1
participant Bob2
end box
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml

```



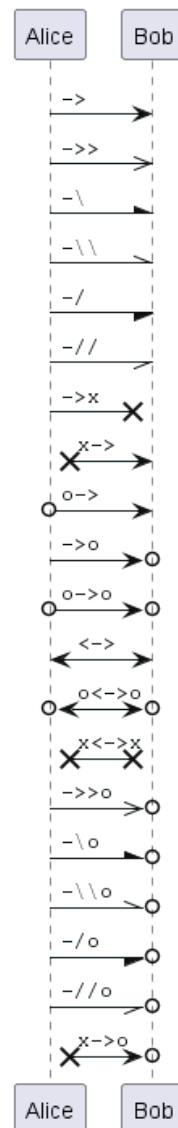
1.38 付録：全種類の矢印の例

1.38.1 通常の矢印

```

@startuml
participant Alice as a
participant Bob as b
a -> b : ""-> ""
a ->> b : ""->> ""
a -\ b : ""-\ ""
a -\\ b : ""-\\ ""
a -/ b : ""-/ ""
a -// b : ""-// ""
a ->x b : ""->x ""
a x-> b : ""x-> ""
a o-> b : ""o-> ""
a ->o b : ""->o ""
a o->o b : ""o->o ""
a <-> b : ""<-> ""
a o<->o b : ""o<->o ""
a x<->x b : ""x<->x ""
a ->>o b : ""->>o ""
a -\o b : ""-\o ""
a -\\o b : ""-\\o ""
a -/o b : ""-/o ""
a -//o b : ""-//o ""
a x->o b : ""x->o ""
@enduml

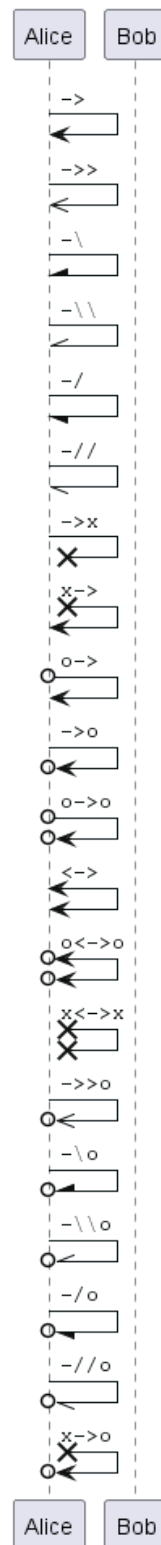
```



1.38.2 自分自身への矢印

```
@startuml
participant Alice as a
participant Bob as b
a -> a : ""-> ""
a ->> a : ""->> ""
a -\ a : ""-\ ""
a -\\ a : ""-\\\\ ""
a -/ a : ""-/ ""
a -// a : ""-// ""
a ->x a : ""->x ""
a x-> a : ""x-> ""
a o-> a : ""o-> ""
a ->o a : ""->o ""
a o->o a : ""o->o ""
a <-> a : ""<-> ""
a o<->o a : ""o<->o""
a x<->x a : ""x<->x""
a ->>o a : ""->>o ""
a -\o a : ""-\o ""
a -\\o a : ""-\\\\o""
```

```
a -/o      a : ""-/o  ""
a -//o     a : ""-//o ""
a x->o      a : ""x->o ""
@enduml
```



1.38.3 インとアウトのメッセージ ('[', ']')

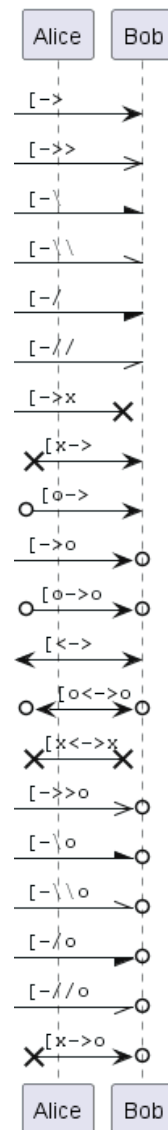
1.38.4 インのメッセージ (‘[’)

```
@startuml
```

```

participant Alice as a
participant Bob as b
[->      b : "" [->  ""
[->>     b : "" [->> ""
[-\      b : "" [-\   ""
[-\\     b : "" [-\\\\ ""
[-/      b : "" [-/   ""
[-//     b : "" [-//  ""
[->x     b : "" [->x  ""
[x->     b : "" [x->  ""
[o->     b : "" [o->  ""
[->o     b : "" [->o  ""
[o->o     b : "" [o->o ""
[<->     b : "" [<-> ""
[o<->o   b : "" [o<->o""
[x<->x   b : "" [x<->x""
[->>o    b : "" [->>o ""
[-\o     b : "" [-\o  ""
[-\\o    b : "" [-\\\\o""
[-/o     b : "" [-/o  ""
[-//o    b : "" [-//o ""
[x->o     b : "" [x->o ""
@enduml

```



1.38.5 アウトのメッセージ (']')

```

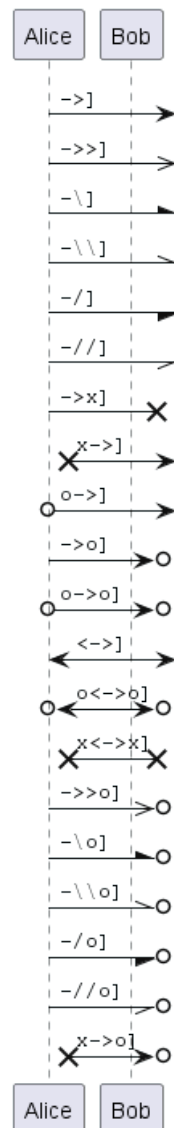
@startuml
participant Alice as a
participant Bob as b
a ->]      : ""->]      ""
a ->>]     : ""->>]     ""
a -\]      : ""-\]      ""
a -\\]     : ""-\\]     ""
a -/]      : ""-/]      ""
a -//]     : ""-//]     ""
a ->x]     : ""->x]     ""
a x->]     : ""x->]     ""
a o->]     : ""o->]     ""
a ->o]     : ""->o]     ""
a o->o]    : ""o->o]    ""
a <->]     : ""<->]     ""
a o<->o]   : ""o<->o]   ""
a x<->x]   : ""x<->x]   ""
a ->>o]    : ""->>o]    ""
a -\o]     : ""-\o]     ""
a -\\o]    : ""-\\o]    ""

```

```

a -/o]      : ""-/o]  ""
a -//o]     : ""-//o] ""
a x->o]     : ""x->o]  ""
@enduml

```



1.38.6 短いインとアウトのメッセージ ('?')

1.38.7 短いインのメッセージ ('?')

```

@startuml
participant Alice as a
participant Bob as b
a -> b : //Long long label//
?-> b : ""?->  ""
?->> b : ""?->>  ""
?-\ b : ""?-\  ""
?-\\ b : ""?-\\  ""
?-/ b : ""?-/  ""
?-// b : ""?-//  ""
?->x b : ""?->x  ""
?x-> b : ""?x->  ""
?o-> b : ""?o->  ""
?->o b : ""?->o  ""

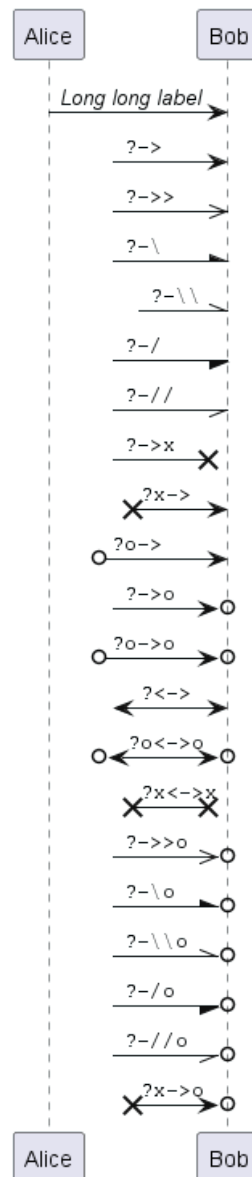
```



```

?o->o    b : ""?o->o ""
?<->     b : ""?<-> ""
?o<->o    b : ""?o<->o""
?x<->x    b : ""?x<->x""
?->>o     b : ""?->>o ""
?-\o      b : ""?-\o  ""
?-\ \o    b : ""?-\ \o ""
?-/o      b : ""?-/o  ""
?-//o     b : ""?-//o ""
?x->o     b : ""?x->o ""
@enduml

```



1.38.8 短いアウトのメッセージ ('?')

```

@startuml
participant Alice as a
participant Bob as b
a->>b : //Long long label//
a->? : ""->? ""
a->>? : ""->>? ""
a->\? : ""->\? ""

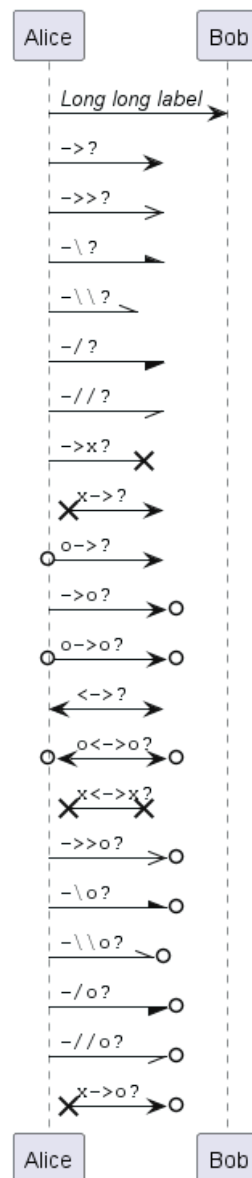
```



```

a -\\?      : "\"-\\\\\\?\""
a -/?       : "\"-/?  \""
a -//?      : "\"-//?  \""
a ->x?      : "\"->x?  \""
a x->?      : "\"x->?  \""
a o->?      : "\"o->?  \""
a ->o?      : "\"->o?  \""
a o->o?     : "\"o->o?  \""
a <->?      : "\"<->?  \""
a o<->o?    : "\"o<->o?\""
a x<->x?    : "\"x<->x?\""
a ->>o?     : "\"->>o?  \""
a -\\o?     : "\"-\\o?  \""
a -\\\\o?    : "\"-\\\\\\o?\""
a -/o?      : "\"-/o?  \""
a -//o?     : "\"-//o?  \""
a x->o?     : "\"x->o?  \""
@enduml

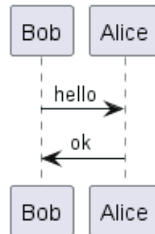
```



1.39 特有の skinparam

1.39.1 デフォルト

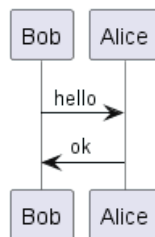
```
@startuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



1.39.2 ライフラインの設定 (lifelineStrategy)

- nosolid (デフォルト)

```
@startuml
skinparam lifelineStrategy nosolid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```

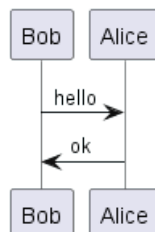


[Ref. QA-9016]

- solid

シーケンス図のライフラインを実線で表示するには、`skinparam lifelineStrategy solid` を設定します：

```
@startuml
skinparam lifelineStrategy solid
Bob -> Alice : hello
Alice -> Bob : ok
@enduml
```



[Ref. QA-2794]

1.39.3 厳密な UML スタイル (style strictuml)

厳密な UML に準拠する（矢印の端を矢じり形ではなく三角形にする等）には、次のようにします：

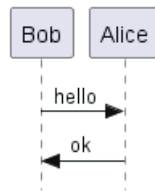
- `skinparam style strictuml`



```

@startuml
skinparam style strictuml
Bob -> Alice : hello
Alice -> Bob : ok
@enduml

```



[Ref. QA-1047]

1.40 未接続の分類子を表示しない

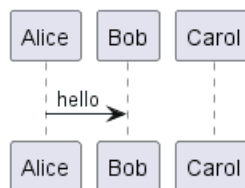
デフォルトでは、すべての分類子が表示されます。

```

@startuml
participant Alice
participant Bob
participant Carol

Alice -> Bob : hello
@enduml

```



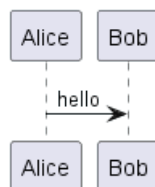
hide unlinked を指定すると、接続されていない分類子を非表示にできます。

```

@startuml
hide unlinked
participant Alice
participant Bob
participant Carol

Alice -> Bob : hello
@enduml

```



[Ref. QA-4247]

1.41 グループメッセージに色を付ける

グループメッセージに色を付けることができます:

```

@startuml
Alice -> Bob: Authentication Request
alt#Gold #LightBlue Successful case
    Bob -> Alice: Authentication Accepted
end

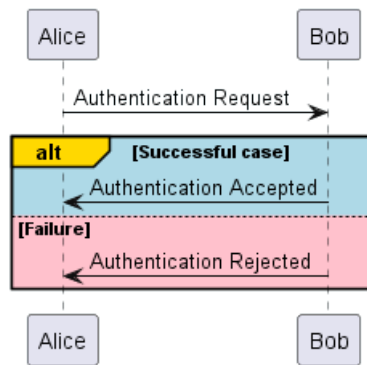
```



```

else #Pink Failure
    Bob -> Alice: Authentication Rejected
end
@enduml

```



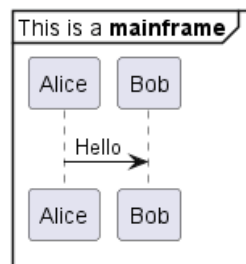
[Ref. QA-4750 and QA-6410]

1.42 メインフレーム

```

@startuml
mainframe This is a **mainframe**
Alice->Bob : Hello
@enduml

```



[Ref. QA-4019 and Issue#148]

1.43 Slanted or odd arrows

You can use the (nn) option (before or after arrow) to make the arrows slanted, where *nn* is the number of shift pixels.

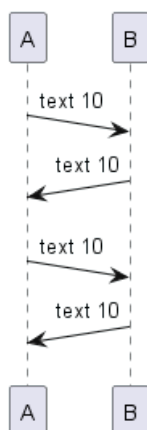
[Available only after v1.2022.6beta+]

```

@startuml
A ->(10) B: text 10
B ->(10) A: text 10

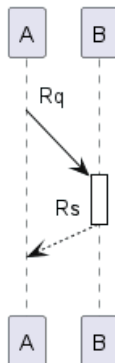
A ->(10) B: text 10
A (10)<- B: text 10
@enduml

```



```

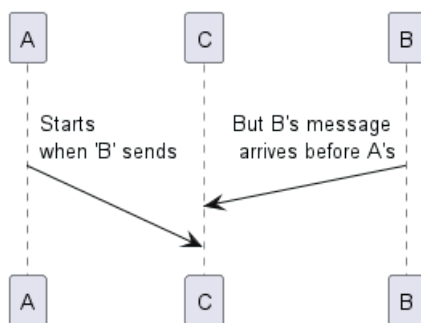
@startuml
A ->(40) B++: Rq
B -->(20) A--: Rs
@enduml
  
```



[Ref. QA-14145]

```

@startuml
!pragma teoz true
A ->(50) C: Starts\nwhen 'B' sends
& B ->(25) C: \nBut B's message\n arrives before A's
@enduml
  
```



[Ref. QA-6684]

```

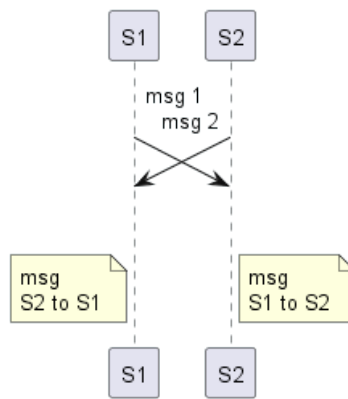
@startuml
!pragma teoz true

S1 ->(30) S2: msg 1\n
& S2 ->(30) S1: msg 2

note left S1: msg\nS2 to S1
& note right S2: msg\nS1 to S2
  
```



@endum1



[Ref. QA-1072]

2 ユースケース図

いくつかの例を示します。

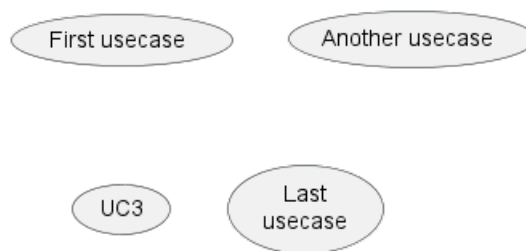
PlantUML offers a unique approach to creating use case diagrams through its text-based language. One of the primary advantages of using PlantUML is its **simplicity and efficiency**. Instead of manually drawing shapes and connections, users can define their diagrams using intuitive and concise textual descriptions. This not only speeds up the diagram creation process but also ensures **consistency and accuracy**. The ability to integrate with various documentation platforms and its wide range of supported output formats make PlantUML a versatile tool for both developers and non-developers. Lastly, being **open-source**, PlantUML boasts a [strong community](https://forum.plantuml.net/) that continually contributes to its improvement and offers a wealth of resources for users at all levels.

2.1 ユースケース

ユースケースは丸括弧で囲んで使います (丸括弧の対は楕円に似ているからです)。

`usecase` キーワードを使ってユースケースを定義することもできます。 `as` キーワードを使ってエイリアスを定義することもできます。このエイリアスはあとで、ユースケースの関係を定義するために使います。

```
@startuml
(First usecase)
(Another usecase) as (UC2)
usecase UC3
usecase (Last\nusecase) as UC4
@enduml
```



2.2 アクター

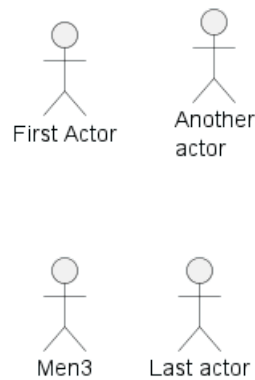
アクターは 2 つのコロンで囲まれます。

`actor` キーワードを使ってアクターを定義することもできます。 `as` キーワードを使ってエイリアスを定義することもできます。このエイリアスはあとで、ユースケースの関係を定義するために使います。

後から説明しますが、アクターの定義は必須ではありません。

```
@startuml
:First Actor:
:Another\nactor: as Men2
actor Men3
actor :Last actor: as Men4
@enduml
```





2.3 アクターのスタイルを変更する

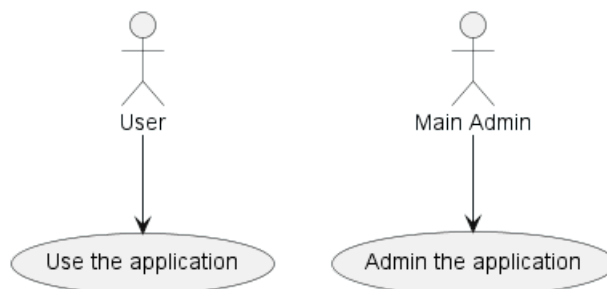
アクターのスタイルを、デフォルトの棒人間以外に変更できます：

- `skinparam actorStyle awesome` コマンドで、awesome man スタイル
- `skinparam actorStyle hollow` コマンドで、hollow man スタイル

2.3.1 棒人間 (デフォルト)

```

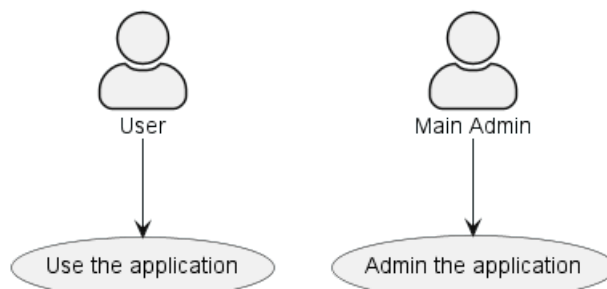
@startuml
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
  
```



2.3.2 Awesome man

```

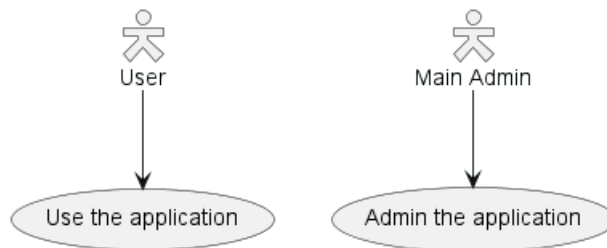
@startuml
skinparam actorStyle awesome
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
  
```



[Ref. QA-10493]

2.3.3 Hollow man

```
@startuml
skinparam actorStyle Hollow
:User: --> (Use)
"Main Admin" as Admin
"Use the application" as (Use)
Admin --> (Admin the application)
@enduml
```



[Ref. PR#396]

2.4 ユースケースの説明

クオート記号を使うことにより、複数行にわたる説明を記述できます。

また、次の区切り記号を使用できます：

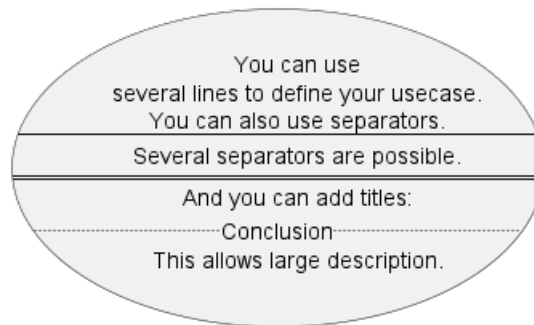
- -- (ダッシュ)
- .. (ピリオド)
- == (イコール)
- __ (アンダースコア)

これらのペアで囲んで、その間にテキストを記述することで、区切り記号の中にタイトルを記入できます。

```
@startuml

usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.
--
Several separators are possible.
==
And you can add titles:
..Conclusion..
This allows large description."

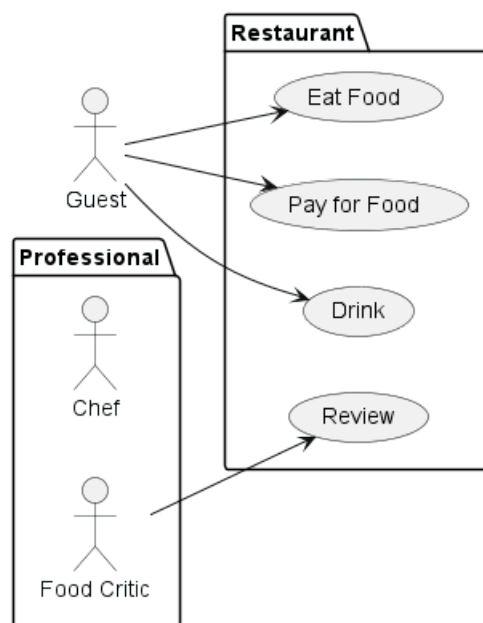
@enduml
```



2.5 パッケージ

パッケージを使用して、アクターやユースケースをグループ化できます。

```
@startuml
left to right direction
actor Guest as g
package Professional {
    actor Chef as c
    actor "Food Critic" as fc
}
package Restaurant {
    usecase "Eat Food" as UC1
    usecase "Pay for Food" as UC2
    usecase "Drink" as UC3
    usecase "Review" as UC4
}
fc --> UC4
g --> UC1
g --> UC2
g --> UC3
@enduml
```



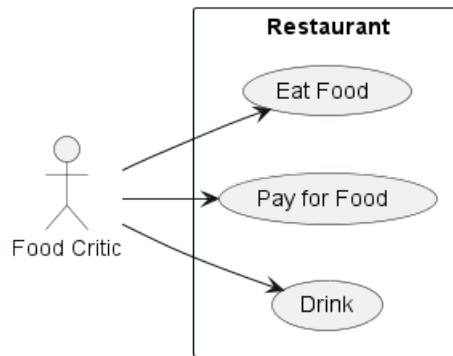
rectangle を使用するとパッケージの見た目を変更できます。

```
@startuml
left to right direction
```

```

actor "Food Critic" as fc
rectangle Restaurant {
    usecase "Eat Food" as UC1
    usecase "Pay for Food" as UC2
    usecase "Drink" as UC3
}
fc --> UC1
fc --> UC2
fc --> UC3
@enduml

```



2.6 簡単な例

アクターとユースケースを繋げるには --> 矢印を使います。

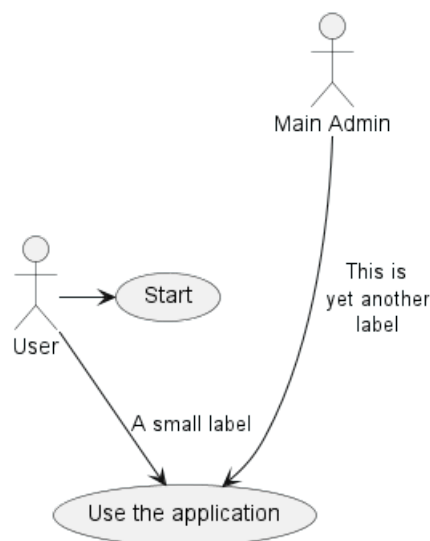
矢印に使うハイフン - の数を増やすと矢印を長くできます。矢印の定義に : を使うことにより矢印にラベルをつけることができます。

以下の例では *User* は定義なしにアクターとして使われています。

```

@startuml
User -> (Start)
User --> (Use the application) : A small label
:Main Admin: ---> (Use the application) : This is\nyet another\nlabel
@enduml

```



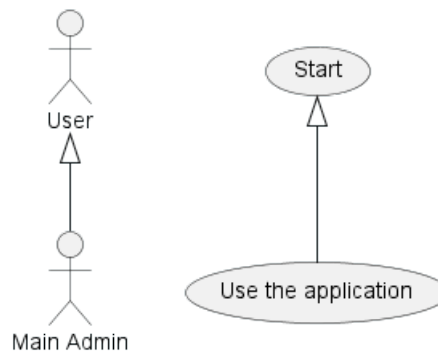
2.7 継承

もしアクターやユースケースが継承をする場合には、<|-- 記号を使います。

```
@startuml
:Main Admin: as Admin
(Use the application) as (Use)
```

```
User <|-- Admin
(Start) <|-- (Use)
```

```
@enduml
```



2.8 ノートの使用方法

オブジェクトに関連のあるノートを作成するには `note left of`、`note right of`、`note top of`、`note bottom of` キーワードを使います。

または `note` キーワードを使ってノートを作成し、`..` 記号を使ってオブジェクトに紐づけることができます。

```
@startuml
:Main Admin: as Admin
(Use the application) as (Use)
```

```
User -> (Start)
```

```
User --> (Use)
```

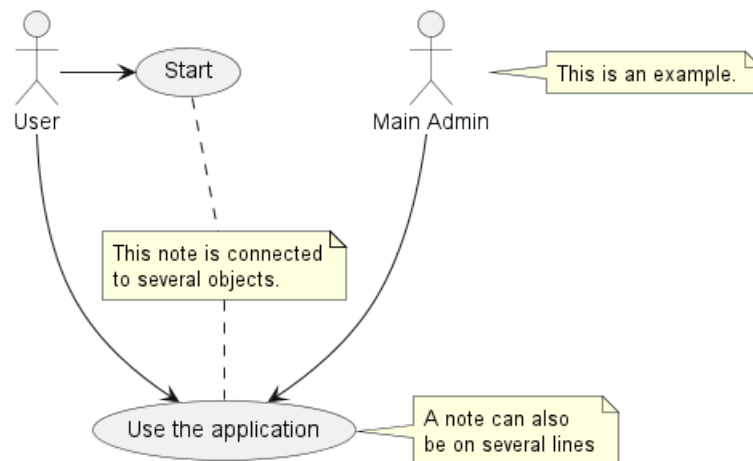
```
Admin ---> (Use)
```

```
note right of Admin : This is an example.
```

```
note right of (Use)
  A note can also
  be on several lines
end note
```

```
note "This note is connected\nto several objects." as N2
(Start) .. N2
N2 .. (Use)
@enduml
```





2.9 ステレオタイプ

<< と >> を使い、アクターとユースケースを定義中にステレオタイプを追加できます。

```

@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

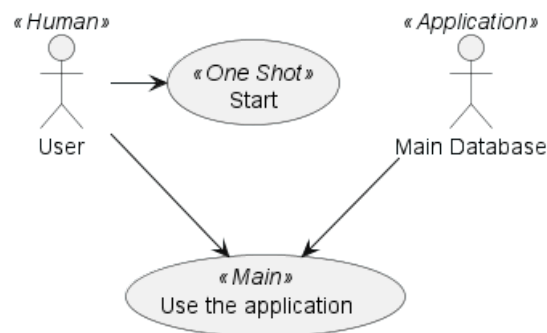
```

```
User -> (Start)
```

```
User --> (Use)
```

```
MySql --> (Use)
```

```
@enduml
```



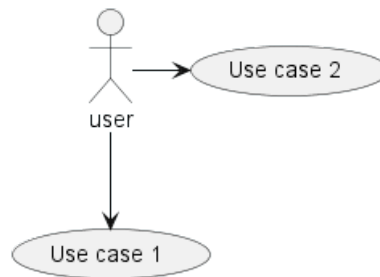
2.10 矢印の方向を変えるには

デフォルトでは、クラス間の線は 2 個のハイフン -- で表され、縦方向につながります。横方向の線を描くには以下のようにハイフン 1 つかドット 1 つを書きます。

```

@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml

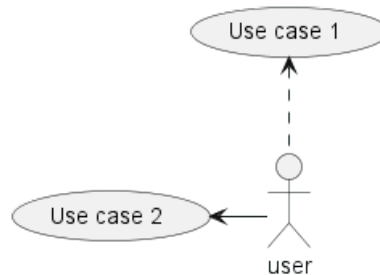
```



線を反対にすることでも方向を変えることができます。

```

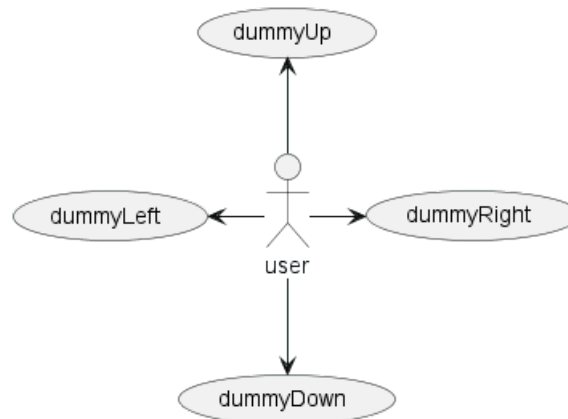
@startuml
  (Use case 1) <.. :user:
  (Use case 2) <- :user:
@enduml
  
```



矢印の内側に left、right、up、down を書くことによって線の方向を変えられます。

```

@startuml
  :user: -left-> (dummyLeft)
  :user: -right-> (dummyRight)
  :user: -up-> (dummyUp)
  :user: -down-> (dummyDown)
@enduml
  
```



例えば、`-down-` ではなく `-d-` など、各方向の頭文字、または頭 2 文字 (`-do-`) だけ使って矢印を短く記述することも出来ます。

ただし、この機能の使いすぎには注意しましょう。ほとんどの場合、特別なことをしなくても *Graphviz* がその場にあった表示を選びます。

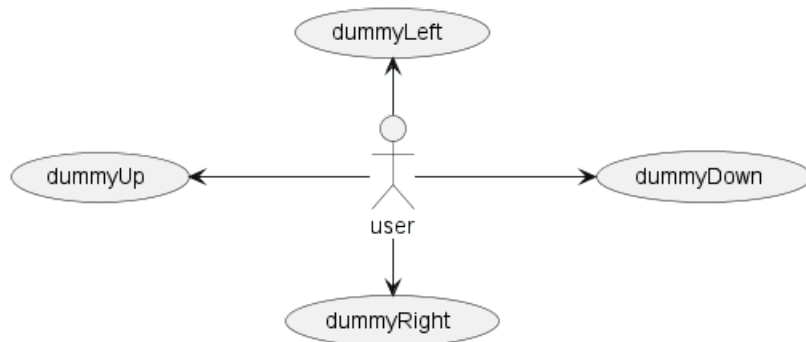
`left to right direction` パラメータを使用した場合は、次のようになります：

```

@startuml
  left to right direction
  :user: -left-> (dummyLeft)
  :user: -right-> (dummyRight)
  
```



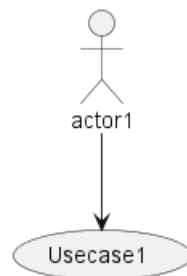
```
:user: -up-> (dummyUp)
:user: -down-> (dummyDown)
@enduml
```



2.11 図を分割する

newpage キーワードは、いくつかのページや画像に図を分割します。

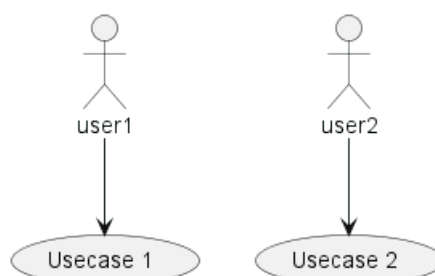
```
@startuml
:actor1: --> (Usecase1)
newpage
:actor2: --> (Usecase2)
@enduml
```



2.12 左から右に描画する

デフォルトの作図方向は **top to bottom** となっています。

```
@startuml
'default
top to bottom direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)
@enduml
```



作図方向を **left to right** に変更するには **left to right direction** コマンドを使います。

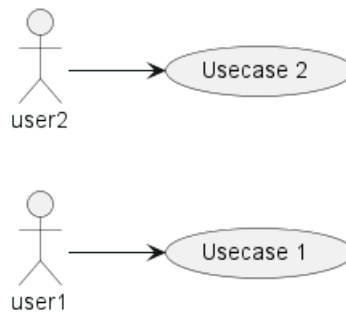
```
@startuml
```

```

left to right direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)

```

```
@enduml
```



2.13 スキン設定 (Skinparam)

ダイアグラムの色やフォントを変更するには skinparam コマンドを使用します。

このコマンドは以下の場面で使用できます。

- ダイアグラム定義内で他のコマンドを同様に。
- インクルードされたファイル内。
- 設定ファイルのコマンドライン内や ANT タスク内。

個別のステレオタイプ付きアクターやユースケースにそれぞれ色やフォントを定義することができます。

```

@startuml
skinparam handwritten true

skinparam usecase {
  BackgroundColor DarkSeaGreen
  BorderColor DarkSlateGray

  BackgroundColor<< Main >> YellowGreen
  BorderColor<< Main >> YellowGreen

  ArrowColor Olive
  ActorBorderColor black
  ActorFontName Courier

  ActorBackgroundColor<< Human >> Gold
}

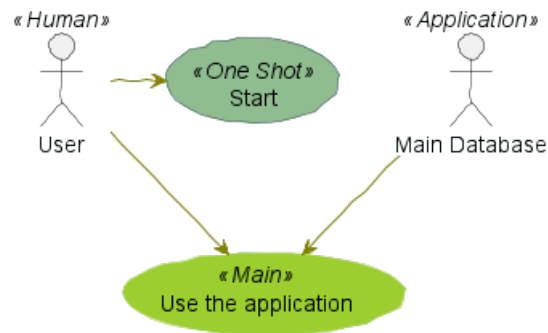
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)

@enduml

```

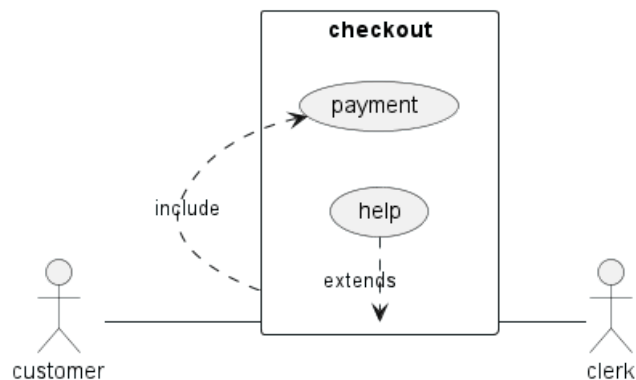


2.14 完全な例

```

@startuml
left to right direction
skinparam packageStyle rectangle
actor customer
actor clerk
rectangle checkout {
  customer -- (checkout)
  (checkout) .> (payment) : include
  (help).> (checkout) : extends
  (checkout) -- clerk
}
@enduml

```



2.15 ビジネスユースケース

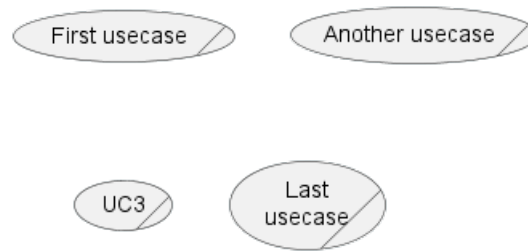
/を加えると、ビジネスユースケースを作成できます。

2.15.1 ビジネスユースケース

```

@startuml
(First usecase)/
(Another usecase)/ as UC2
usecase/ UC3
usecase/ (Last\nusecase) as UC4
@enduml

```

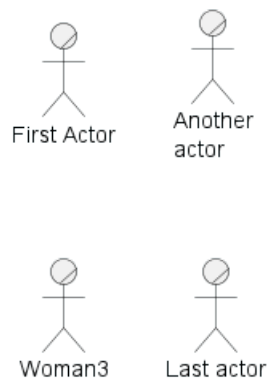


2.15.2 ビジネスアクター

@startuml

```
:First Actor:/
:Another\nactor:/ as Man2
actor/ Woman3
actor/ :Last actor: as Person1
```

@enduml



[Ref. QA-12179]

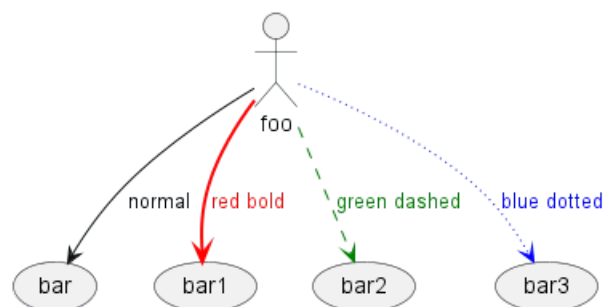
2.16 矢印の色とスタイルを変更する (インラインスタイル)

個別の矢印ごとに色とスタイルを変更するには、次の記法を使用します：

- #color;line.[bold|dashed|dotted];text:color

@startuml

```
actor foo
foo --> (bar) : normal
foo --> (bar1) #line:red;line.bold;text:red : red bold
foo --> (bar2) #green;line.dashed;text:green : green dashed
foo --> (bar3) #blue;line.dotted;text:blue : blue dotted
@enduml
```



[Ref. QA-3770 and QA-3816] [配置図、クラス図の同様の機能を参照]

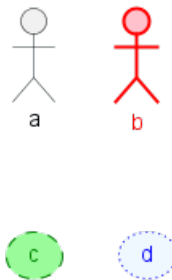


2.17 要素の色とスタイルを変更する（インラインスタイル）

個別の要素ごとに色とスタイルを変更するには、次の記法を使用します：

- `#[color|back:color];line:color;line.[bold|dashed|dotted];text:color`

```
@startuml
actor a
actor b #pink;line:red;line.bold;text:red
usecase c #palegreen;line:green;line.dashed;text:green
usecase d #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



[Ref. QA-5340 and adapted from QA-6852]

2.18 Display JSON Data on Usecase diagram

2.18.1 Simple example

```
@startuml
allowmixing

actor Actor
usecase Usecase

json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.



3 クラス図

クラス図は、プログラミング言語で伝統的に採用されている構文を真似て設計されています。

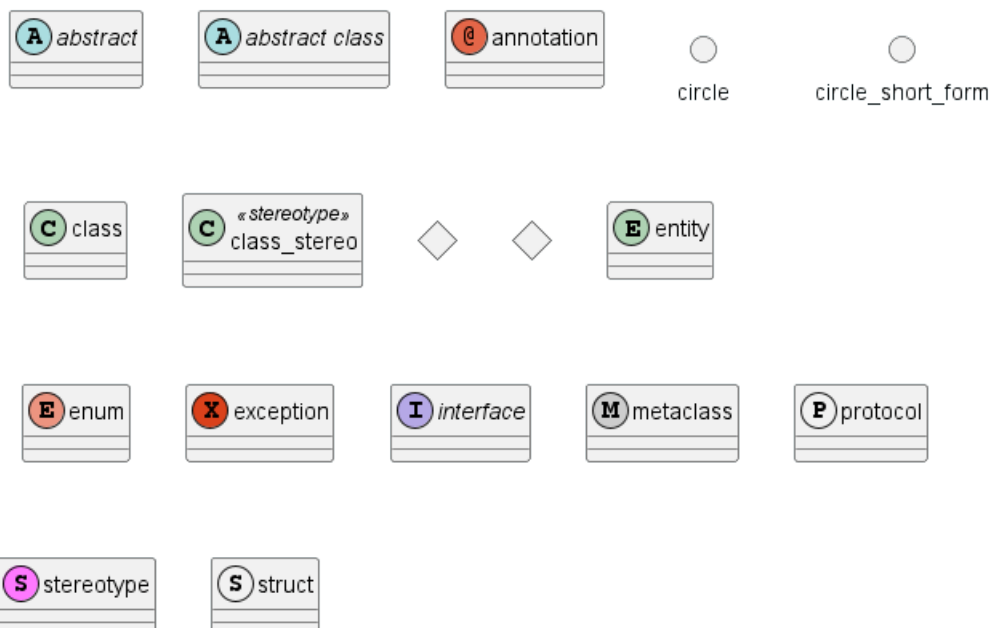
この設計手法は簡潔であるだけでなく、簡潔かつ表現力豊かな表現を作成することができます。さらに、シーケンス図と同じ構文によってクラス間の関係を表示することができ、クラスの相互作用を流動的かつ洞察的に描写するための道を開きます。

構造的な表現や関係的な表現だけでなく、クラス図の構文では、注釈の挿入や色の適用などのさらなる拡張がサポートされており、ユーザーは情報量が多く視覚的に魅力的な図を作成することができます。

図の作成体験を向上させる PlantUML の一般的なコマンドについて、さらに詳しく知ることができます。

3.1 宣言する要素

```
@startuml
abstract          abstract
abstract class    "abstract class"
annotation        annotation
circle            circle
()                circle_short_form
class             class
class             class_stereo <<stereotype>>
diamond           diamond
<>               diamond_short_form
entity            entity
enum              enum
exception          exception
interface          interface
metaclass          metaclass
protocol           protocol
stereotype         stereotype
struct            struct
@enduml
```



[*protocol, struct* の場合:GH-1028、*exception* の場合:QA-16258] を参照。



3.2 クラス間の関係

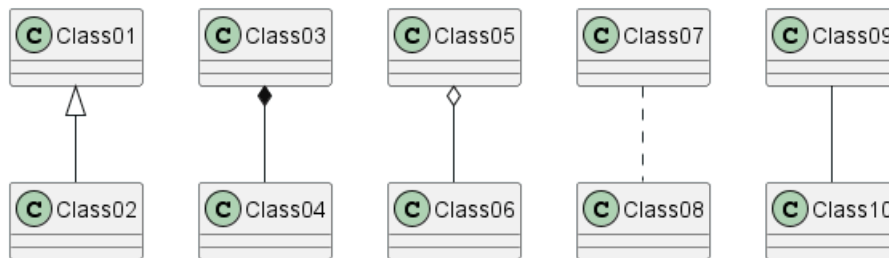
クラス間の関係は次の記号を使用して定義されています:

Type	Symbol	Drawing
Extension	< --	
Composition	*--	
Aggregation	o--	

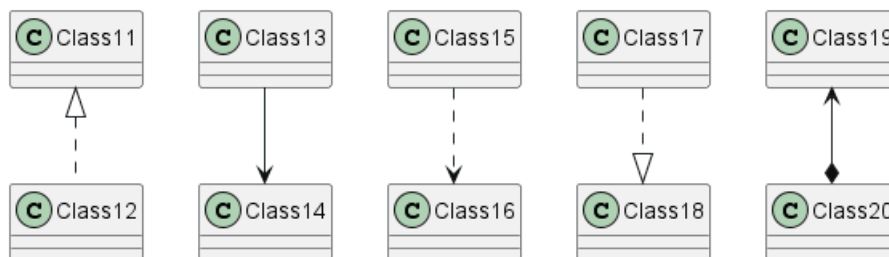
-- を .. に置き換えると点線にできます。

これらのルールを知ること、以下の図面を描くことができます:

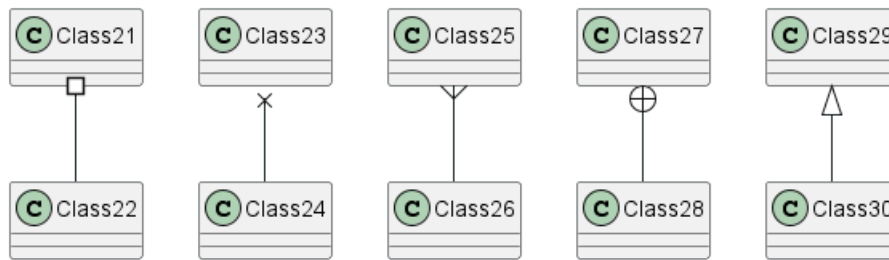
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```



```
@startuml
Class21 #-- Class22
Class23 x-- Class24
Class25 }-- Class26
Class27 +-- Class28
Class29 ^-- Class30
@enduml
```



3.3 関係のラベル

: にテキストを続けることによって、関係へラベルを追加することが可能です。

多重度を示す為に関係のそれぞれの側にダブルクォーテーション"" を使うことができます。

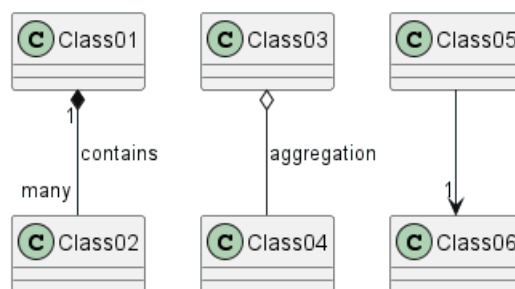
```
@startuml
```

```
Class01 "1" *-- "many" Class02 : contains
```

```
Class03 o-- Class04 : aggregation
```

```
Class05 --> "1" Class06
```

```
@enduml
```



ラベルの最初または最後に < か > を使って、他のオブジェクトへの関係を示す矢印を追加できます。

```
@startuml
```

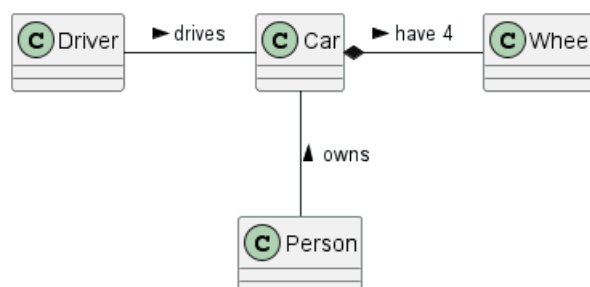
```
class Car
```

```
Driver - Car : drives >
```

```
Car *- Wheel : have 4 >
```

```
Car -- Person : < owns
```

```
@enduml
```



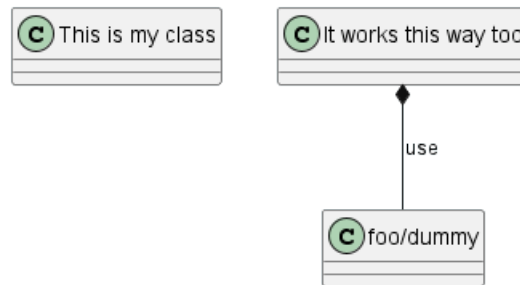
3.4 要素名と関係のラベルでの非文字の使用

クラス（または列挙型など）の表示に文字以外を使用したい場合は、次のいずれかの方法ですることができます：

- クラス定義でキーワード `as` を使用して別名を付ける
- クラス名の前後に引用符 `"` を入れる

```
@startuml
class "This is my class" as class1
class class2 as "It works this way too"

class2 *-- "foo/dummy" : use
@enduml
```



要素に別名を付けた場合は、ファイルのその他の場所では別名を使って要素を参照する必要があります。

3.4.1 \$ で始まる名前

名前が `$` で始まる場合、後で要素を非表示にしたり削除したりできません。なぜなら `hide` と `remove` のコマンドは `$` で始まる文字列をコンポーネントの名前ではなくタグ (`$tag`) として扱うからです。そのような要素を削除するには、別名を付けるかタグを付ける必要があります。

```
@startuml
class $C1
class $C2 $C2
class "$C2" as dollarC2
remove $C1
remove $C2
remove dollarC2
@enduml
```



また、`$` で始まる名前は有効ですが、そのような要素に別名を付けるには、クォーテーション`"` で囲む必要があることにも注意してください。

3.5 メソッドの追加

`:` に続けてフィールド名やメソッド名を記述すると、フィールドやメソッドを宣言できます。

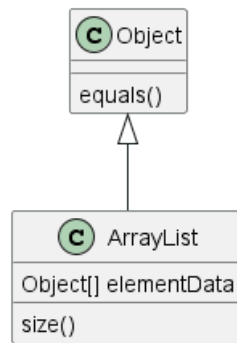
システムは括弧をチェックしてメソッドとフィールドのどちらなのかを選択します。

```
@startuml
Object <|-- ArrayList

Object : equals()
ArrayList : Object[] elementData
ArrayList : size()

@enduml
```





波括弧 `{}` を使って、フィールドやメソッドをくくることができます。

構文はタイプや名前の順番について非常に柔軟であることに注意してください。

```

@startuml
class Dummy {
    String data
    void methods()
}

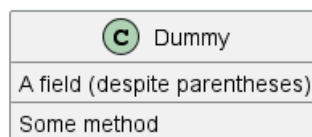
class Flight {
    flightNumber : Integer
    departureTime : Date
}
@enduml
  
```



`{field}` や `{method}` 修飾子を用いれば、構文によりフィールドやメソッドだと通常は解釈されるものを強制的に変更することができます。

```

@startuml
class Dummy {
    {field} A field (despite parentheses)
    {method} Some method
}
@enduml
  
```



3.6 可視性の定義

メソッドやフィールドを定義するときに対応する項目の可視性を定義する記号を使用することができます。

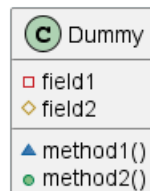
文字	フィールドのアイコン	メソッドのアイコン	可視性
-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public



```

@startuml
class Dummy {
    -field1
    #field2
    ~method1()
    +method2()
}
@enduml

```

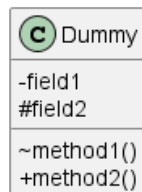


コマンド `skinparam classAttributeIconSize 0` を使用してこの機能を切ることができます。

```

@startuml
skinparam classAttributeIconSize 0
class Dummy {
    -field1
    #field2
    ~method1()
    +method2()
}
@enduml

```



3.7 Abstract と Static

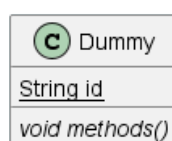
静的または抽象的なメソッドまたはフィールドは `{static}` または `{abstract}` 修飾子を使用することで定義することができます。

これらの修飾子は行の始めまたは終りに使用することができます。`{static}` の代わりに `{classifier}` もまた使用できます。

```

@startuml
class Dummy {
    {static} String id
    {abstract} void methods()
}
@enduml

```



3.8 高等なクラス本体

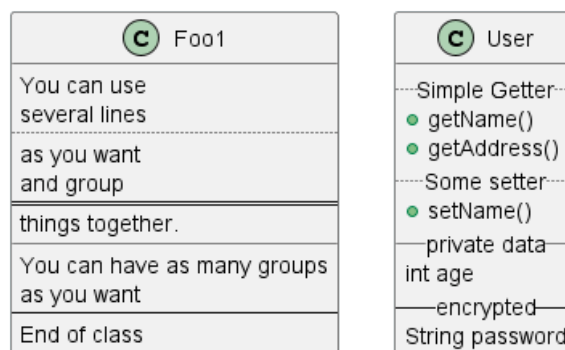
デフォルトでは、メソッドやフィールドは PlantUML によって自動再編成されます。メソッドやフィールドに独自の順序付けを定義するためのセパレータを使用できます。以下のセパレータが使用できます: -- .. == __

セパレータ内でタイトルを使用することもできます:

```
@startuml
class Foo1 {
    You can use
    several lines
    ..
    as you want
    and group
    ==
    things together.
    --
    You can have as many groups
    as you want
    --
    End of class
}

class User {
    .. Simple Getter ..
    + getName()
    + getAddress()
    .. Some setter ..
    + setName()
    __ private data __
    int age
    -- encrypted --
    String password
}

@enduml
```



3.9 注釈とステレオタイプ

ステレオタイプは、キーワード `class` に `<<` と `>>` で定義されます。

注釈の定義には、キーワード `note left of`, `<code>note right of</code>`

, `note top of`, `note bottom of` も使用できます。

クラス定義の最後には `note left`, `note right`, `note top`, `note bottom` も使用できます。

注釈は、キーワード `note` とで単独に定義することができ、記号 `..` を使用して他のオブジェクトとリンクすることもできます。



```

@startuml
class Object << general >>
Object <|--- ArrayList

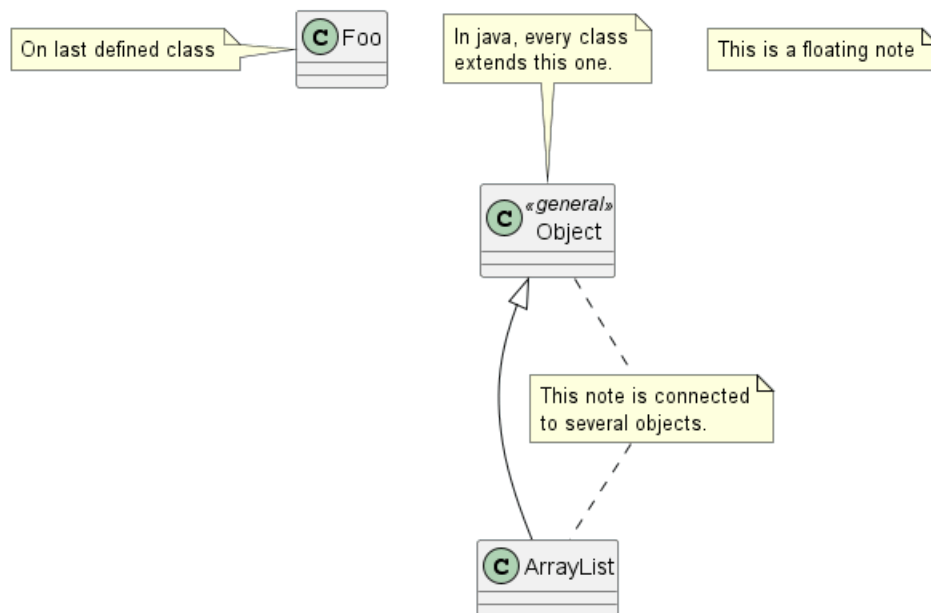
note top of Object : In java, every class\nextends this one.

note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml

```



3.10 注釈の詳細

次のようないくつかの HTML タグを使用することも可能です (Creole 表現を参照) :

- `<b>`
- `<u>`
- `<i>`
- `<s>`, `<del>`, `<strike>`
- `<font color="#AAAAAA">` or `<font color="colorName">`
- `<color:#AAAAAA>` or `<color:colorName>`
- `<size:nn>` to change font size
- `` or `<img:file>`: the file must be accessible by the filesystem

また、複数行にまたがる注釈も可能です。

クラス定義の最後には `note left`, `note right`, `note top`, `note bottom` も使用できます。

```

@startuml

class Foo
note left: On last defined class

```



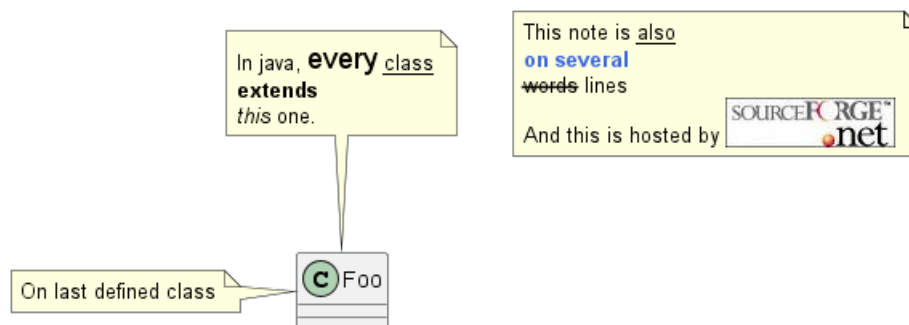
```

note top of Foo
  In java, <size:18>every</size> <u>class</u>
  <b>extends</b>
  <i>this</i> one.
end note

note as N1
  This note is <u>also</u>
  <b><color:royalBlue>on several</color>
  <s>words</s> lines
  And this is hosted by <img:sourceforge.jpg>
end note

@enduml

```



3.11 フィールド (フィールド、属性、メンバー) またはメソッドへの注釈

フィールド (フィールド、属性、メンバー) またはメソッドに注釈を追加することができます。

3.11.1 制限事項

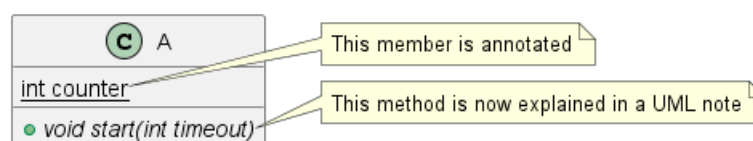
- top, bottom を指定することはできません (left と right のみ実装されています)
- 名前空間の区切り文字列 (namespaceSeparator) を :: に設定した場合、この機能は使えません。

3.11.2 フィールドまたはメンバーへの注釈

```

@startuml
class A {
{static} int counter
+void {abstract} start(int timeout)
}
note right of A::counter
  This member is annotated
end note
note right of A::start
  This method is now explained in a UML note
end note
@enduml

```

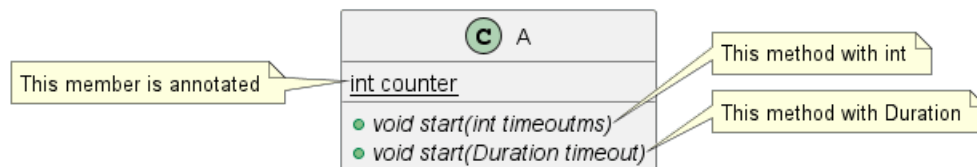


3.11.3 同名のメソッドへの注釈

```

@startuml
class A {
{static} int counter
+void {abstract} start(int timeouts)
+void {abstract} start(Duration timeout)
}
note left of A::counter
  This member is annotated
end note
note right of A::"start(int timeouts)"
  This method with int
end note
note right of A::"start(Duration timeout)"
  This method with Duration
end note
@enduml

```



[Ref. QA-3474 and QA-5835]

3.12 リンクへの注釈

リンク定義の直後に `note on link` を使用して、リンクに注釈を加えることが可能です。

もし注釈の相対位置を変えたい場合には、ラベル `note left on link`, `note right on link`, `note top on link`, `note bottom on link` も使用できます。

```

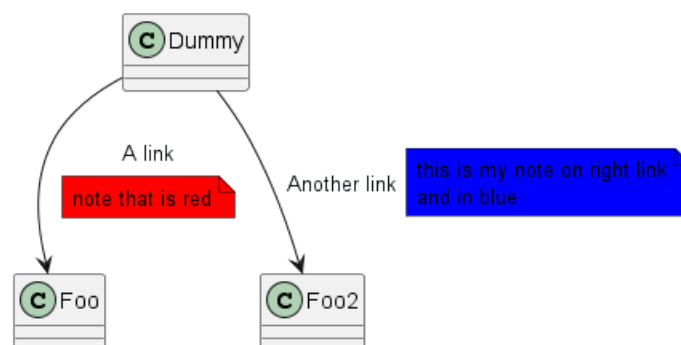
@startuml

class Dummy
Dummy --> Foo : A link
note on link #red: note that is red

Dummy --> Foo2 : Another link
note right on link #blue
this is my note on right link
and in blue
end note

@enduml

```



3.13 抽象クラスとインタフェース

抽象クラスは、キーワード `abstract` または `abstract class` を使用して宣言できます。

そのクラスはイタリック体で印字されます。

キーワード `interface`, `annotation` と `enum` も使用できます。

```
@startuml

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

List <|-- AbstractList
Collection <|-- AbstractCollection

Collection <|-- List
AbstractCollection <|-- AbstractList
AbstractList <|-- ArrayList

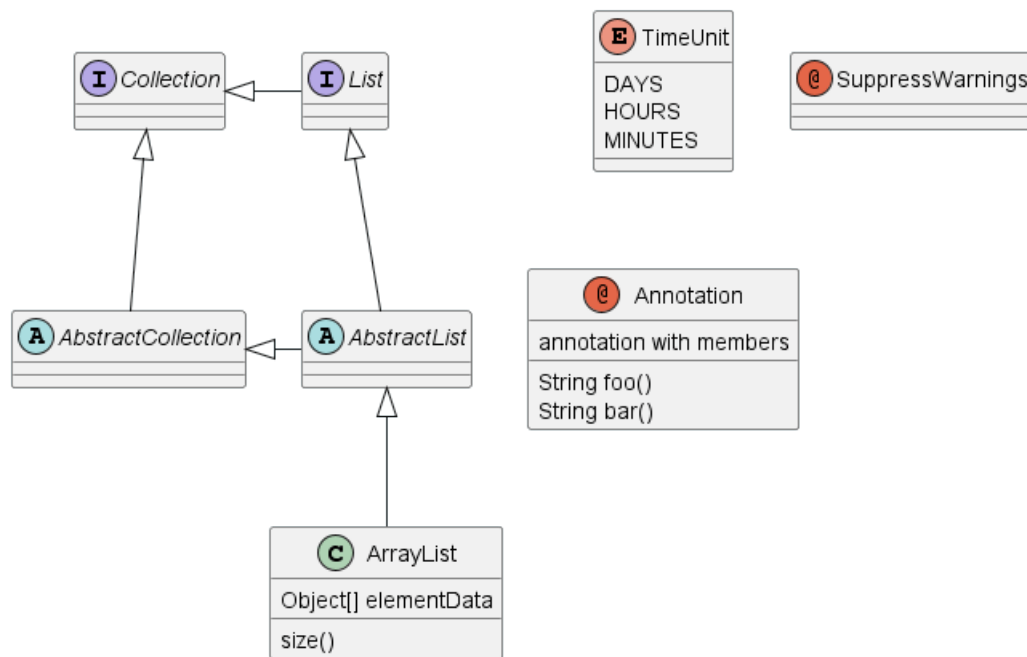
class ArrayList {
    Object[] elementData
    size()
}

enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}

annotation SuppressWarnings

annotation Annotation {
    annotation with members
    String foo()
    String bar()
}

@enduml
```



[Ref. 'Annotation with members' Issue#458]

3.14 属性、メソッド等の非表示

コマンド `hide/show` を使用して、クラスの表示をパラメータ化できます。

基本のコマンドは `hide empty members` です。このコマンドは属性やメソッドが空の場合に非表示にします。

`empty members` の代わりに使用することができます：

- `empty fields` または `empty attributes` は空のフィールドに、
- `empty methods` は空のメソッドに、
- `fields` または `attributes` は、それらが記述されていても非表示になります、
- `methods` はメソッドが記述されていても非表示になります、
- `members` はフィールドとメソッドが記述されていても非表示になります、
- `circle` はクラス名の前の丸で囲んだ文字に、
- `stereotype` はステレオタイプに。

キーワード `hide` または `show` のすぐ後ろに提供することもできます：

- `class` は全てのクラスに、
- `interface` は全てのインタフェースに、
- `enum` は全ての列挙型に、
- `<<foo1>>` は `foo1` でステレオタイプ化されたクラスに、
- 既存のクラス名。

コマンド `show/hide` をルールや例外の定義にそれぞれ使用することができます。

@startuml

```

class Dummy1 {
    +myMethods()
}
  
```

```

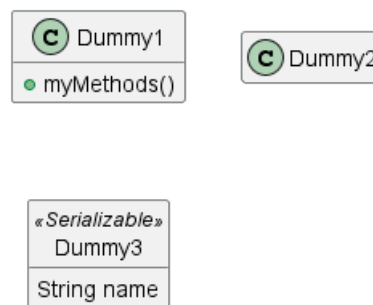
class Dummy2 {
    +hiddenMethod()
}

class Dummy3 <<Serializable>> {
    String name
}

hide members
hide <<Serializable>> circle
show Dummy1 methods
show <<Serializable>> fields

@enduml

```



3.15 非表示クラス

コマンド `show/hide` でクラスを非表示にすることができます。

これは大規模なインクルードファイルを定義する場合で、ファイルのインクルードの後でいくつかのクラスを非表示にしたい場合に有用である可能性が有ります。

```

@startuml

class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml

```



3.16 クラスの削除

コマンド `remove` でクラスを削除することができます。

これは大規模なインクルードファイルを定義する場合で、ファイルのインクルードの後でいくつかのクラスを削除したい場合に有用である可能性が有ります。

```

@startuml

```

```

class Foo1
class Foo2

Foo2 *-- Foo1

remove Foo2

@enduml

```



3.17 タグ付き要素またはワイルドカードの非表示、削除、復元

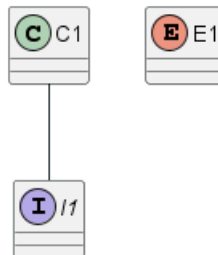
\$ を使用して要素にタグ (\$tags) を付けると、個別またはタグ単位でコンポーネントを削除、非表示、復元できます。

デフォルトでは、すべてのコンポーネントが表示されます：

```

@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1
@enduml

```



しかし、次のことができます：

- `hide $tag13` でコンポーネントを非表示：

```

@startuml
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1

hide $tag13
@enduml

```



- `remove $tag13` でコンポーネントを削除：

```

@startuml

```



```
class C1 $tag13
enum E1
interface I1 $tag13
C1 -- I1

remove $tag13
@enduml
```



- remove \$tag13 と restore \$tag1 でコンポーネントを復元 :

```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1

remove $tag13
restore $tag1
@enduml
```



- remove * と restore \$tag1 でコンポーネントを復元 :

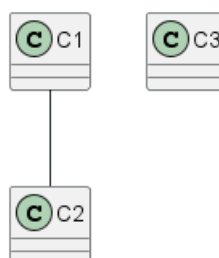
```
@startuml
class C1 $tag13 $tag1
enum E1
interface I1 $tag13
C1 -- I1

remove *
restore $tag1
@enduml
```

3.18 孤立したクラスを非表示または削除する

デフォルトでは、すべてのクラスが表示されます :

```
@startuml
class C1
class C2
class C3
C1 -- C2
@enduml
```



- しかし、hide @unlinked で、孤立したクラスを非表示にすることができます :

```
@startuml
class C1
class C2
class C3
C1 -- C2

hide @unlinked
@enduml
```



- もしくは、`remove @unlinked` で、孤立したクラスを削除できます：

```
@startuml
class C1
class C2
class C3
C1 -- C2

remove @unlinked
@enduml
```



[Adapted from QA-11052]

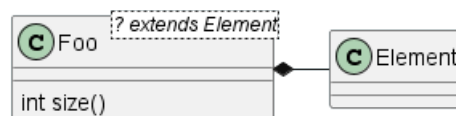
3.19 ジェネリクスの使用

括弧 < と > を使用してジェネリクスの使用をクラスに定義できます。

```
@startuml

class Foo<? extends Element> {
    int size()
}
Foo *- Element

@enduml
```



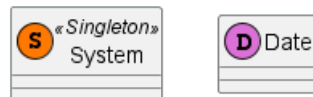
この描画は `skinparam genericDisplay old` コマンドにより非表示にすることができます。

3.20 特殊な目印

通常、目印文字 (C,I,E,A) は、クラス、インターフェイス、列挙型と抽象クラスのために使用されます。しかし、つぎの例のように単一の文字と色を追加し、ステレオタイプを定義するクラスに独自の目印を作成することができます：

```
@startuml
```

```
class System << (S,#FF7700) Singleton >>
class Date << (D,orchid) >>
@enduml
```



3.21 パッケージ

キーワード **package** を使用してパッケージを定義でき、必要に応じてパッケージの背景色 (HTML カラーコードまたは名前) を宣言します。

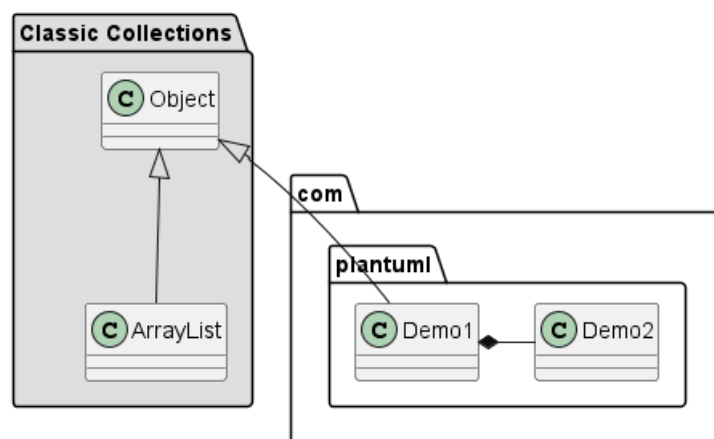
パッケージ定義は入れ子にできることに注意してください。

```
@startuml
```

```
package "Classic Collections" #DDDDDD {
    Object <|-- ArrayList
}
```

```
package com.plantum1 {
    Object <|-- Demo1
    Demo1 *-- Demo2
}
```

```
@enduml
```



3.22 パッケージスタイル

パッケージに利用可能なさまざまなスタイルがあります。

コマンド **skinparam packageStyle** を使用してデフォルトのスタイルを設定する、またはパッケージのステレオタイプを使用する、のどちらかで指定することができます。or by using a stereotype on the package:

```
@startuml
```

```
scale 750 width
```



```

package foo1 <<Node>> {
    class Class1
}

package foo2 <<Rectangle>> {
    class Class2
}

package foo3 <<Folder>> {
    class Class3
}

package foo4 <<Frame>> {
    class Class4
}

package foo5 <<Cloud>> {
    class Class5
}

package foo6 <<Database>> {
    class Class6
}

@enduml

```



次の例のように、パッケージ間のリンクを定義することもできます：

```

@startuml

skinparam packageStyle rectangle

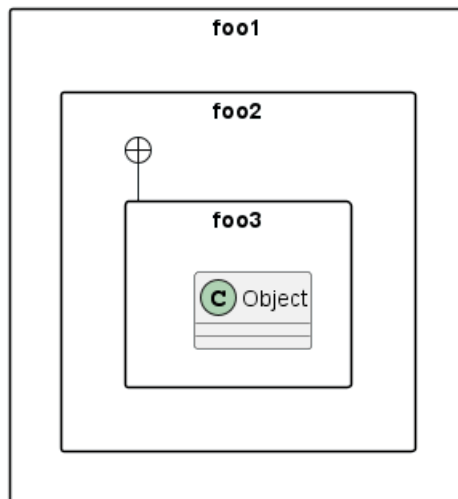
package foo1.foo2 {

package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml

```



3.23 名前空間

バージョン 1.2023.2（ベータ版としてオンライン中）から、PlantUML は名前空間とパッケージの扱いを変えます。

名前空間とパッケージの違いはもうありません：両方のキーワードは今や同義語です。

3.24 自動的にパッケージを作成する

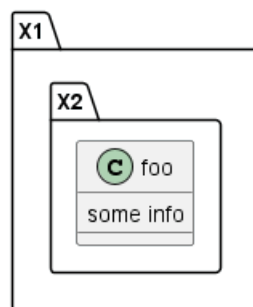
コマンド `set separator ???` を使用して、（ドット以外の）別の区切り文字を定義できます。

```

@startuml

set separator ::
class X1::X2::foo {
    some info
}

@enduml
  
```



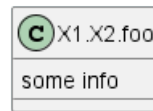
コマンド `set separator none` を使用して、自動的に名前空間を作成する機能を無効にできます。

```

@startuml

set separator none
class X1.X2.foo {
    some info
}

@enduml
  
```



3.25 ロリポップ (棒付きキャンディー) インタフェース

次の構文を使用して、クラスにロリポップインタフェースを定義することもできます：

- `bar ()- foo`
- `bar ()-- foo`
- `foo -() bar`

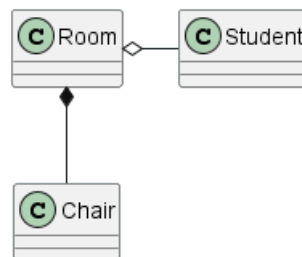
```
@startuml
class foo
bar ()- foo
@enduml
```



3.26 矢印の向きを変える

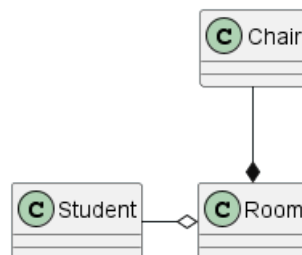
デフォルトではクラス間のリンクは2つのダッシュ -- を持っており、垂直に配向されています。次のように単一のダッシュ (またはドット) を置くことによって水平方向にリンクを使用することが可能です。

```
@startuml
Room o- Student
Room *-- Chair
@enduml
```



リンクをひっくり返すことにより向きを変えることができます:

```
@startuml
Student -o Room
Chair --* Room
@enduml
```



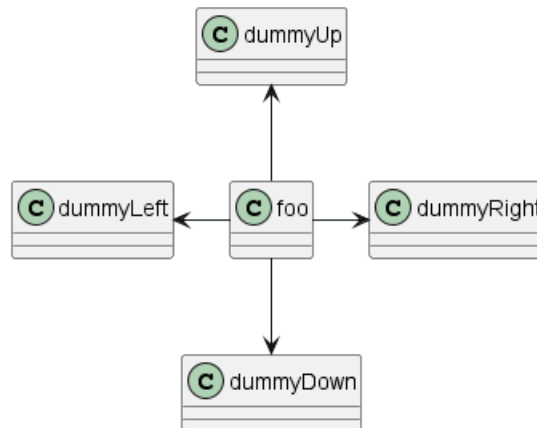
キーワード `left`, `right`, `up`, `down` を矢印の内側に置くことにより、矢印の方向を変えることも可能です：



```

@startuml
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml

```



方向の最初の文字を使用して矢印を短縮することができます（例えば、**-d-** を **-down-** の代わりに、または、最初の 2 文字 (**-do-**)。

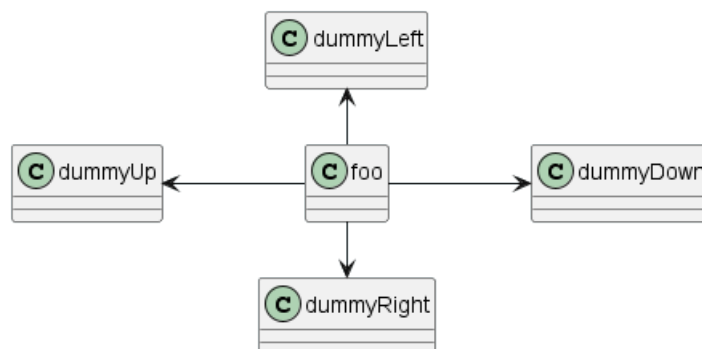
この機能を悪用してはならないことに注意してください。Graphviz は微調整のいらない良い結果を通常は与えてくれます。

left to right direction パラメータを使用した場合は次のようになります。

```

@startuml
left to right direction
foo -left-> dummyLeft
foo -right-> dummyRight
foo -up-> dummyUp
foo -down-> dummyDown
@enduml

```



3.27 関連クラス

この例のように、2 つのクラスの関係性を定義した後で 関連クラスを定義することができます。

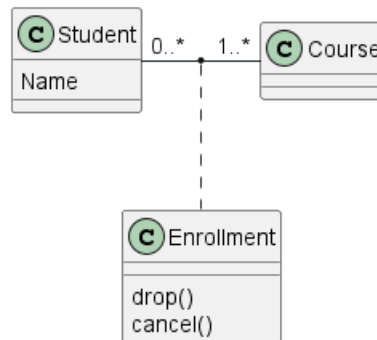
```

@startuml
class Student {
    Name
}
Student "0..*" -- "1..*" Course
(Student, Course) .. Enrollment

```



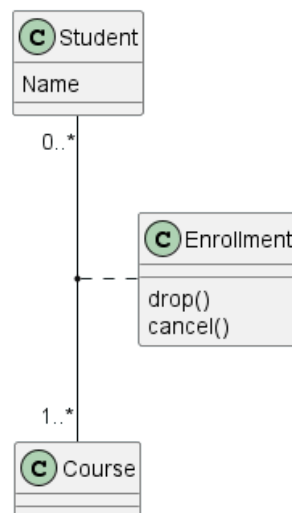
```
class Enrollment {
    drop()
    cancel()
}
@enduml
```



別の方向にそれを定義することができます：

```
@startuml
class Student {
    Name
}
Student "0..*" -- "1..*" Course
(Student, Course) . Enrollment

class Enrollment {
    drop()
    cancel()
}
@enduml
```



3.28 同一クラスに複数の関連

```
@startuml
class Station {
    +name: string
}

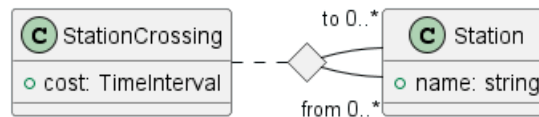
class StationCrossing {
    +cost: TimeInterval
}

```



```
<> diamond
```

```
StationCrossing . diamond
diamond - "from 0..*" Station
diamond - "to 0..*" Station
@enduml
```



[Ref. Incubation: Associations]

3.29 化粧をする

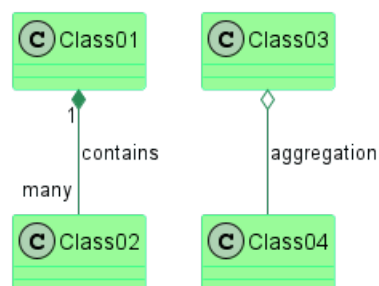
ダイアグラムの色やフォントを変更するには skinparam コマンドを使用します。

このコマンドは以下の場面で使用できます。

- ダイアグラム定義内で他のコマンドを同様に。
- インクルードされたファイル内。
- 設定ファイルのコマンドライン内や Ant タスク内。

```
@startuml
skinparam class {
  BackgroundColor PaleGreen
  ArrowColor SeaGreen
  BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
@enduml
```



3.30 ステレオタイプの化粧

ステレオタイプクラスに特定の色やフォントを定義することができます。

```
@startuml
skinparam class {
  BackgroundColor PaleGreen
  ArrowColor SeaGreen
  BorderColor SpringGreen
}
```

```

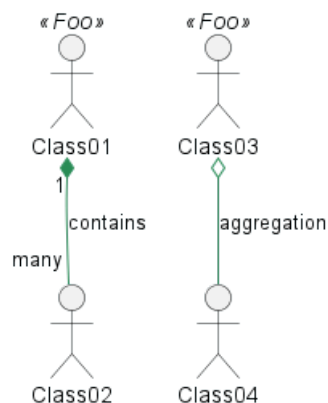
BackgroundColor<<Foo>> Wheat
BorderColor<<Foo>> Tomato
}
skinparam stereotypeCBackgroundColor YellowGreen
skinparam stereotypeCBackgroundColor<< Foo >> DimGray

Class01 <<Foo>>
Class03 <<Foo>>
Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation

@enduml

```



3.31 色のグラデーション

表記を使用して、クラスや注釈に個別の色を宣言することが可能です。

標準的な色の名前または RGB コードのいずれかを様々な記法で 사용할ことができます。色を参照してください。

次の構文で背景に色のグラデーションを設定することもできます。2 つの色の名前を次のいずれかで区切って記述してください：

- |
- /
- \
- -

グラデーションの方向に応じて記号を使い分けてください。

例：

```

@startuml

skinparam backgroundcolor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

class Foo #red-green
note left of Foo #blue\9932CC
    this is my
    note on this class
end note

package example #GreenYellow/LightGoldenRodYellow {
    class Dummy

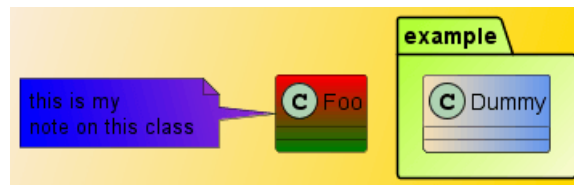
```



```

}
@enduml

```



3.32 レイアウトの手助け

ときには、デフォルトのレイアウトでは完璧とは言えないことがあります…

together キーワードを使って複数のクラスをグループにまとめることができます: レイアウトエンジンは、それらのクラスを（あたかも同じパッケージにあるかのように）グループにまとめようとします。

hidden リンクを使ってレイアウトを強制することも可能です。

```

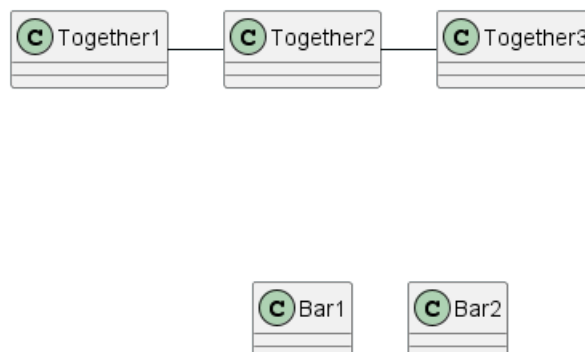
@startuml
class Bar1
class Bar2
together {
    class Together1
    class Together2
    class Together3
}
Together1 - Together2
Together2 - Together3
Together2 -[hidden]--> Bar1
Bar1 -[hidden]> Bar2

```

```

@enduml

```



3.33 大きなファイルの分割

時には、ある非常に大きな画像ファイルを受け取ることがあるでしょう。

生成された画像を複数のファイルに分割するコマンド **page (hpages)x(vpages)** を使用することができます:

hpages は横方向のページ数を示すコマンドであり、そして **vpages** は縦方向のページ数を示すコマンドです。

特定のスキンパラメータ設定を使用して、分割されたページに罫線を配置することもできます（例を参照）。



```

@startuml
' Split into 4 pages
page 2x2
skinparam pageMargin 10
skinparam pageExternalColor gray
skinparam pageBorderColor black

class BaseClass

namespace net.dummy #DDDDDD {
    .BaseClass <|-- Person
    Meeting o-- Person

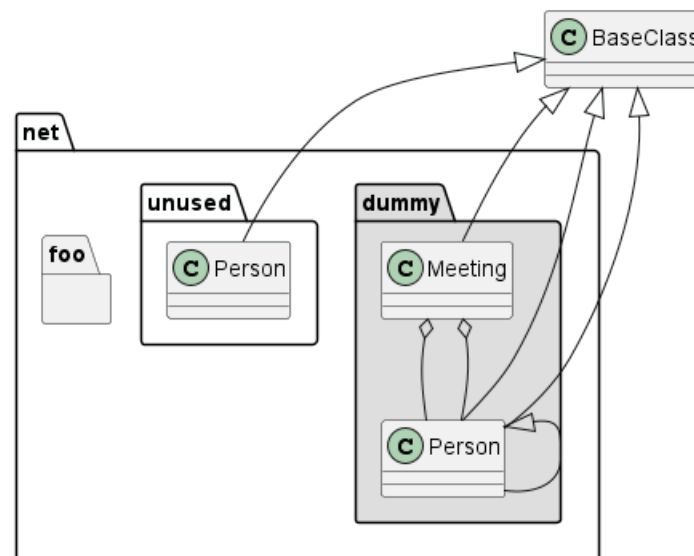
    .BaseClass <|-- Meeting
}

namespace net.foo {
    net.dummy.Person <|-- Person
    .BaseClass <|-- Person

    net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person
@enduml

```



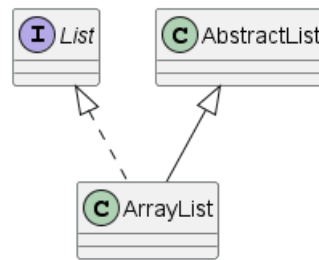
3.34 継承 (extends) と実装 (implements)

`extends` キーワードと `implements` キーワードを使用することができます。

```

@startuml
class ArrayList implements List
class ArrayList extends AbstractList
@enduml

```



3.35 角括弧を使用した関係（リンク、矢印）のスタイル

3.35.1 線のスタイル

関係（リンク、矢印）に **bold**、**dashed**、**dotted**、**hidden**、**plain** のスタイルを指定することができます。

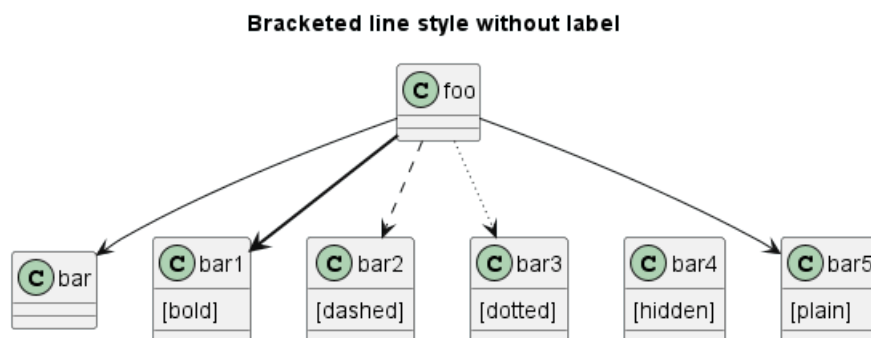
- ラベル無し

```

@startuml
title Bracketed line style without label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml

```



- ラベル有り

```

@startuml
title Bracketed line style with label
class foo
class bar
bar1 : [bold]
bar2 : [dashed]
bar3 : [dotted]
bar4 : [hidden]
bar5 : [plain]

foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml

```

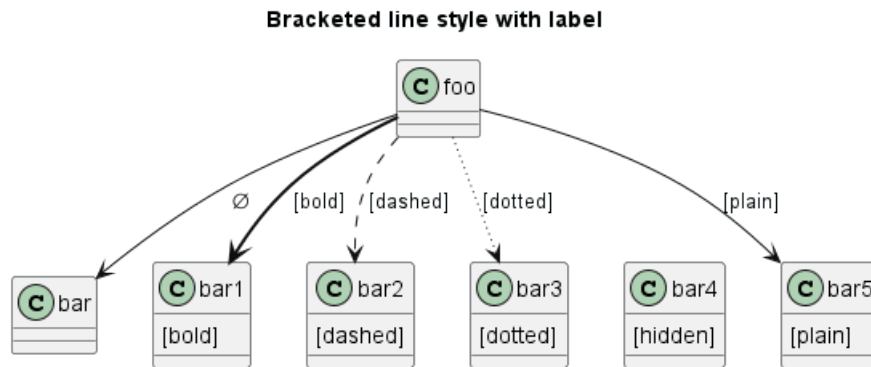


```

foo --> bar          :
foo -[bold]-> bar1    : [bold]
foo -[dashed]-> bar2  : [dashed]
foo -[dotted]-> bar3  : [dotted]
foo -[hidden]-> bar4  : [hidden]
foo -[plain]-> bar5   : [plain]

```

```
@enduml
```



[Adapted from QA-4181]

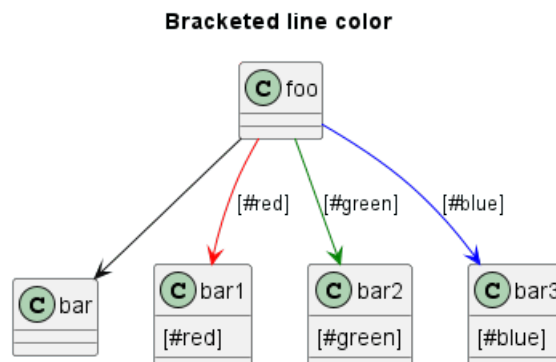
3.35.2 線の色

```

@startuml
title Bracketed line color
class foo
class bar
bar1 : [#red]
bar2 : [#green]
bar3 : [#blue]

foo --> bar
foo -[#red]-> bar1 : [#red]
foo -[#green]-> bar2 : [#green]
foo -[#blue]-> bar3 : [#blue]
'foo -[#blue;#yellow;#green]-> bar4
@enduml

```



3.35.3 線の太さ

```

@startuml
title Bracketed line thickness
class foo

```



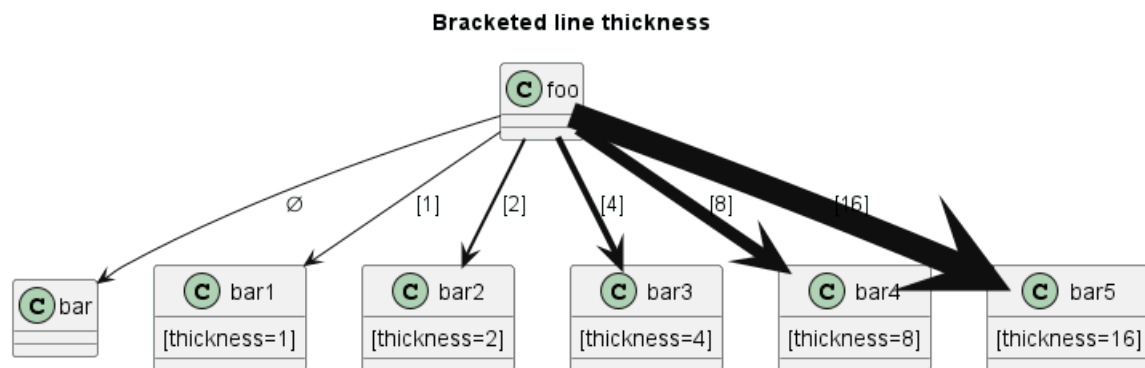
```

class bar
bar1 : [thickness=1]
bar2 : [thickness=2]
bar3 : [thickness=4]
bar4 : [thickness=8]
bar5 : [thickness=16]

foo --> bar
      :
foo -[thickness=1]-> bar1 : [1]
foo -[thickness=2]-> bar2 : [2]
foo -[thickness=4]-> bar3 : [4]
foo -[thickness=8]-> bar4 : [8]
foo -[thickness=16]-> bar5 : [16]

@enduml

```



[Ref. QA-4949]

3.35.4 混合

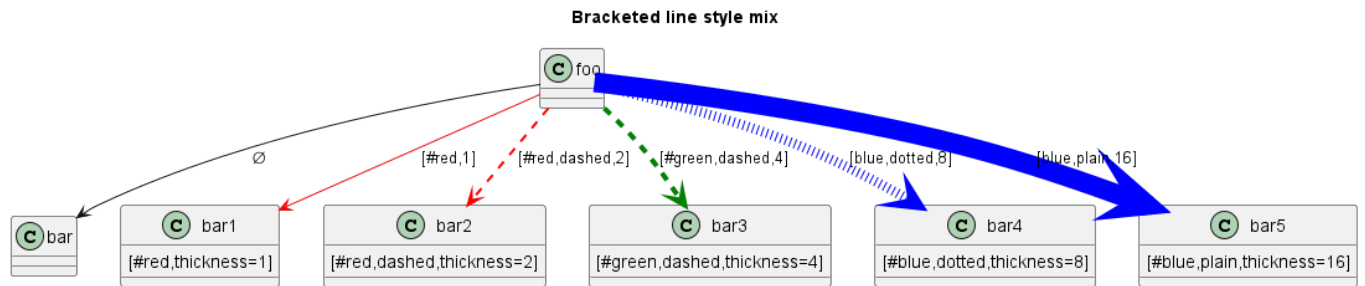
```

@startuml
title Bracketed line style mix
class foo
class bar
bar1 : [#red,thickness=1]
bar2 : [#red,dashed,thickness=2]
bar3 : [#green,dashed,thickness=4]
bar4 : [#blue,dotted,thickness=8]
bar5 : [#blue,plain,thickness=16]

foo --> bar
      :
foo -[#red,thickness=1]-> bar1 : [#red,1]
foo -[#red,dashed,thickness=2]-> bar2 : [#red,dashed,2]
foo -[#green,dashed,thickness=4]-> bar3 : [#green,dashed,4]
foo -[#blue,dotted,thickness=8]-> bar4 : [blue,dotted,8]
foo -[#blue,plain,thickness=16]-> bar5 : [blue,plain,16]

@enduml

```

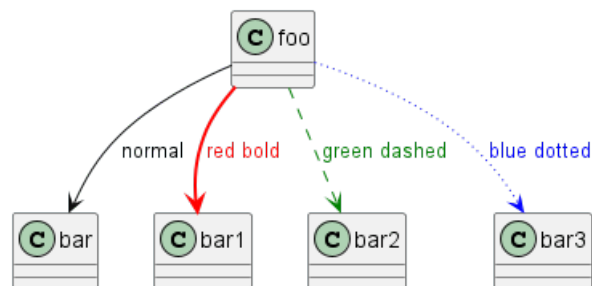


3.36 関係 (リンク、矢印) の色とスタイルを変更する (インラインスタイル)

個別の関係ごとに色とスタイルを変更するには、次の記法を使用します：

- `#color;line.[bold|dashed|dotted];text:color`

```
@startuml
class foo
foo --> bar : normal
foo --> bar1 : #line:red;line.bold;text:red : red bold
foo --> bar2 : #green;line.dashed;text:green : green dashed
foo --> bar3 : #blue;line.dotted;text:blue : blue dotted
@enduml
```



[See similar feature on deployment]

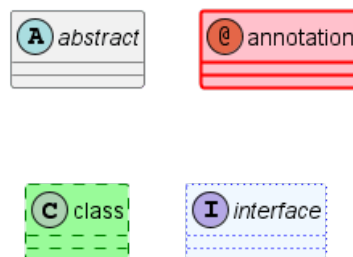
3.37 クラスの色とスタイルを変更する (インラインスタイル)

個別のクラスごとに色とスタイルを変更するには、次の記法を使用します：

- `#color ##[style]color`

最初に背景色 (`#color`)、次に線の色 (`##[style]color`) を指定します。

```
@startuml
abstract abstract
annotation annotation #pink ##[bold]red
class class #palegreen ##[dashed]green
interface interface #aliceblue ##[dotted]blue
@enduml
```

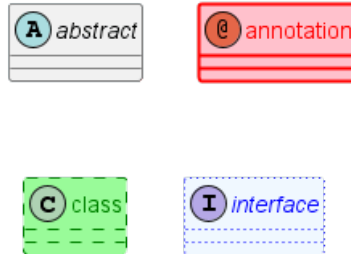


[Ref. QA-1487]



- `#[color|back:color];header:color;line:color;line.[bold|dashed|dotted];text:color`

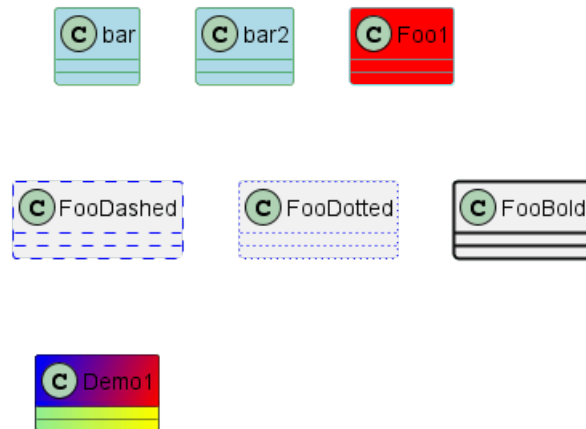
```
@startuml
abstract    abstract
annotation  annotation #pink;line:red;line.bold;text:red
class       class      #palegreen;line:green;line.dashed;text:green
interface   interface  #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



例:

```
@startuml
class bar #line:green;back:lightblue
class bar2 #lightblue;line:green

class Foo1 #back:red;line:00FFFF
class FooDashed #line.dashed:blue
class FooDotted #line.dotted:blue
class FooBold #line.bold
class Demo1 #back:lightgreen|yellow;header:blue/red
@enduml
```



[Ref. QA-3770]

3.38 クラスメンバ間の矢印

```
@startuml
class Foo {
+ field1
+ field2
}

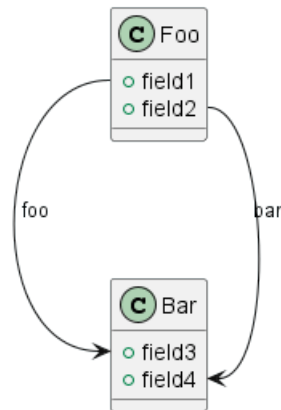
class Bar {
+ field3
+ field4
}
```



```

Foo::field1 --> Bar::field3 : foo
Foo::field2 --> Bar::field4 : bar
@enduml

```



[Ref. QA-3636]

```

@startuml
left to right direction

```

```

class User {
    id : INTEGER
    ..
    other_id : INTEGER
}

```

```

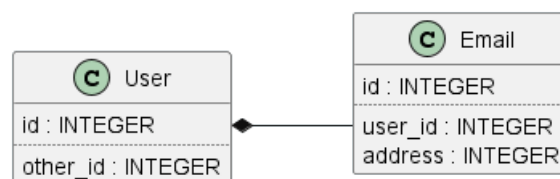
class Email {
    id : INTEGER
    ..
    user_id : INTEGER
    address : INTEGER
}

```

```

User::id *-- Email::user_id
@enduml

```



[Ref. QA-5261]

3.39 継承の矢印のグループ化

skinparam groupInheritance を使用すると、複数の矢印の矢じりを統合することができます。また、閾値をパラメータとして指定できます。

3.39.1 groupInheritance 1 (グループ化しない)

```

@startuml
skinparam groupInheritance 1

```

```

A1 <|-- B1

```

```

A2 <|-- B2

```



```
A2 <|-- C2
```

```
A3 <|-- B3
```

```
A3 <|-- C3
```

```
A3 <|-- D3
```

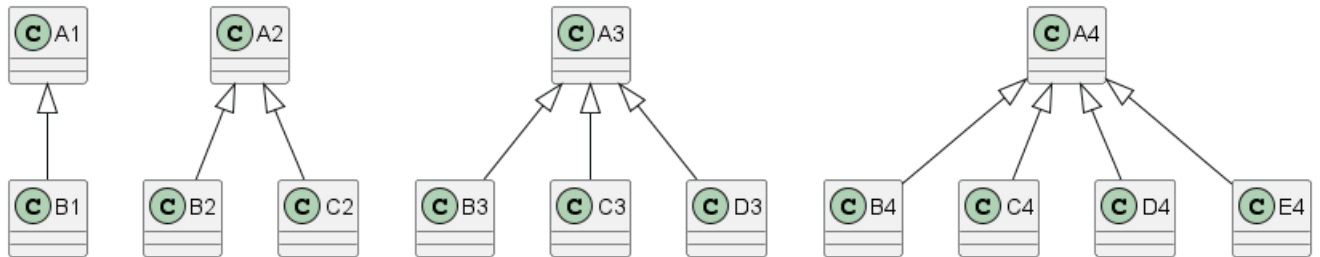
```
A4 <|-- B4
```

```
A4 <|-- C4
```

```
A4 <|-- D4
```

```
A4 <|-- E4
```

```
@enduml
```



3.39.2 groupInheritance 2 (2 つ以上の場合にグループ化)

```
@startuml
```

```
skinparam groupInheritance 2
```

```
A1 <|-- B1
```

```
A2 <|-- B2
```

```
A2 <|-- C2
```

```
A3 <|-- B3
```

```
A3 <|-- C3
```

```
A3 <|-- D3
```

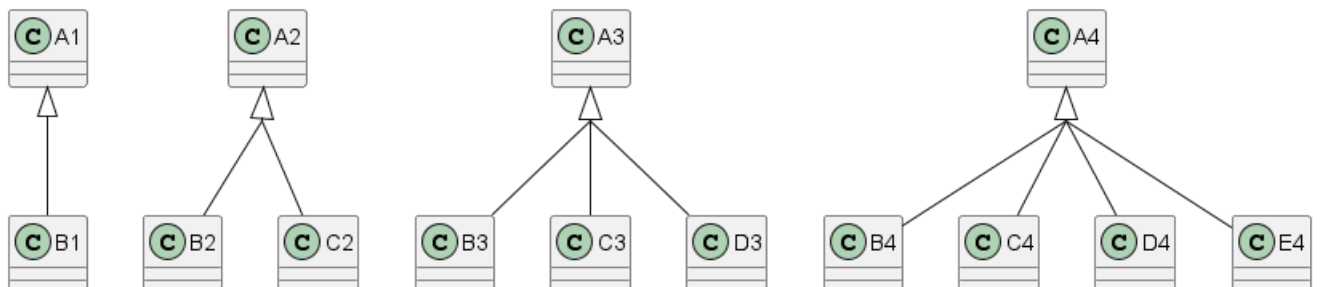
```
A4 <|-- B4
```

```
A4 <|-- C4
```

```
A4 <|-- D4
```

```
A4 <|-- E4
```

```
@enduml
```



3.39.3 groupInheritance 3 (3 つ以上の場合にグループ化)

```
@startuml
```

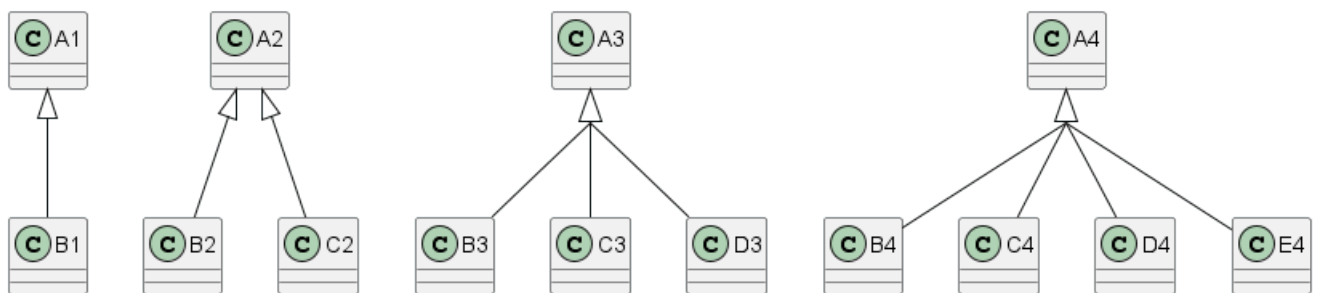
```
skinparam groupInheritance 3
```

```
A1 <|-- B1
```

```
A2 <|-- B2
A2 <|-- C2
```

```
A3 <|-- B3
A3 <|-- C3
A3 <|-- D3
```

```
A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```



3.39.4 groupInheritance 4 (4 つ以上の場合にグループ化)

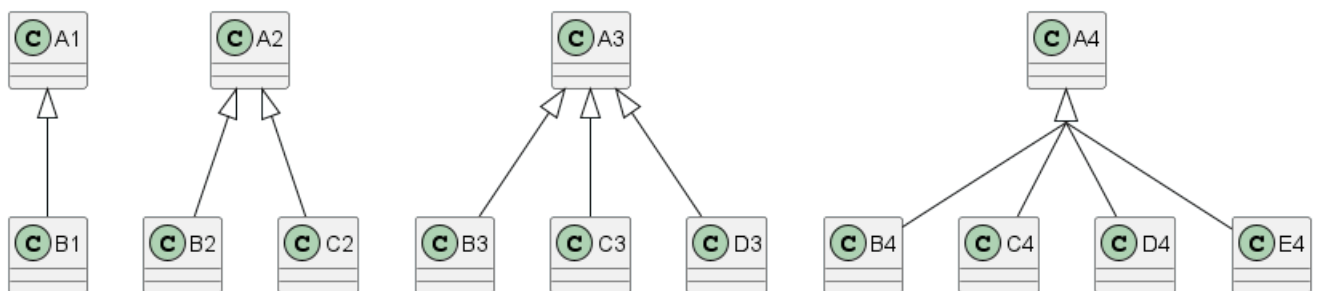
```
@startuml
skinparam groupInheritance 4
```

```
A1 <|-- B1
```

```
A2 <|-- B2
A2 <|-- C2
```

```
A3 <|-- B3
A3 <|-- C3
A3 <|-- D3
```

```
A4 <|-- B4
A4 <|-- C4
A4 <|-- D4
A4 <|-- E4
@enduml
```

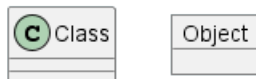


[Ref. QA-3193, and Defect QA-13532]

3.40 Display JSON Data on Class or Object diagram

3.40.1 Simple example

```
@startuml
class Class
object Object
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

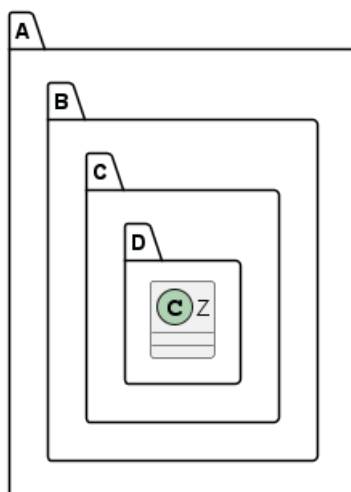
[Ref. QA-15481]

For another example, see on JSON page.

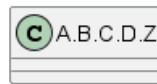
3.41 Packages and Namespaces Enhancement

[From V1.2023.2+, and V1.2023.5]

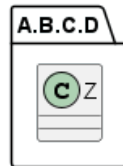
```
@startuml
class A.B.C.D.Z {
}
@enduml
```



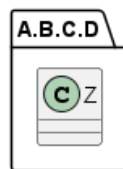
```
@startuml
set separator none
class A.B.C.D.Z {
}
@enduml
```



```
@startuml
!pragma useIntermediatePackages false
class A.B.C.D.Z {
}
@enduml
```



```
@startuml
set separator none
package A.B.C.D {
    class Z {
    }
}
@enduml
```



[Ref. GH-1352]

4 オブジェクト図

PlantUML offers a simple and intuitive way to create object diagrams using plain text. Its user-friendly syntax allows for quick diagram creation without the need for complex GUI tools. Moreover, the [PlantUML forum](https://forum.plantuml.net/) provides a platform for users to discuss, share, and seek assistance, fostering a collaborative community. By choosing PlantUML, users benefit from both the efficiency of markdown-based diagramming and the support of an active community.

4.1 オブジェクトの定義

オブジェクトのインスタンスを、キーワード `object` を使用して定義します。

```
@startuml
object firstObject
object "My Second Object" as o2
@enduml
```



4.2 オブジェクト間の関係

オブジェクト間の関係は次の記号を用いて定義します:

Type	Symbol	Image
Extension	< --	
Composition	*--	
Aggregation	o--	

-- を .. に置き換えることで点線を示すことができます。

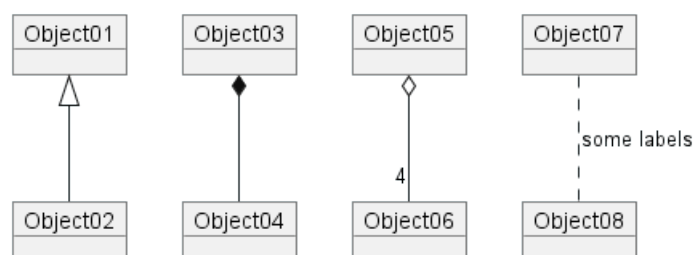
これらのルールを知ることで、以下の図を描くことができます。

関係にラベルをつけることができ、: を用い、ラベルの文字列を続けます。

関係の各側のスペースを含む文字列を引用符 "" で囲むことができます。

```
@startuml
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
@enduml
```



4.3 n-項関連

```
@startuml
object o1
object o2
diamond dia
object o3

o1 --> dia
o2 --> dia
dia --> o3
@enduml
```



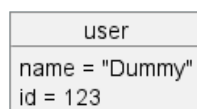
4.4 フィールドの追加

フィールドを宣言するには、シンボル：にフィールド名を続けます。

```
@startuml
object user

user : name = "Dummy"
user : id = 123

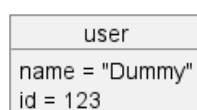
@enduml
```



全てのフィールドを括弧 {} で括って範囲を示すことも可能です。

```
@startuml
object user {
    name = "Dummy"
    id = 123
}

@enduml
```



4.5 クラス図と共通の機能

- 属性、メソッド等の非表示
- 注釈の詳細
- 非文字の使用
- 化粧をする

4.6 マップテーブル (連想配列)

map キーワードとセパレータ => を使って、マップテーブル (連想配列) を定義することができます。

```
@startuml
map CapitalCity {
  UK => London
  USA => Washington
  Germany => Berlin
}
@enduml
```

CapitalCity	
UK	London
USA	Washington
Germany	Berlin

```
@startuml
map "Map **Contry => CapitalCity**" as CC {
  UK => London
  USA => Washington
  Germany => Berlin
}
@enduml
```

Map Contry => CapitalCity	
UK	London
USA	Washington
Germany	Berlin

```
@startuml
map "map: Map<Integer, String>" as users {
  1 => Alice
  2 => Bob
  3 => Charlie
}
@enduml
```

map: Map<Integer, String>	
1	Alice
2	Bob
3	Charlie

オブジェクトにリンクを追加します。

```
@startuml
object London

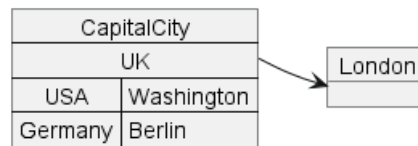
map CapitalCity {
  UK *-> London
}
```



```

USA => Washington
Germany => Berlin
}
@enduml

```



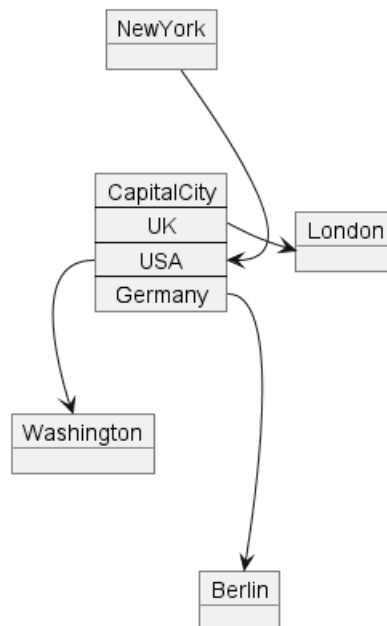
```

@startuml
object London
object Washington
object Berlin
object NewYork

map CapitalCity {
  UK *-> London
  USA *--> Washington
  Germany *---> Berlin
}

NewYork --> CapitalCity::USA
@enduml

```



[Ref. #307]

```

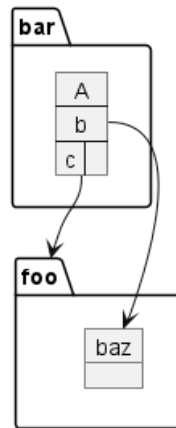
@startuml
package foo {
  object baz
}

package bar {
  map A {
    b *-> foo.baz
    c =>
  }
}

```



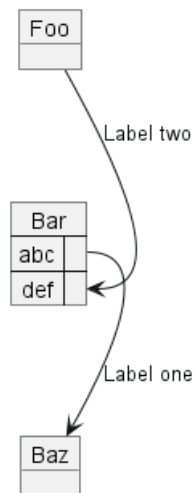
```
A::c --> foo
@enduml
```



[Ref. QA-12934]

```
@startuml
object Foo
map Bar {
  abc=>
  def=>
}
object Baz
```

```
Bar::abc --> Baz : Label one
Foo --> Bar::def : Label two
@enduml
```



[Ref. #307]

4.7 map を利用して PERT (Program Evaluation and Review Technique) 図を作成する

map table を利用して PERT (Program (or Project) Evaluation and Review Technique) 図を作成できます。

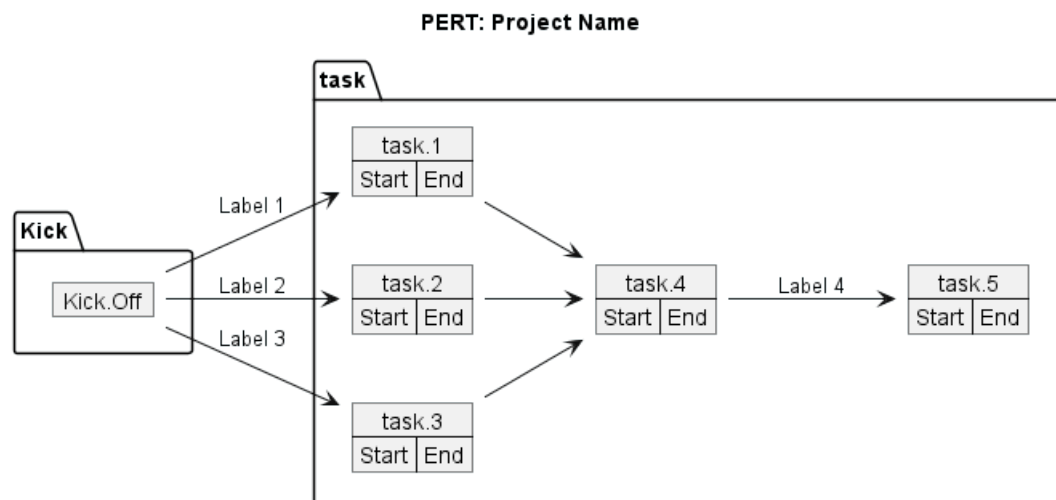
```
@startuml PERT
left to right direction
' Horizontal lines: -->, <-->, <-->
' Vertical lines: ->, <-, <->
title PERT: Project Name
```



```

map Kick.Off {
}
map task.1 {
    Start => End
}
map task.2 {
    Start => End
}
map task.3 {
    Start => End
}
map task.4 {
    Start => End
}
map task.5 {
    Start => End
}
Kick.Off --> task.1 : Label 1
Kick.Off --> task.2 : Label 2
Kick.Off --> task.3 : Label 3
task.1 --> task.4
task.2 --> task.4
task.3 --> task.4
task.4 --> task.5 : Label 4
@enduml

```



[Ref. QA-12337]

4.8 Display JSON Data on Class or Object diagram

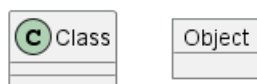
4.8.1 Simple example

```

@startuml
class Class
object Object
json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml

```





JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

For another example, see on JSON page.

5 アクティビティ図 (レガシー版)

これはアクティビティ図の古い記法です。現行の新しいバージョンは、アクティビティ図を参照してください。

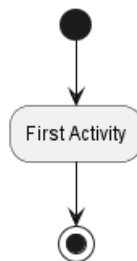
5.1 単純なアクティビティ

(*) をアクティビティ図の開始点と終了点に使用します。

場合によっては、(*top) を使用して開始点を図の一番上に置くこともできます。

--> で矢印を表します。

```
@startuml
(*) --> "First Activity"
"First Activity" --> (*)
@enduml
```

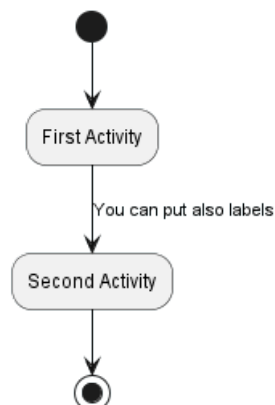


5.2 矢印のラベル

デフォルトで、矢印は最後に書いたアクティビティを起点に描かれます。

矢印にラベルを付けるには、矢印の定義の直後に角括弧 [と] を使います。

```
@startuml
(*) --> "First Activity"
-->[You can put also labels] "Second Activity"
--> (*)
@enduml
```



5.3 矢印の方向を変える

水平矢印には -> を使用できます。次の構文を使用して矢印の方向を強制することができます。

- -down-> (デフォルトの矢印)

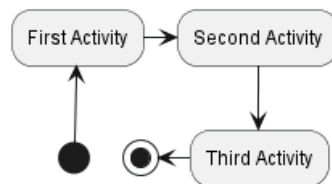


- -right-> or ->
- -left->
- -up->

```
@startuml
```

```
(*) -up-> "First Activity"
-right-> "Second Activity"
--> "Third Activity"
-left-> (*)
```

```
@enduml
```



5.4 分岐

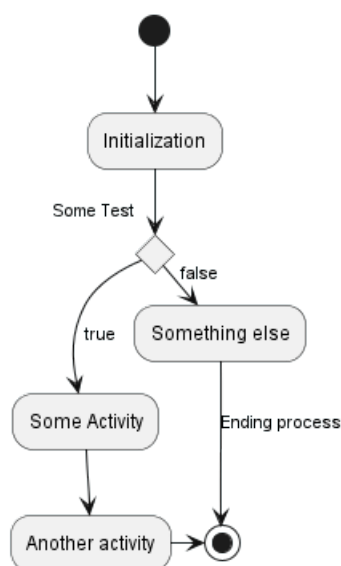
キーワード `if/then/else` を使用してブランチを定義することができます。

```
@startuml
```

```
(*) --> "Initialization"
```

```
if "Some Test" then
  -->[true] "Some Activity"
  --> "Another activity"
  -right-> (*)
else
  ->[false] "Something else"
  -->[Ending process] (*)
endif
```

```
@enduml
```



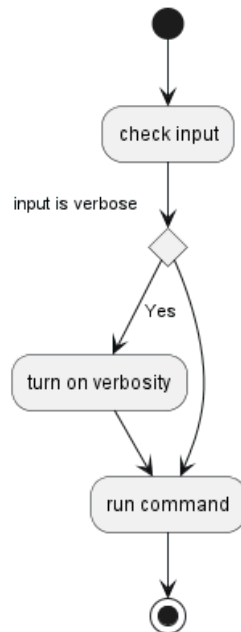
残念ながら、図のテキストで同じアクティビティを繰り返すことがあります：

```
@startuml
```

```

(*) --> "check input"
If "input is verbose" then
--> [Yes] "turn on verbosity"
--> "run command"
else
--> "run command"
Endif
-->(*)
@enduml

```



5.5 もっと分岐

デフォルトでは、分岐は最後に定義されたアクティビティに接続されますが、これを上書きしてキーワード `if` でリンクを定義することは可能です。

分岐をネストすることも可能です。

```

@startuml
(*) --> if "Some Test" then
    -->[true] "activity 1"
    if "" then
        -> "activity 3" as a3
    else
        if "Other test" then
            -left-> "activity 5"
        else
            --> "activity 6"
        endif
    endif
endif
else
    -->[false] "activity 2"
endif

```

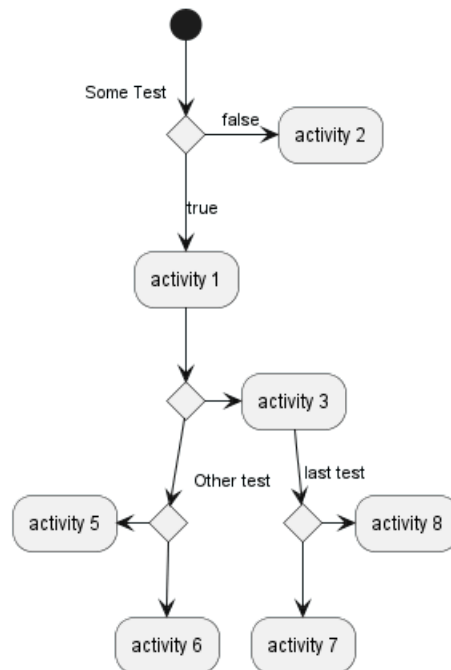


```

a3 --> if "last test" then
  --> "activity 7"
else
  --> "activity 8"
endif

@enduml

```



5.6 同期

=== code === を使用して同期バーを表示できます。

```

@startuml

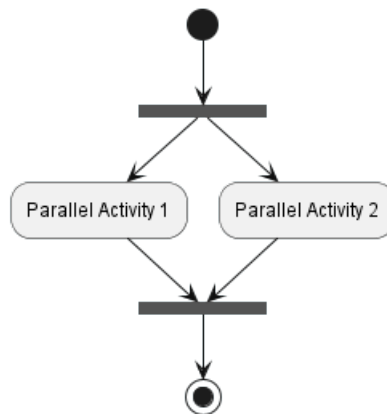
(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

===B1=== --> "Parallel Activity 2"
--> ===B2===

--> (*)

@enduml

```



5.7 長いアクティビティの記述

アクティビティを宣言するとき、説明文を複数の行にまたがせることができます。説明に `as` を追加することもできます。

キーワード `as` を使ってアクティビティに短いコードを与えることもできます。このコードは、図の説明の後半で使用できます。

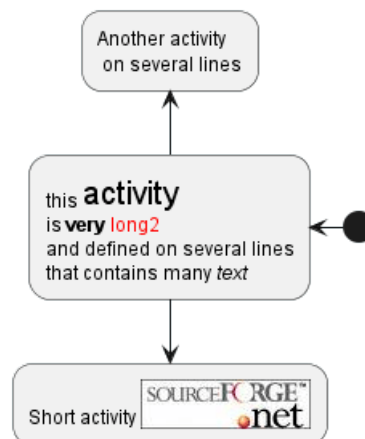
```

@startuml
(*) -left-> "this <size:20>activity</size>
is <b>very</b> <color:red>long2</color>
and defined on several lines
that contains many <i>text</i>" as A1

-up-> "Another activity\n on several lines"

A1 --> "Short activity <img:sourceforge.jpg>"
@enduml

```



5.8 注釈

注釈をつけるアクティビティの説明の直後にあるコマンド `note left`, `note right`, `note top` or `note bottom`, を使用して、アクティビティに注釈を追加することができます。

開始点に注釈を付ける場合は、図の説明の最初に注釈を定義します。

キーワード `endnote` を使用して、複数の行に注釈を付けることもできます。

```

@startuml

(*) --> "Some Activity"
note right: This activity has to be defined

endnote

```

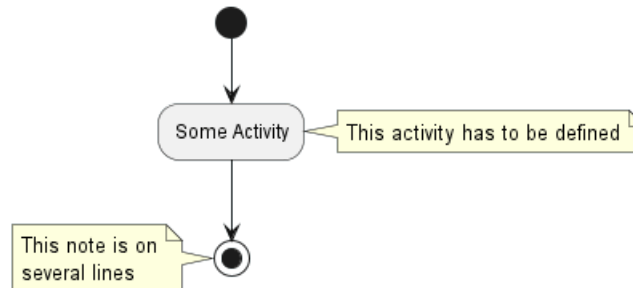


```

"Some Activity" --> (*)
note left
  This note is on
  several lines
end note

@enduml

```



5.9 パーティション

キーワード **partition** を使用してパーティションを定義し、必要に応じてパーティションの背景色を宣言することができます (HTML カラーコードまたは名前を使用)。

アクティビティを宣言すると、自動的に最後に使用されたパーティションに配置されます。

閉じ括弧 **}** を使用してパーティション定義を閉じることができます。

```

@startuml

partition Conductor {
  (*) --> "Climbs on Platform"
  --> === S1 ===
  --> Bows
}

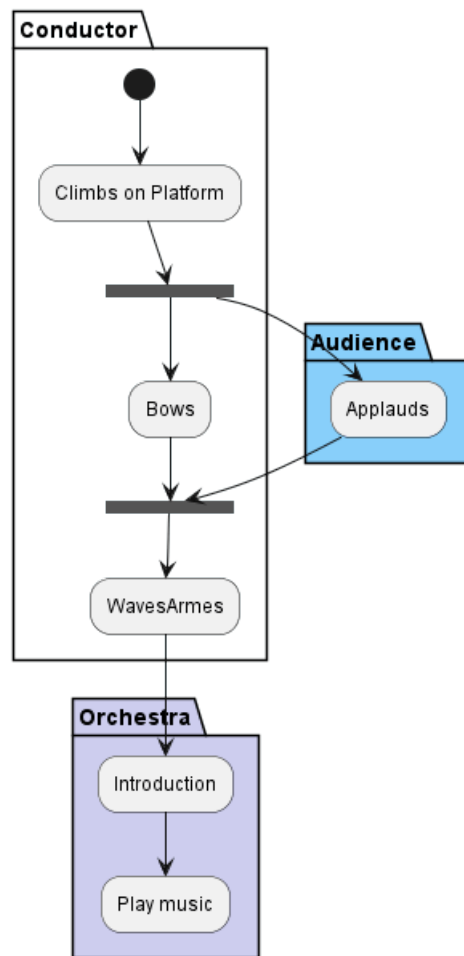
partition Audience #LightSkyBlue {
  === S1 === --> Applauds
}

partition Conductor {
  Bows --> === S2 ===
  --> WavesArmes
  Applauds --> === S2 ===
}

partition Orchestra #CCCCEE {
  WavesArmes --> Introduction
  --> "Play music"
}

@enduml

```



5.10 スキンパラメータ

コマンド `skinparam` を使用して、図面の色とフォントを変更することができます。

このコマンドを使用することができます：

- 図の定義中では、他のコマンドと同様に、
- インクルードされたファイルの中で、
- コマンドラインまたは ANT タスクで提供される構成ファイルの中で。

定型アクティビティには、特定の色とフォントを定義できます。

@startuml

```

skinparam backgroundColor #AAFFFF
skinparam activity {
  StartColor red
  BarColor SaddleBrown
  EndColor Silver
  BackgroundColor Peru
  BackgroundColor<< Begin >> Olive
  BorderColor Peru
  FontName Impact
}
  
```

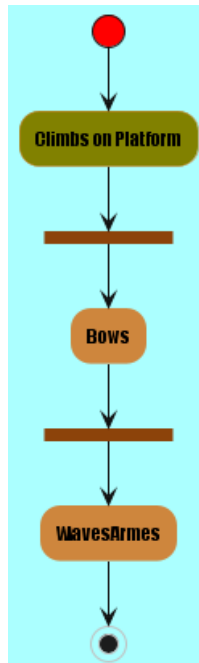
```

(*) --> "Climbs on Platform" << Begin >>
--> === S1 ===
--> Bows
  
```



```
--> === S2 ===
--> WavesArmes
--> (*)
```

```
@enduml
```



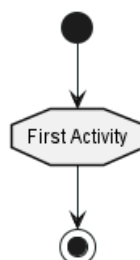
5.11 八角形

コマンド `skinparam activityShape octagon` を使用して、アクティビティの形状を八角形に変更できます。

```
@startuml
'Default is skinparam activityShape roundBox
skinparam activityShape octagon
```

```
(*) --> "First Activity"
"First Activity" --> (*)
```

```
@enduml
```



5.12 完全な例

```
@startuml
title Servlet Container

(*) --> "ClickServlet.handleRequest()"
--> "new Page"

if "Page.onSecurityCheck" then
```



```
->[true] "Page.onInit()"

if "isForward?" then
  ->[no] "Process controls"

  if "continue processing?" then
    -->[yes] ===RENDERING===
  else
    -->[no] ===REDIRECT_CHECK===
  endif

else
  -->[yes] ===RENDERING===
endif

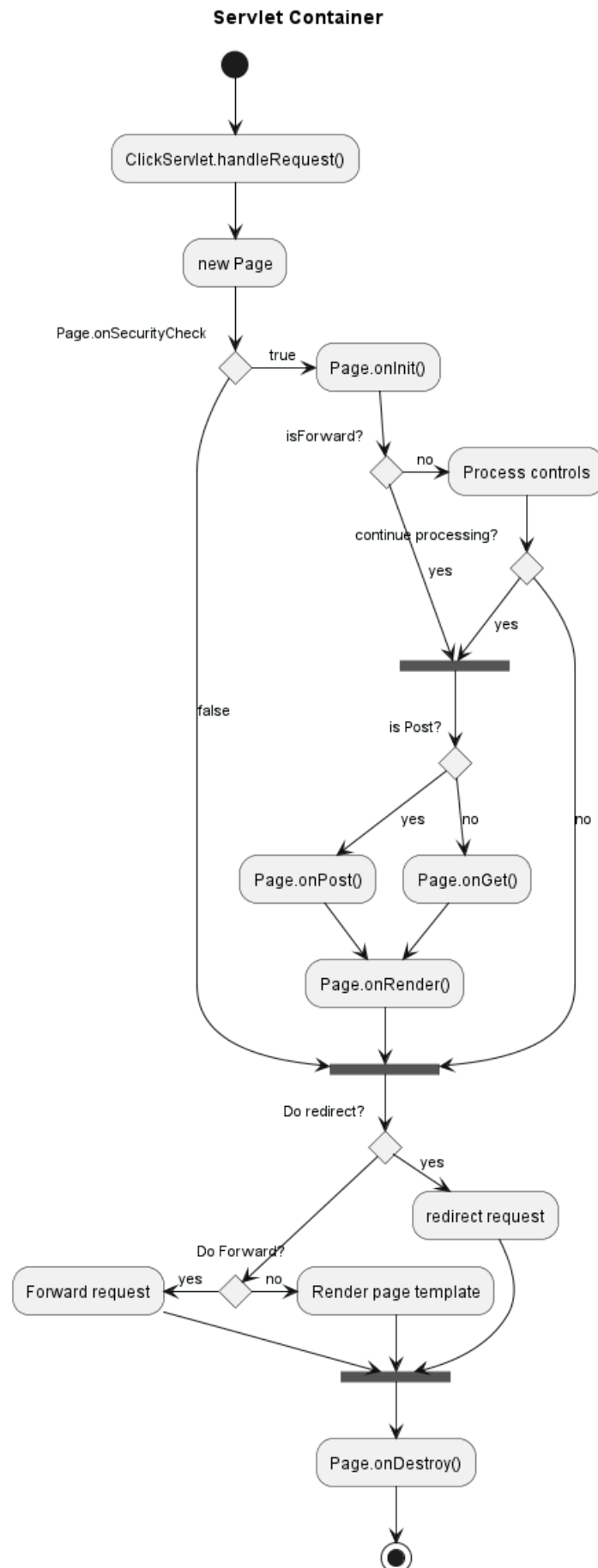
if "is Post?" then
  -->[yes] "Page.onPost()"
  --> "Page.onRender()" as render
  --> ===REDIRECT_CHECK===
else
  -->[no] "Page.onGet()"
  --> render
endif

else
  -->[false] ===REDIRECT_CHECK===
endif

if "Do redirect?" then
  ->[yes] "redirect request"
  --> ==BEFORE_DESTROY==
else
  if "Do Forward?" then
    -left->[yes] "Forward request"
    --> ==BEFORE_DESTROY==
  else
    -right->[no] "Render page template"
    --> ==BEFORE_DESTROY==
  endif
endif

--> "Page.onDestroy()"
-->(*)

@enduml
```



6 アクティビティ図 (新しい構文)

アクティビティ図に使用されていた以前の構文には、いくつかの制限や保守性の問題がありました。

6.0.1 新しい構文の利点

- Graphviz に依存しない：* シーケンス図と同様に、新しい構文では Graphviz をインストールする必要がないため、セットアッププロセスが簡素化されます。
- メンテナンスの容易さ：

6.0.2 新シンタックスへの移行

互換性を維持するため、旧シンタックスも引き続きサポートしますが、強化された機能と利点を活用するために、新シンタックスに移行することを強くお勧めします。

- 今すぐ移行して、新しいアクティビティ図のシンタックスで、より合理的で効率的なダイアグラム作成プロセスを体験してください。

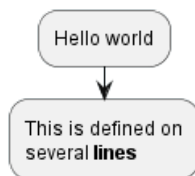
6.1 単純なアクティビティ

アクティビティのラベルは: で開始し; で終了します。

テキストの書式設定は、Creole 記法の Wiki 構文を使用して行うことができます。

それらは定義順に暗黙的にリンクされます。

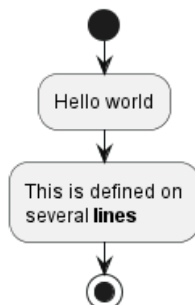
```
@startuml
:Hello world;
:This is defined on
several **lines**;
@enduml
```



6.2 開始／終了

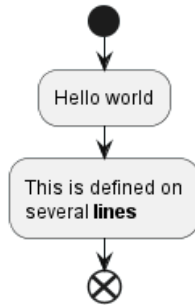
図の開始と終了を示すために、キーワード **start** と **stop** を使用できます。

```
@startuml
start
:Hello world;
:This is defined on
several **lines**;
stop
@enduml
```



キーワード `end` もまた使用できます。

```
@startuml
start
:Hello world;
:This is defined on
several **lines**;
end
@enduml
```



6.3 条件文

図に条件分岐を追加したい場合は、キーワード `if`、`then` そして `else` を使用することができます。ラベルは括弧を使用することで与えることができます。

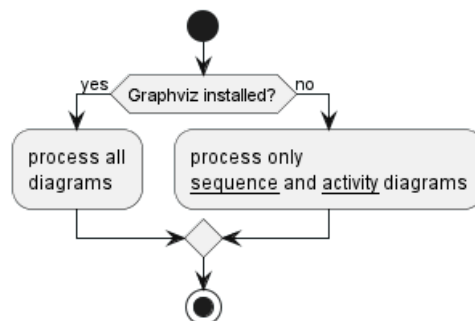
3 種類の構文を使うことができます。

- `if (...) then (...)`

```
@startuml
start

if (Graphviz installed?) then (yes)
:process all\ndiagrams;
else (no)
:process only
__sequence__ and __activity__ diagrams;
endif

stop
@enduml
```



- `if (...) is (...) then`

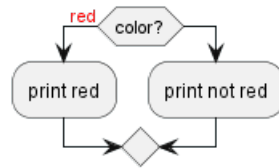
```
@startuml
if (color?) is (<color:red>red) then
:print red;
```



```

else
:print not red;
@enduml

```

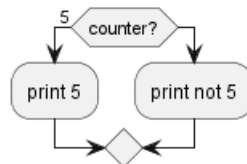


- if (...) equals (...) then

```

@startuml
if (counter?) equals (5) then
:print 5;
else
:print not 5;
@enduml

```



[Ref. QA-301]

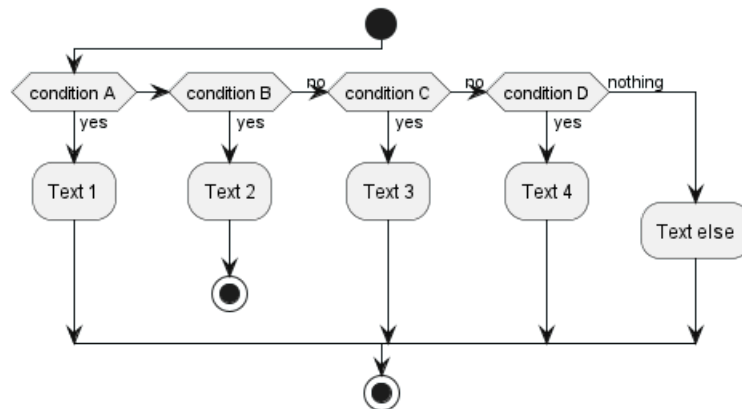
6.3.1 複数条件 (水平モード)

いくつもの条件分岐がある場合には、キーワード **elseif** を使用できます。(デフォルトで水平モードになります) :

```

@startuml
start
if (condition A) then (yes)
:Text 1;
elseif (condition B) then (yes)
:Text 2;
stop
(no) elseif (condition C) then (yes)
:Text 3;
(no) elseif (condition D) then (yes)
:Text 4;
else (nothing)
:Text else;
endif
stop
@enduml

```

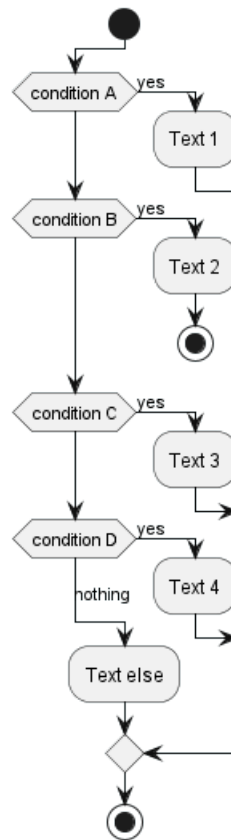


6.3.2 複数条件 (垂直モード)

!pragma useVerticalIf on コマンドを使用すると、垂直モードの分岐になります:

```

@startuml
!pragma useVerticalIf on
start
if (condition A) then (yes)
  :Text 1;
elseif (condition B) then (yes)
  :Text 2;
  stop
elseif (condition C) then (yes)
  :Text 3;
elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml
  
```



コマンドラインオプション-P を使用して pragma を指定することもできます：

```
java -jar plantuml.jar -PuseVerticalIf=on
```

[Refs. QA-3931, issue-582]

6.4 スイッチとケース [switch, case, endswitch]

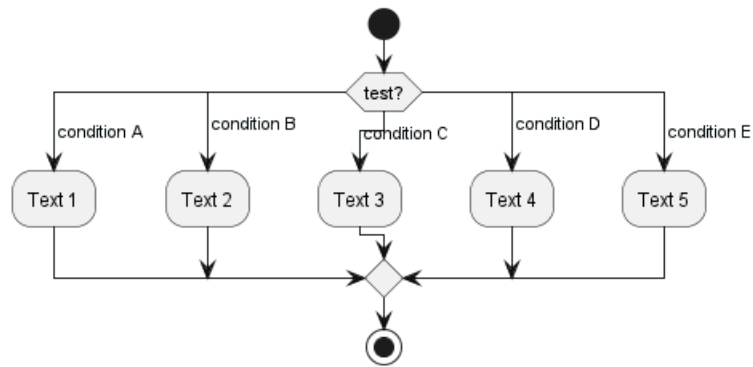
switch,case,endswitch キーワードを使って、図の中にスイッチを入れることができます。

ラベルは括弧を使って提供できます。

```

@startuml
start
switch (test?)
case ( condition A )
  :Text 1;
case ( condition B )
  :Text 2;
case ( condition C )
  :Text 3;
case ( condition D )
  :Text 4;
case ( condition E )
  :Text 5;
endswitch
stop
@enduml
  
```



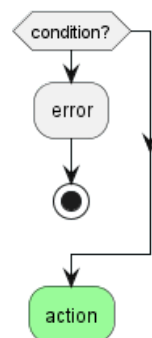


6.5 アクションの停止を伴う条件文 [kill, detach]

if 節内でアクションを停止できます。

```

@startuml
if (condition?) then
    :error;
    stop
endif
#palegreen:action;
@enduml
  
```

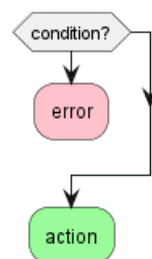


ただし、明確なアクションで停止したい場合は、キーワード「kill」または「detach」を使用できます：

- kill

```

@startuml
if (condition?) then
    #pink:error;
    kill
endif
#palegreen:action;
@enduml
  
```



[Ref. QA-265]

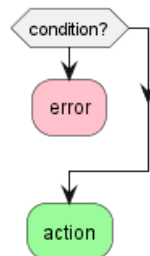
- detach



```

@startuml
if (condition?) then
  #pink:error;
  detach
endif
#palegreen:action;
@enduml

```



6.6 繰り返し（後判定）

繰り返し処理（後判定）がある場合には、キーワード `repeat` と `repeat while` を使用できます。

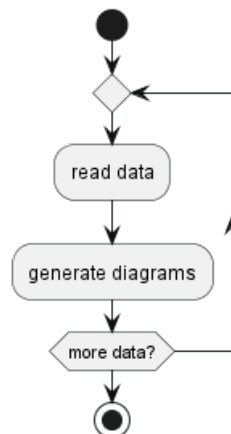
```

@startuml
start

repeat
  :read data;
  :generate diagrams;
repeat while (more data?)

stop
@enduml

```



アクティビティを `repeat` の戻り先にもすることもできます。また、`backward` キーワードを使用して、戻りのパスにアクティビティを挿入することもできます。

```

@startuml
start

repeat :foo as starting label;
  :read data;
  :generate diagrams;

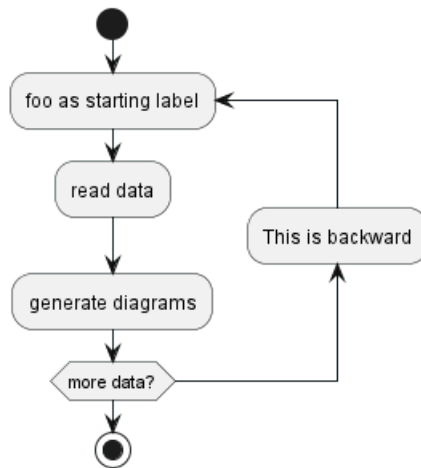
```



```
backward:This is backward;
repeat while (more data?)

stop

@enduml
```

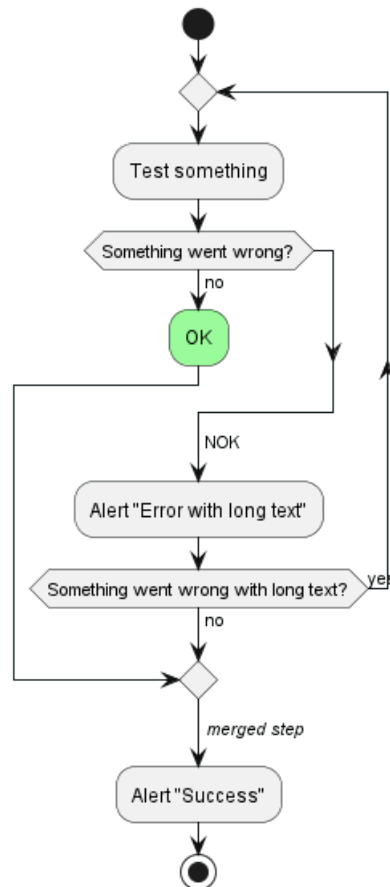


[Ref. QA-5826]

6.7 repeat 節を中断する [break]

アクションの後で **break** キーワードを使うと、ループを中断することができます。

```
@startuml
start
repeat
  :Test something;
  if (Something went wrong?) then (no)
    #palegreen:OK;
    break
  endif
  ->NOK;
  :Alert "Error with long text";
repeat while (Something went wrong with long text?) is (yes) not (no)
->//merged step//;
:Alert "Success";
stop
@enduml
```



[Ref. QA-6105]

6.8 Goto and Label Processing [label, goto]

It is currently only experimental

You can use `label` and `goto` keywords to denote goto processing, with:

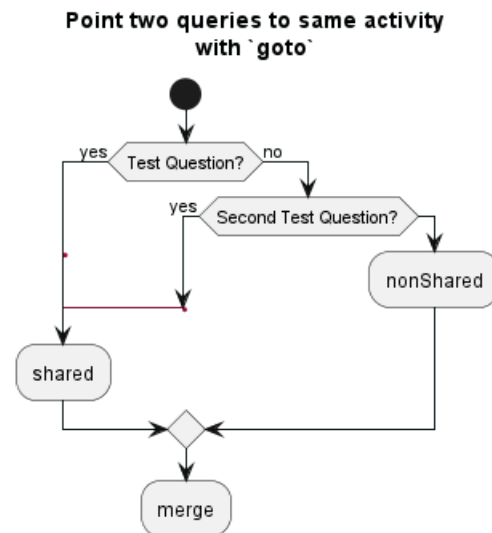
- `label <label_name>`
- `goto <label_name>`

```

@startuml
title Point two queries to same activity\nwith `goto`
start
if (Test Question?) then (yes)
'space label only for alignment
label sp_lab0
label sp_lab1
'real label
label lab
:shared;
else (no)
if (Second Test Question?) then (yes)
label sp_lab2
goto sp_lab1
else
:nonShared;
endif
endif
:merge;
  
```



@enduml



[Ref. QA-15026, QA-12526 and initially QA-1626]

6.9 繰り返し（前判定）

繰り返し処理（前判定）がある場合には、キーワード `while` と `endwhile` を使用できます。

@startuml

start

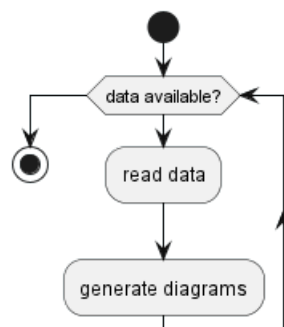
```

while (data available?)
  :read data;
  :generate diagrams;
endwhile

```

stop

@enduml



キーワード `endwhile` の後ろ、または、キーワード `is` を使用することで、ラベルを与えることができます。

@startuml

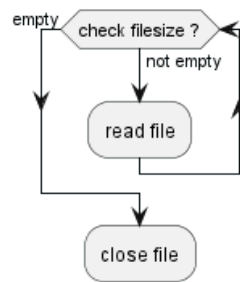
```

while (check filesize ?) is (not empty)
  :read file;
endwhile (empty)
:close file;

```



```
@enduml
```

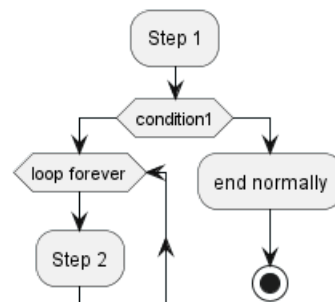


`detach` を使用して無限ループを作る場合は、`-[hidden]->` を使用して不要な矢印を隠すと良いでしょう。

```

@startuml
:Step 1;
if (condition1) then
  while (loop forever)
    :Step 2;
  endwhile
  -[hidden]->
  detach
else
  :end normally;
  stop
endif
@enduml

```



6.10 並列処理

キーワード `fork`、`fork again` そして `end fork` または `end merge` を使用して、並列処理を記述することができます。

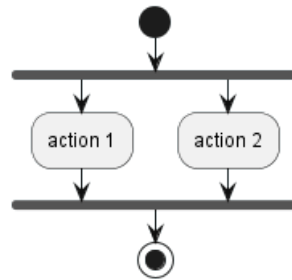
6.10.1 単純な `fork`

```

@startuml
start
fork
  :action 1;
fork again
  :action 2;
end fork
stop
@enduml

```

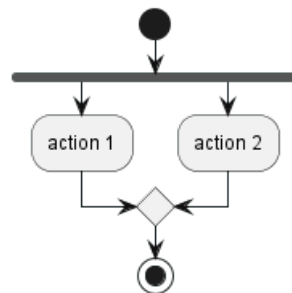




6.10.2 end merge を使った fork

```

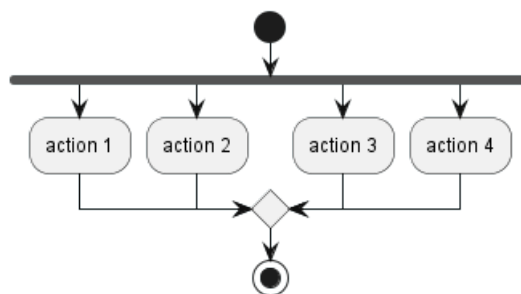
@startuml
start
fork
    :action 1;
fork again
    :action 2;
end merge
stop
@enduml
  
```



[Ref. QA-5320]

```

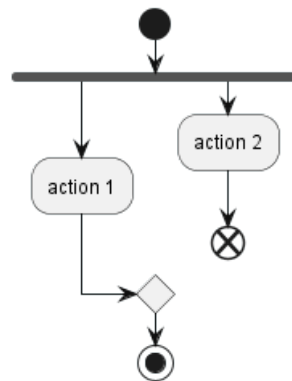
@startuml
start
fork
    :action 1;
fork again
    :action 2;
fork again
    :action 3;
fork again
    :action 4;
end merge
stop
@enduml
  
```



```

@startuml
start
fork
    :action 1;
fork again
    :action 2;
end
end merge
stop
@enduml

```



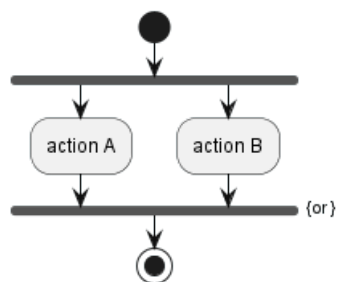
[Ref. QA-13731]

6.10.3 end fork のラベル、または join 仕様 (UML joinspec):

```

@startuml
start
fork
    :action A;
fork again
    :action B;
end fork {or}
stop
@enduml

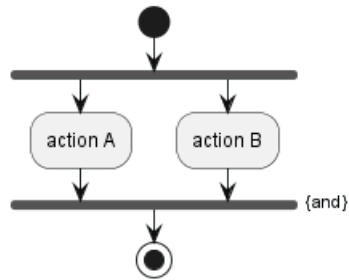
```



```

@startuml
start
fork
    :action A;
fork again
    :action B;
end fork {and}
stop
@enduml

```

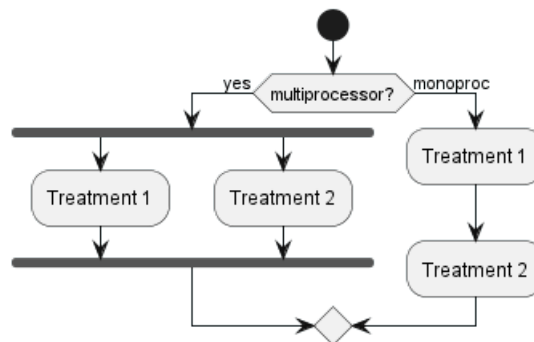


[Ref. QA-5346]

6.10.4 その他の例

```

@startuml
start
if (multiprocessor?) then (yes)
  fork
    :Treatment 1;
  fork again
    :Treatment 2;
  end fork
else (monoproc)
  :Treatment 1;
  :Treatment 2;
endif
@enduml
  
```



6.11 処理の分岐

6.11.1 Split

split、split again、end split キーワードを使って、プロセスの分岐を表すことができます。

```

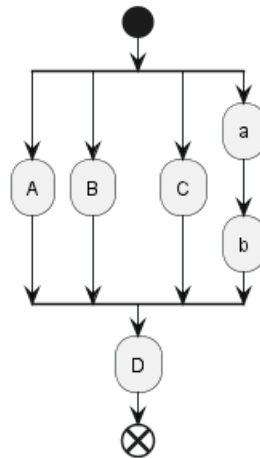
@startuml
start
split
  :A;
split again
  :B;
split again
  :C;
split again
  :a;
  
```



```

    :b;
end split
:D;
end
@enduml

```



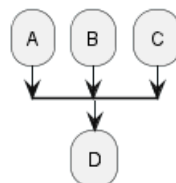
6.11.2 入力の分岐 (複数開始)

入力の分岐を表現するには、`hidden` で矢印を隠します。

```

@startuml
split
-[hidden]->
:A;
split again
-[hidden]->
:B;
split again
-[hidden]->
:C;
end split
:D;
@enduml

```

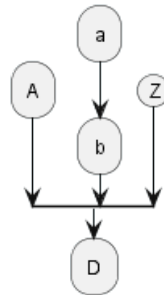


```

@startuml
split
-[hidden]->
:A;
split again
-[hidden]->
:a;
:b;
split again
-[hidden]->
(Z)
end split

```

```
:D;
@enduml
```

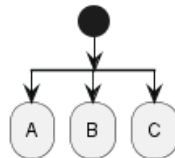


[Ref. QA-8662]

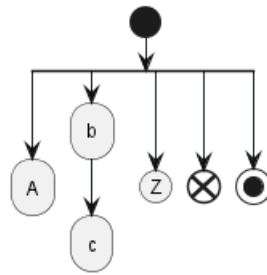
6.11.3 出力の分岐 (複数終了)

出力の分岐を表現するには、**kill** または **detach** を使用します。

```
@startuml
start
split
  :A;
  kill
split again
  :B;
  detach
split again
  :C;
  kill
end split
@enduml
```



```
@startuml
start
split
  :A;
  kill
split again
  :b;
  :c;
  detach
split again
  (Z)
  detach
split again
  end
split again
  stop
end split
@enduml
```



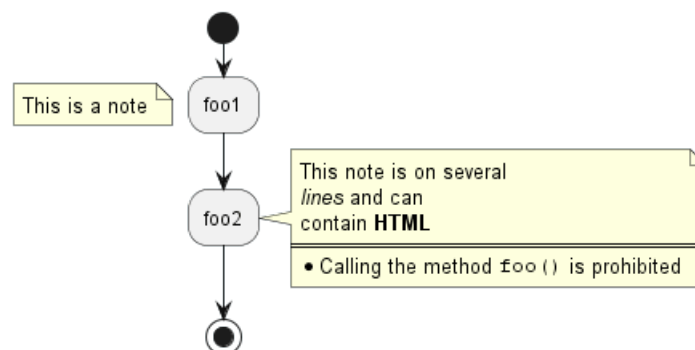
6.12 注釈

Creole 表記の Wiki 構文を使用することで、テキストの書式設定ができます。

キーワード `floating` を使用し、注釈を遊離させることもできます。

```

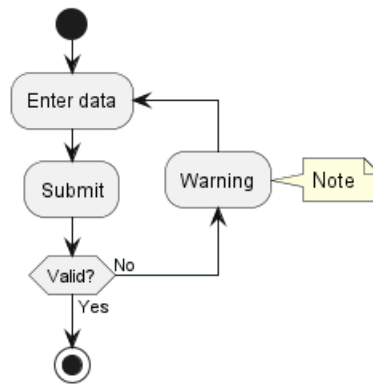
@startuml
start
:foo1;
floating note left: This is a note
:foo2;
note right
  This note is on several
  //lines// and can
  contain <b>HTML</b>
  ====
  * Calling the method "foo()" is prohibited
end note
stop
@enduml
  
```



戻り方向 (backward) のアクティビティに注釈をつけることもできます。

```

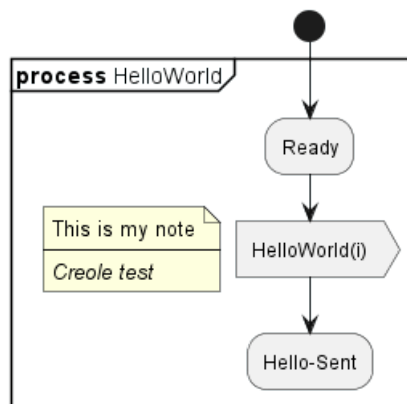
@startuml
start
repeat :Enter data;
:Submit;
backward :Warning;
note right: Note
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
  
```



パーティションにノートを追加することもできます。

```

@startuml
start
partition "**process** HelloWorld" {
    note
        This is my note
        ----
        //Creole test//
    end note
    :Ready;
    :HelloWorld(i)>
    :Hello-Sent;
}
@enduml
  
```



[Ref. QA-2398]

6.13 色指定

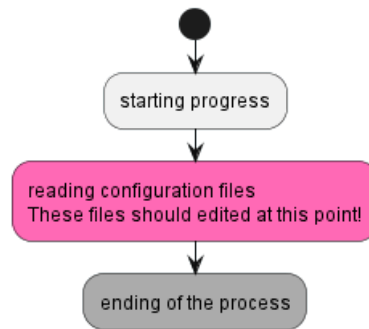
各アクティビティに、色を指定することができます。

```

@startuml
start
:starting progress;
#HotPink:reading configuration files
These files should edited at this point!;
#AAAAAA:ending of the process;

@enduml
  
```

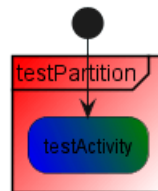




グラデーションを使用することもできます。

```

@startuml
start
partition #red/white testPartition {
    #blue\green:testActivity;
}
@enduml
  
```



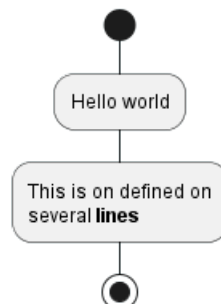
[Ref. QA-4906]

6.14 矢印無しの線

`skinparam ArrowHeadColor none` を指定すると、アクティビティの接続線を矢印無しにすることができます。

```

@startuml
skinparam ArrowHeadColor none
start
:Hello world;
:This is on defined on
several **lines**;
stop
@enduml
  
```

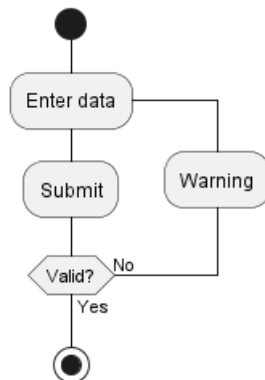


```

@startuml
skinparam ArrowHeadColor none
start
repeat :Enter data;
:Submit;
  
```



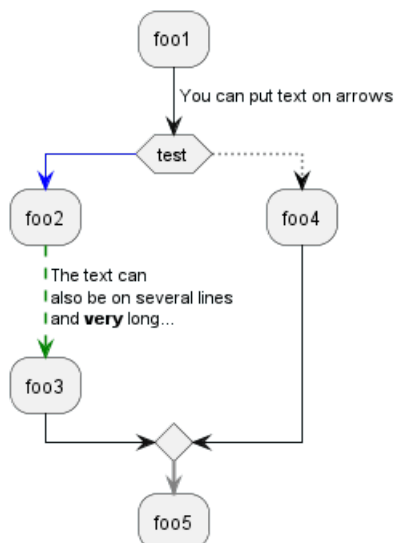
```
backward :Warning;
repeat while (Valid?) is (No) not (Yes)
stop
@enduml
```



6.15 矢印

記号-> を用いて、矢印にテキストを添えることができ、また、色を変えることもできます。
点線、破線、太線、または、矢印なし、もまた可能です。

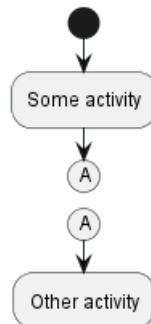
```
@startuml
:foo1;
-> You can put text on arrows;
if (test) then
-[#blue]->
:foo2;
-[#green,dashed]-> The text can
also be on several lines
and very long...;
:foo3;
else
-[#black,dotted]->
:foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
```



6.16 コネクタ

半角括弧を使用して、コネクタを記述することができます。

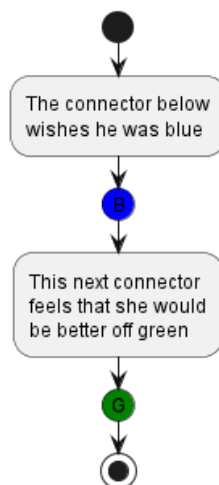
```
@startuml
start
:Some activity;
(A)
detach
(A)
:Other activity;
@enduml
```



6.17 コネクタの色

コネクタに色を設定することができます。

```
@startuml
start
:The connector below
wishes he was blue;
#blue:(B)
:This next connector
feels that she would
be better off green;
#green:(G)
stop
@enduml
```



[Ref. QA-10077]

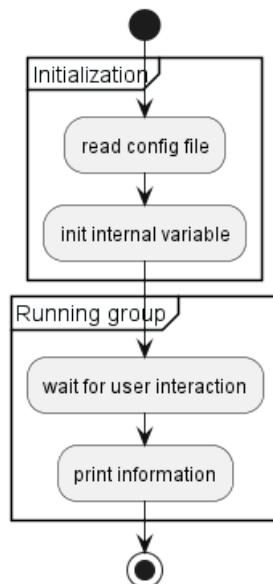
6.18 グループ化 (パーティション)

6.18.1 グループ

グループを定義して複数のアクティビティをまとめることができます:

```
@startuml
start
group Initialization
    :read config file;
    :init internal variable;
end group
group Running group
    :wait for user interaction;
    :print information;
end group

stop
@enduml
```

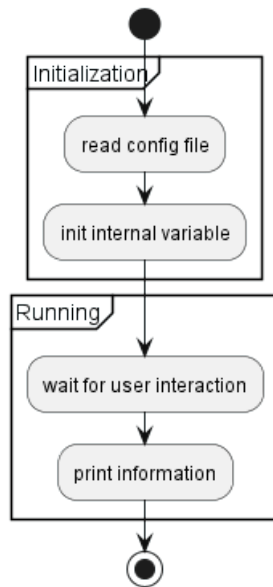


6.18.2 パーティション

パーティションを定義して複数のアクティビティをまとめることができます:

```
@startuml
start
partition Initialization {
    :read config file;
    :init internal variable;
}
partition Running {
    :wait for user interaction;
    :print information;
}

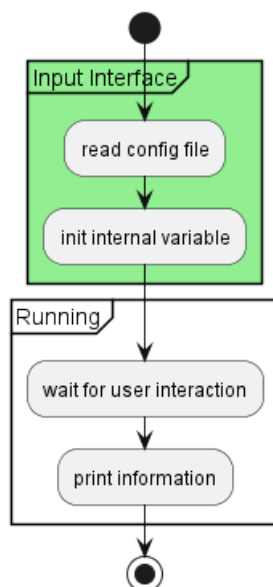
stop
@enduml
```



パーティションの色を変更することができます:

```

@startuml
start
partition #lightGreen "Input Interface" {
    :read config file;
    :init internal variable;
}
partition Running {
    :wait for user interaction;
    :print information;
}
stop
@enduml
  
```



[Ref. QA-2793]

パーティションにリンクを追加することもできます:

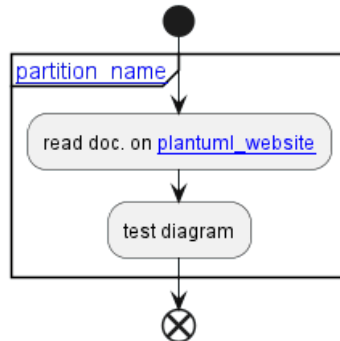
```

@startuml
start
  
```

```

partition "[[http://plantuml.com partition_name]]" {
    :read doc. on [[http://plantuml.com plantuml_website]];
    :test diagram;
}
end
@enduml

```



[Ref. QA-542]

6.18.3 グループ、パーティション、パッケージ、四角形、カード

- グループ (group)、
- パーティション (partition)、
- パッケージ (package)、
- 四角形 (rectangle)、
- カード (card)

を定義して複数のアクティビティをまとめることができます:

```

@startuml
start
group Group
    :Activity;
end group
floating note: Note on Group

partition Partition {
    :Activity;
}
floating note: Note on Partition

package Package {
    :Activity;
}
floating note: Note on Package

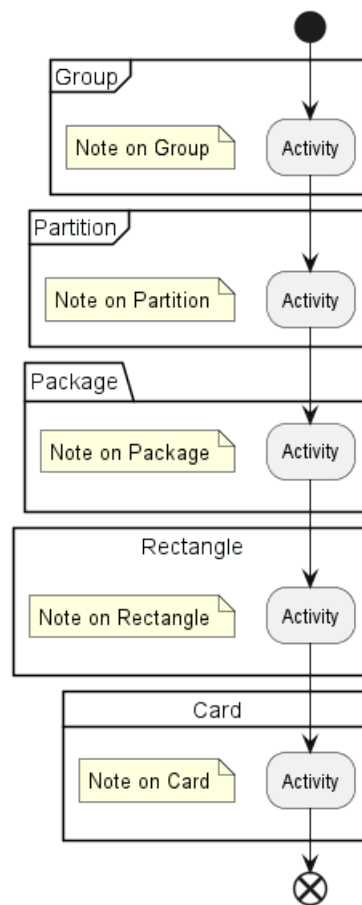
rectangle Rectangle {
    :Activity;
}
floating note: Note on Rectangle

card Card {
    :Activity;
}
floating note: Note on Card
end

```



@enduml



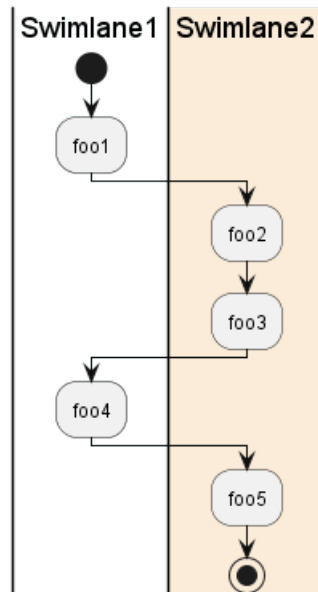
6.19 スイムレーン

パイプ記号 | を用いて、複数のスイムレーンを定義することができます。

さらに、スイムレーン毎に色を変えることができます。

```

@startuml
|Swimlane1|
start
:foo1;
|#AntiqueWhite|Swimlane2|
:foo2;
:foo3;
|Swimlane1|
:foo4;
|Swimlane2|
:foo5;
stop
@enduml
  
```

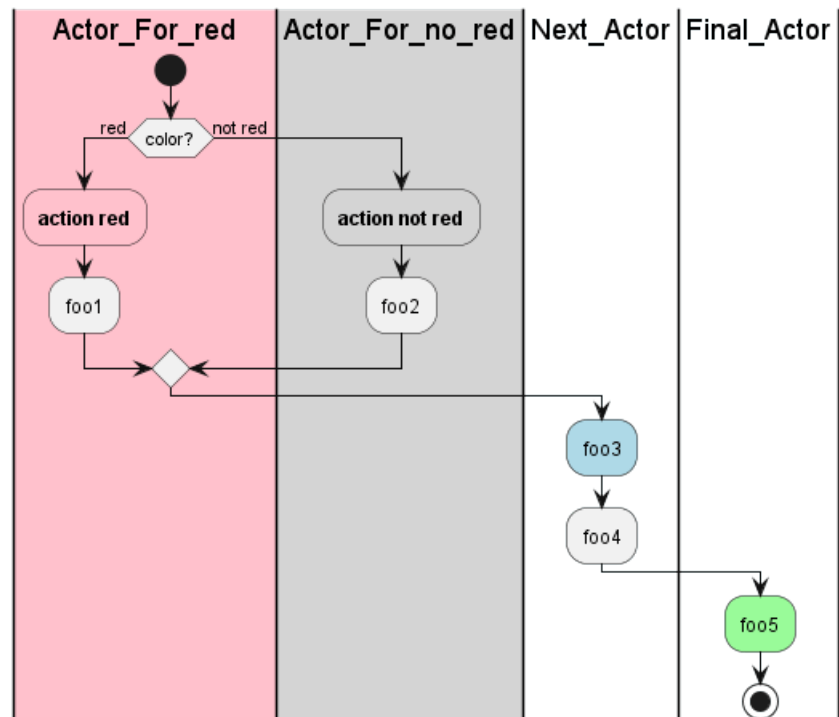


スイムレーンの中で、if 条件文や repeat、while のループを使用できます。

```

@startuml
|#pink|Actor_For_red|
start
if (color?) is (red) then
#pink:**action red**;
:foo1;
else (not red)
|#lightgray|Actor_For_no_red|
#lightgray:**action not red**;
:foo2;
endif
|Next_Actor|
#lightblue:foo3;
:foo4;
|Final_Actor|
#palegreen:foo5;
stop
@enduml

```



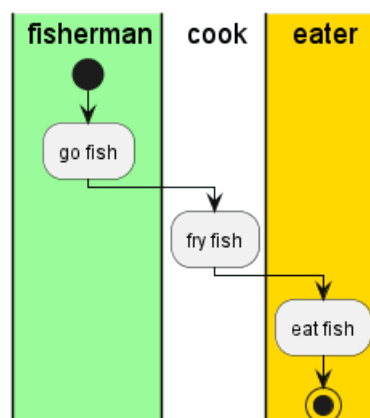
次の構文で、スイムレーンに別名 (alias) を付けることができます。

- |[#<color>]|<swimlane_alias>| <swimlane_title>

```

@startuml
|#palegreen|f| fisherman
|c| cook
|#gold|e| eater
|f|
start
:go fish;
|c|
:fry fish;
|e|
:eat fish;
stop
@enduml

```

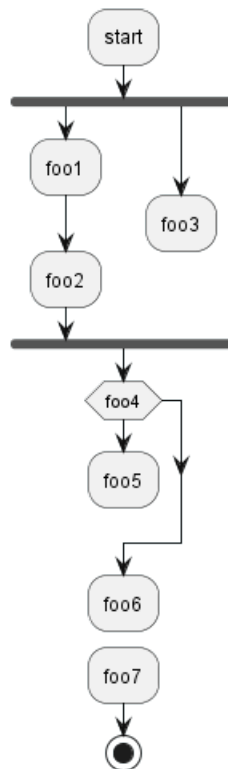


[Ref. QA-2681]

6.20 矢印の除去 (**detach**, **kill**)

キーワード **detach** または **kill** を使用して、矢印を取り除くことができます。

```
@startuml
: start;
fork
: foo1;
: foo2;
fork again
: foo3;
detach
endfork
if (foo4) then
: foo5;
detach
endif
: foo6;
detach
: foo7;
stop
@enduml
```



- **kill**

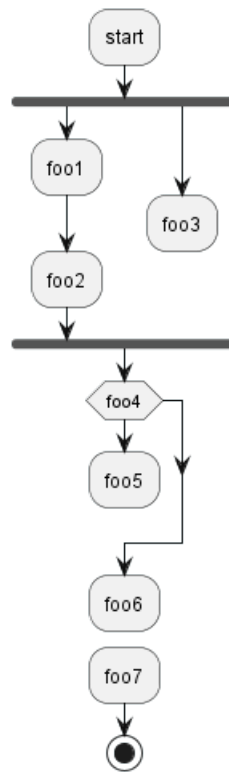
```
@startuml
: start;
fork
: foo1;
: foo2;
fork again
: foo3;
kill
endfork
if (foo4) then
: foo5;
endif
: foo6;
: foo7;
stop
@enduml
```



```

:foo5;
kill
endif
:foo6;
kill
:foo7;
stop
@enduml

```



6.21 SDL 図

終端記号; を置き換えることで、アクティビティの表現形式を変えることができます：

- |
- <
- >
- /
- \\\
-]
- }

```

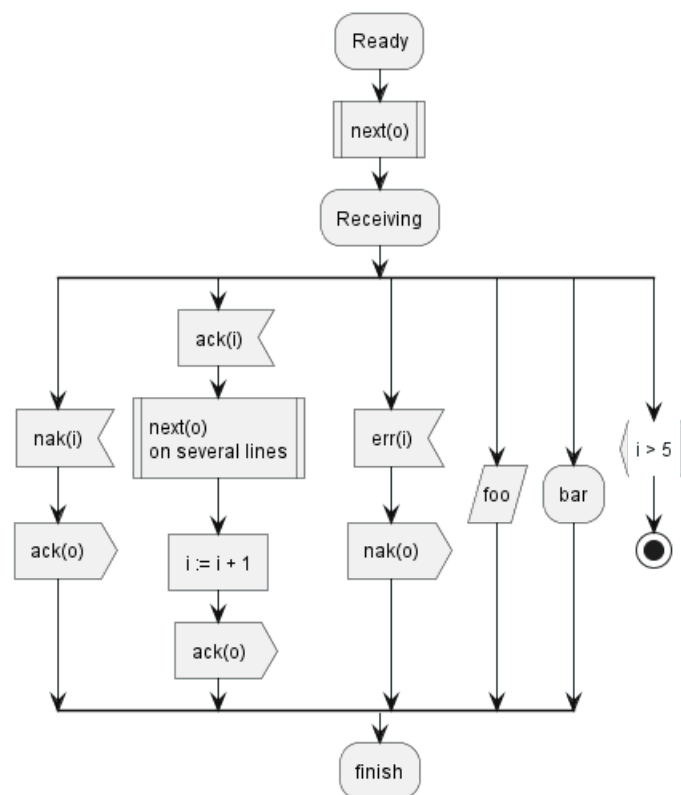
@startuml
:Ready;
:next(o)|
:Receiving;
split
:nak(i)<
:ack(o)>
split again
:ack(i)<
:next(o)

```

```

on several lines|
:i := i + 1]
:ack(o)>
split again
:err(i)<
:nak(o)>
split again
:foo/
split again
:bar\\
split again
:i > 5}
stop
end split
:finish;
@enduml

```



6.22 完全な例

```

@startuml
start
:ClickServlet.handleRequest();
:new page;
if (Page.onSecurityCheck) then (true)
:Page.onInit();
if (isForward?) then (no)
:Process controls;
if (continue processing?) then (no)
stop
endif
endif

```

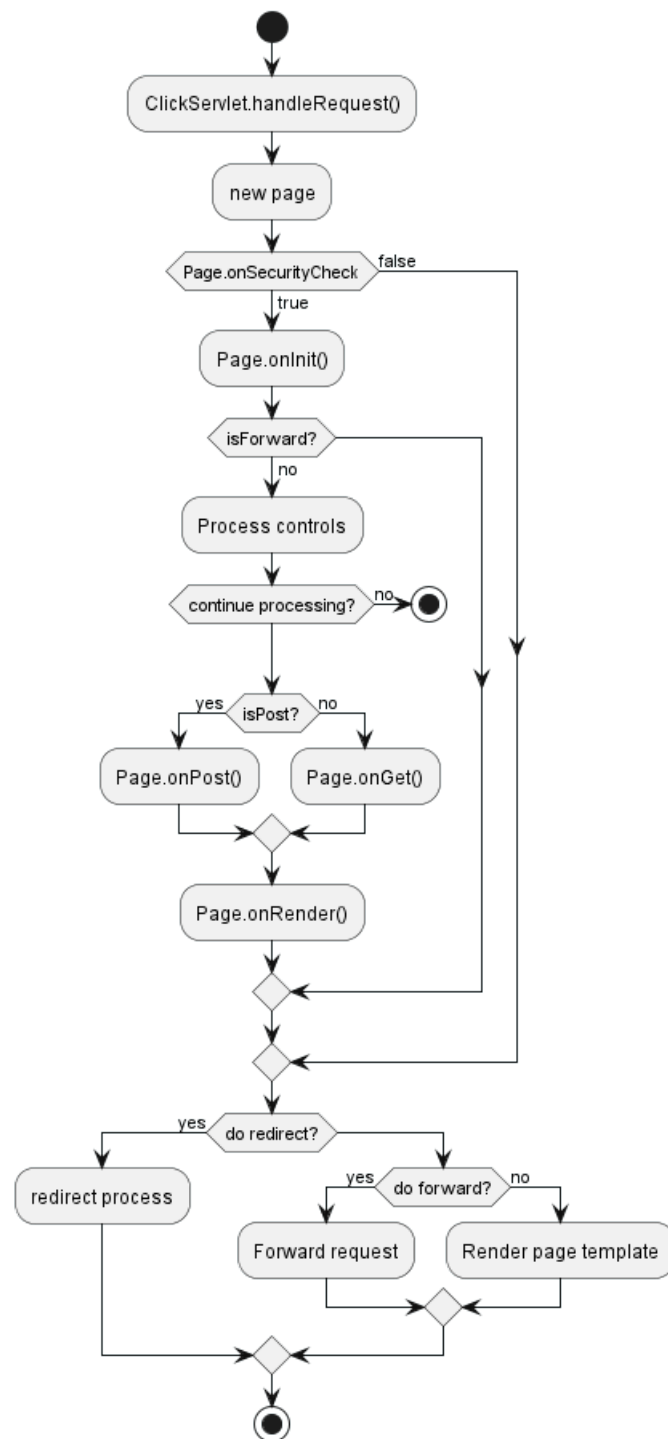


```
    if (isPost?) then (yes)
        :Page.onPost();
    else (no)
        :Page.onGet();
    endif
    :Page.onRender();
endif
else (false)
endif

if (do redirect?) then (yes)
    :redirect process;
else
    if (do forward?) then (yes)
        :Forward request;
    else (no)
        :Render page template;
    endif
endif

stop

@enduml
```



6.23 条件のスタイル

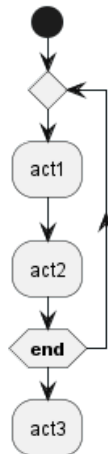
6.23.1 inside スタイル (デフォルト)

```

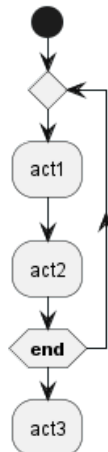
@startuml
skinparam conditionStyle inside
start
repeat
    :act1;
    :act2;
repeatwhile (<b>end)
:act3;
  
```



```
@enduml
```

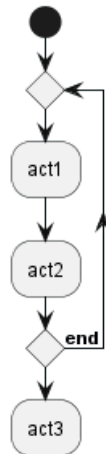


```
@startuml
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



6.23.2 diamond スタイル

```
@startuml
skinparam conditionStyle diamond
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
```



6.23.3 InsideDiamond (または *foo1*) スタイル

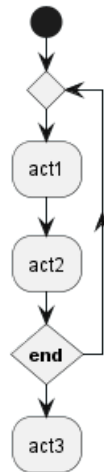
```

@startuml
skinparam conditionStyle InsideDiamond
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
  
```



```

@startuml
skinparam conditionStyle foo1
start
repeat
  :act1;
  :act2;
repeatwhile (<b>end)
:act3;
@enduml
  
```



[Ref. QA-1290 and #400]

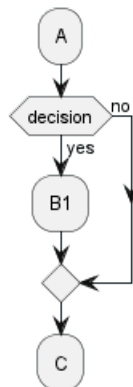
6.24 条件終了のスタイル

6.24.1 diamond スタイル (デフォルト)

- 分岐が1つの場合

```

@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
    :B1;
else (no)
endif
:C;
@enduml
  
```

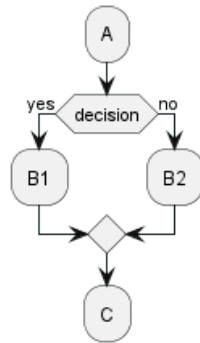


- 分岐が2つ (B1, B2) の場合

```

@startuml
skinparam ConditionEndStyle diamond
:A;
if (decision) then (yes)
    :B1;
else (no)
    :B2;
endif
:C;
@enduml
  
```



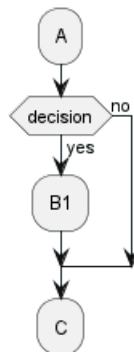


6.24.2 水平ライン (hline) スタイル

- 分岐が1つの場合

```

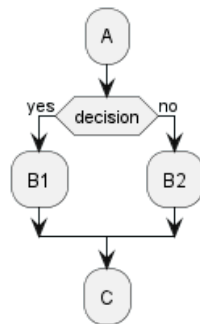
@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
    :B1;
else (no)
endif
:C;
@enduml
  
```



- 分岐が2つ (B1, B2) の場合

```

@startuml
skinparam ConditionEndStyle hline
:A;
if (decision) then (yes)
    :B1;
else (no)
    :B2;
endif
:C;
@enduml
  
```



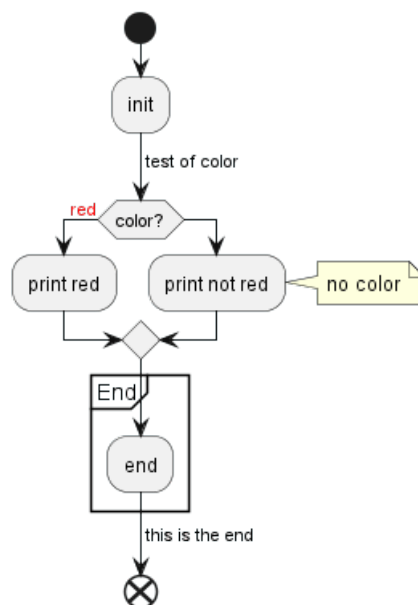
[Ref. QA-4015]

6.25 グローバル (global) スタイルの使用

6.25.1 スタイル無し (デフォルト)

```

@startuml
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
  
```



6.25.2 スタイル有り

スタイルを指定して要素の見た目を変更することができます。

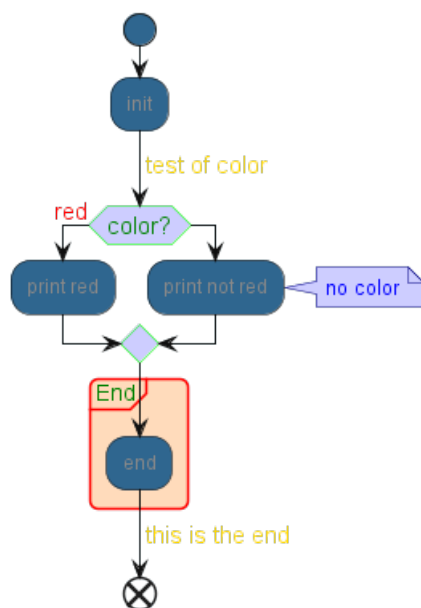
```

@startuml
  
```



```
<style>
activityDiagram {
    BackgroundColor #33668E
    BorderColor #33668E
    FontColor #888
    FontName arial

    diamond {
        BackgroundColor #ccf
        LineColor #00FF00
        FontColor green
        FontName arial
        FontSize 15
    }
    arrow {
        FontColor gold
        FontName arial
        FontSize 15
    }
    partition {
        LineColor red
        FontColor green
        RoundCorner 10
        BackgroundColor PeachPuff
    }
    note {
        FontColor Blue
        LineColor Navy
        BackgroundColor #ccf
    }
}
document {
    BackgroundColor transparent
}
</style>
start
:init;
-> test of color;
if (color?) is (<color:red>red) then
:print red;
else
:print not red;
note right: no color
endif
partition End {
:end;
}
-> this is the end;
end
@enduml
```



7 コンポーネント図

いくつかの例をご覧ください。

Advantages of PlantUML:

- **Simplicity:** With PlantUML, you can create component diagrams using simple and intuitive text-based descriptions, eliminating the need for complex drawing tools.
- **Integration:** PlantUML seamlessly integrates with various tools and platforms, making it a versatile choice for developers and architects.
- **Collaboration:** The PlantUML forum offers a platform for users to discuss, share, and seek assistance on their diagrams, fostering a collaborative community.

7.1 コンポーネント

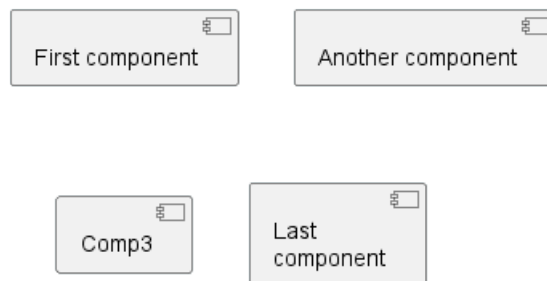
コンポーネントは括弧でくくります。

また、`component` キーワードでもコンポーネントを定義できます。この場合、コンポーネント名に空白や特殊文字を含まなければ括弧を省略することができます。

コンポーネントには `as` キーワードにより別名をつけることができます。この別名は、後でリレーションを定義するときに使えます。

```
@startuml
[First component]
[Another component] as Comp2
component Comp3
component [Last\ncomponent] as Comp4
```

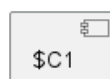
@enduml



7.1.1 名前に関する注意

コンポーネント名が `$` で始まる場合は、後で非表示にしたり削除したりすることができません。これは、`hide` および `remove` コマンドが `$tag` のような名称をコンポーネント名ではなくタグであると解釈するためです。そのようなコンポーネントを後から削除するには、別名を付けるか、タグを付けてください。

```
@startuml
component [$C1]
component [$C2] $C2
component [$C2] as dollarC2
remove $C1
remove $C2
remove dollarC2
@enduml
```



7.2 インタフェース

インタフェースは丸括弧 () でシンボルを囲うことで定義できます。(何故なら見た目が丸いからです。)

もちろん `interface` キーワードを使って定義することもできます。 `as` キーワードでエイリアスを定義できます。このエイリアスは後で、関係を定義する時に使えます。

後で説明されますが、インタフェースの定義は省略可能です。

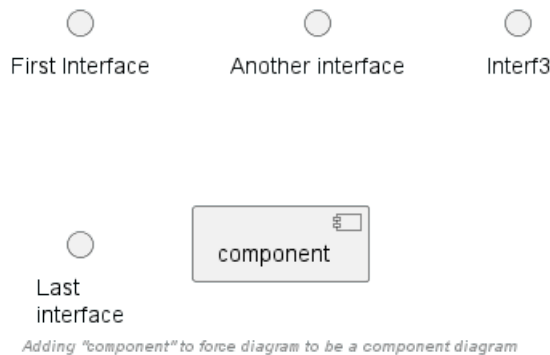
```
@startuml
```

```
() "First Interface"
() "Another interface" as Interf2
interface Interf3
interface "Last\ninterface" as Interf4
```

```
[component]
```

```
footer //Adding "component" to force diagram to be a **component diagram**//
```

```
@enduml
```



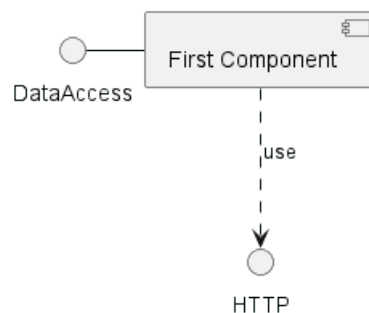
7.3 基本的な例

要素間の関係は、破線 (..)、直線 (--), 矢印 (-->) の組合せで構成されます。

```
@startuml
```

```
DataAccess - [First Component]
[First Component] ..> HTTP : use
```

```
@enduml
```



7.4 ノートの使用方法

オブジェクトに関連のあるノートを作成するには `note left of`、`note right of`、`note top of`、`note bottom of` キーワードを使います。

```
@startuml
```

```
[Component] as C
```



```

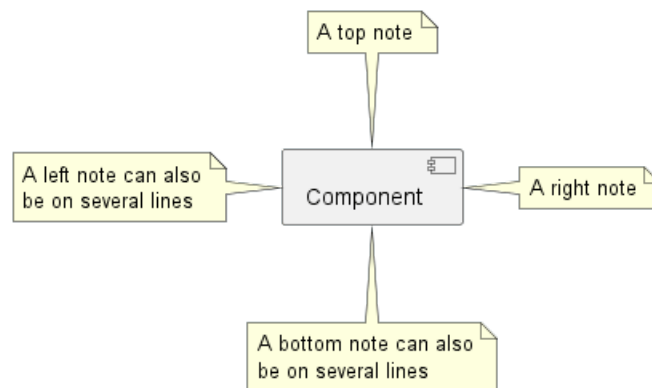
note top of C: A top note

note bottom of C
  A bottom note can also
  be on several lines
end note

note left of C
  A left note can also
  be on several lines
end note

note right of C: A right note
@enduml

```



または `note` キーワードを使ってノートを作成し、... 記号 (または任意の矢印記号 (-、--、...)) を使ってオブジェクトに紐づけることができます。

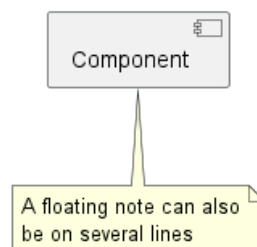
```

@startuml
[Component] as C

note as N
  A floating note can also
  be on several lines
end note

C .. N
@enduml

```



別の例:

```

@startuml

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

```

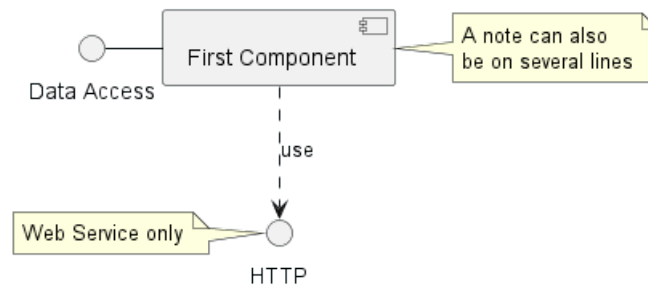


```
note left of HTTP : Web Service only
```

```
note right of [First Component]
```

```
  A note can also  
  be on several lines  
end note
```

```
@enduml
```



7.5 コンポーネントのグループ化

いくつかのキーワードをグループコンポーネントやインタフェースに使用することができます：

- package
- node
- folder
- frame
- cloud
- database

```
@startuml
```

```
package "Some Group" {
  HTTP - [First Component]
  [Another Component]
}
```

```
node "Other Groups" {
  FTP - [Second Component]
  [First Component] --> FTP
}
```

```
cloud {
  [Example 1]
}
```

```
database "MySQL" {
  folder "This is my folder" {
    [Folder 3]
  }
  frame "Foo" {
    [Frame 4]
  }
}
```

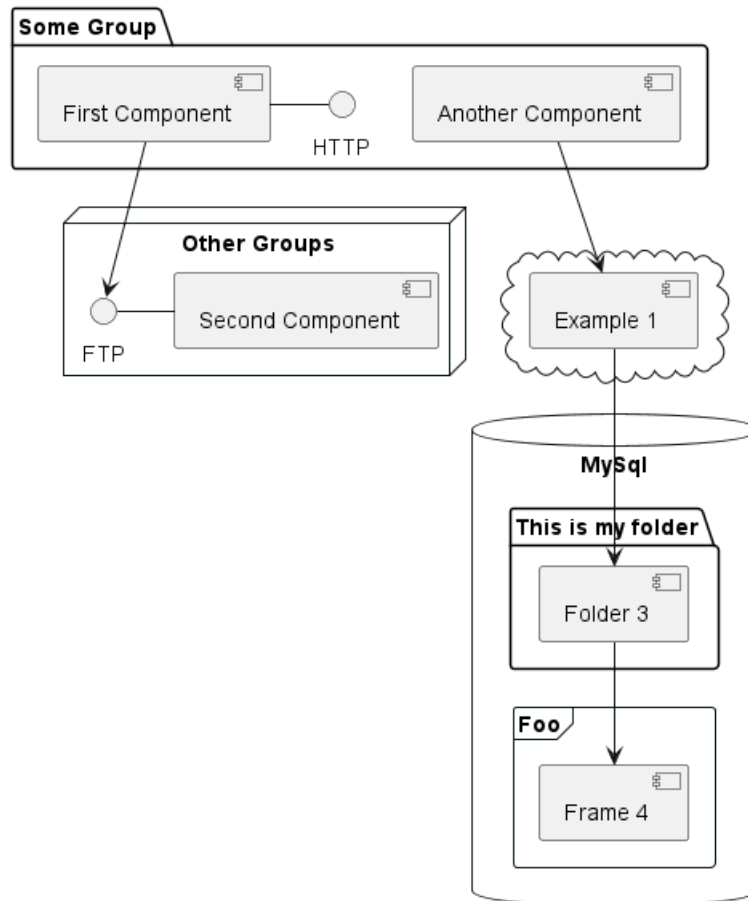


```

[Another Component] --> [Example 1]
[Example 1] --> [Folder 3]
[Folder 3] --> [Frame 4]

```

```
@enduml
```



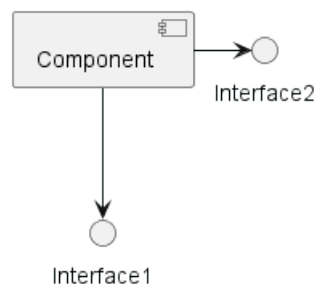
7.6 矢印の方向を変える

デフォルトではクラス間のリンクは2つのダッシュ -- を持っており垂直方向に配向されています。次のように単一のダッシュ（またはドット）を置くことによって水平方向のリンクを使用することが可能です：

```

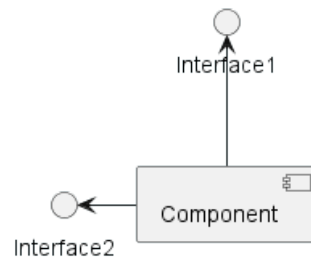
@startuml
[Component] --> Interface1
[Component] -> Interface2
@enduml

```



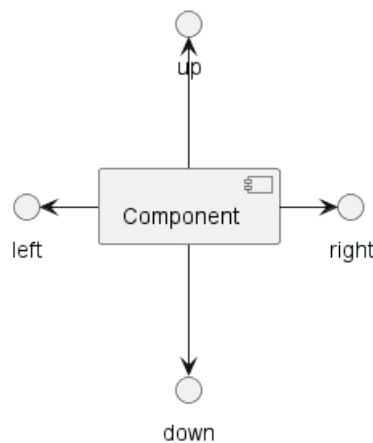
リンクを反対にすることで方向を変更することもできます。

```
@startuml
Interface1 <-- [Component]
Interface2 <- [Component]
@enduml
```



また、**left** を加えることで矢印の向きを変更することもできます。**right, up, down** などのキーワードを矢印の間に記述します。:

```
@startuml
[Component] -left-> left
[Component] -right-> right
[Component] -up-> up
[Component] -down-> down
@enduml
```

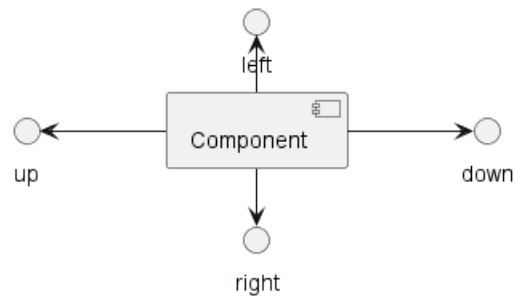


方向の最初の文字のみを使用して矢印を短くすることができます（例えば、**-down-** の代わりに **-d-**）、または最初の 2 文字 (**-do-**)。

この機能を悪用してはならないことに注意してください: *Graphviz* は微調整の必要がない良い結果を通常は与えてくれます。

left to right direction パラメータを指定した場合は、次のようになります。:

```
@startuml
left to right direction
[Component] -left-> left
[Component] -right-> right
[Component] -up-> up
[Component] -down-> down
@enduml
```



7.7 UML2 表記の使用

デフォルトでは、UML2 表記が使用されます (*v1.2020.13-14* 以降)。

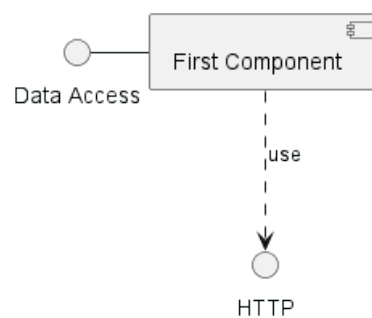
```
@startuml
```

```
interface "Data Access" as DA
```

```
DA - [First Component]
```

```
[First Component] ..> HTTP : use
```

```
@enduml
```



7.8 UML1 表記の使用

コマンド `skinparam componentStyle uml1` は、UML1 表記に切り替えるために使用されます。

```
@startuml
```

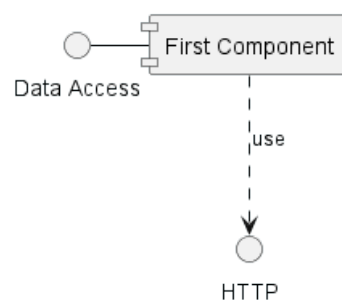
```
skinparam componentStyle uml1
```

```
interface "Data Access" as DA
```

```
DA - [First Component]
```

```
[First Component] ..> HTTP : use
```

```
@enduml
```



7.9 四角形表記の使用 (UML 表記をしない)

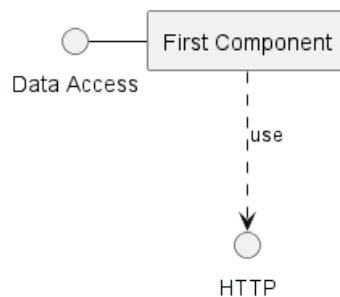
`skinparam componentStyle rectangle` コマンドを使用すると、UML 表記ではなく四角形による表記を行うことができます。

```
@startuml
skinparam componentStyle rectangle

interface "Data Access" as DA

DA - [First Component]
[First Component] ..> HTTP : use

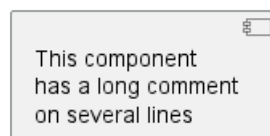
@enduml
```



7.10 長い説明

角括弧を使用して説明を複数行で記述することができます。

```
@startuml
component comp1 [
This component
has a long comment
on several lines
]
@enduml
```



7.11 個々の色

コンポーネント定義のあとに色を指定することができます。

```
@startuml
component [Web Server] #Yellow
@enduml
```



7.12 ステレオタイプでスプライトを使用

ステレオタイプのコンポーネント内にスプライトを使用することができます。

```
@startuml
sprite $businessProcess [16x16/16] {
```



```

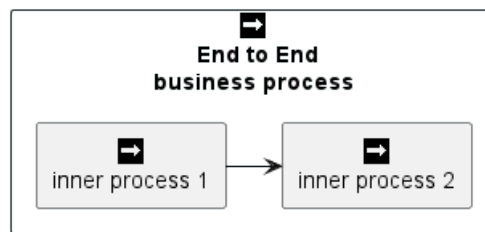
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFF0FFFFF
FFFFFFFFF00FFFF
FF000000000000FF
FF000000000000FF
FF000000000000FF
FFFFFFFFF00FFFF
FFFFFFFFF00FFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}

```

```

rectangle " End to End\nbusiness process" <<$businessProcess>> {
  rectangle "inner process 1" <<$businessProcess>> as src
  rectangle "inner process 2" <<$businessProcess>> as tgt
  src -> tgt
}
@enduml

```



7.13 見かけを変える

ダイアグラムの色やフォントを変更するには `skinparam` コマンドを使用します。

このコマンドは以下の場面で使用できます。

- ダイアグラム定義内で他のコマンドを同様に。
- インクルードされたファイル内。
- 設定ファイルのコマンドライン内や Ant タスク内。

ステレオタイプのコンポーネントおよびインタフェースのための特定の色とフォントを定義することができます。

```

@startuml

skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

skinparam component {
  FontSize 13
  BackgroundColor<<Apache>> Pink
  BorderColor<<Apache>> #FF6655
  FontName Courier
}

```

```

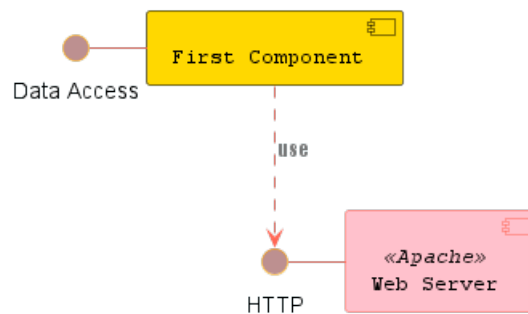
    BorderColor black
    BackgroundColor gold
    ArrowFontName Impact
    ArrowColor #FF6655
    ArrowFontColor #777777
}

() "Data Access" as DA
Component "Web Server" as WS << Apache >>

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - WS

@enduml

```



```

@startuml

skinparam component {
    backgroundColor<<static_lib>> DarkKhaki
    backgroundColor<<shared_lib>> Green
}

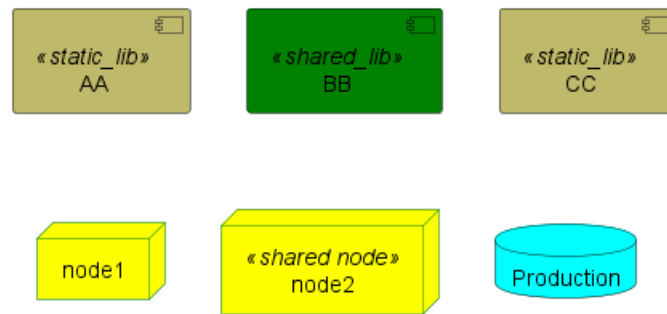
skinparam node {
    borderColor Green
    backgroundColor Yellow
    backgroundColor<<shared_node>> Magenta
}
skinparam databaseBackgroundColor Aqua

[AA] <<static_lib>>
[BB] <<shared_lib>>
[CC] <<static_lib>>

node node1
node node2 <<shared node>>
database Production

@enduml

```

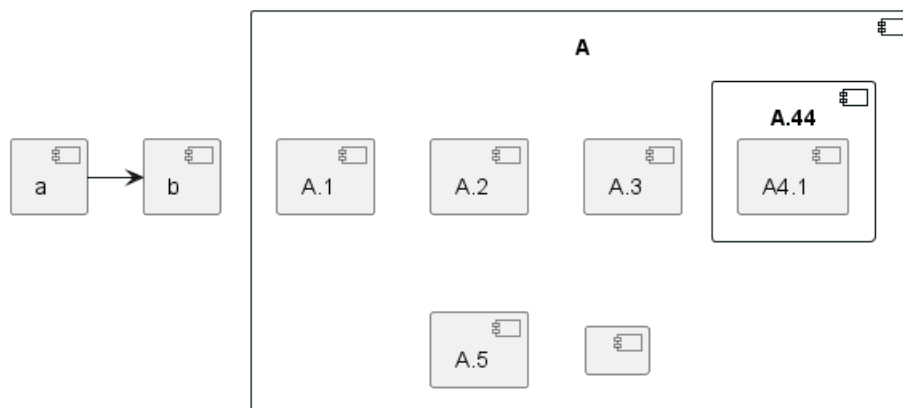


7.14 特有の skinparam

7.14.1 componentStyle

- デフォルトで (もしくは `skinparam componentStyle uml2` を指定することで)、コンポーネントにはアイコンが表示されます。

```
@startuml
skinparam BackgroundColor transparent
skinparam componentStyle uml2
component A {
    component "A.1" {
    }
    component A.44 {
        [A4.1]
    }
    component "A.2"
        [A.3]
    component A.5 [
A.5]
    component A.6 [
]
}
[a]->[b]
@enduml
```



- これを非表示にして四角形だけを表示するには、`skinparam componentStyle rectangle` を指定します。

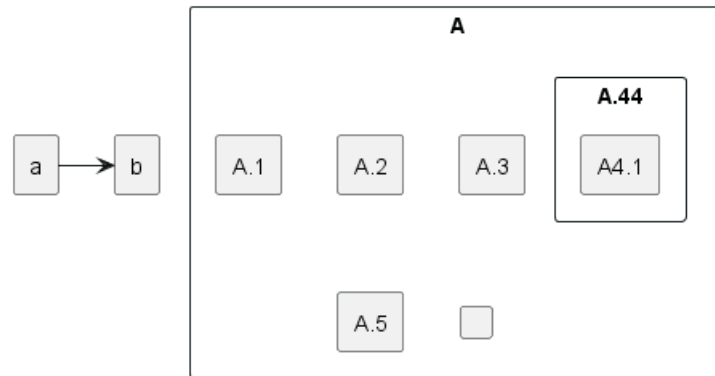
```
@startuml
skinparam BackgroundColor transparent
skinparam componentStyle rectangle
component A {
    component "A.1" {
    }
}
```



```

    component A.44 {
        [A4.1]
    }
    component "A.2"
    [A.3]
    component A.5 [
A.5]
    component A.6 [
]
    }
    [a]->[b]
@enduml

```



[Ref. 10798]

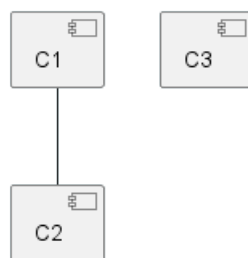
7.15 孤立したコンポーネントを非表示または削除する

デフォルトでは、すべてのコンポーネントが表示されます：

```

@startuml
component C1
component C2
component C3
C1 -- C2
@enduml

```



- しかし、`hide @unlinked` で、孤立したコンポーネントを非表示にできます：

```

@startuml
component C1
component C2
component C3
C1 -- C2

hide @unlinked
@enduml

```



- もしくは、`remove @unlinked` で、孤立したコンポーネントを削除できます：

```

@startuml
component C1
component C2
component C3
C1 -- C2

remove @unlinked
@enduml

```



[Ref. QA-11052]

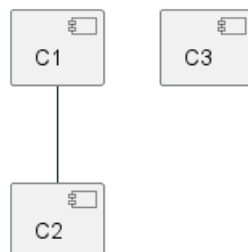
7.16 タグ付け、またはワイルドカードによるコンポーネントの非表示、削除、復元
 \$ を使って、コンポーネントにタグ付けすることができます。それから、コンポーネント個別、またはタグごとに、非表示 (hide)、削除 (remove)、復元 (restore) を指定することができます。

デフォルトでは、すべてのコンポーネントが表示されます：

```

@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2
@enduml

```



次のような指定が可能です：

- `$tag13` のコンポーネントを非表示にする：

```

@startuml
component C1 $tag13
component C2
component C3 $tag13

```



```
C1 -- C2
```

```
hide $tag13
@enduml
```



- \$tag13 のコンポーネントを削除する:

```
@startuml
component C1 $tag13
component C2
component C3 $tag13
C1 -- C2
```

```
remove $tag13
@enduml
```



- \$tag13 のコンポーネントを削除するが \$tag1 は表示する:

```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2
```

```
remove $tag13
restore $tag1
@enduml
```



- *(すべてのコンポーネント) を削除するが \$tag1 は表示する:

```
@startuml
component C1 $tag13 $tag1
component C2
component C3 $tag13
C1 -- C2
```

```
remove *
restore $tag1
@enduml
```



[Ref. QA-7337 and QA-11052]

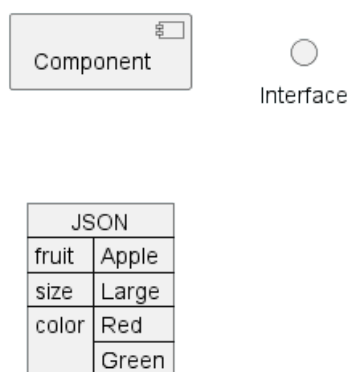
7.17 Display JSON Data on Component diagram

7.17.1 Simple example

```
@startuml
allowmixing

component Component
()      Interface

json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-15481]

For another example, see on JSON page.

7.18 Port [port, portIn, portOut]

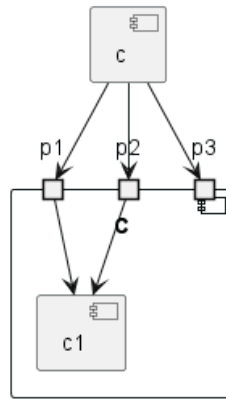
You can add **port** with port, portin and portout keywords.

7.18.1 Port

```
@startuml
[c]
component C {
    port p1
    port p2
    port p3
    component c1
}

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
```





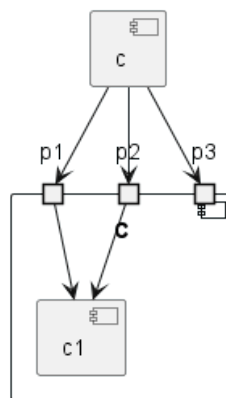
7.18.2 PortIn

```

@startuml
[c]
component C {
    portin p1
    portin p2
    portin p3
    component c1
}
  
```

```

c --> p1
c --> p2
c --> p3
p1 --> c1
p2 --> c1
@enduml
  
```



7.18.3 PortOut

```

@startuml
component C {
    portout p1
    portout p2
    portout p3
    component c1
}
  
```

```

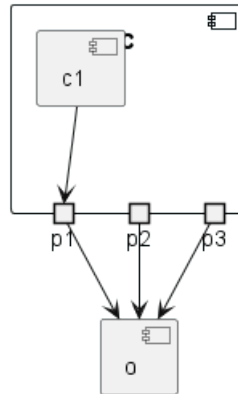
[o]
p1 --> o
p2 --> o
p3 --> o
  
```



```

c1 --> p1
@enduml

```



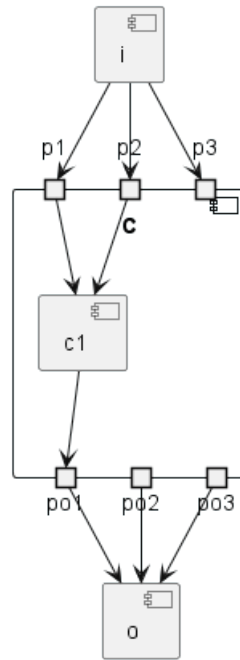
7.18.4 Mixing PortIn & PortOut

```

@startuml
[i]
component C {
    portin p1
    portin p2
    portin p3
    portout po1
    portout po2
    portout po3
    component c1
}
[o]

i --> p1
i --> p2
i --> p3
p1 --> c1
p2 --> c1
po1 --> o
po2 --> o
po3 --> o
c1 --> po1
@enduml

```



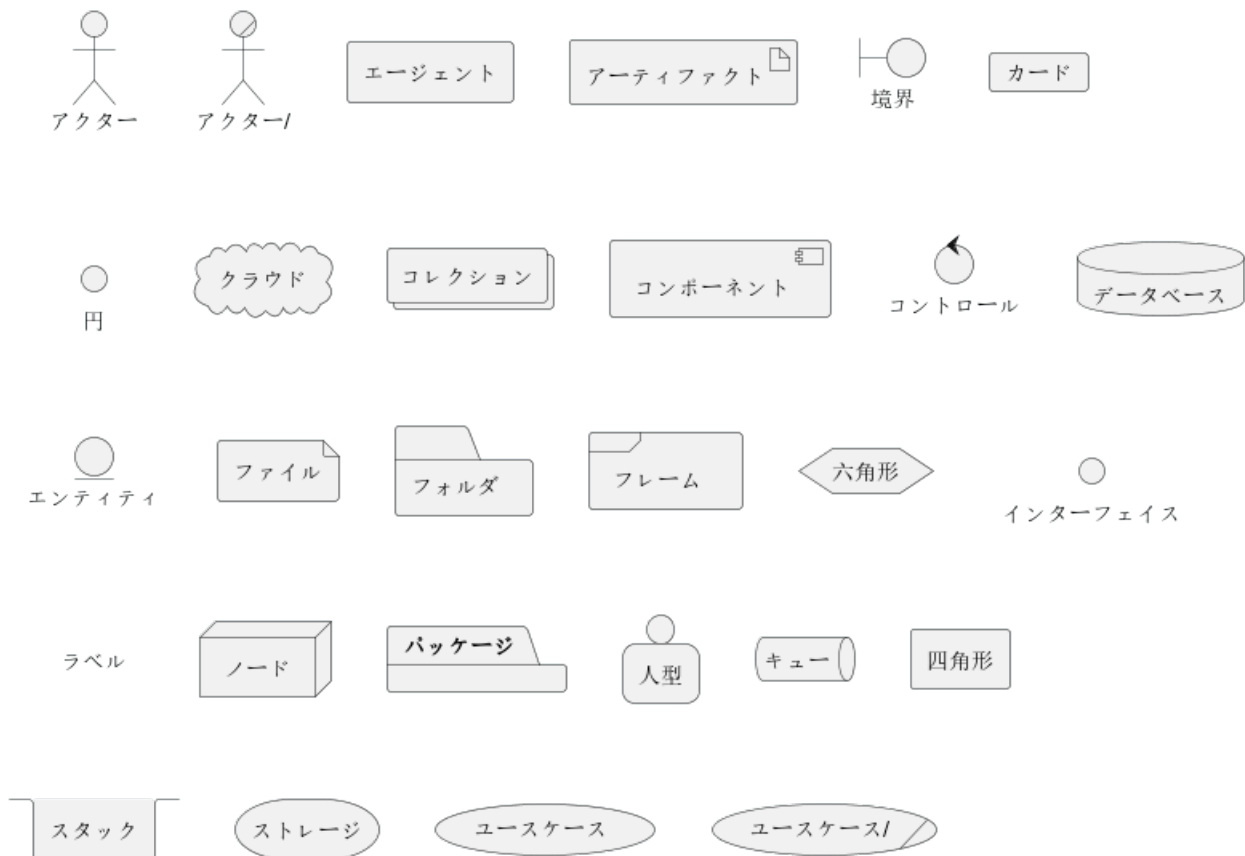
8 配置図

With [PlantUML](<https://plantuml.com/en/deployment-diagram>), creating deployment diagrams becomes a breeze. The platform offers a simple and intuitive way to design these diagrams using plain text, ensuring rapid iterations and easy version control. Moreover, the [PlantUML forum](<https://forum.plantuml.net/>) provides a vibrant community where users can seek help, share ideas, and collaborate on diagramming challenges. One of the key advantages of PlantUML is its ability to integrate seamlessly with various tools and platforms, making it a preferred choice for professionals and enthusiasts alike.

8.1 要素の宣言

```
@startuml
actor アクター
actor/ "アクター/"
agent エージェント
artifact アーティファクト
boundary 境界
card カード
circle 円
cloud クラウド
collections コレクション
component コンポーネント
control コントロール
database データベース
entity エンティティ
file ファイル
folder フォルダ
frame フレーム
hexagon 六角形
interface インターフェイス
label ラベル
node ノード
package パッケージ
person 人型
queue キュー
rectangle 四角形
stack スタック
storage ストレージ
usecase ユースケース
usecase/ "ユースケース/"
@enduml
```





説明文が長くなる場合は、オプションでテキストを [] の中に書くこともできます。

```
@startuml
folder フォルダ [
これは<b>フォルダ</b>です
----
境界線として
====
いろいろな種類の
....
スタイルが使えます
]

node ノード [
これは<b>ノード</b>です
----
境界線として
====
いろいろな種類の
....
スタイルが使えます
]

database データベース [
これは<b>データベース</b>です
----
境界線として
====
いろいろな種類の
....
スタイルが使えます
```



]

usecase ユースケース [
これはユースケースです

境界線として

=====

いろいろな種類の

.....

スタイルが使えます

]

card カード [
これはカードです

境界線として

=====

いろいろな種類の

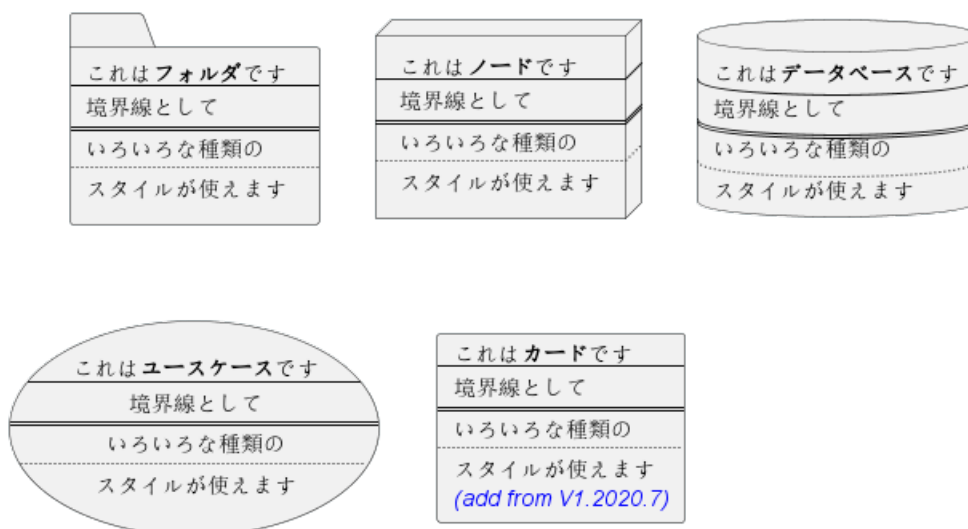
.....

スタイルが使えます

<i><color:blue>(add from V1.2020.7)</color></i>

]

@enduml



8.2 要素の宣言 (省略記法)

いくつかの省略記法を使って要素を宣言することができます。

通常記法のキーワード	省略記法のキーワード	通常記法
actor	: a :	actor
component	[c]	comp
interface	() i	inte
usecase	(u)	usec

8.2.1 アクター

@startuml

actor アクター1
:アクター2:

@enduml





注意: 二重山括弧 (guillemet) を使用したアクターの古い記法がありますが、現在は非推奨であり、削除される予定です。今後は使用しないでください。

8.2.2 コンポーネント

```
@startuml
```

```
component コンポーネント1
[コンポーネント2]
```

```
@enduml
```



8.2.3 インターフェース

```
@startuml
```

```
interface インターフェース1
() "インターフェース2"
```

```
label "//interface example//"
@enduml
```



interface example

8.2.4 ユースケース

```
@startuml
```

```
usecase ユースケース1
(ユースケース2)
```

```
@enduml
```



8.3 リンク、矢印

要素の間をシンプルなリンクで結ぶことができます。リンクにラベルを付けることもできます。

```
@startuml
```

```
node ノード1
node ノード2
node ノード3
node ノード4
```

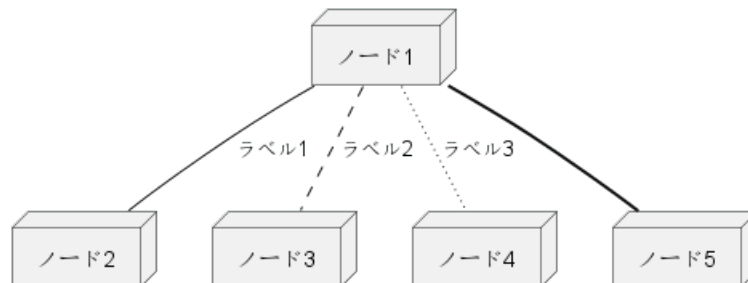


```

node ノード5
ノード1 -- ノード2 : ラベル1
ノード1 .. ノード3 : ラベル2
ノード1 ~~ ノード4 : ラベル3
ノード1 == ノード5

```

```
@enduml
```



複数の種類のリンクを使うこともできます。

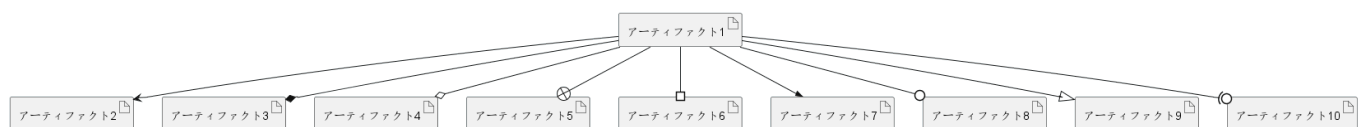
```
@startuml
```

```

artifact アーティファクト1
artifact アーティファクト2
artifact アーティファクト3
artifact アーティファクト4
artifact アーティファクト5
artifact アーティファクト6
artifact アーティファクト7
artifact アーティファクト8
artifact アーティファクト9
artifact アーティファクト10
アーティファクト1 --> アーティファクト2
アーティファクト1 --* アーティファクト3
アーティファクト1 --o アーティファクト4
アーティファクト1 --+ アーティファクト5
アーティファクト1 --# アーティファクト6
アーティファクト1 -->> アーティファクト7
アーティファクト1 --0 アーティファクト8
アーティファクト1 --^ アーティファクト9
アーティファクト1 --(0 アーティファクト10

```

```
@enduml
```



次のような種類のリンクも使用できます。

```
@startuml
```

```

cloud クラウド1
cloud クラウド2
cloud クラウド3
cloud クラウド4
cloud クラウド5
クラウド1 -0- クラウド2

```

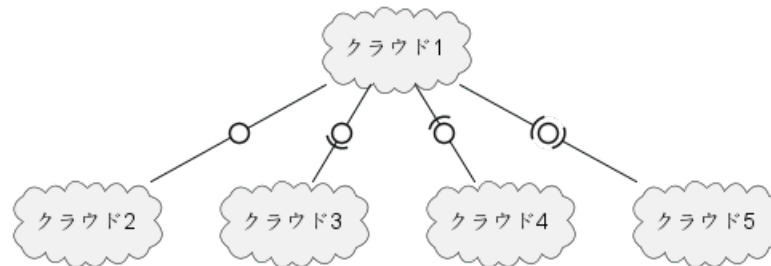


```

クラウド1 -0)- クラウド3
クラウド1 -(0- クラウド4
クラウド1 -(0)- クラウド5

```

```
@enduml
```



別の例:

```

@startuml
actor foo1
actor foo2
foo1 <-0-> foo2
foo1 <-(0)-> foo2

(ac1) -le(0)-> left1
ac1 -ri(0)-> right1
ac1 .up(0).> up1
ac1 ~up(0)~> up2
ac1 -do(0)-> down1
ac1 -do(0)-> down2

actor1 -0)- actor2

component comp1
component comp2
comp1 *-0)-+ comp2
[comp3] <--> [comp4]

boundary b1
control c1
b1 -(0)- c1

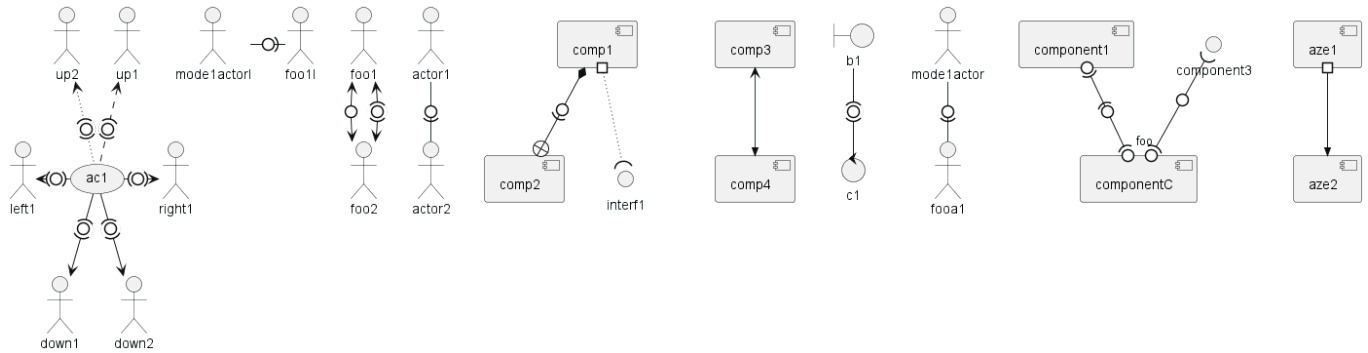
component comp1
interface interf1
comp1 #~~( interf1

:modelactor: -0)- fooa1
:modelactorl: -ri0)- foo1l

[component1] 0)-(0-(0 [componentC]
() component3 )-0-(0 "foo" [componentC]

[aze1] #--> [aze2]
@enduml

```



[Ref. QA-547 and QA-1736]

すべての種類は付録を参照。

8.4 各括弧を使用した矢印のスタイル

(角括弧を使用したクラスの関係 (リンク、矢印) のスタイルと同様)

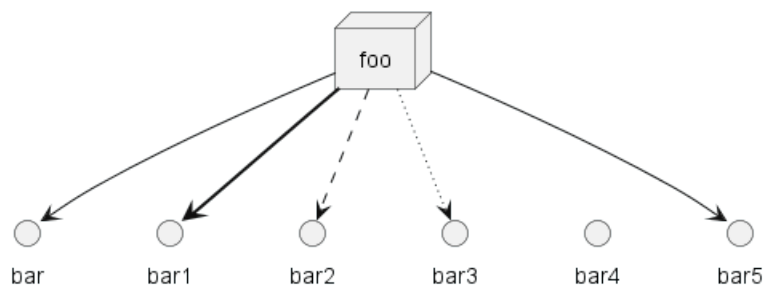
8.4.1 線のスタイル

矢印に **bold**、**dashed**、**dotted**、**hidden**、**plain** のスタイルを指定することができます：

- ラベル無し

```
@startuml
node foo
title Bracketed line style without label
foo --> bar
foo -[bold]-> bar1
foo -[dashed]-> bar2
foo -[dotted]-> bar3
foo -[hidden]-> bar4
foo -[plain]-> bar5
@enduml
```

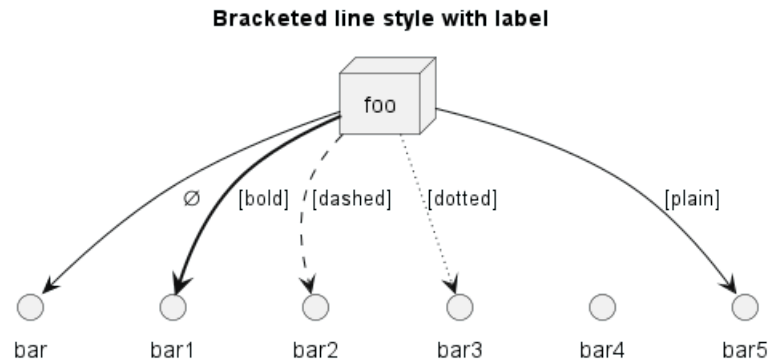
Bracketed line style without label



- ラベル有り

```
@startuml
title Bracketed line style with label
node foo
foo --> bar : 
foo -[bold]-> bar1 : [bold]
foo -[dashed]-> bar2 : [dashed]
foo -[dotted]-> bar3 : [dotted]
foo -[hidden]-> bar4 : [hidden]
foo -[plain]-> bar5 : [plain]
@enduml
```



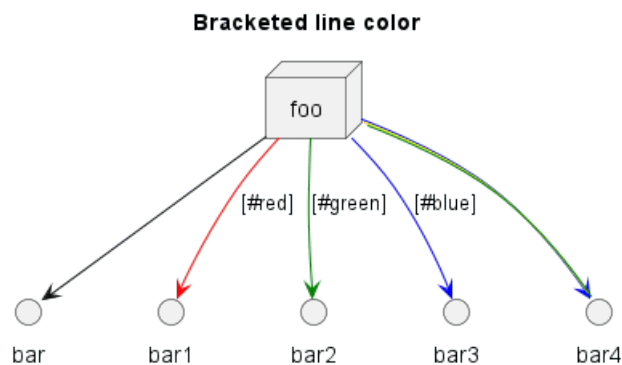


[Adapted from QA-4181]

8.4.2 線の色

```

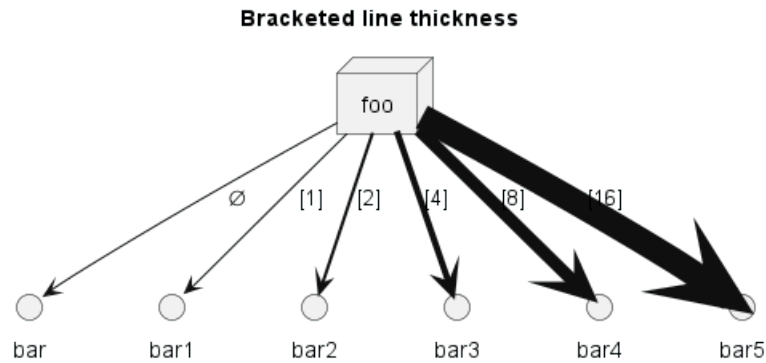
@startuml
title Bracketed line color
node foo
foo --> bar
foo -[#red]-> bar1 : [#red]
foo -[#green]-> bar2 : [#green]
foo -[#blue]-> bar3 : [#blue]
foo -[#blue;#yellow;#green]-> bar4
@enduml
  
```



8.4.3 線の太さ

```

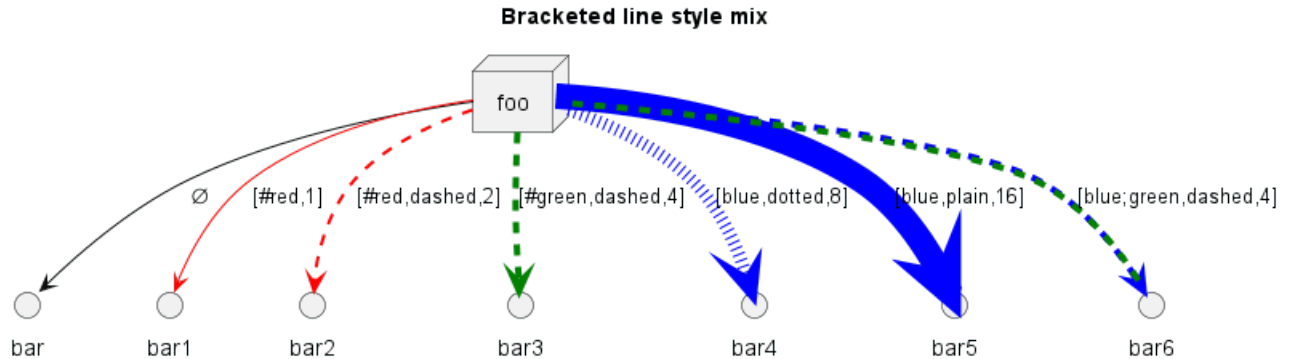
@startuml
title Bracketed line thickness
node foo
foo --> bar : 
foo -[thickness=1]-> bar1 : [1]
foo -[thickness=2]-> bar2 : [2]
foo -[thickness=4]-> bar3 : [4]
foo -[thickness=8]-> bar4 : [8]
foo -[thickness=16]-> bar5 : [16]
@enduml
  
```



[Adapted from QA-4949]

8.4.4 混合

```
@startuml
title Bracketed line style mix
node foo
foo --> bar : 
foo -[#red,thickness=1]-> bar1 : [#red,1]
foo -[#red,dashed,thickness=2]-> bar2 : [#red,dashed,2]
foo -[#green,dashed,thickness=4]-> bar3 : [#green,dashed,4]
foo -[#blue,dotted,thickness=8]-> bar4 : [blue,dotted,8]
foo -[#blue,plain,thickness=16]-> bar5 : [blue,plain,16]
foo -[#blue;#green,dashed,thickness=4]-> bar6 : [blue;green,dashed,4]
@enduml
```



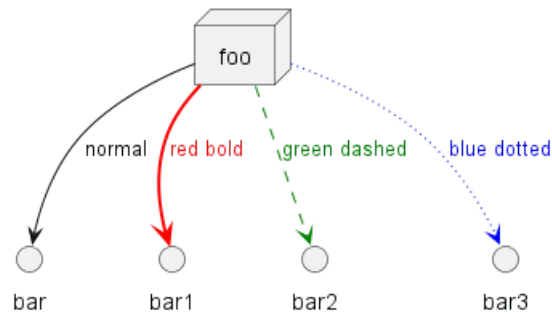
8.5 矢印の色とスタイルを変更する（インラインスタイル）

個別の矢印ごとに色とスタイルを変更するには、次の記法を使用します：

- #color;line.[bold|dashed|dotted];text:color

```
@startuml
node foo
foo --> bar : normal
foo --> bar1 #line:red;line.bold;text:red : red bold
foo --> bar2 #green;line.dashed;text:green : green dashed
foo --> bar3 #blue;line.dotted;text:blue : blue dotted
@enduml
```



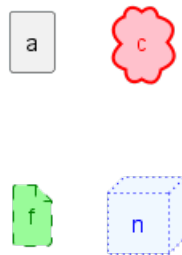


[Ref. QA-3770 and QA-3816] [See similar feature on class diagram]

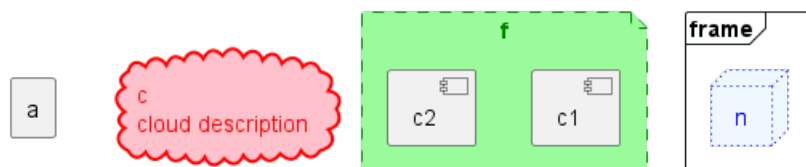
8.6 要素の色とスタイルを変更する（インラインスタイル）

それぞれ個別の要素について、色とスタイルを変更するには、次の記法を使用します：`#color;line:color;line.[bold]`

```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red
file f #palegreen;line:green;line.dashed;text:green
node n #aliceblue;line:blue;line.dotted;text:blue
@enduml
```



```
@startuml
agent a
cloud c #pink;line:red;line.bold;text:red [
c
cloud description
]
file f #palegreen;line:green;line.dashed;text:green {
[c1]
[c2]
}
frame frame {
node n #aliceblue;line:blue;line.dotted;text:blue
}
@enduml
```



[Ref. QA-6852]

8.7 入れ子にできる要素

次の要素は入れ子にできます：



```

@startuml
artifact アーティファクト {
}
card カード {
}
cloud クラウド {
}
component コンポーネント {
}
database データベース {
}
file ファイル {
}
folder フォルダ {
}
frame フレーム {
}
hexagon 六角形 {
}
node ノード {
}
package パッケージ {
}
queue キュー {
}
rectangle 四角形 {
}
stack スタック {
}
storage ストレージ {
}
@enduml

```



8.8 パッケージと入れ子要素

8.8.1 一階層の例

```

@startuml
artifact      artifactVeryL00000000000000000000g      as "アーティファクト" {
file f1
}
card          cardVeryL00000000000000000000g        as "カード" {
file f2
}
cloud         cloudVeryL00000000000000000000g       as "クラウド" {
file f3
}
component     componentVeryL00000000000000000000g   as "コンポーネント" {
file f4
}
database      databaseVeryL00000000000000000000g    as "データベース" {
file f5
}
file          fileVeryL00000000000000000000g        as "ファイル" {
file f6
}

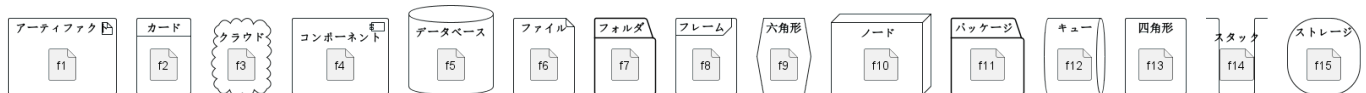
```



```

folder      folderVeryL00000000000000000000g      as "フォルダ" {
file f7
}
frame       frameVeryL00000000000000000000g      as "フレーム" {
file f8
}
hexagon     hexagonVeryL00000000000000000000g    as "六角形" {
file f9
}
node        nodeVeryL00000000000000000000g      as "ノード" {
file f10
}
package     packageVeryL00000000000000000000g    as "パッケージ" {
file f11
}
queue       queueVeryL00000000000000000000g     as "キュー" {
file f12
}
rectangle   rectangleVeryL0000000000000000000g  as "四角形" {
file f13
}
stack       stackVeryL00000000000000000000g     as "スタック" {
file f14
}
storage     storageVeryL00000000000000000000g    as "ストレージ" {
file f15
}
@enduml

```



8.8.2 他の例

```

@startuml
artifact Foo1 {
    folder Foo2
}

folder Foo3 {
    artifact Foo4
}

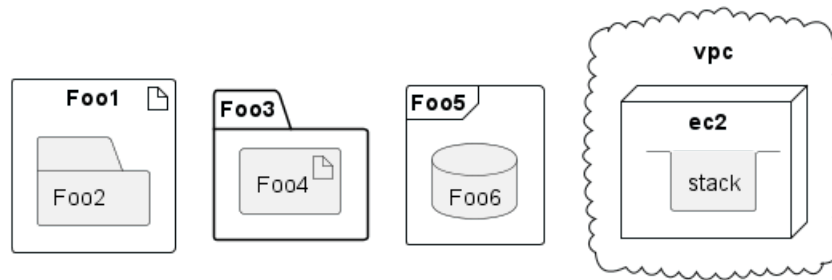
frame Foo5 {
    database Foo6
}

cloud vpc {
    node ec2 {
        stack stack
    }
}

@enduml

```





```

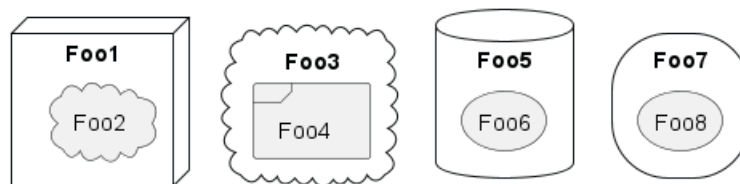
@startuml
node Foo1 {
  cloud Foo2
}

cloud Foo3 {
  frame Foo4
}

database Foo5 {
  storage Foo6
}

storage Foo7 {
  storage Foo8
}
@enduml

```



8.8.3 すべて入れ子にした例

すべての入れ子要素の例です:

- アルファベット順:

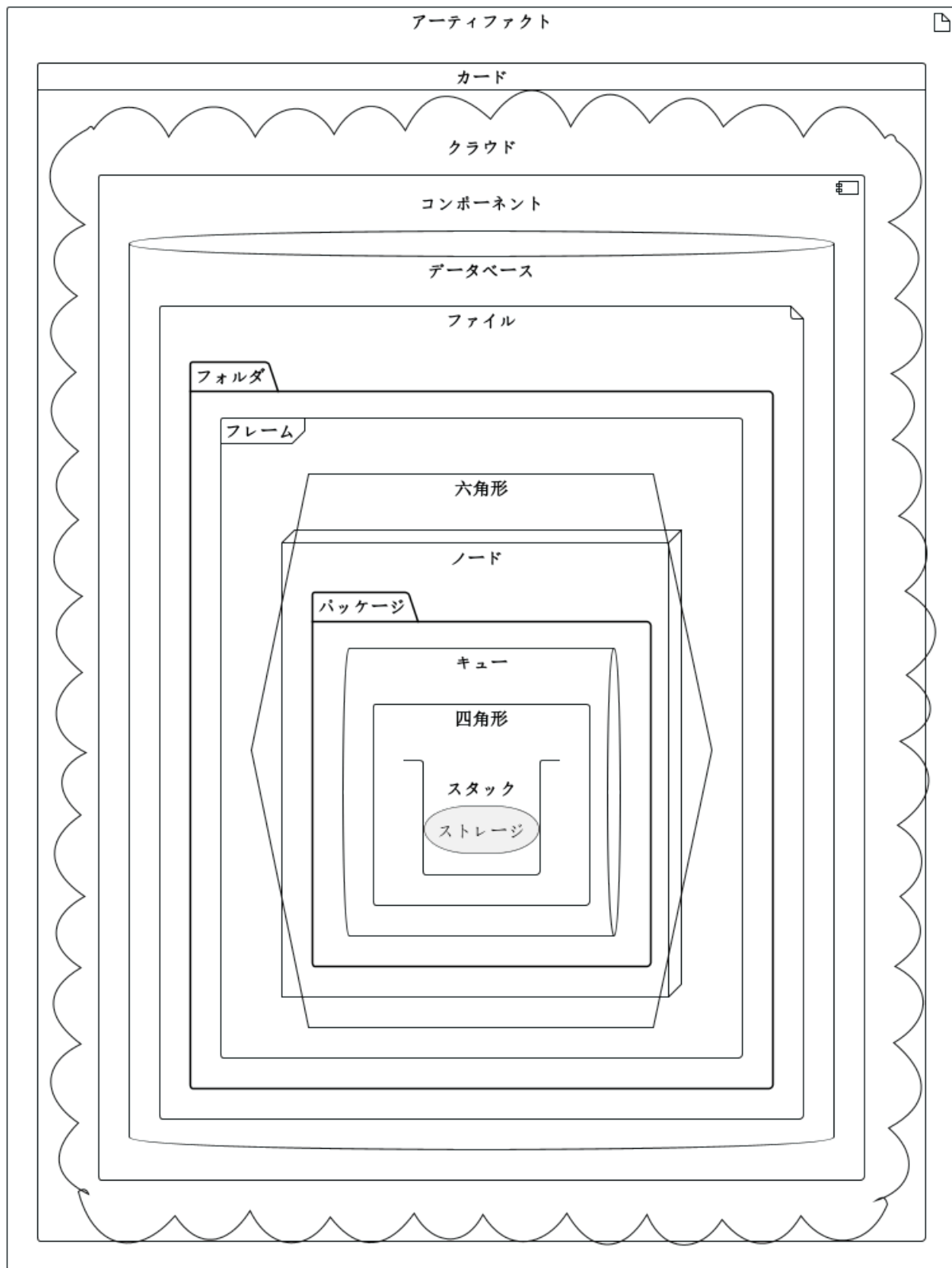
```

@startuml
artifact アーティファクト {
  card カード {
    cloud クラウド {
      component コンポーネント {
        database データベース {
          file ファイル {
            folder フォルダ {
              frame フレーム {
                hexagon 六角形 {
                  node ノード {
                    package パッケージ {
                      queue キュー {
                        rectangle 四角形 {
                          stack スタック {
                            storage ストレージ {
                              }
                            }
                          }
                        }
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}

```

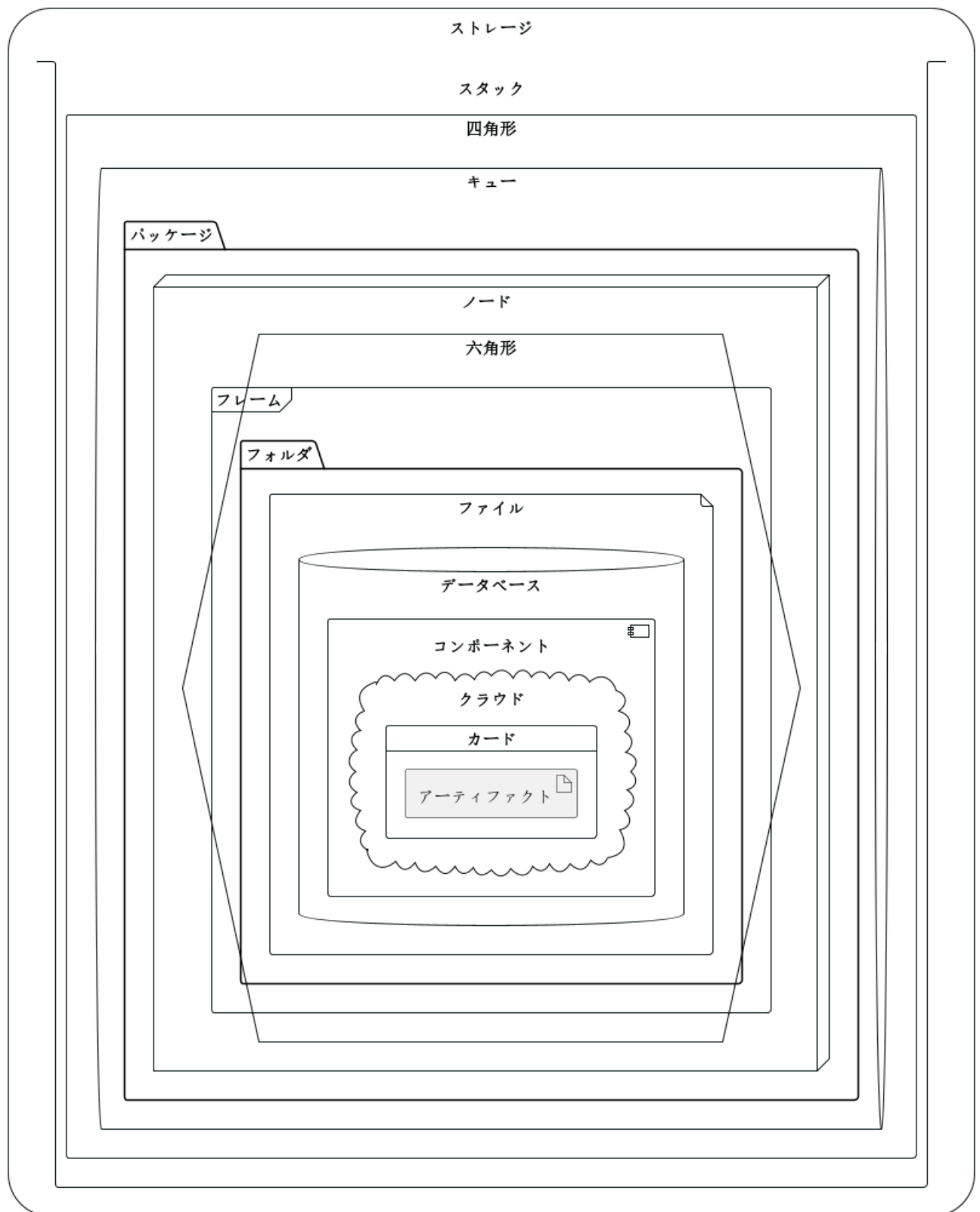


```
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
@enduml
```



- アルファベットの逆順

```
@startuml
storage ストレージ {
stack スタック {
rectangle 四角形 {
queue キュー {
package パッケージ {
```

8.9 別名

8.9.1 as による単純な別名

```
@startuml
node ノード1 as n1
node "ノード 2" as n2
```

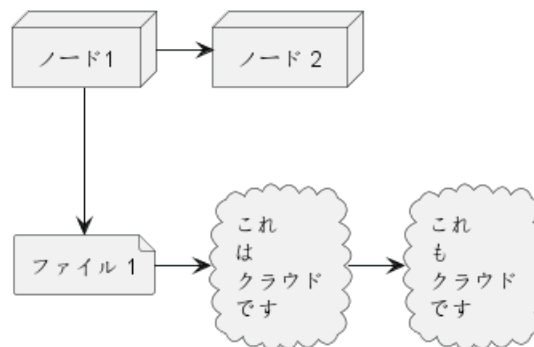


```

file f1 as "ファイル 1"
cloud c1 as "これは
クラウド
です"
cloud c2 [これは
も
クラウド
です]

n1 -> n2
n1 --> f1
f1 -> c1
c1 -> c2
@enduml

```

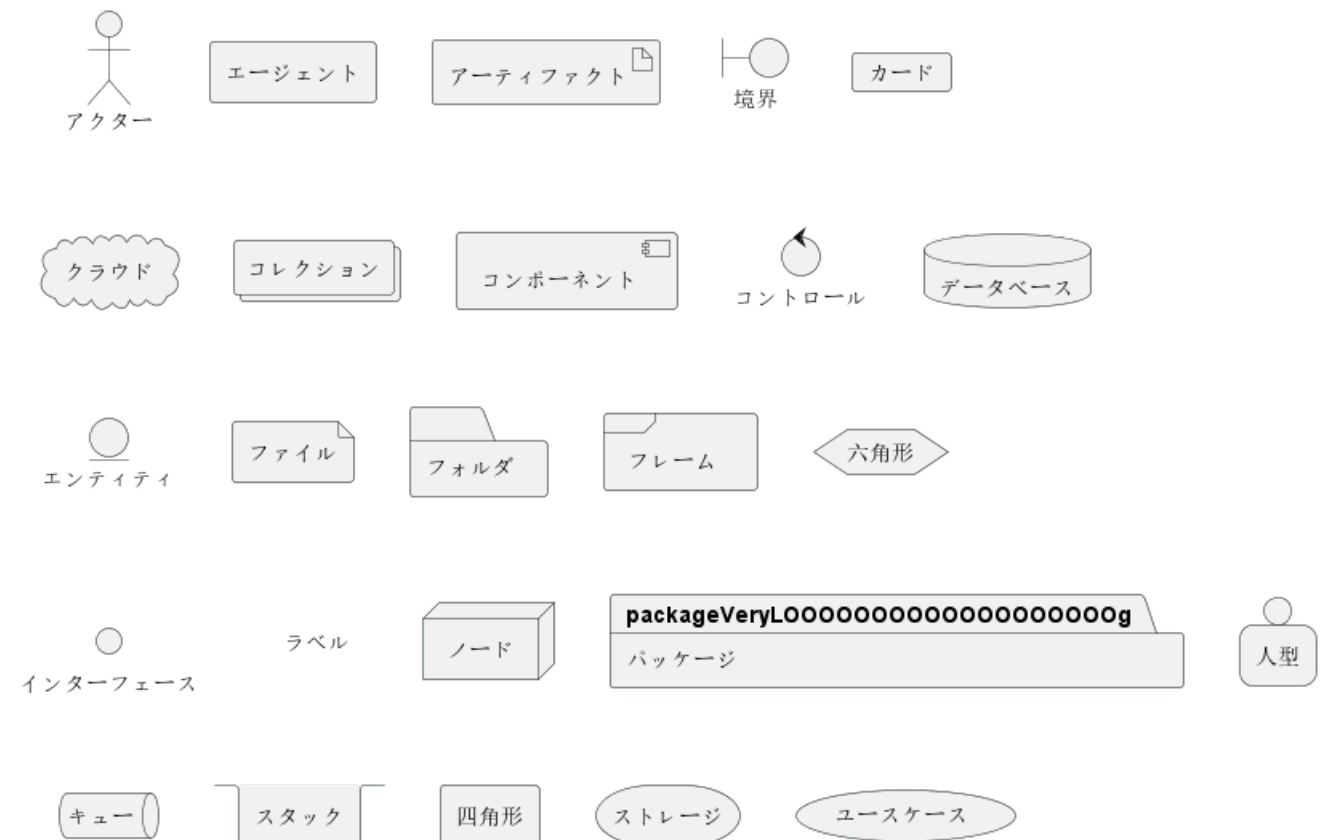


8.9.2 長い別名の例

```

@startuml
actor      "アクター"      as actorVeryL00000000000000000000g
agent      "エージェント"   as agentVeryL00000000000000000000g
artifact   "アーティファクト" as artifactVeryL00000000000000000000g
boundary   "境界"          as boundaryVeryL00000000000000000000g
card       "カード"         as cardVeryL00000000000000000000g
cloud      "クラウド"       as cloudVeryL00000000000000000000g
collections "コレクション"  as collectionsVeryL00000000000000000000g
component  "コンポーネント" as componentVeryL00000000000000000000g
control     "コントロール"  as controlVeryL00000000000000000000g
database   "データベース"   as databaseVeryL00000000000000000000g
entity     "エンティティ"   as entityVeryL00000000000000000000g
file       "ファイル"       as fileVeryL00000000000000000000g
folder     "フォルダ"       as folderVeryL00000000000000000000g
frame      "フレーム"       as frameVeryL00000000000000000000g
hexagon    "六角形"        as hexagonVeryL00000000000000000000g
interface  "インターフェース" as interfaceVeryL00000000000000000000g
label      "ラベル"        as labelVeryL00000000000000000000g
node       "ノード"        as nodeVeryL00000000000000000000g
package    "パッケージ"     as packageVeryL00000000000000000000g
person     "人型"          as personVeryL00000000000000000000g
queue      "キュー"        as queueVeryL00000000000000000000g
stack      "スタック"      as stackVeryL00000000000000000000g
rectangle  "四角形"        as rectangleVeryL00000000000000000000g
storage    "ストレージ"     as storageVeryL00000000000000000000g
usecase    "ユースケース"  as usecaseVeryL00000000000000000000g
@enduml

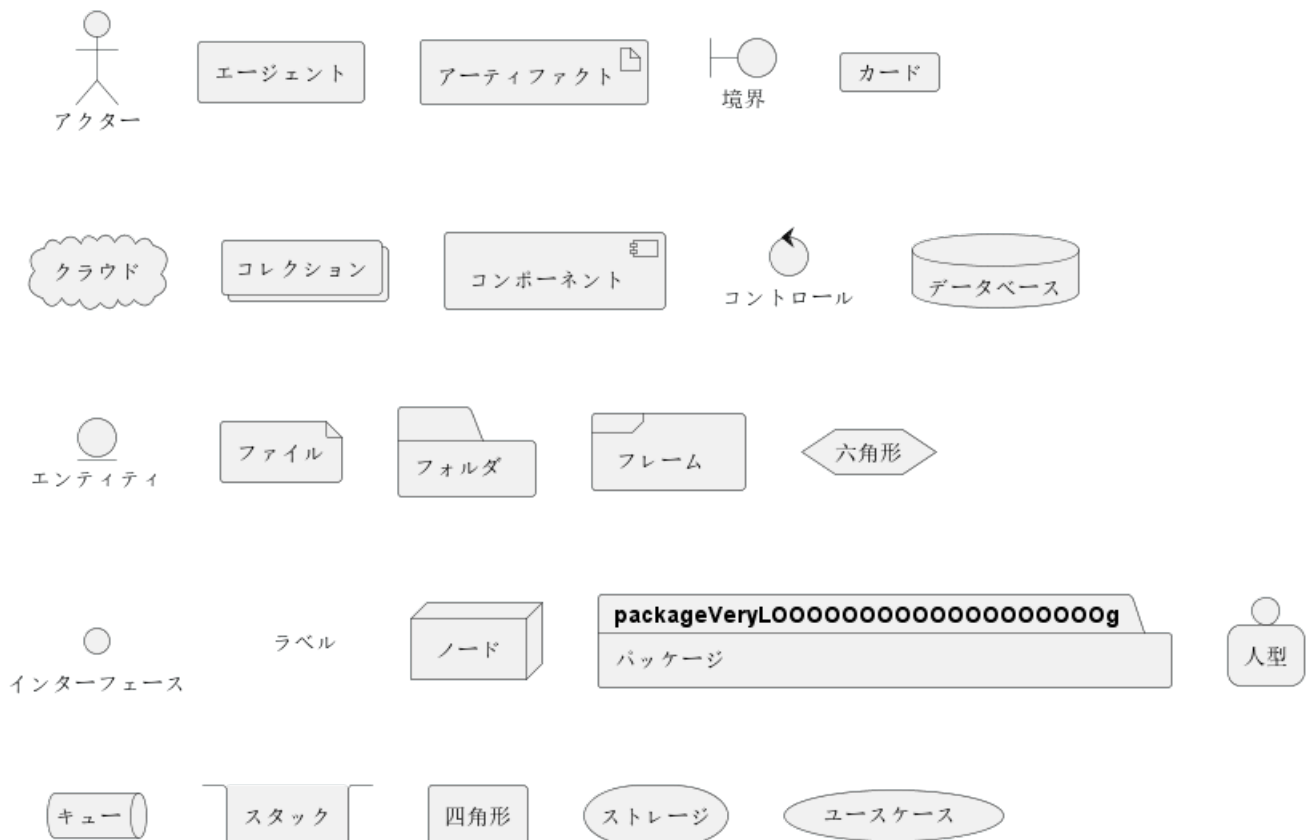
```



```

@startuml
actor      actorVeryL0000000000000000000g      as "アクター"
agent      agentVeryL0000000000000000000g      as "エージェント"
artifact   artifactVeryL0000000000000000000g   as "アーティファクト"
boundary   boundaryVeryL0000000000000000000g   as "境界"
card       cardVeryL0000000000000000000g       as "カード"
cloud      cloudVeryL0000000000000000000g      as "クラウド"
collections collectionsVeryL0000000000000000000g as "コレクション"
component  componentVeryL0000000000000000000g as "コンポーネント"
control    controlVeryL0000000000000000000g    as "コントロール"
database   databaseVeryL0000000000000000000g   as "データベース"
entity     entityVeryL0000000000000000000g     as "エンティティ"
file       fileVeryL0000000000000000000g       as "ファイル"
folder     folderVeryL0000000000000000000g     as "フォルダ"
frame      frameVeryL0000000000000000000g      as "フレーム"
hexagon    hexagonVeryL0000000000000000000g    as "六角形"
interface  interfaceVeryL0000000000000000000g  as "インターフェース"
label      labelVeryL0000000000000000000g      as "ラベル"
node       nodeVeryL0000000000000000000g       as "ノード"
package    packageVeryL0000000000000000000g    as "パッケージ"
person     personVeryL0000000000000000000g     as "人型"
queue      queueVeryL0000000000000000000g      as "キュー"
stack      stackVeryL0000000000000000000g      as "スタック"
rectangle  rectangleVeryL0000000000000000000g  as "四角形"
storage    storageVeryL0000000000000000000g    as "ストレージ"
usecase    usecaseVeryL0000000000000000000g    as "ユースケース"
@enduml

```

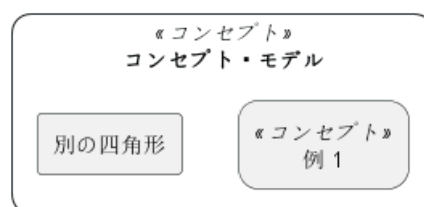


[Ref. QA-12082]

8.10 角に丸みをつける

```
@startuml
skinparam rectangle {
    roundCorner<<コンセプト>> 25
}

rectangle "コンセプト・モデル" <<コンセプト>> {
    rectangle "例 1" <<コンセプト>> as ex1
    rectangle "別の四角形"
}
@enduml
```



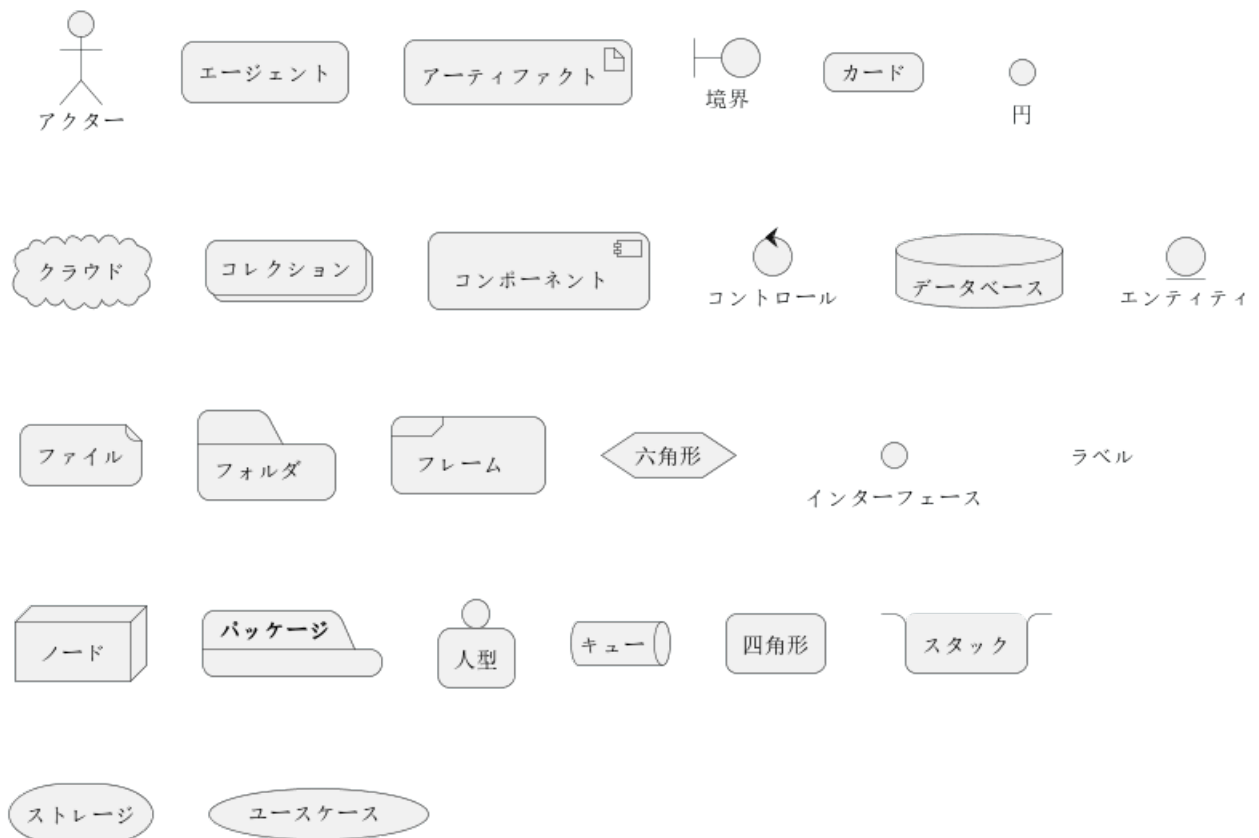
8.11 特有の skinparam

8.11.1 roundCorner

```
@startuml
skinparam roundCorner 15
actor アクター
agent エージェント
```



artifact アーティファクト
 boundary 境界
 card カード
 circle 円
 cloud クラウド
 collections コレクション
 component コンポーネント
 control コントロール
 database データベース
 entity エンティティ
 file ファイル
 folder フォルダ
 frame フレーム
 hexagon 六角形
 interface インターフェース
 label ラベル
 node ノード
 package パッケージ
 person 人型
 queue キュー
 rectangle 四角形
 stack スタック
 storage ストレージ
 usecase ユースケース
 @enduml



[Ref. QA-5299, QA-6915, QA-11943]

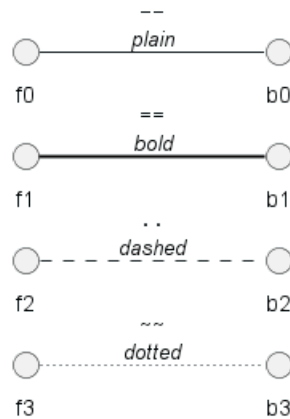
8.12 付録：線の種類の一覧

@startuml



```
left to right direction
skinparam nodesep 5
```

```
f3 ~~ b3 : ""~~""\n//dotted//
f2 .. b2 : ""..""\n//dashed//
f1 == b1 : ""==""\n//bold//
f0 -- b0 : ""--""\n//plain//
@enduml
```

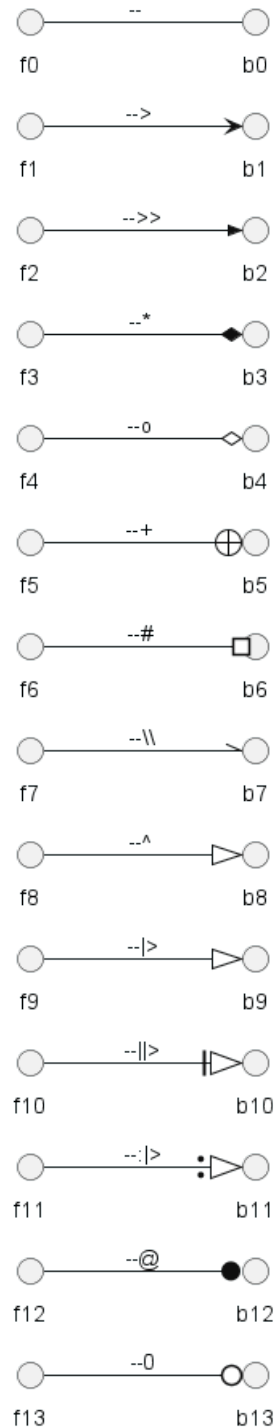


8.13 付録：矢印の先端と'0' 矢印の一覧

8.13.1 矢印の先端

```
@startuml
left to right direction
skinparam nodesep 5

f13 --0 b13 : ""--0""
f12 --@ b12 : ""--@""
f11 --:|> b11 : ""--:|>""
f10 --||> b10 : ""--||>""
f9 --|> b9 : ""--|>""
f8 --^ b8 : ""--^ ""
f7 --\\ b7 : ""--\\\\ ""
f6 --# b6 : ""--# ""
f5 --+ b5 : ""--+ ""
f4 --o b4 : ""--o ""
f3 --* b3 : ""--* ""
f2 -->> b2 : ""-->>""
f1 --> b1 : ""--> ""
f0 -- b0 : ""-- ""
@enduml
```



8.13.2 丸形の矢印 ('0' 矢印)

```
@startuml
left to right direction
skinparam nodesep 5

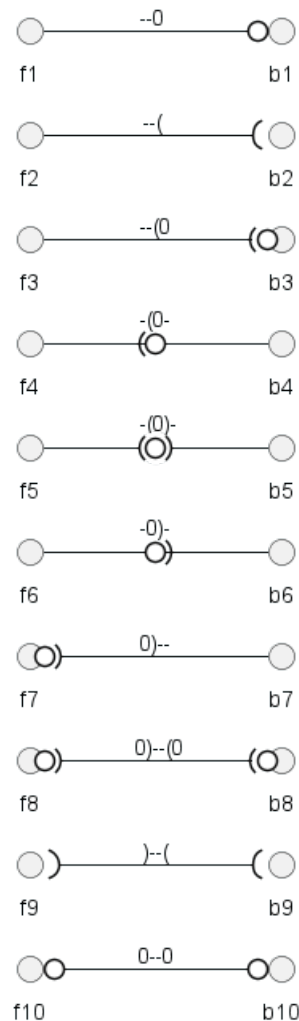
f10 0--0 b10 : "" 0--0 ""
f9 )--( b9 : "" )--( ""
f8 0)--(0 b8 : "" 0)--(0""
f7 0)-- b7 : "" 0)-- ""
f6 -0)- b6 : "" -0)- ""
f5 -(0)- b5 : "" -(0)-""
```



```

f4 -(0- b4 : "" -(0- ""
f3 --(0 b3 : "" --(0 ""
f2 --( b2 : "" --( ""
f1 --0 b1 : "" --0 ""
@enduml

```



8.14 付録：すべての要素に対するインラインスタイルのテスト

8.14.1 シンプルな要素

```

@startuml
actor アクター                                #aliceblue;line:blue;line.dotted;text:blue
actor/ "アクター/"                          #aliceblue;line:blue;line.dotted;text:blue
agent エージェント                          #aliceblue;line:blue;line.dotted;text:blue
artifact アーティファクト                  #aliceblue;line:blue;line.dotted;text:blue
boundary 境界                              #aliceblue;line:blue;line.dotted;text:blue
card カード                                #aliceblue;line:blue;line.dotted;text:blue
circle 円                                  #aliceblue;line:blue;line.dotted;text:blue
cloud クラウド                             #aliceblue;line:blue;line.dotted;text:blue
collections コレクション                  #aliceblue;line:blue;line.dotted;text:blue
component コンポーネント                  #aliceblue;line:blue;line.dotted;text:blue
control コントロール                      #aliceblue;line:blue;line.dotted;text:blue
database データベース                     #aliceblue;line:blue;line.dotted;text:blue
entity エンティティ                       #aliceblue;line:blue;line.dotted;text:blue
file ファイル                             #aliceblue;line:blue;line.dotted;text:blue
folder フォルダ                           #aliceblue;line:blue;line.dotted;text:blue

```



```

frame フレーム #aliceblue;line:blue;line.dotted;text:blue
hexagon 六角形 #aliceblue;line:blue;line.dotted;text:blue
interface インターフェース #aliceblue;line:blue;line.dotted;text:blue
label ラベル #aliceblue;line:blue;line.dotted;text:blue
node ノード #aliceblue;line:blue;line.dotted;text:blue
package パッケージ #aliceblue;line:blue;line.dotted;text:blue
person 人型 #aliceblue;line:blue;line.dotted;text:blue
queue キュー #aliceblue;line:blue;line.dotted;text:blue
rectangle 四角形 #aliceblue;line:blue;line.dotted;text:blue
stack スタック #aliceblue;line:blue;line.dotted;text:blue
storage ストレージ #aliceblue;line:blue;line.dotted;text:blue
usecase ユースケース #aliceblue;line:blue;line.dotted;text:blue
usecase/ "ユースケース/" #aliceblue;line:blue;line.dotted;text:blue
@enduml

```



8.14.2 入れ子の要素

8.14.3 サブ要素無し

```

@startuml
artifact アーティファクト #aliceblue;line:blue;line.dotted;text:blue {
}
card カード #aliceblue;line:blue;line.dotted;text:blue {
}
cloud クラウド #aliceblue;line:blue;line.dotted;text:blue {
}
component コンポーネント #aliceblue;line:blue;line.dotted;text:blue {
}
database データベース #aliceblue;line:blue;line.dotted;text:blue {
}

```



```

file ファイル #aliceblue;line:blue;line.dotted;text:blue {
}
folder フォルダ #aliceblue;line:blue;line.dotted;text:blue {
}
frame フレーム #aliceblue;line:blue;line.dotted;text:blue {
}
hexagon 六角形 #aliceblue;line:blue;line.dotted;text:blue {
}
node ノード #aliceblue;line:blue;line.dotted;text:blue {
}
package パッケージ #aliceblue;line:blue;line.dotted;text:blue {
}
queue キュー #aliceblue;line:blue;line.dotted;text:blue {
}
rectangle 四角形 #aliceblue;line:blue;line.dotted;text:blue {
}
stack スタック #aliceblue;line:blue;line.dotted;text:blue {
}
storage ストレージ #aliceblue;line:blue;line.dotted;text:blue {
}
@enduml

```



8.14.4 サブ要素有り

```

@startuml
artifact      artifactVeryL00000000000000000000g      as "アーティファクト" #aliceblue;line:blue;line.dotted;text:blue {
file f1
}
card          cardVeryL00000000000000000000g        as "カード" #aliceblue;line:blue;line.dotted;text:blue {
file f2
}
cloud         cloudVeryL00000000000000000000g       as "クラウド" #aliceblue;line:blue;line.dotted;text:blue {
file f3
}
component     componentVeryL00000000000000000000g    as "コンポーネント" #aliceblue;line:blue;line.dotted;text:blue {
file f4
}
database      databaseVeryL00000000000000000000g    as "データベース" #aliceblue;line:blue;line.dotted;text:blue {
file f5
}
file          fileVeryL00000000000000000000g        as "ファイル" #aliceblue;line:blue;line.dotted;text:blue {
file f6
}
folder        folderVeryL00000000000000000000g      as "フォルダ" #aliceblue;line:blue;line.dotted;text:blue {
file f7
}
frame         frameVeryL00000000000000000000g       as "フレーム" #aliceblue;line:blue;line.dotted;text:blue {
file f8
}
hexagon       hexagonVeryL00000000000000000000g     as "六角形" #aliceblue;line:blue;line.dotted;text:blue {
file f9
}
node          nodeVeryL00000000000000000000g        as "ノード" #aliceblue;line:blue;line.dotted;text:blue {
file f10
}

```



```

package      packageVeryL00000000000000000000g      as "パッケージ" #aliceblue;line:blue;line.dotted;text:blue
file f11
}
queue        queueVeryL00000000000000000000g        as "キュー" #aliceblue;line:blue;line.dotted;text:blue
file f12
}
rectangle    rectangleVeryL00000000000000000000g      as "四角形" #aliceblue;line:blue;line.dotted;text:blue
file f13
}
stack        stackVeryL00000000000000000000g          as "スタック" #aliceblue;line:blue;line.dotted;text:blue
file f14
}
storage      storageVeryL00000000000000000000g        as "ストレージ" #aliceblue;line:blue;line.dotted;text:blue
file f15
}
@enduml

```



8.15 付録：すべての要素に対するスタイルのテスト

8.15.1 シンプルな要素

8.15.2 グローバルスタイル (componentDiagram)

```

@startuml
<style>
componentDiagram {
    BackgroundColor palegreen
    LineThickness 1
    LineColor red
}
document {
    BackgroundColor white
}
</style>
actor アクター
actor/ "アクター/"
agent エージェント
artifact アーティファクト
boundary 境界
card カード
circle 円
cloud クラウド
collections コレクション
component コンポーネント
control コントロール
database データベース
entity エンティティ
file ファイル
folder フォルダ
frame フレーム
hexagon 六角形
interface インターフェース
label ラベル
node ノード

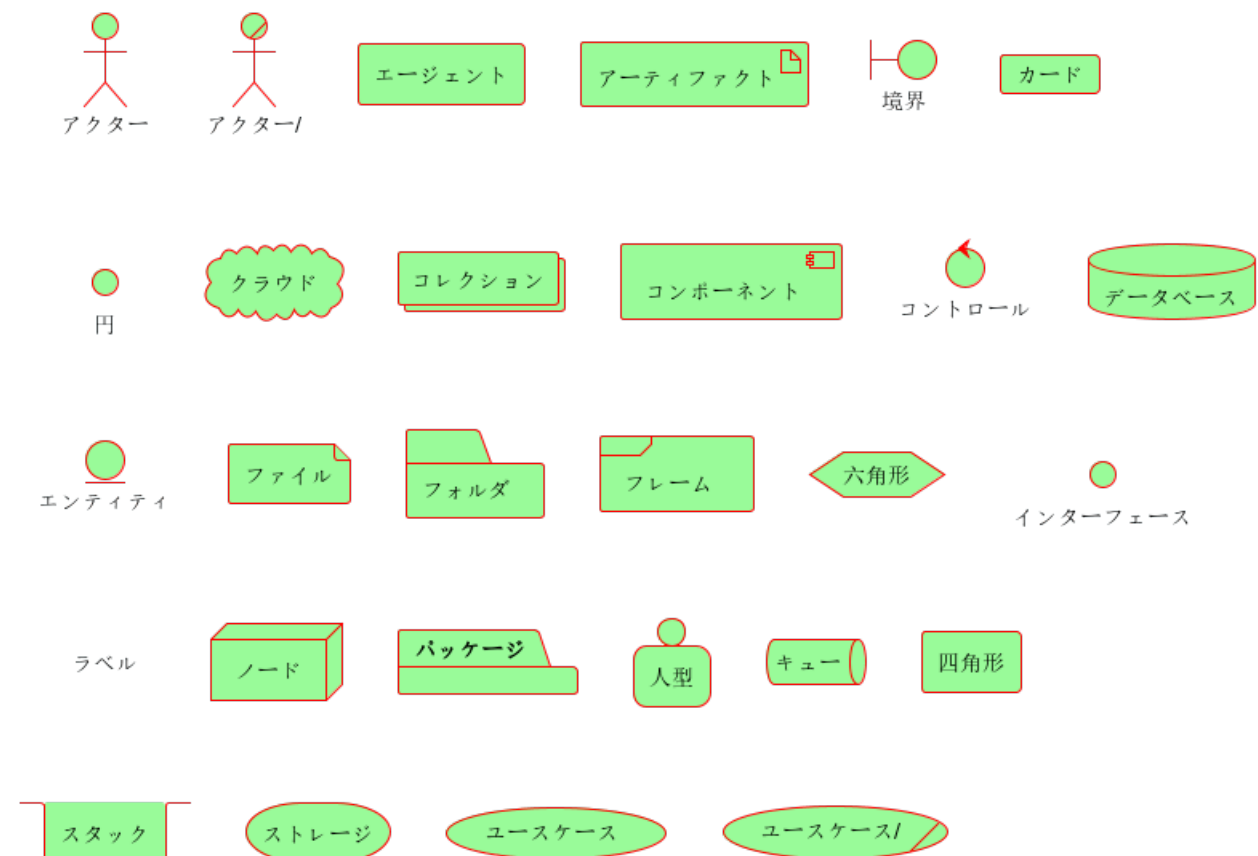
```



```

package パッケージ
person 人型
queue キュー
rectangle 四角形
stack スタック
storage ストレージ
usecase ユースケース
usecase/ "ユースケース/"
@enduml

```



8.15.3 エlement毎のスタイル

```

@startuml
<style>
actor {
  BackGroundColor #f80c12
  LineThickness 1
  LineColor black
}
agent {
  BackGroundColor #f80c12
  LineThickness 1
  LineColor black
}
artifact {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
boundary {

```



```
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
circle {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
collections {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
control {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
entity {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
```



```

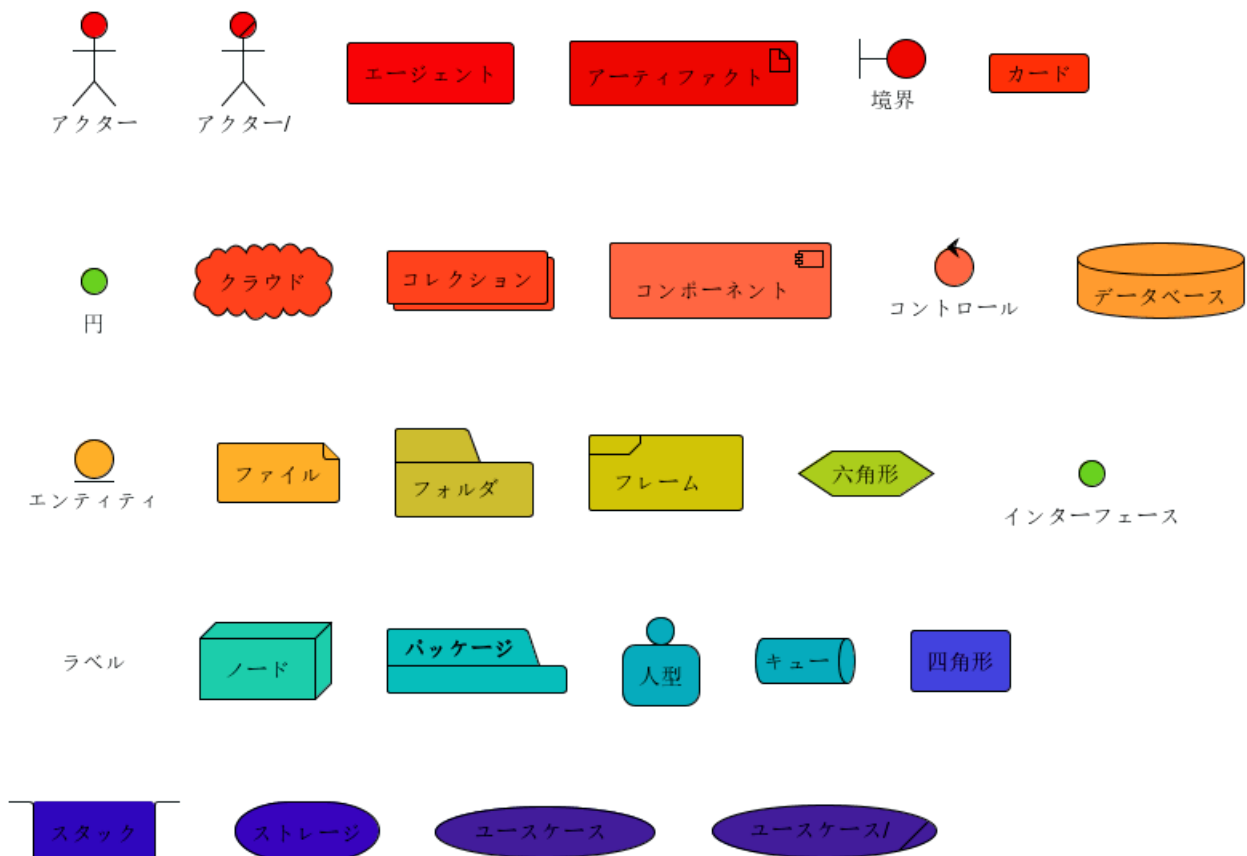
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
interface {
    BackGroundColor #69d025
    LineThickness 1
    LineColor black
}
label {
    BackGroundColor black
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
person {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}
usecase {
    BackGroundColor #442299
    LineThickness 1
    LineColor black
}
}
</style>
actor アクター

```

```

actor/ "アクター/"
agent エージェント
artifact アーティファクト
boundary 境界
card カード
circle 円
cloud クラウド
collections コレクション
component コンポーネント
control コントロール
database データベース
entity エンティティ
file ファイル
folder フォルダ
frame フレーム
hexagon 六角形
interface インターフェース
label ラベル
node ノード
package パッケージ
person 人型
queue キュー
rectangle 四角形
stack スタック
storage ストレージ
usecase ユースケース
usecase/ "ユースケース/"
@enduml

```



[Ref. QA-13261]



8.15.4 入れ子要素 (階層無し)

8.15.5 グローバルスタイル (componentDiagram)

```

@startuml
<style>
componentDiagram {
  BackGroundColor palegreen
  LineThickness 2
  LineColor red
}
</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml

```



8.15.6 入れ子要素ごとのスタイル

```

@startuml
<style>
artifact {
  BackGroundColor #ee1100
  LineThickness 1
  LineColor black
}
card {
  BackGroundColor #ff3311
  LineThickness 1
}

```



```
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
```



```

    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}

</style>
artifact artifact {
}
card card {
}
cloud cloud {
}
component component {
}
database database {
}
file file {
}
folder folder {
}
frame frame {
}
hexagon hexagon {
}
node node {
}
package package {
}
queue queue {
}
rectangle rectangle {
}
stack stack {
}
storage storage {
}
@enduml

```



8.15.7 入れ子要素（一階層）

8.15.8 グローバルスタイル（componentDiagram）

```

@startuml
<style>
componentDiagram {
    BackGroundColor palegreen
    LineThickness 1
    LineColor red
}
document {
    BackGroundColor white
}

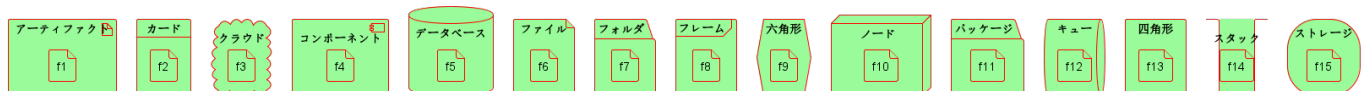
```



```

}
</style>
artifact e1 as "アーティファクト" {
file f1
}
card e2 as "カード" {
file f2
}
cloud e3 as "クラウド" {
file f3
}
component e4 as "コンポーネント" {
file f4
}
database e5 as "データベース" {
file f5
}
file e6 as "ファイル" {
file f6
}
folder e7 as "フォルダ" {
file f7
}
frame e8 as "フレーム" {
file f8
}
hexagon e9 as "六角形" {
file f9
}
node e10 as "ノード" {
file f10
}
package e11 as "パッケージ" {
file f11
}
queue e12 as "キュー" {
file f12
}
rectangle e13 as "四角形" {
file f13
}
stack e14 as "スタック" {
file f14
}
storage e15 as "ストレージ" {
file f15
}
@enduml

```



8.15.9 入れ子要素ごとのスタイル

```

@startuml
<style>
artifact {

```



```
    BackGroundColor #ee1100
    LineThickness 1
    LineColor black
}
card {
    BackGroundColor #ff3311
    LineThickness 1
    LineColor black
}
cloud {
    BackGroundColor #ff4422
    LineThickness 1
    LineColor black
}
component {
    BackGroundColor #ff6644
    LineThickness 1
    LineColor black
}
database {
    BackGroundColor #ff9933
    LineThickness 1
    LineColor black
}
file {
    BackGroundColor #feae2d
    LineThickness 1
    LineColor black
}
folder {
    BackGroundColor #ccbb33
    LineThickness 1
    LineColor black
}
frame {
    BackGroundColor #d0c310
    LineThickness 1
    LineColor black
}
hexagon {
    BackGroundColor #aacc22
    LineThickness 1
    LineColor black
}
node {
    BackGroundColor #22ccaa
    LineThickness 1
    LineColor black
}
package {
    BackGroundColor #12bdb9
    LineThickness 1
    LineColor black
}
queue {
    BackGroundColor #11aabb
    LineThickness 1
    LineColor black
}
```



```

}
rectangle {
    BackGroundColor #4444dd
    LineThickness 1
    LineColor black
}
stack {
    BackGroundColor #3311bb
    LineThickness 1
    LineColor black
}
storage {
    BackGroundColor #3b0cbd
    LineThickness 1
    LineColor black
}
</style>
artifact e1 as "アーティファクト" {
file f1
}
card e2 as "カード" {
file f2
}
cloud e3 as "クラウド" {
file f3
}
component e4 as "コンポーネント" {
file f4
}
database e5 as "データベース" {
file f5
}
file e6 as "ファイル" {
file f6
}
folder e7 as "フォルダ" {
file f7
}
frame e8 as "フレーム" {
file f8
}
hexagon e9 as "六角形" {
file f9
}
node e10 as "ノード" {
file f10
}
package e11 as "パッケージ" {
file f11
}
queue e12 as "キュー" {
file f12
}
rectangle e13 as "四角形" {
file f13
}
stack e14 as "スタック" {
file f14
}

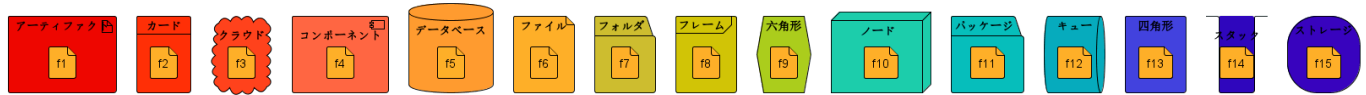
```



```

}
storage e15 as "ストレージ" {
file f15
}
}
@enduml

```



8.16 付録: すべての要素にスタイル指定した場合のステレオタイプのテスト

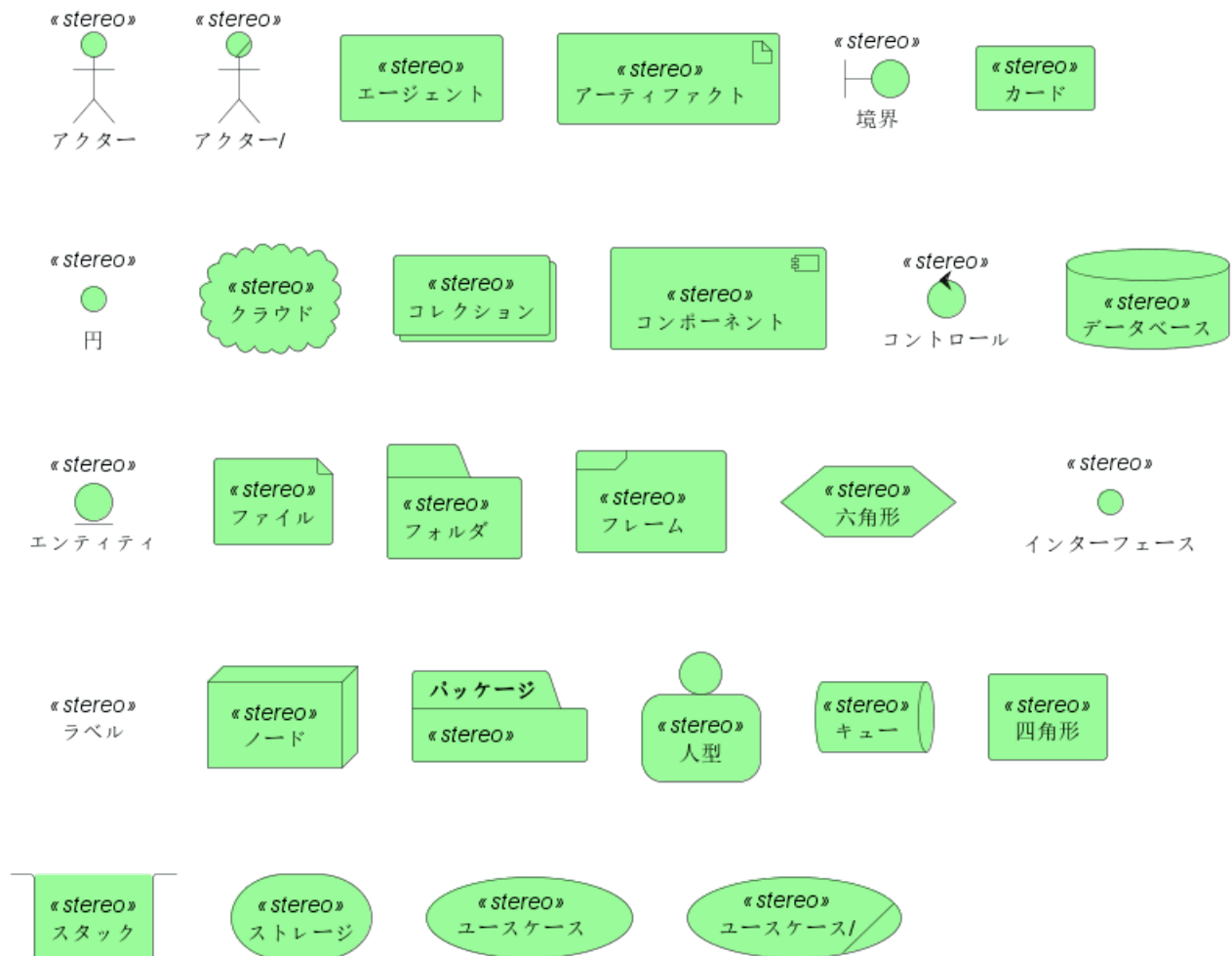
8.16.1 単純な要素

```

@startuml
<style>
.stereo {
  BackgroundColor palegreen
}
</style>
actor アクター << stereo >>
actor/ "アクター/" << stereo >>
agent エージェント << stereo >>
artifact アーティファクト << stereo >>
boundary 境界 << stereo >>
card カード << stereo >>
circle 円 << stereo >>
cloud クラウド << stereo >>
collections コレクション << stereo >>
component コンポーネント << stereo >>
control コントロール << stereo >>
database データベース << stereo >>
entity エンティティ << stereo >>
file ファイル << stereo >>
folder フォルダ << stereo >>
frame フレーム << stereo >>
hexagon 六角形 << stereo >>
interface インターフェース << stereo >>
label ラベル << stereo >>
node ノード << stereo >>
package パッケージ << stereo >>
person 人型 << stereo >>
queue キュー << stereo >>
rectangle 四角形 << stereo >>
stack スタック << stereo >>
storage ストレージ << stereo >>
usecase ユースケース << stereo >>
usecase/ "ユースケース/" << stereo >>
@enduml

```





8.17 Display JSON Data on Deployment diagram

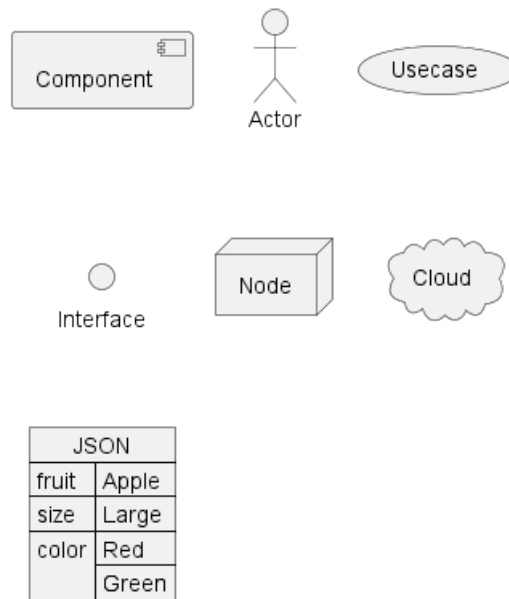
8.17.1 Simple example

```
@startuml
allowmixing

component Component
actor Actor
usecase Usecase
() Interface
node Node
cloud Cloud

json JSON {
    "fruit": "Apple",
    "size": "Large",
    "color": ["Red", "Green"]
}

@enduml
```



[Ref. QA-15481]

For another example, see on JSON page.

8.18 Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram

In order to add a Deployment element or a State element within a Class or Object diagram, you can use the `allowmixing` or `allow_mixing` directive.

8.18.1 Mixing all elements

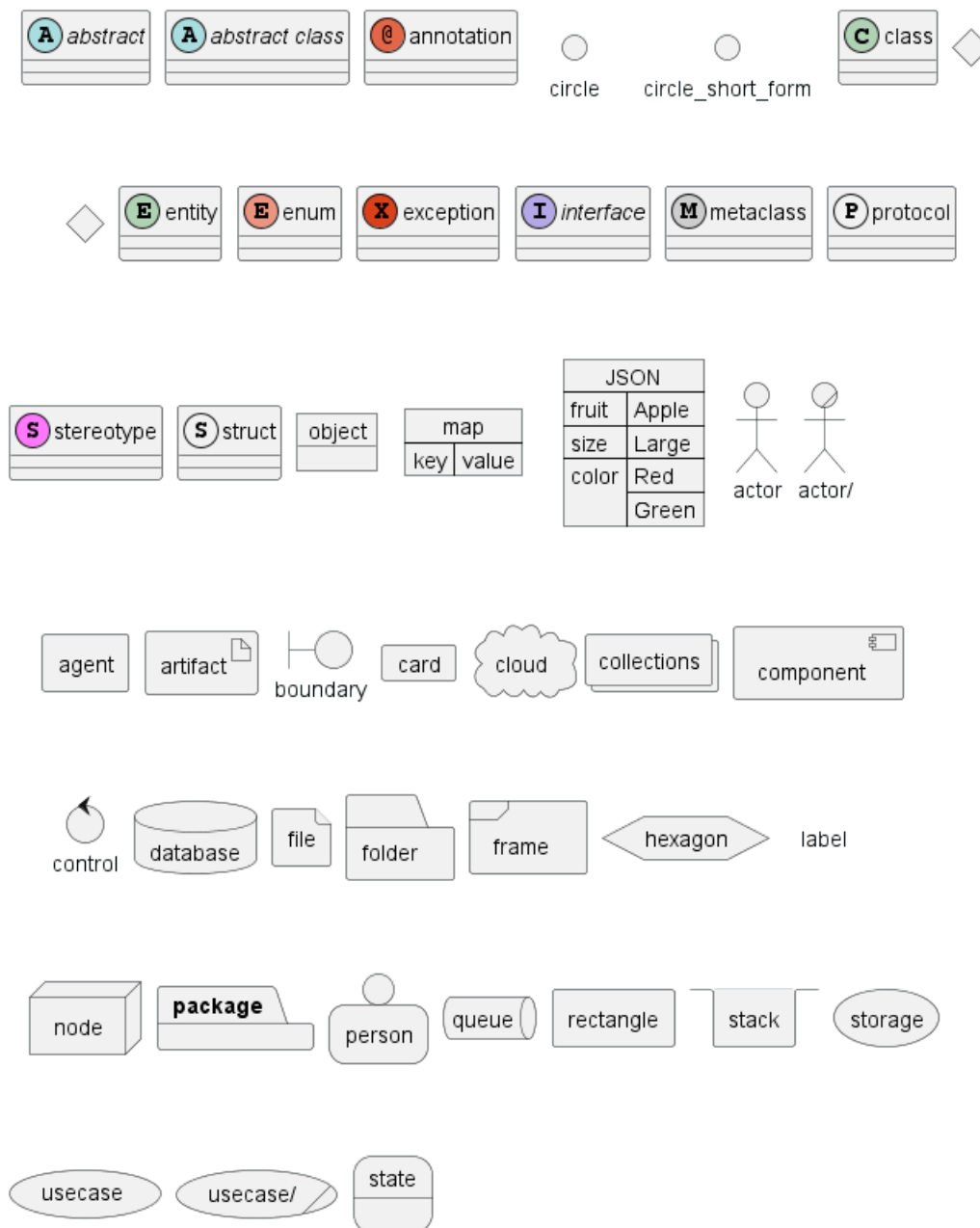
```
@startuml
```

```
allowmixing
```

```
skinparam nodesep 10
abstract      abstract
abstract class "abstract class"
annotation    annotation
circle        circle
()            circle_short_form
class         class
diamond       diamond
<>           diamond_short_form
entity        entity
enum          enum
exception     exception
interface     interface
metaclass     metaclass
protocol      protocol
stereotype    stereotype
struct        struct
object        object
map map {
  key => value
}
json JSON {
  "fruit":"Apple",
```



```
        "size": "Large",
        "color": ["Red", "Green"]
    }
    actor actor
    actor/ "actor/"
    agent agent
    artifact artifact
    boundary boundary
    card card
    circle circle
    cloud cloud
    collections collections
    component component
    control control
    database database
    entity entity
    file file
    folder folder
    frame frame
    hexagon hexagon
    interface interface
    label label
    node node
    package package
    person person
    queue queue
    rectangle rectangle
    stack stack
    storage storage
    usecase usecase
    usecase/ "usecase/"
    state state
    @enduml
```



[Ref. QA-2335 and QA-5329]

8.19 Port [port, portIn, portOut]

You can added **port** with port, portin and portout keywords.

8.19.1 Port

```
@startuml
[c]
node node {
  port p1
  port p2
  port p3
  file f1
}

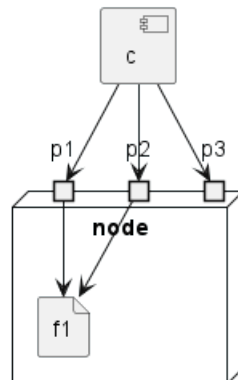
c --> p1
```



```

c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml

```



8.19.2 PortIn

```

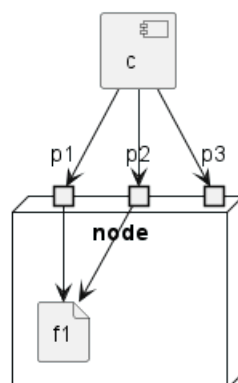
@startuml
[c]
node node {
    portin p1
    portin p2
    portin p3
    file f1
}

```

```

c --> p1
c --> p2
c --> p3
p1 --> f1
p2 --> f1
@enduml

```



8.19.3 PortOut

```

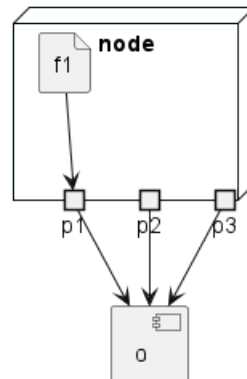
@startuml
node node {
    portout p1
    portout p2
    portout p3
    file f1
}

```

```

}
[o]
p1 --> o
p2 --> o
p3 --> o
f1 --> p1
@enduml

```



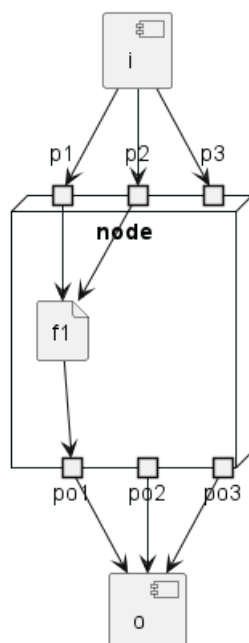
8.19.4 Mixing PortIn & PortOut

```

@startuml
[i]
node node {
    portin p1
    portin p2
    portin p3
    portout po1
    portout po2
    portout po3
    file f1
}
[o]

i --> p1
i --> p2
i --> p3
p1 --> f1
p2 --> f1
po1 --> o
po2 --> o
po3 --> o
f1 --> po1
@enduml

```



9 ステート図

状態遷移図 (ステート図) とは、システムの振る舞いを抽象化して表現するために使われます。This behavior is represented as a series of events that can occur in one or more possible states.

Using [PlantUML](https://plantuml.com/) to create state diagrams offers several advantages:

- **Text-Based Language:** Quickly define and visualize the states and transitions without the hassle of manual drawing.
- **Efficiency and Consistency:** Ensure streamlined diagram creation and easy version control.
- **Versatility:** Integrate with various documentation platforms and support multiple output formats.
- **Open-Source & Community Support:** Backed by a [strong community](https://forum.plantuml.net/) that continuously contributes to its enhancements and offers invaluable resources.

9.1 簡単なステート

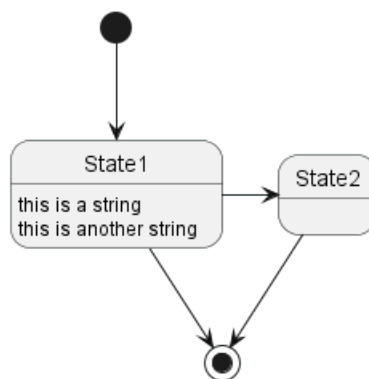
ステート図の始点と終点は、[*] で示します。

矢印は、--> で示します。

```
@startuml
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```



9.2 ステートの表現を変える

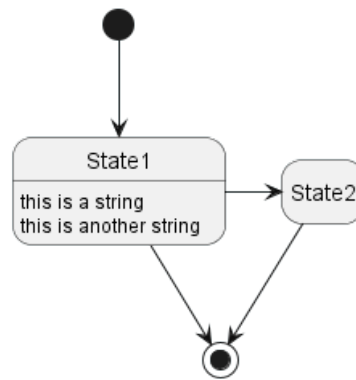
hide empty description を使用し、ステート枠をシンプルにできます。

```
@startuml
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```





9.3 合成状態

状態は合成することもできます。キーワード `state` と中括弧を使用して定義します。

9.3.1 内部サブ状態

```

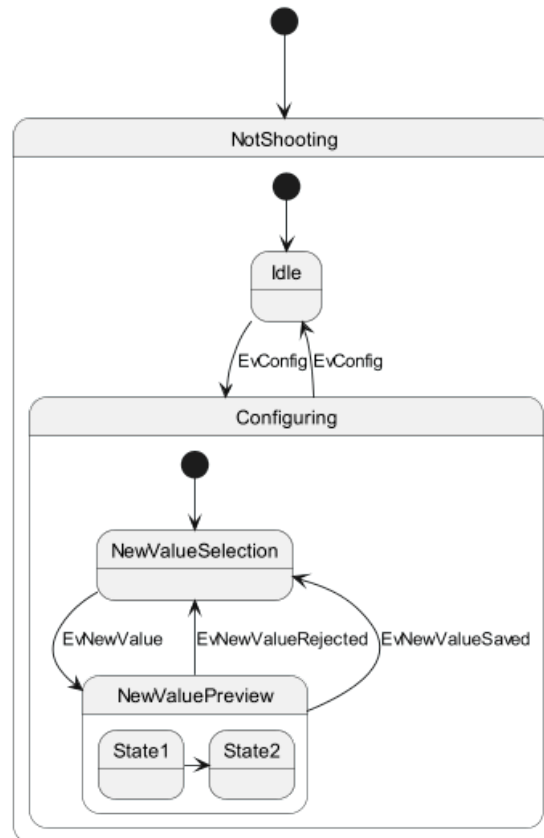
@startuml
scale 350 width
[*] --> NotShooting

state NotShooting {
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

state Configuring {
    [*] --> NewValueSelection
    NewValueSelection --> NewValuePreview : EvNewValue
    NewValuePreview --> NewValueSelection : EvNewValueRejected
    NewValuePreview --> NewValueSelection : EvNewValueSaved

    state NewValuePreview {
        State1 -> State2
    }
}

@enduml
  
```



9.3.2 サブ状態からサブ状態へ

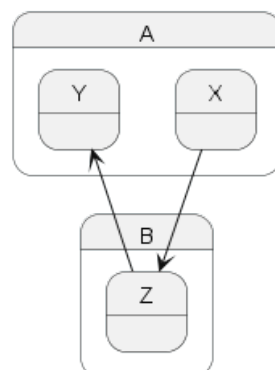
```

@startuml
state A {
    state X {
    }
    state Y {
    }
}

state B {
    state Z {
    }
}

X --> Z
Z --> Y
@enduml

```



[Ref. QA-3300]

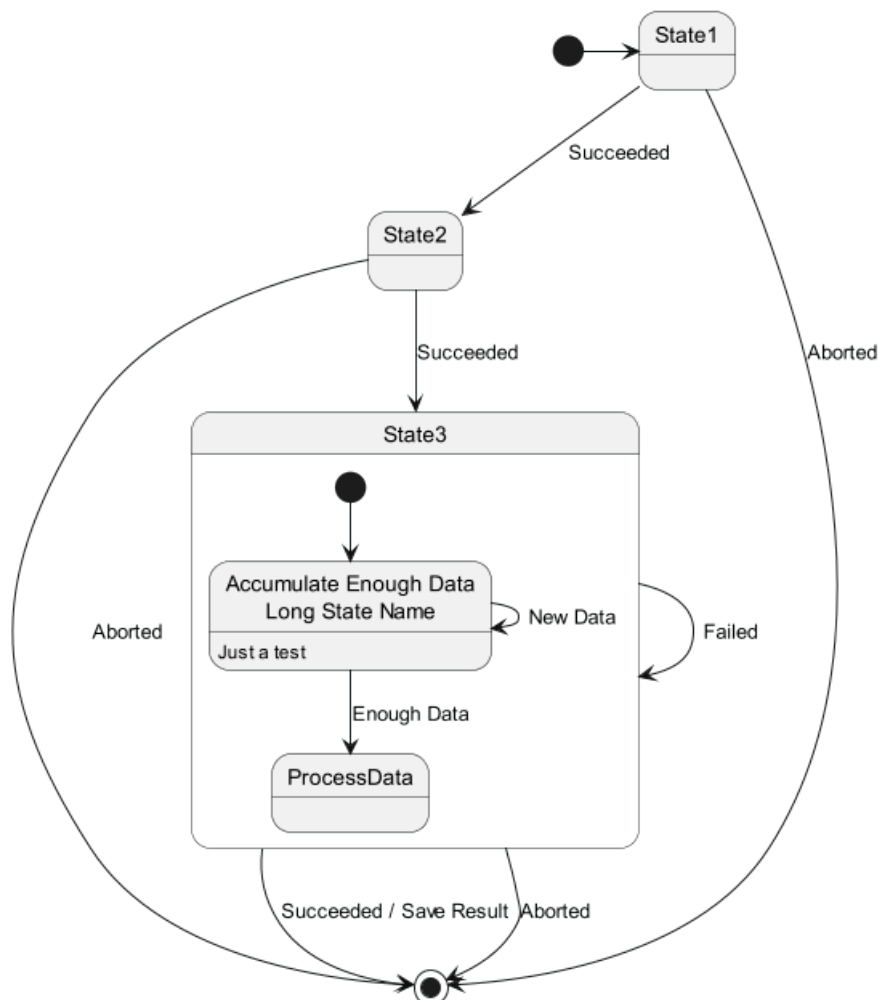
9.4 長い名前

キーワード `state` によって、状態についての長めの記述を使用することができます。

```
@startuml
scale 600 width

[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
  state "Accumulate Enough Data\nLong State Name" as long1
  long1 : Just a test
  [*] --> long1
  long1 --> long1 : New Data
  long1 --> ProcessData : Enough Data
}
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted

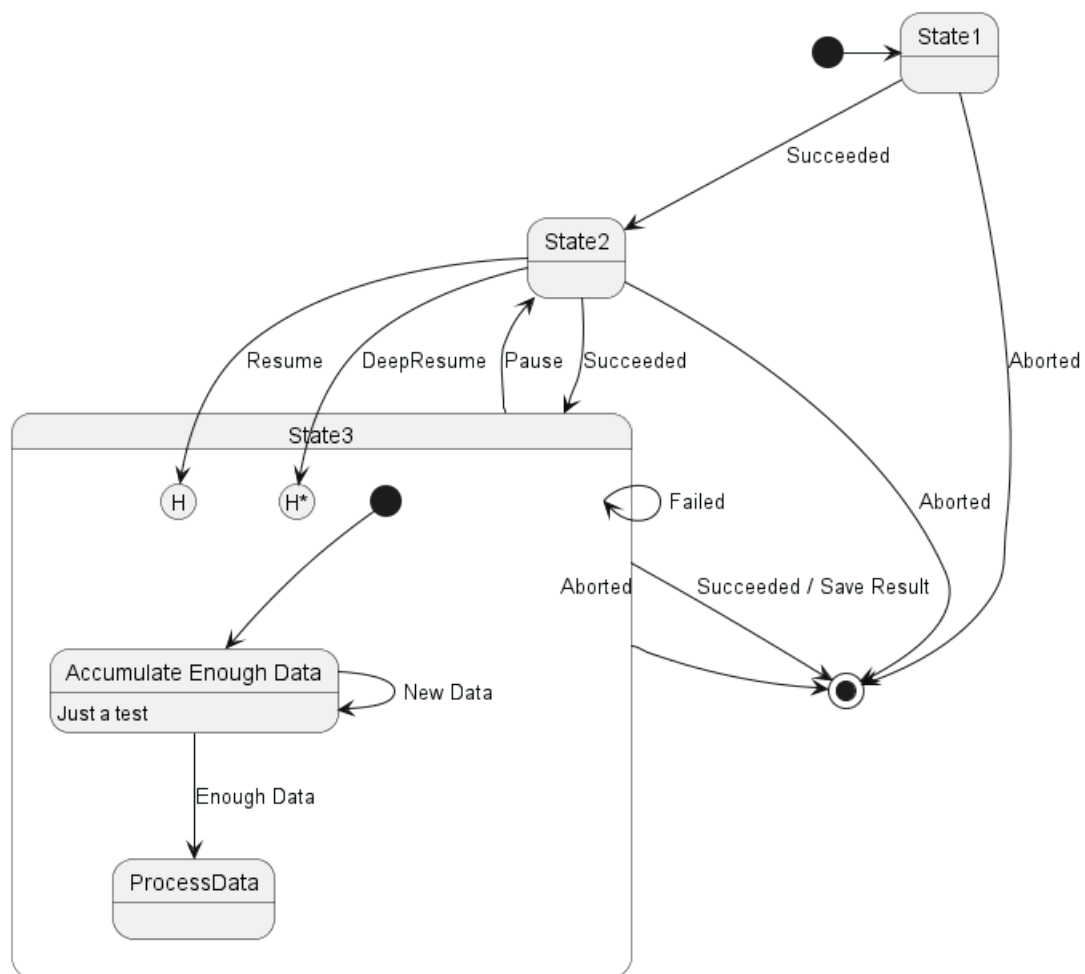
@enduml
```



9.5 履歴

[H] でサブ状態の履歴、[H*] でサブ状態の深い履歴 (deep history) を定義します。

```
@startuml
[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
    state "Accumulate Enough Data" as long1
    long1 : Just a test
    [*] --> long1
    long1 --> long1 : New Data
    long1 --> ProcessData : Enough Data
    State2 --> [H]: Resume
}
State3 --> State2 : Pause
State2 --> State3[H*]: DeepResume
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted
@enduml
```



9.6 フォーク (非同期実行)

<<fork>> と <<join>> stereotypes を使うことで、フォークノードとジョインノードを表せます。



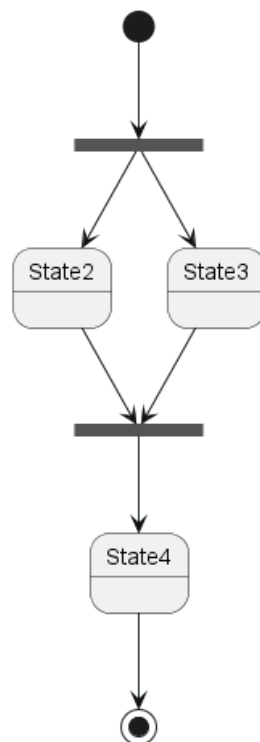
```

@startuml
state fork_state <<fork>>
[*] --> fork_state
fork_state --> State2
fork_state --> State3

state join_state <<join>>
State2 --> join_state
State3 --> join_state
join_state --> State4
State4 --> [*]

@enduml

```



9.7 同時状態

記号--または || で分離することで、合成状態の中に同時状態を定義することができます。

9.7.1 水平セパレータ--

```

@startuml
[*] --> Active

state Active {
    [*] -> NumLockOff
    NumLockOff --> NumLockOn : EvNumLockPressed
    NumLockOn --> NumLockOff : EvNumLockPressed
    --
    [*] -> CapsLockOff
    CapsLockOff --> CapsLockOn : EvCapsLockPressed
    CapsLockOn --> CapsLockOff : EvCapsLockPressed
    --
    [*] -> ScrollLockOff
    ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
}

```

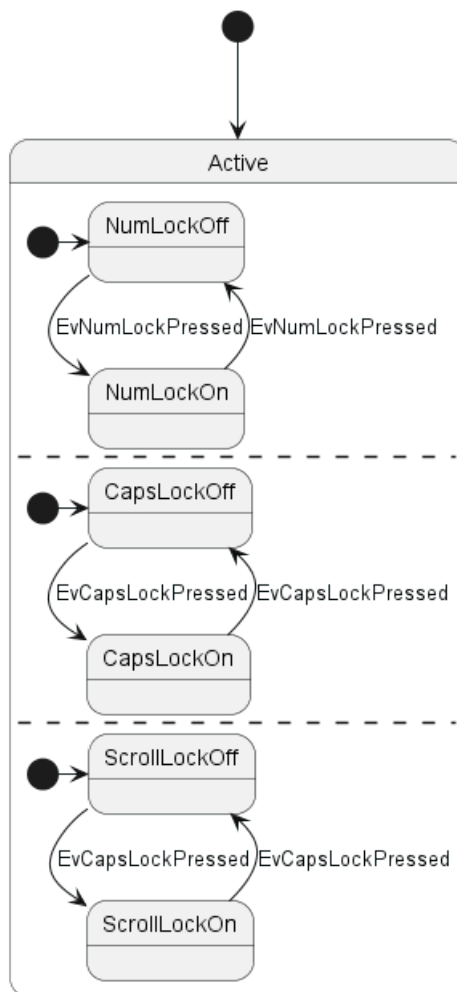


```

    ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml

```



9.7.2 垂直セパレータ ||

```

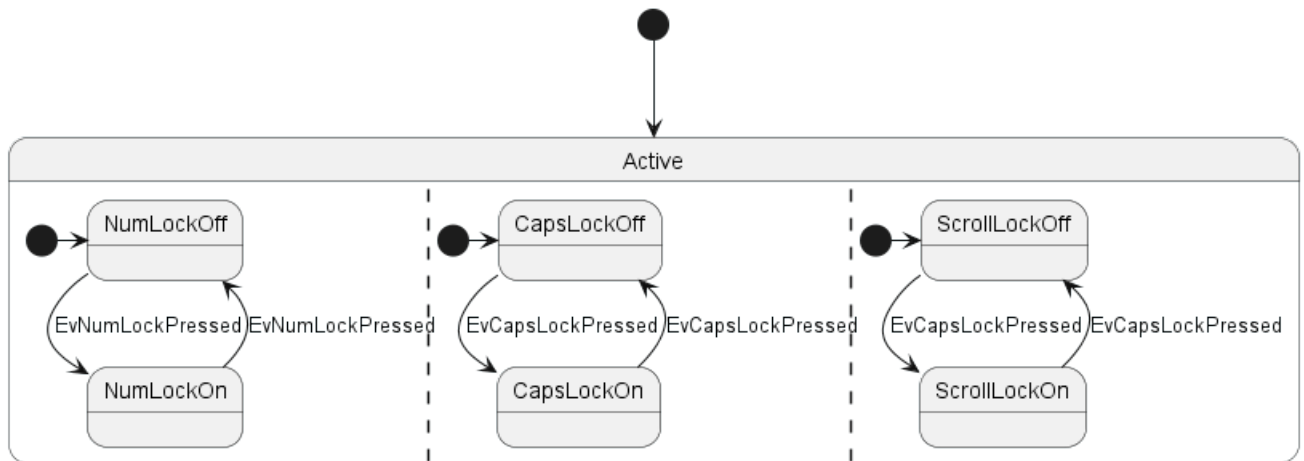
@startuml
[*] --> Active

state Active {
    [*] -> NumLockOff
    NumLockOff --> NumLockOn : EvNumLockPressed
    NumLockOn --> NumLockOff : EvNumLockPressed
    ||
    [*] -> CapsLockOff
    CapsLockOff --> CapsLockOn : EvCapsLockPressed
    CapsLockOn --> CapsLockOff : EvCapsLockPressed
    ||
    [*] -> ScrollLockOff
    ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
    ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml

```





[Ref. QA-3086]

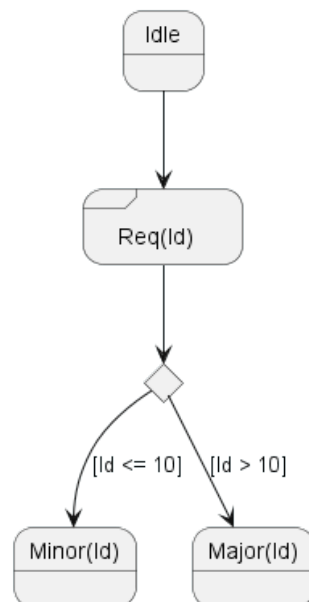
9.8 条件

<<choice>> ステレオタイプで条件付きの状態を表すことができます。

```
@startuml
state "Req(Id)" as ReqId <<sdlreceive>>
state "Minor(Id)" as MinorId
state "Major(Id)" as MajorId

state c <<choice>>

Idle --> ReqId
ReqId --> c
c --> MinorId : [Id <= 10]
c --> MajorId : [Id > 10]
@enduml
```



9.9 全ステレオタイプの例 (choice, fork, join, end)

```
@startuml
state choice1 <<choice>>
state fork1 <<fork>>
```



```

state join2    <<join>>
state end3     <<end>>

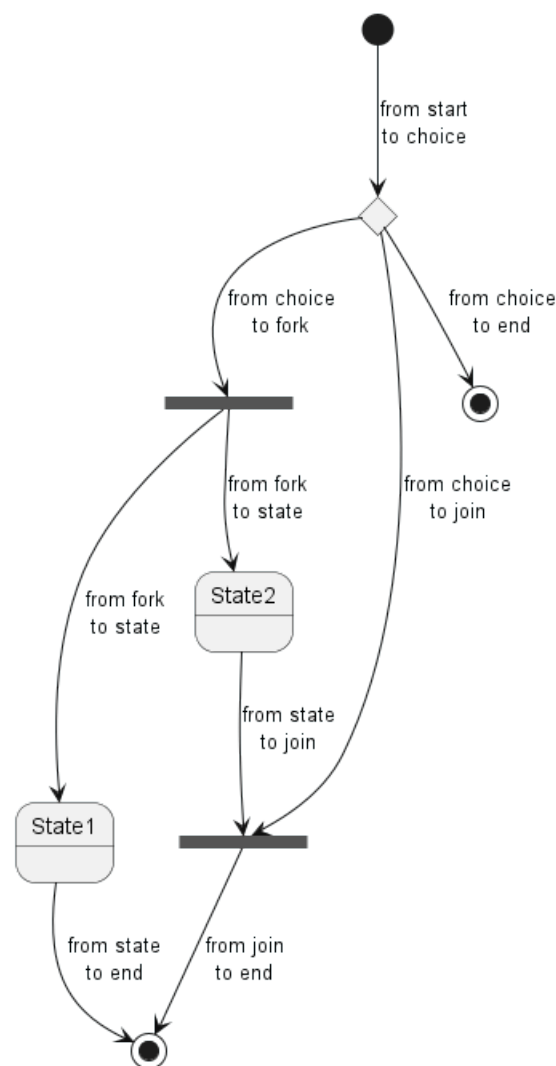
[*]    --> choice1 : from start\nto choice
choice1 --> fork1   : from choice\nto fork
choice1 --> join2   : from choice\nto join
choice1 --> end3    : from choice\nto end

fork1   ---> State1 : from fork\nto state
fork1   ---> State2 : from fork\nto state

State2 --> join2   : from state\nto join
State1 --> [*]    : from state\nto end

join2   --> [*]    : from join\nto end
@enduml

```



[Ref. QA-404 and QA-1159]

[Ref. QA-404, QA-1159 and GH-887]

9.10 入場点と退場点

<<entryPoint>> ステレオタイプで入場点、<<exitPoint>> ステレオタイプで退場点を追加することができます。

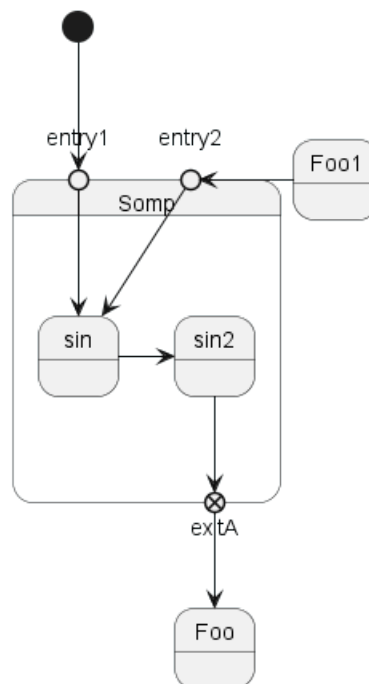


```

@startuml
state Somp {
  state entry1 <<entryPoint>>
  state entry2 <<entryPoint>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<exitPoint>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml

```



9.11 入力ピンと出力ピン

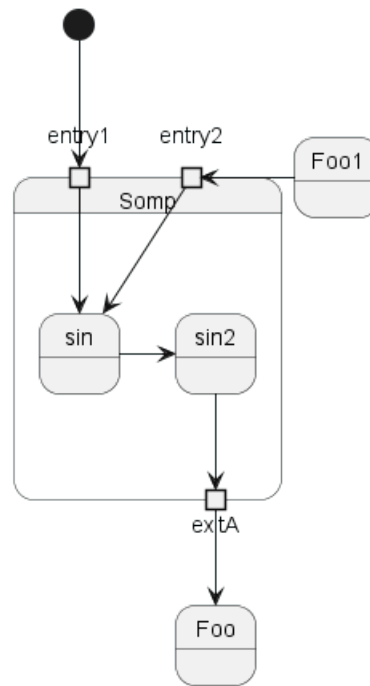
`<<inputPin>>` ステレオタイプで入力ピン、`<<outputPin>>` ステレオタイプで出力ピンを追加することができます。

```

@startuml
state Somp {
  state entry1 <<inputPin>>
  state entry2 <<inputPin>>
  state sin
  entry1 --> sin
  entry2 -> sin
  sin -> sin2
  sin2 --> exitA <<outputPin>>
}

[*] --> entry1
exitA --> Foo
Foo1 -> entry2
@enduml

```



[Ref. QA-4309]

9.12 展開

<<expansionInput>> と <<expansionOutput>> ステレオタイプにより、展開 (expansion) を追加することができます。

```

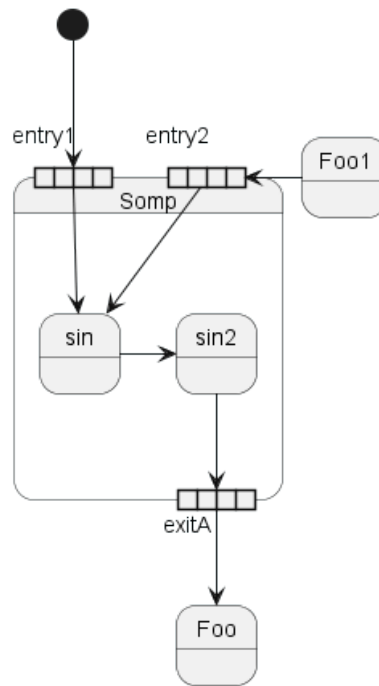
@startuml
state Somp {
    state entry1 <<expansionInput>>
    state entry2 <<expansionInput>>
    state sin
    entry1 --> sin
    entry2 --> sin
    sin --> sin2
    sin2 --> exitA <<expansionOutput>>
}

```

```

[*] --> entry1
exitA --> Foo
Foo1 --> entry2
@enduml

```



[Ref. QA-4309]

9.13 矢印の方向

記号-> を水平矢印として使用でき、以下の構文を使用することで、矢印の方向を指定することができます。

- -down-> or -->
- -right-> or -> (デフォルトの矢印)
- -left->
- -up->

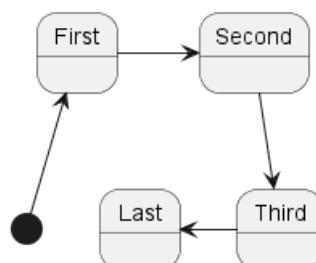
@startuml

```

[*] -up-> First
First -right-> Second
Second --> Third
Third -left-> Last

```

@enduml



方向を示す単語の、最初の文字だけ（例：-down-の代わりに-d-）、または 2 文字（-do-）を使用することで、矢印の記述を短くすることができます。

この機能を乱用しないよう注意しなくてはなりません：通常、*Graphviz* は微調整なしでよい結果をもたらしてくれます。

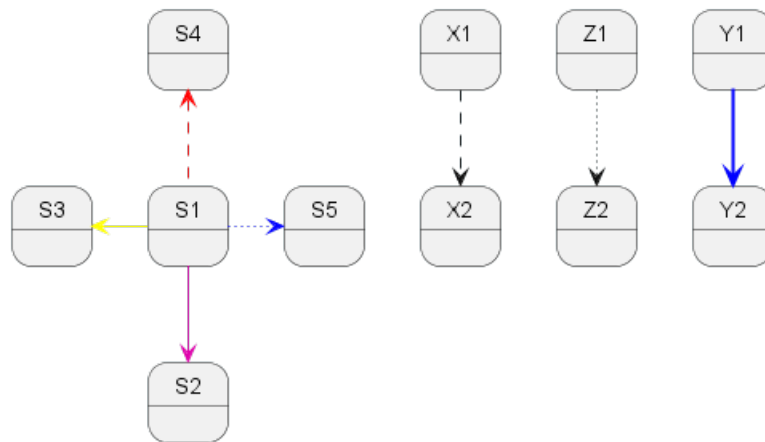


9.14 線の色とスタイルを変更する

線の色とスタイルを変更することができます。

```
@startuml
State S1
State S2
S1 -[#DD00AA]-> S2
S1 -left[#yellow]-> S3
S1 -up[#red,dashed]-> S4
S1 -right[dotted,#blue]-> S5

X1 -[dashed]-> X2
Z1 -[dotted]-> Z2
Y1 -[#blue,bold]-> Y2
@enduml
```



[Ref. Incubation: Change line color in state diagrams]

9.15 注釈

キーワード `note left of`, `note right of`, `note top of`, `note bottom of` を使用して注釈を定義することができます。

また、複数行の注釈を定義することもできます。

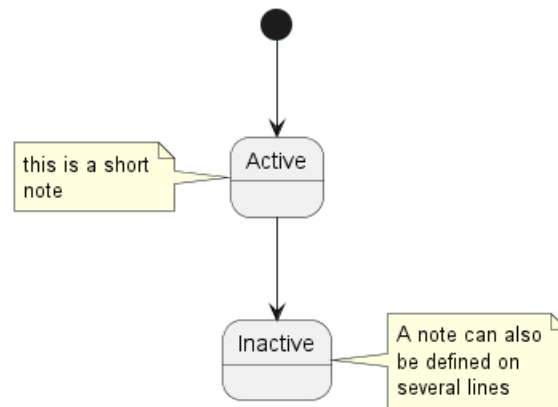
```
@startuml

[*] --> Active
Active --> Inactive

note left of Active : this is a short\nnote

note right of Inactive
  A note can also
  be defined on
  several lines
end note

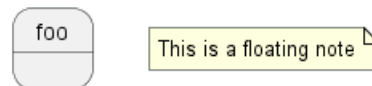
@enduml
```



さらに、状態にひもづかない注釈を定義できます。

```

@startuml
state foo
note "This is a floating note" as N1
@enduml
  
```

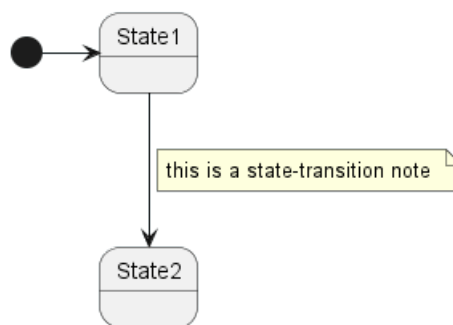


9.16 リンクへの注釈

`note on link` キーワードで、状態遷移（リンク）に注釈を追加することができます。

```

@startuml
[*] -> State1
State1 --> State2
note on link
    this is a state-transition note
end note
@enduml
  
```



9.17 その他の注釈

合成状態にも注釈をつけることができます。

```

@startuml
[*] --> NotShooting

state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle
    state "Configuring mode" as Configuring
  }
  
```



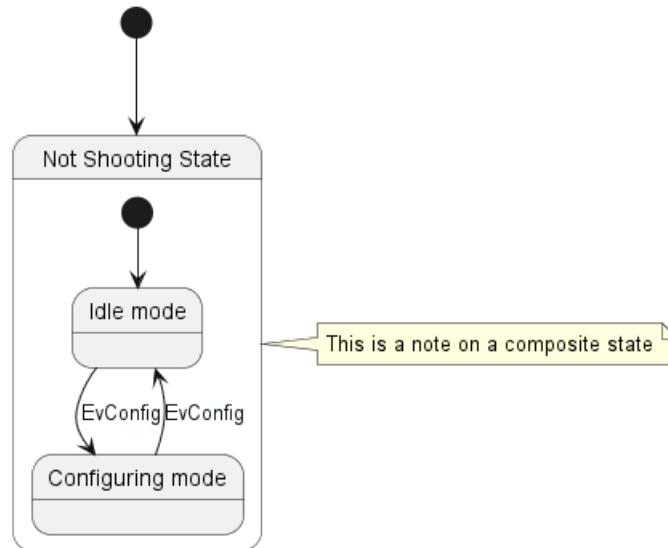
```

[*] --> Idle
Idle --> Configuring : EvConfig
Configuring --> Idle : EvConfig
}

note right of NotShooting : This is a note on a composite state

@enduml

```

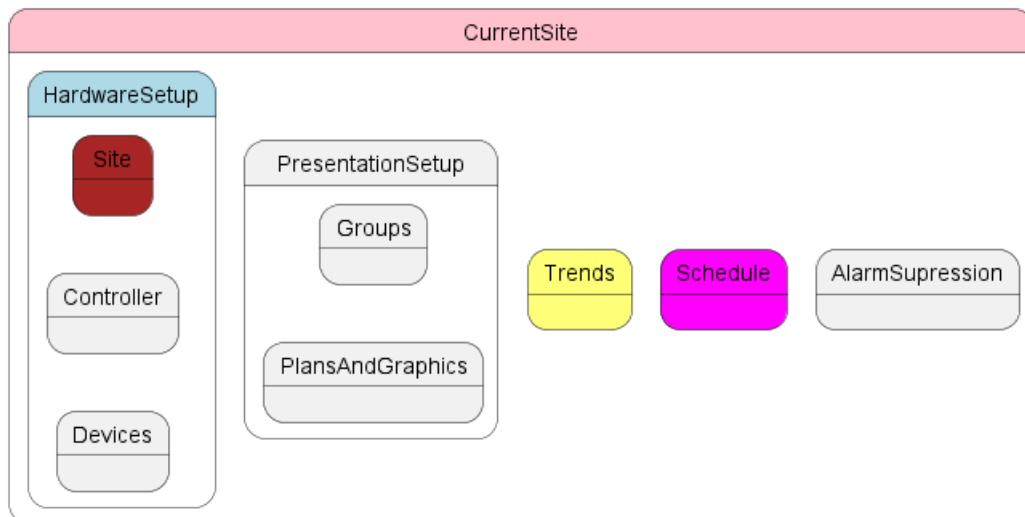


9.18 インライン色指定

```

@startuml
state CurrentSite #pink {
    state HardwareSetup #lightblue {
        state Site #brown
        Site -[hidden]-> Controller
        Controller -[hidden]-> Devices
    }
    state PresentationSetup{
        Groups -[hidden]-> PlansAndGraphics
    }
    state Trends #FFFF77
    state Schedule #magenta
    state AlarmSupression
}
@enduml

```



[Ref. QA-1812]

9.19 skinparam

skinparam コマンドを使用すると、ダイアグラムの色やフォントを変更できます。

このコマンドは以下の場面で使用できます。

- 他のコマンドと同様にダイアグラム定義内で
- インクルードされたファイル内
- コマンドラインや Ant タスクで指定された設定ファイル内

ステレオタイプが指定された状態に対して、特定の色とフォントを定義することができます。

```

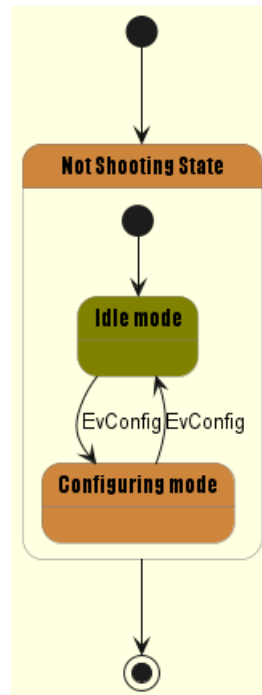
@startuml
skinparam backgroundColor LightYellow
skinparam state {
  StartColor MediumBlue
  EndColor Red
  BackgroundColor Peru
  BackgroundColor<<Warning>> Olive
  BorderColor Gray
  FontName Impact
}

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle <<Warning>>
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml

```



9.19.1 すべての状態遷移図特有の **skinparam** のテスト

```

@startuml
skinparam State {
  AttributeFontColor blue
  AttributeFontName serif
  AttributeFontSize 9
  AttributeFontStyle italic
  BackgroundColor palegreen
  BorderColor violet
  EndColor gold
  FontColor red
  FontName Sanserif
  FontSize 15
  FontStyle bold
  StartColor silver
}

```

```

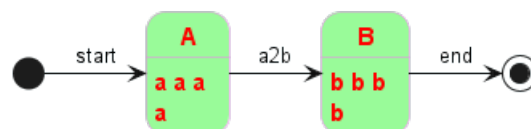
state A : a a a\na
state B : b b b\nb

```

```

[*] -> A : start
A -> B : a2b
B -> [*] : end
@enduml

```



9.20 スタイル変更

スタイルを変更できます。

```

@startuml

```



```

<style>
stateDiagram {
  BackgroundColor Peru
  'LineColor Gray
  FontName Impact
  FontColor Red
  arrow {
    FontSize 13
    LineColor Blue
  }
}
</style>

```

```

[*] --> NotShooting

```

```

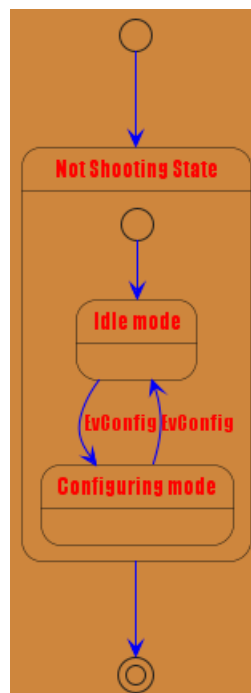
state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle <<Warning>>
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

```

```

NotShooting --> [*]
@enduml

```



[Ref. GH-880]

9.21 状態の色とスタイルを変更する (インラインスタイル)

個別の状態ごとに色とスタイルを変更するには、次の記法を使用します：

- #color ##[style]color

最初に背景色 (#color) を書き、次に線のスタイルと色 (##[style]color) を書きます。



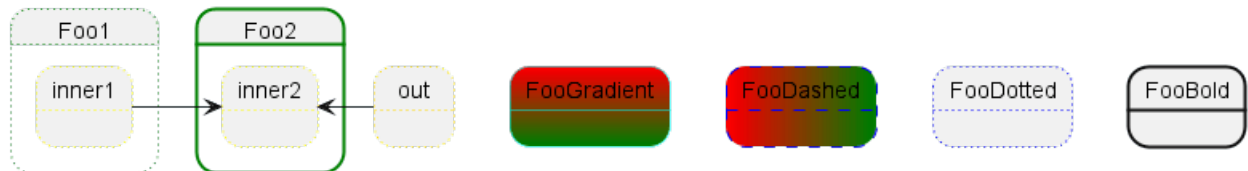
```

@startuml
state FooGradient #red-green ##00FFFF
state FooDashed #red|green ##[dashed]blue {
}
state FooDotted ##[dotted]blue {
}
state FooBold ##[bold] {
}
state Foo1 ##[dotted]green {
state inner1 ##[dotted]yellow
}

state out ##[dotted]gold

state Foo2 ##[bold]green {
state inner2 ##[dotted]yellow
}
inner1 -> inner2
out -> inner2
@enduml

```



[Ref. QA-1487]

- #color;line:color;line.[bold|dashed|dotted];text:color

TODO: FIXME text:color seems not to be taken into account **TODO: FIXME**

```

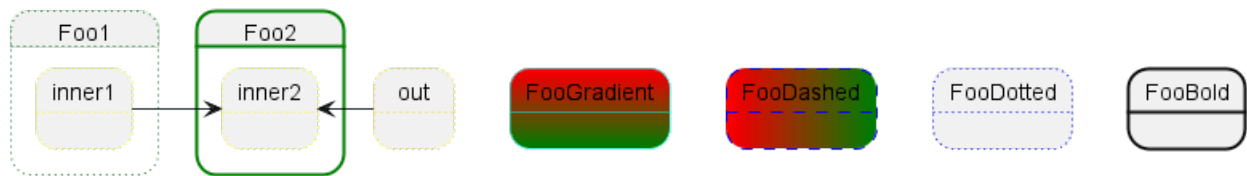
@startuml
@startuml
state FooGradient #red-green;line:00FFFF
state FooDashed #red|green;line.dashed;line:blue {
}
state FooDotted #line.dotted;line:blue {
}
state FooBold #line.bold {
}
state Foo1 #line.dotted;line:green {
state inner1 #line.dotted;line:yellow
}

state out #line.dotted;line:gold

state Foo2 #line.bold;line:green {
state inner2 #line.dotted;line:yellow
}
inner1 -> inner2
out -> inner2
@enduml
@enduml

```





```

@startuml
state s1 : s1 description
state s2 #pink;line:red;line.bold;text:red : s2 description
state s3 #palegreen;line:green;line.dashed;text:green : s3 description
state s4 #aliceblue;line:blue;line.dotted;text:blue : s4 description
@enduml

```



[Adapted from QA-3770]

9.22 別名

alias を使用して状態に別名を付けることができます:

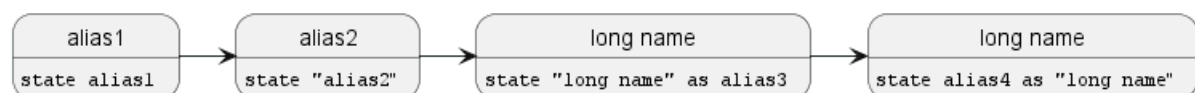
```

@startuml
state alias1
state "alias2"
state "long name" as alias3
state alias4 as "long name"

alias1 : ""state alias1""
alias2 : ""state "alias2"""
alias3 : ""state "long name" as alias3""
alias4 : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml

```



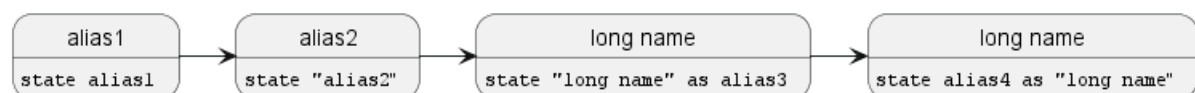
もしくは、

```

@startuml
state alias1 : ""state alias1""
state "alias2" : ""state "alias2"""
state "long name" as alias3 : ""state "long name" as alias3""
state alias4 as "long name" : ""state alias4 as "long name"""

alias1 -> alias2
alias2 -> alias3
alias3 -> alias4
@enduml

```

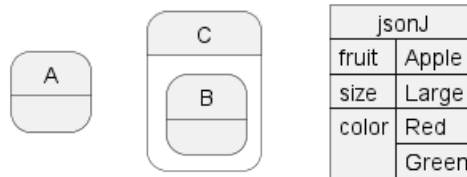


9.23 Display JSON Data on State diagram

9.23.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
  state B
}

json jsonJ {
  "fruit":"Apple",
  "size":"Large",
  "color": ["Red", "Green"]
}
@enduml
```



[Ref. QA-17275]

For another example, see on JSON page.

10 タイミング図

タイミング図は現在作成中です。必要な機能があれば提案してください。

10.1 要素、ライフラインの定義

以下のキーワードを使用してライフラインを定義します。表示方法に応じてキーワードを選択します。

キーワード	説明
concise	データの動きを表すための単純化された信号（メッセージに最適です）
robust	状態の遷移を表すための複雑な線（複数の状態を作れます）
clock	period の時間間隔で high と low の状態を繰り返し遷移するクロック信号（ pulse , offset を指定する）
binary	2 状態（バイナリ）に制限された信号

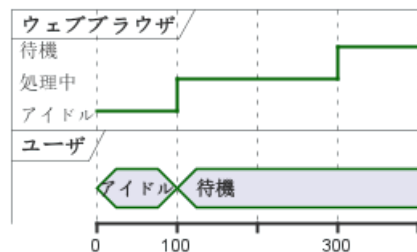
@ と is を用いて、状態の変化を記述できます。

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
@0
WU is アイドル
WB is アイドル
```

```
@100
WU is 待機
WB is 処理中
```

```
@300
WB is 待機
@enduml
```



```
@startuml
clock "Clock_0" as C0 with period 50
clock "Clock_1" as C1 with period 50 pulse 15 offset 10
binary "Binary" as B
concise "Concise" as C
robust "Robust" as R
```

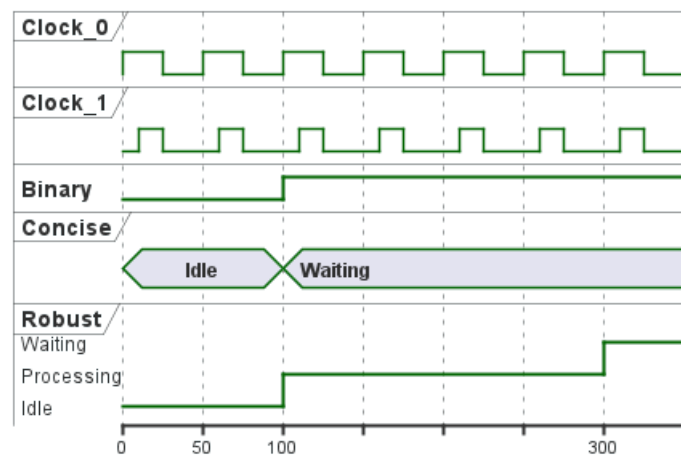
```
@0
C is Idle
R is Idle
```

```
@100
B is high
C is Waiting
R is Processing
```

```
@300
R is Waiting
```



@enduml



[Ref. QA-14631 and QA-14647]

[Ref. QA-14631, QA-14647 and QA-11288]

10.2 バイナリとクロック

次のキーワードでバイナリ信号とクロック信号を定義することができます:

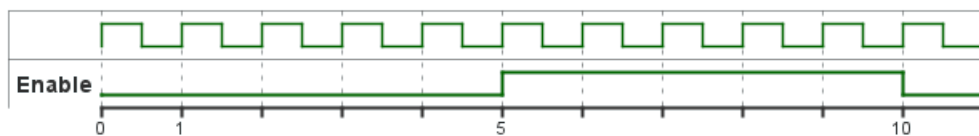
- binary
- clock

```
@startuml
clock clk with period 1
binary "Enable" as EN
```

```
@0
EN is low
```

```
@5
EN is high
```

```
@10
EN is low
@enduml
```



10.3 メッセージ（相互作用）

メッセージは、矢印構文を使います。

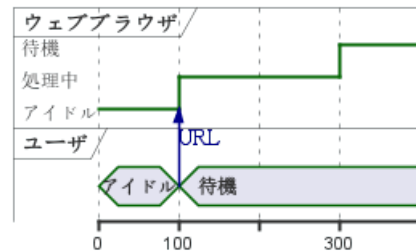
```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
@0
WU is アイドル
WB is アイドル
```



```
@100
WU -> WB : URL
WU is 待機
WB is 処理中
```

```
@300
WB is 待機
@enduml
```



10.4 相対時間での指定

@ で時間を指定するとき、相対的な時間の指定の仕方ができます。

```
@startuml
robust "DNS Resolver" as DNS
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

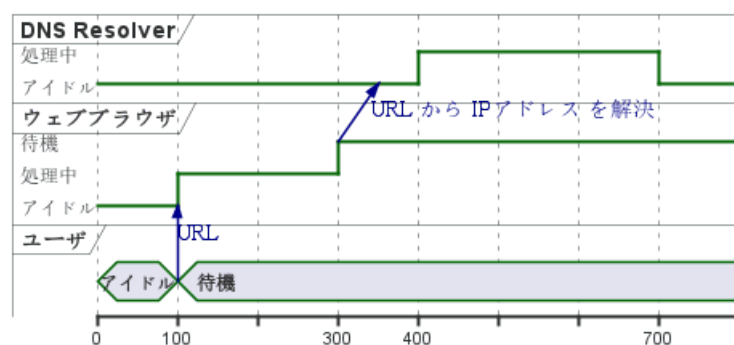
```
@0
WU is アイドル
WB is アイドル
DNS is アイドル
```

```
@+100
WU -> WB : URL
WU is 待機
WB is 処理中
```

```
@+200
WB is 待機
WB -> DNS@+50 : URL から IPアドレス を解決
```

```
@+100
DNS is 処理中
```

```
@+300
DNS is アイドル
@enduml
```



10.5 アンカーポイント

絶対時間、相対時間を使用する代わりに、`as` キーワードと`:`で開始する名前を使用してアンカーポイントを定義することができます。

```
@XX as :<anchor point name>

@startuml
clock clk with period 1
binary "enable" as EN
concise "dataBus" as db

@0 as :start
@5 as :en_high
@10 as :en_low
@:en_high-2 as :en_highMinus2

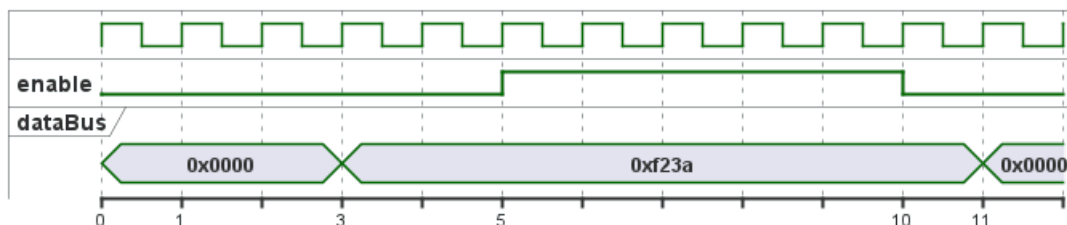
@:start
EN is low
db is "0x0000"

@:en_high
EN is high

@:en_low
EN is low

@:en_highMinus2
db is "0xf23a"

@:en_high+6
db is "0x0000"
@enduml
```



10.6 インスタンス指向

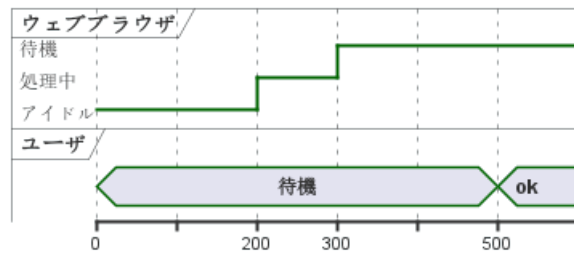
時系列順での定義ではなく、インスタンス毎（ライフライン毎）に定義できます。

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU

@WB
0 is アイドル
+200 is 処理中
+100 is 待機

@WU
0 is 待機
+500 is ok
@enduml
```



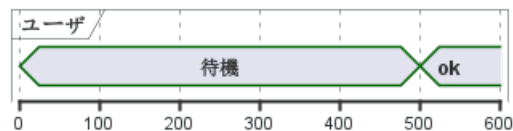


10.7 スケールの設定

スケール（目盛りの数値の表示）を指定できます。以下の例では、「目盛りを 100 ずつ表示、1 目盛りの幅を 50px にする」設定になります。

```
@startuml
concise "ユーザ" as WU
scale 100 as 50 pixels
```

```
@WU
0 is 待機
+500 is ok
@enduml
```



10.8 初期状態

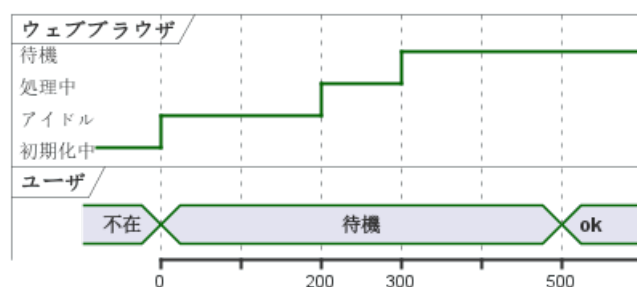
「初期状態」を設定できます。

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
WB is 初期化中
WU is 不在
```

```
@WB
0 is アイドル
+200 is 処理中
+100 is 待機
```

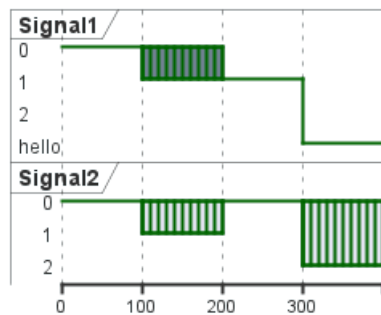
```
@WU
0 is 待機
+500 is ok
@enduml
```



10.9 複雑な状態

信号をいくつかの不定状態とすることができます。

```
@startuml
robust "Signal1" as S1
robust "Signal2" as S2
S1 has 0,1,2,hello
S2 has 0,1,2
@0
S1 is 0
S2 is 0
@100
S1 is {0,1} #SlateGrey
S2 is {0,1}
@200
S1 is 1
S2 is 0
@300
S1 is hello
S2 is {0,2}
@enduml
```



[Ref. [QA-11936](<https://forum.plantuml.net/11936>) and [QA-15933](<https://forum.plantuml.net/15933>)]

10.10 状態の非表示

いくつかの状態を非表示にすることもできます。

```
@startuml
concise "Web User" as WU

@0
WU is {-}

@100
WU is A1

@200
WU is {-}

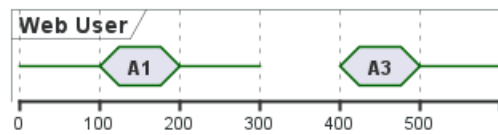
@300
WU is {hidden}

@400
WU is A3

@500
WU is {-}
```



```
@enduml
```



```
@startuml
scale 1 as 50 pixels
```

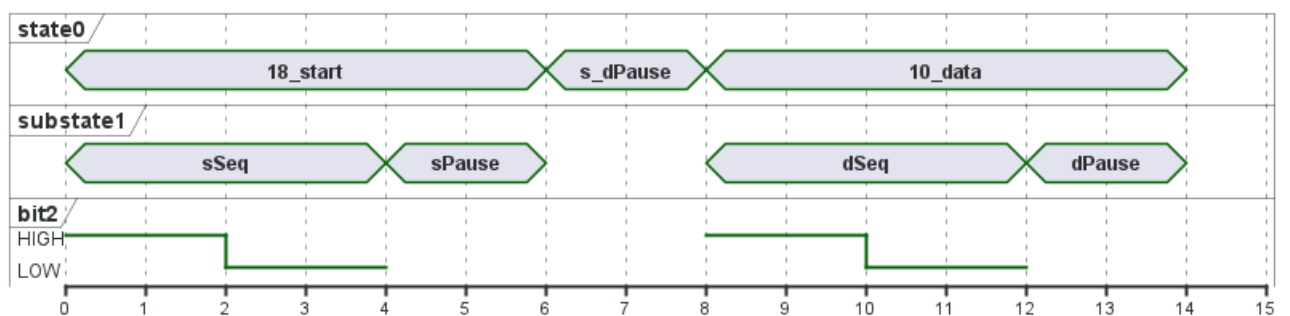
```
concise state0
concise substate1
robust bit2
```

```
bit2 has HIGH,LOW
```

```
@state0
0 is 18_start
6 is s_dPause
8 is 10_data
14 is {hidden}
```

```
@substate1
0 is sSeq
4 is sPause
6 is {hidden}
8 is dSeq
12 is dPause
14 is {hidden}
```

```
@bit2
0 is HIGH
2 is LOW
4 is {hidden}
8 is HIGH
10 is LOW
12 is {hidden}
@enduml
```



[Ref. QA-12222]

10.11 時間軸を非表示にする

時間軸を非表示にすることができます。

```
@startuml
hide time-axis
```



```
concise "ユーザ" as WU
```

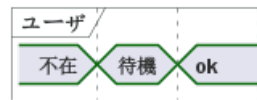
```
WU is 不在
```

```
@WU
```

```
0 is 待機
```

```
+500 is ok
```

```
@enduml
```



10.12 時刻と日付の使用

時刻または日付を使用することができます。

```
@startuml
```

```
robust "ウェブブラウザ" as WB
```

```
concise "ユーザ" as WU
```

```
@2019/07/02
```

```
WU is アイドル
```

```
WB is アイドル
```

```
@2019/07/04
```

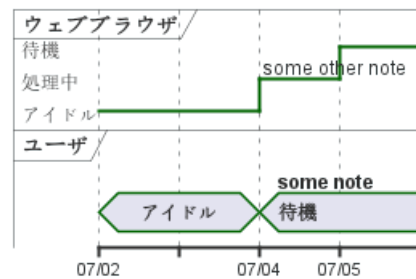
```
WU is 待機 : some note
```

```
WB is 処理中 : some other note
```

```
@2019/07/05
```

```
WB is 待機
```

```
@enduml
```



```
@startuml
```

```
robust "ウェブブラウザ" as WB
```

```
concise "ユーザ" as WU
```

```
@1:15:00
```

```
WU is アイドル
```

```
WB is アイドル
```

```
@1:16:30
```

```
WU is 待機 : some note
```

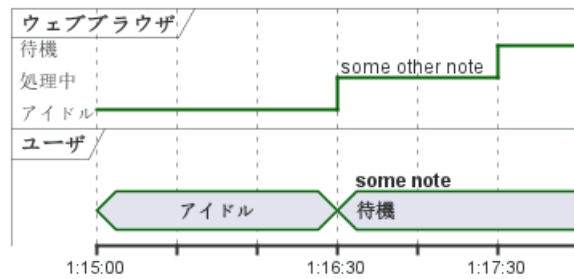
```
WB is 処理中 : some other note
```

```
@1:17:30
```

```
WB is 待機
```

```
@enduml
```





10.13 時間定規 (time constraint) の追加

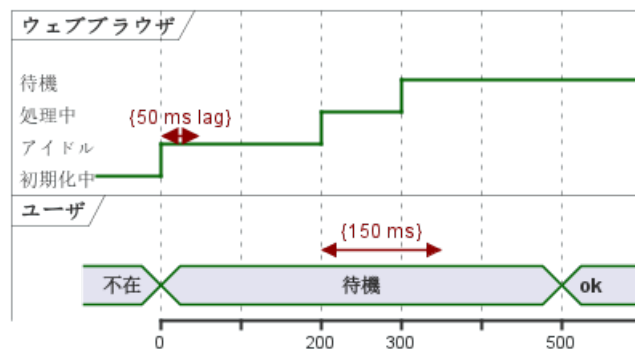
タイムラインの目盛りとは別に、時間の尺度を示す矢印を表示することができます。

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
WB is 初期化中
WU is 不在
```

```
@WB
0 is アイドル
+200 is 処理中
+100 is 待機
WB@0 <-> @50 : {50 ms lag}
```

```
@WU
0 is 待機
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.14 期間のハイライト

図の一部をハイライトすることができます。

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
@0
WU is アイドル
WB is アイドル
```

```
@100
WU -> WB : URL
```



```
WU is 待機 #LightCyan;line:Aqua
```

```
@200
```

```
WB is 処理中
```

```
@300
```

```
WU -> WB@350 : URL2
```

```
WB is 待機
```

```
@+200
```

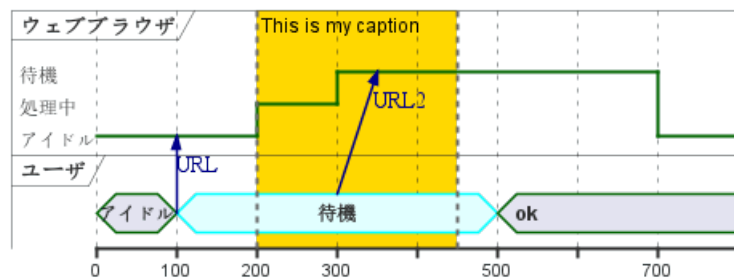
```
WU is ok
```

```
@+200
```

```
WB is アイドル
```

```
highlight 200 to 450 #Gold;line:DimGrey : This is my caption
```

```
@enduml
```



[Ref. QA-10868]

10.15 ノートの使用

note top of および note bottom of キーワードを使用して、1つのオブジェクトまたは参加者に関連するノートを定義できます (concise オブジェクトでのみ使用可能)。

```
@startuml
```

```
robust "Web Browser" as WB
```

```
concise "Web User" as WU
```

```
@0
```

```
WU is Idle
```

```
WB is Idle
```

```
@100
```

```
WU is Waiting
```

```
WB is Processing
```

```
note top of WU : first note\non several\nlines
```

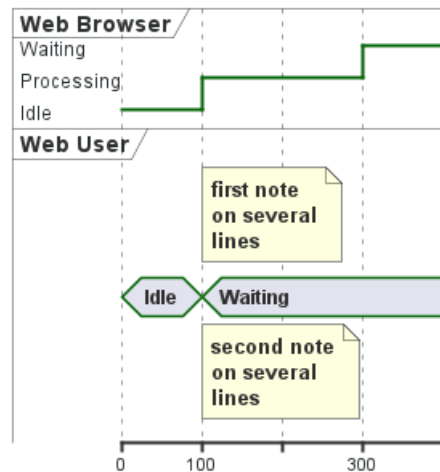
```
note bottom of WU : second note\non several\nlines
```

```
@300
```

```
WB is Waiting
```

```
@enduml
```





[Ref. QA-6877]

[Ref. [QA-6877](<https://forum.plantuml.net/6877>), [GH-1465](<https://github.com/plantuml/plantuml/issues/1465>)]

10.16 タイトルなどを追加する

(他の UML ダイアグラムと同様に) タイトル、ヘッダー／フッター、説明文、キャプションを書くことができます。

```

@startuml
Title これはタイトル
header: ここにヘッダーを書く
footer: ここにフッターを書く
legend
図の説明は、ここに書きます。
複数行かけますよ。
end legend
caption 一行の説明は、caption に書きましょう。
  
```

```

robust "Web ブラウザ" as WB
concise "ユーザ" as WU
  
```

```

@0
WU is アイドル
WB is アイドル
  
```

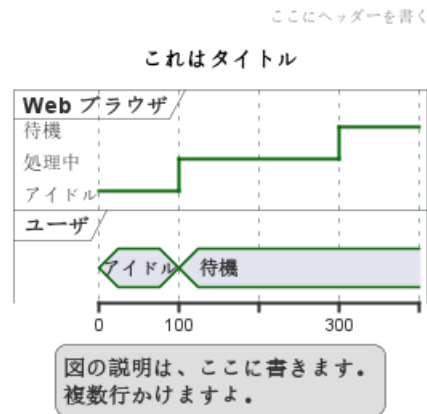
```

@100
WU is 待機
WB is 処理中
  
```

```

@300
WB is 待機
@enduml
  
```





一行の説明は、caption に書きましょう。

ここにフッターを書く

10.17 複雑な例

Adam Rosien による例:

```
@startuml
concise "Client" as Client
concise "Server" as Server
concise "Response freshness" as Cache

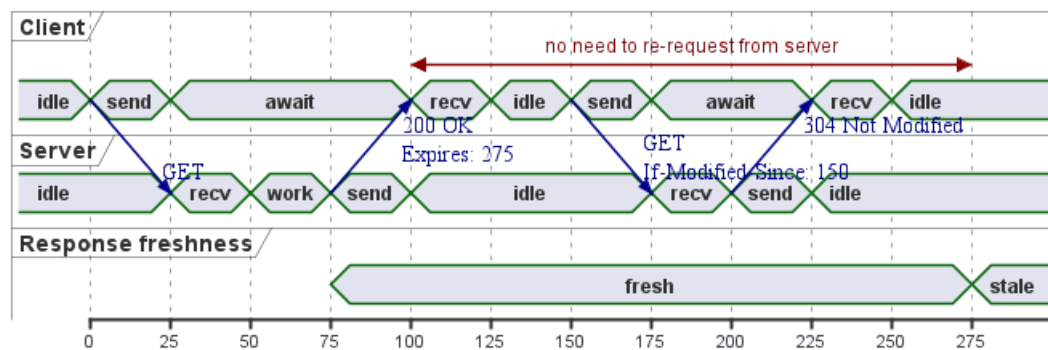
Server is idle
Client is idle

@Client
0 is send
Client -> Server@+25 : GET
+25 is await
+75 is recv
+25 is idle
+25 is send
Client -> Server@+25 : GET\nIf-Modified-Since: 150
+25 is await
+50 is recv
+25 is idle
@100 <-> @275 : no need to re-request from server

@Server
25 is recv
+25 is work
+25 is send
Server -> Client@+25 : 200 OK\nExpires: 275
+25 is idle
+75 is recv
+25 is send
Server -> Client@+25 : 304 Not Modified
+25 is idle

@Cache
75 is fresh
+200 is stale
@enduml
```





10.18 デジタル信号の例

```
@startuml
scale 5 as 150 pixels

clock clk with period 1
binary "enable" as en
binary "R/W" as rw
binary "data Valid" as dv
concise "dataBus" as db
concise "address bus" as addr
```

```
@6 as :write_beg
@10 as :write_end
```

```
@15 as :read_beg
@19 as :read_end
```

```
@0
en is low
db is "0x0"
addr is "0x03f"
rw is low
dv is 0
```

```
@:write_beg-3
  en is high
@:write_beg-2
  db is "0xDEADBEEF"
@:write_beg-1
  dv is 1
@:write_beg
  rw is high
```

```
@:write_end
  rw is low
  dv is low
@:write_end+1
  rw is low
  db is "0x0"
  addr is "0x23"
```

```
@12
dv is high
```



```

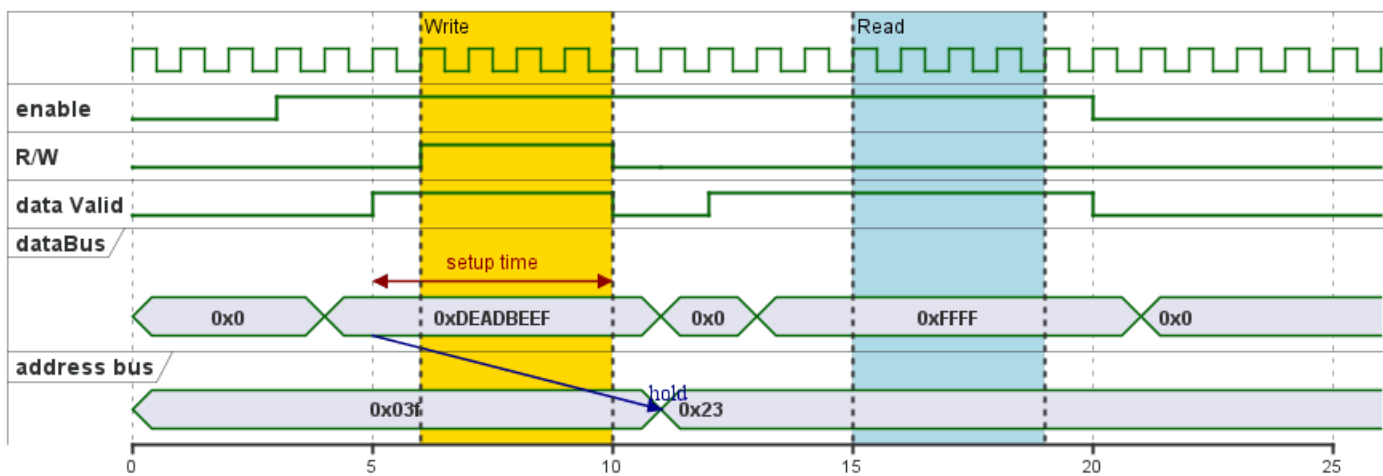
@13
db is "0xFFFF"

@20
en is low
dv is low
@21
db is "0x0"

highlight :write_beg to :write_end #Gold:Write
highlight :read_beg to :read_end #lightBlue:Read

db@:write_beg-1 <-> @:write_end : setup time
db@:write_beg-1 -> addr@:write_end+1 : hold
@enduml

```



10.19 色の追加

色を追加できます。

```

@startuml
concise "LR" as LR
concise "ST" as ST

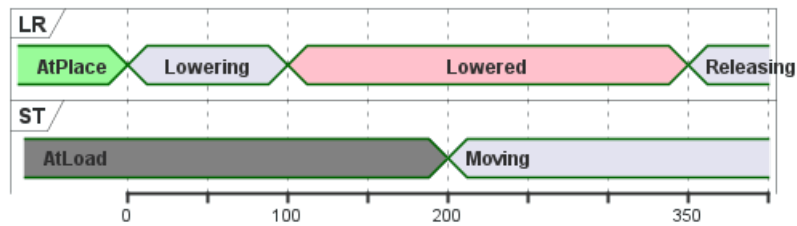
LR is AtPlace #palegreen
ST is AtLoad #gray

@LR
0 is Lowering
100 is Lowered #pink
350 is Releasing

@ST
200 is Moving
@enduml

```





[Ref. QA-5776]

10.20 グローバルスタイルの使用

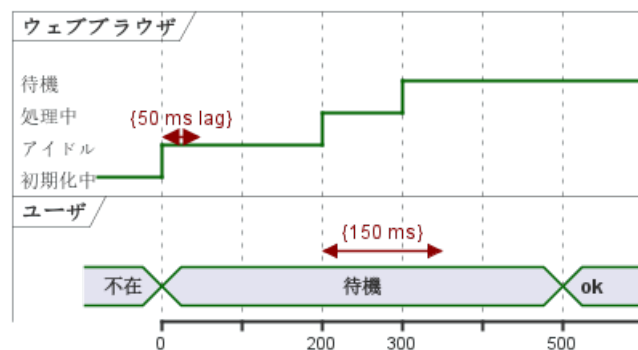
10.20.1 スタイル無し（デフォルト）

```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

WB is 初期化中
WU is 不在

```
@WB
0 is アイドル
+200 is 処理中
+100 is 待機
WB@0 <-> @50 : {50 ms lag}
```

```
@WU
0 is 待機
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```



10.20.2 スタイル有り

スタイルを使用して要素の表示方法を変更することができます。

```
@startuml
<style>
timingDiagram {
  document {
    BackGroundColor SandyBrown
  }
  constraintArrow {
    LineStyle 2-1
    LineThickness 3
    LineColor Blue
  }
}
```



```

}
}
</style>
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU

```

WB is 初期化中

WU is 不在

@WB

0 is アイドル

+200 is 処理中

+100 is 待機

WB@0 <-> @50 : {50 ms lag}

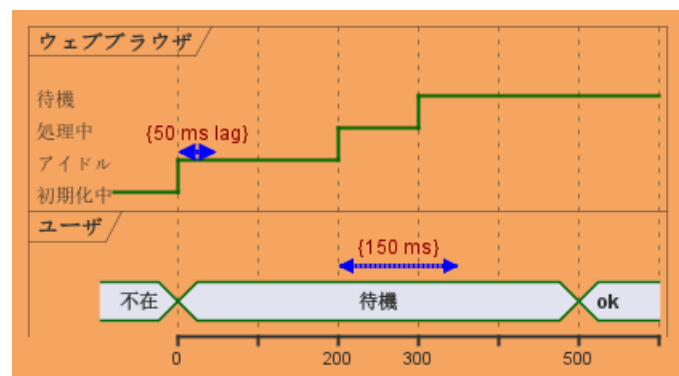
@WU

0 is 待機

+500 is ok

@200 <-> @+150 : {150 ms}

@enduml



[Ref. QA-14340]

10.21 Applying Colors to specific lines

You can use the <style> tags and stereotyping to give a name to line attributes.

```

@startuml
<style>
timingDiagram {
  .red {
    LineColor red
  }
  .blue {
    LineColor blue
    LineThickness 5
  }
}
</style>

clock clk with period 1
binary "Input Signal 1" as IS1
binary "Input Signal 2" as IS2 <<blue>>
binary "Output Signal 1" as OS1 <<red>>

```

@0

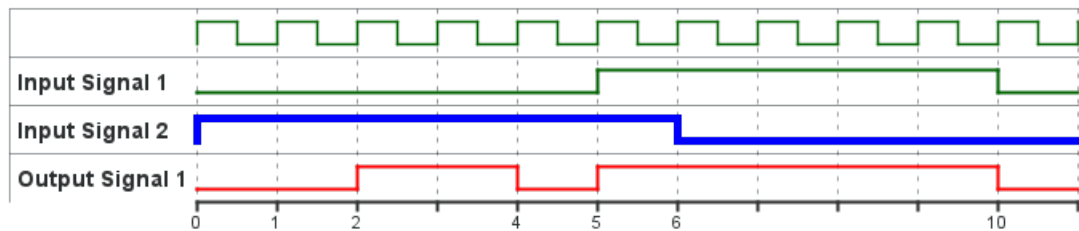
IS1 is low



```

IS2 is high
OS1 is low
@2
OS1 is high
@4
OS1 is low
@5
IS1 is high
OS1 is high
@6
IS2 is low
@10
IS1 is low
OS1 is low
@enduml

```



[Ref. QA-15870]

10.22 Compact mode

You can use `compact` command to compact the timing layout.

10.22.1 By default

```

@startuml
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2

@0
WU is Waiting
WB is Idle
WB2 is Idle

@200
WB is Proc.

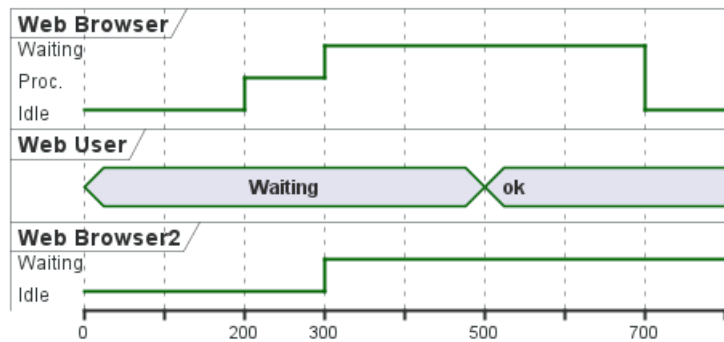
@300
WB is Waiting
WB2 is Waiting

@500
WU is ok

@700
WB is Idle
@enduml

```





10.22.2 Global mode with mode compact

```
@startuml
mode compact
robust "Web Browser" as WB
concise "Web User" as WU
robust "Web Browser2" as WB2
```

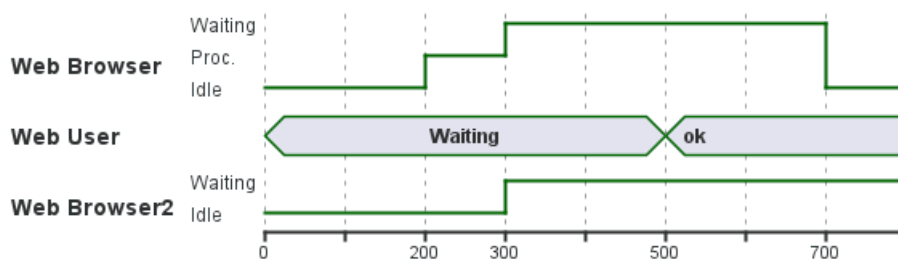
```
@0
WU is Waiting
WB is Idle
WB2 is Idle
```

```
@200
WB is Proc.
```

```
@300
WB is Waiting
WB2 is Waiting
```

```
@500
WU is ok
```

```
@700
WB is Idle
@enduml
```



10.22.3 Local mode with only compact on element

```
@startuml
compact robust "Web Browser" as WB
compact concise "Web User" as WU
robust "Web Browser2" as WB2
```

```
@0
WU is Waiting
WB is Idle
```



WB2 is Idle

@200

WB is Proc.

@300

WB is Waiting

WB2 is Waiting

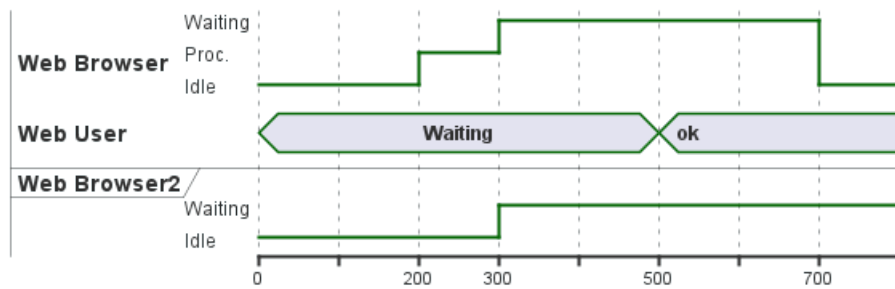
@500

WU is ok

@700

WB is Idle

@enduml



[Ref. QA-11130]

11 JSON データを表示する

JSON 形式は、様々なソフトウェアで使われています。

PlantUML を使って JSON データを可視化することができます。

この機能を使うには、

- `@startjson` キーワードで開始し、
- `@endjson` キーワードで終了する必要があります。

```
@startjson
{
  "fruit": "Apple",
  "size": "Large",
  "color": "Red"
}
@endjson
```

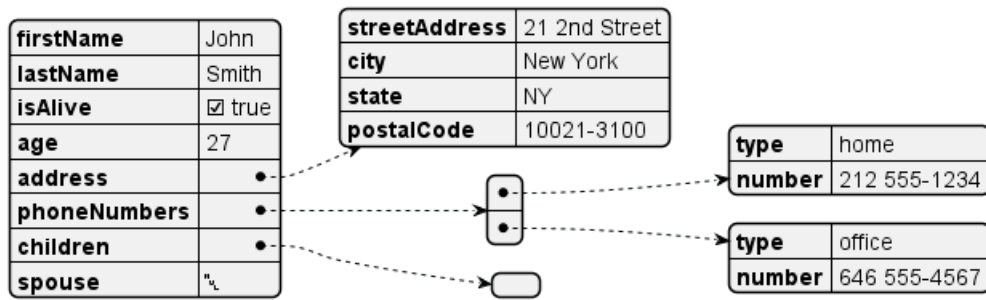
fruit	Apple
size	Large
color	Red

11.1 複雑な例

複雑な JSON 構造を使用することができます。

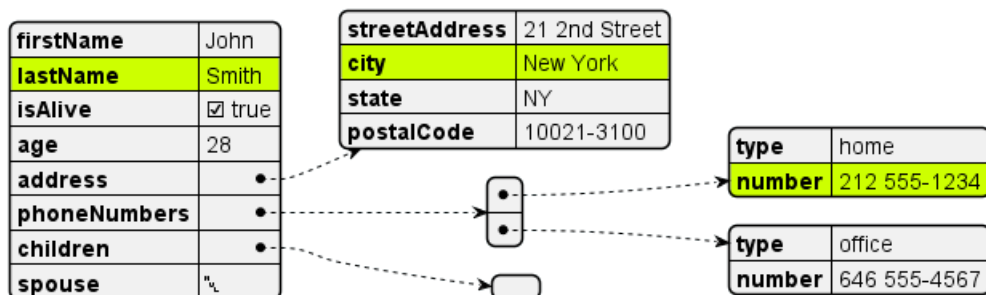
```
@startjson
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```





11.2 一部をハイライトする

```
@startjson
#highlight "lastName"
#highlight "address" / "city"
#highlight "phoneNumbers" / "0" / "number"
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 28,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson
```



11.3 Using different styles for highlight

It is possible to have different styles for different highlights.

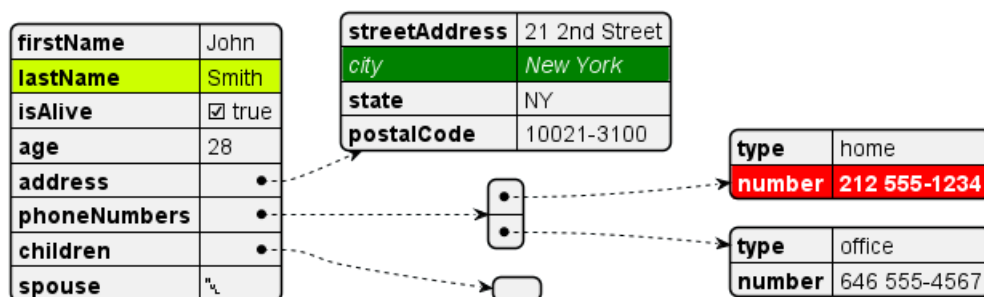
```
@startjson
```



```

<style>
.h1 {
  BackGroundColor green
  FontColor white
  FontStyle italic
}
.h2 {
  BackGroundColor red
  FontColor white
  FontStyle bold
}
</style>
#highlight "lastName"
#highlight "address" / "city" <<h1>>
#highlight "phoneNumbers" / "0" / "number" <<h2>>
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 28,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
@endjson

```



[Ref. QA-15756, GH-1393]

11.4 JSON の基本要素

11.4.1 すべての JSON 基本要素の例

```

@startjson
{

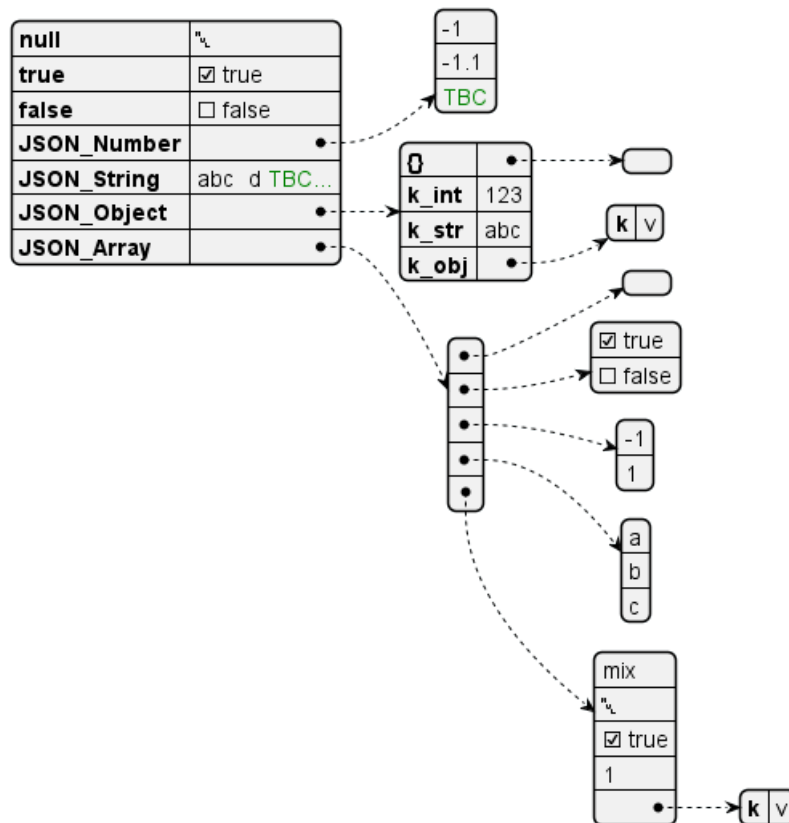
```



```

"null": null,
"true": true,
"false": false,
"JSON_Number": [-1, -1.1, "<color:green>TBC"],
"JSON_String": "a\\nb\\rc\\td <color:green>TBC...",
"JSON_Object": {
  "{}": {},
  "k_int": 123,
  "k_str": "abc",
  "k_obj": {"k": "v"}
},
"JSON_Array" : [
  [],
  [true, false],
  [-1, 1],
  ["a", "b", "c"],
  ["mix", null, true, 1, {"k": "v"}]
]
}
@endjson

```



11.5 JSON 配列またはテーブル

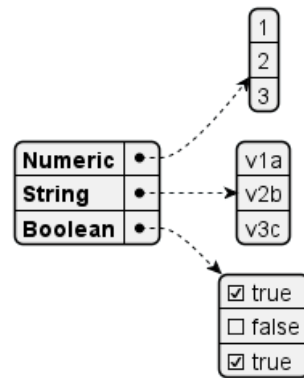
11.5.1 配列型

```

@startjson
{
  "Numeric": [1, 2, 3],
  "String ": ["v1a", "v2b", "v3c"],
  "Boolean": [true, false, true]
}
@endjson

```

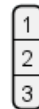




11.5.2 シンプルな配列またはテーブル

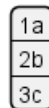
11.5.3 Number 配列

```
@startjson
[1, 2, 3]
@endjson
```



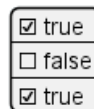
11.5.4 String 配列

```
@startjson
["1a", "2b", "3c"]
@endjson
```



11.5.5 Boolean 配列

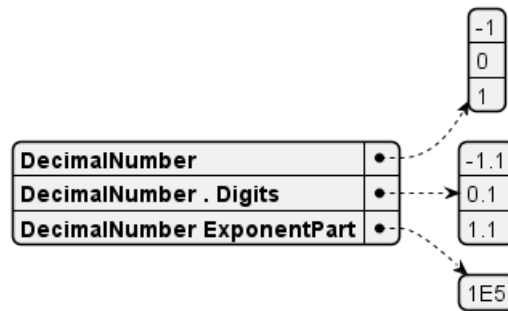
```
@startjson
[true, false, true]
@endjson
```



11.6 number 型

```
@startjson
{
  "DecimalNumber": [-1, 0, 1],
  "DecimalNumber . Digits": [-1.1, 0.1, 1.1],
  "DecimalNumber ExponentPart": [1E5]
}
@endjson
```





11.7 string 型

11.7.1 Unicode

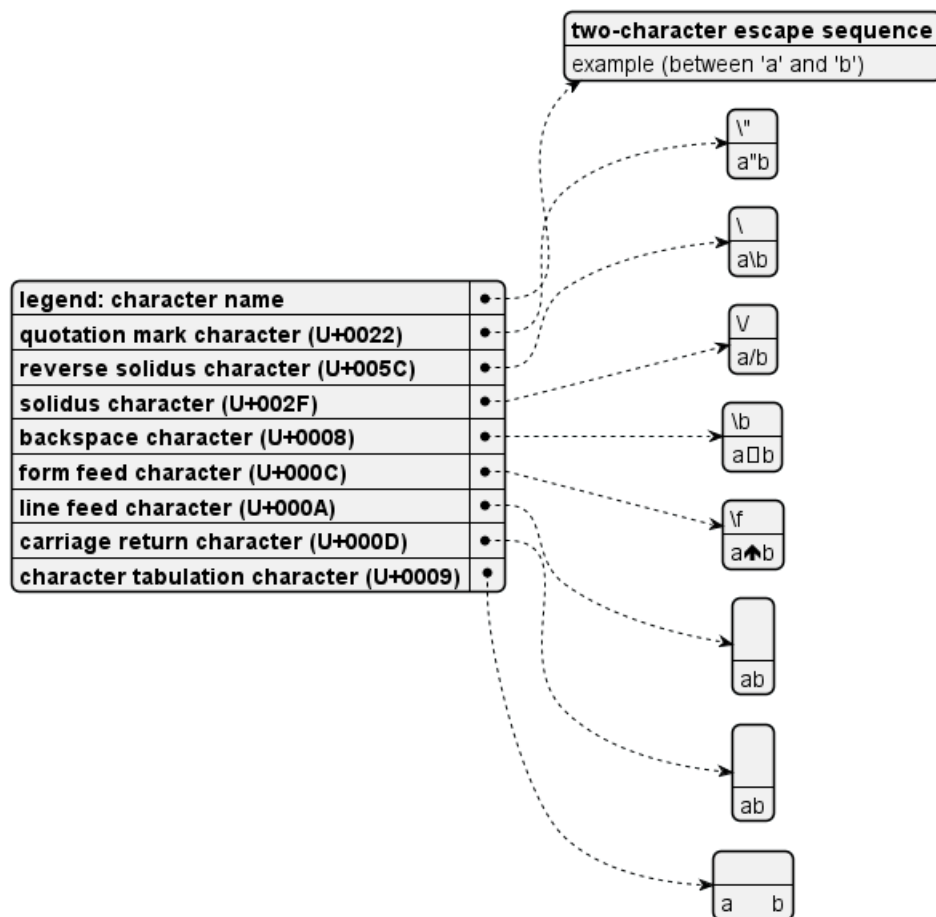
JSON では Unicode を直接記述するか、のような形式でエスケープして記述することができます。

```
@startjson
{
  "<color:blue><b>code": "<color:blue><b>value",
  "a\\u005Cb": "a\\u005Cb",
  "\\uD83D\\uDE10": "\\uD83D\\uDE10",
  " ": " "
}
@endjson
```

code	value
a\\u005Cb	a\b
\\uD83D\\uDE10	😄

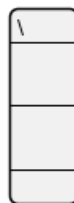
11.7.2 2 文字のエスケープシーケンス

```
@startjson
{
  "**legend**: character name": ["**two-character escape sequence**", "example (between
  "quotation mark character (U+0022)": ["\\\"", "a\\b"],
  "reverse solidus character (U+005C)": ["\\\\", "a\\b"],
  "solidus character (U+002F)": ["\\/", "a\\/b"],
  "backspace character (U+0008)": ["\\b", "a\\bb"],
  "form feed character (U+000C)": ["\\f", "a\\fb"],
  "line feed character (U+000A)": ["\\n", "a\\nb"],
  "carriage return character (U+000D)": ["\\r", "a\\rb"],
  "character tabulation character (U+0009)": ["\\t", "a\\tb"]
}
@endjson
```



TODO: FIXME FIXME or not , on the same item as management in PlantUML See Report Bug on QA-13066 **TODO:** FIXME

```
@startjson
[
  "\\\\",
  "\\n",
  "\\r",
  "\\t"
]
@endjson
```



11.8 最小の JSON の例

```
@startjson
"Hello world!"
@endjson
```

Hello world!



```
@startjson
42
@endjson
```



```
@startjson
true
@endjson
```



(Examples come from STD 90 - Examples)

11.9 Empty table or list

```
@startjson
{
  "empty_tab": [],
  "empty_list": {}
}
@endjson
```

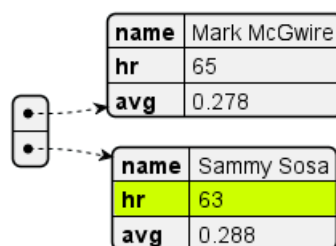


[Ref. QA-14397]

11.10 スタイルを使用する

11.10.1 スタイル無し（デフォルト）

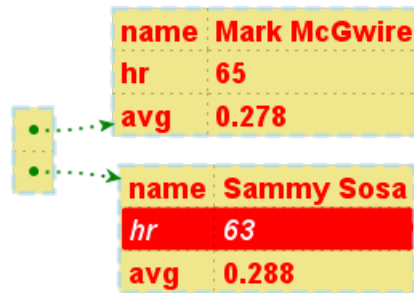
```
@startjson
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
  },
  {
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
  }
]
@endjson
```



11.10.2 スタイル有り

スタイルを指定して、要素の見た目を変更することができます。

```
@startjson
<style>
jsonDiagram {
  node {
    BackGroundColor Khaki
    LineColor lightblue
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    RoundCorner 0
    LineThickness 2
    LineStyle 10-5
    separator {
      LineThickness 0.5
      LineColor black
      LineStyle 1-5
    }
  }
  arrow {
    BackGroundColor lightblue
    LineColor green
    LineThickness 2
    LineStyle 2-5
  }
  highlight {
    BackGroundColor red
    FontColor white
    FontStyle italic
  }
}
</style>
#highlight "1" / "hr"
[
  {
    "name": "Mark McGwire",
    "hr": 65,
    "avg": 0.278
  },
  {
    "name": "Sammy Sosa",
    "hr": 63,
    "avg": 0.288
  }
]
@endjson
```



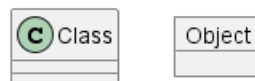
[Adapted from QA-13123 and QA-13288]

11.11 Display JSON Data on Class or Object diagram

11.11.1 Simple example

```

@startuml
class Class
object Object
json JSON {
  "fruit": "Apple",
  "size": "Large",
  "color": ["Red", "Green"]
}
@enduml
  
```



JSON	
fruit	Apple
size	Large
color	Red
	Green

[Ref. QA-15481]

11.11.2 Complex example: with all JSON basic element

```

@startuml
json "<b>JSON basic element" as J {
  "null": null,
  "true": true,
  "false": false,
  "JSON_Number": [-1, -1.1, "<color:green>TBC"],
  "JSON_String": "a\nb\rc\td <color:green>TBC...",
  "JSON_Object": {
    "{}": {},
    "k_int": 123,
    "k_str": "abc",
    "k_obj": {"k": "v"}
  },
  "JSON_Array" : [
    [],
    [true, false],
    [-1, 1],
  ]
}
  
```



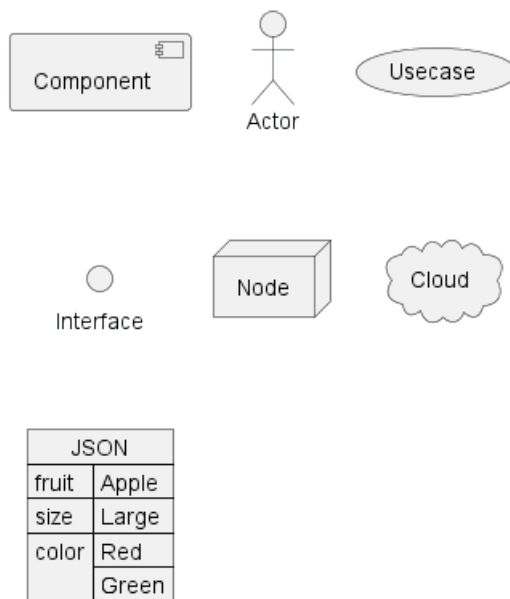
```
["a", "b", "c"],  
["mix", null, true, 1, {"k": "v"}]  
]  
}  
@enduml
```

JSON basic element		
null	null	
true	true	
false	false	
JSON_Number	-1	
	-1.1	
	TBC	
JSON_String	abc d TBC...	
JSON_Object	{}	
	k_int	123
	k_str	abc
	k_obj	k v
JSON_Array	true	
	false	
	-1	
	1	
	a	
	b	
	c	
	mix	
	null	
	true	
	1	
	k v	

11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram

11.12.1 Simple example

```
@startuml  
allowmixing  
  
component Component  
actor Actor  
usecase Usecase  
( ) Interface  
node Node  
cloud Cloud  
  
json JSON {  
    "fruit": "Apple",  
    "size": "Large",  
    "color": ["Red", "Green"]  
}  
@enduml
```



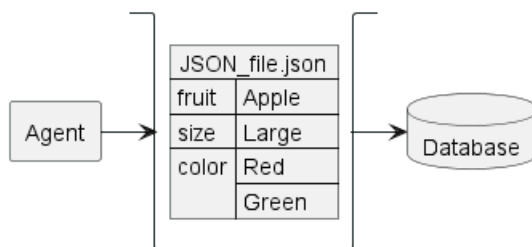
[Ref. QA-15481]

Complex example: with arrow

```
@startuml
allowmixing

agent Agent
stack {
    json "JSON_file.json" as J {
        "fruit":"Apple",
        "size":"Large",
        "color": ["Red", "Green"]
    }
}
database Database

Agent -> J
J -> Database
@enduml
```

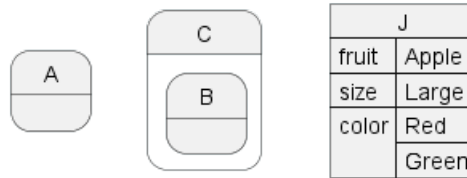


11.13 Display JSON Data on State diagram

11.13.1 Simple example

```
@startuml
state "A" as stateA
state "C" as stateC {
    state B
}
stateA --> stateC
```

```
json J {  
    "fruit": "Apple",  
    "size": "Large",  
    "color": ["Red", "Green"]  
}  
@enduml
```



[Ref. QA-17275]

12 YAML データを表示する

YAML 形式は、様々なソフトウェアで使われています。

PlantUML を使って YAML データを可視化することができます。

この機能を使うには、

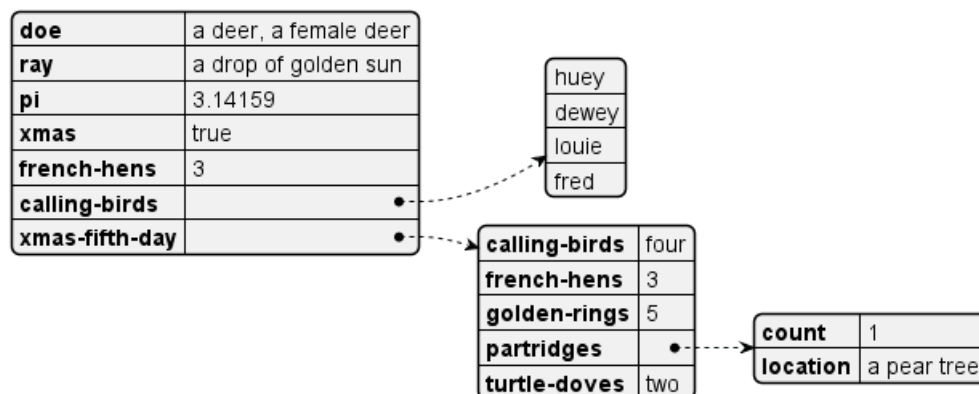
- @startyaml キーワードで開始し、
- @endyaml キーワードで終了する必要があります。

```
@startyaml
fruit: Apple
size: Large
color: Red
@endyaml
```

fruit	Apple
size	Large
color	Red

12.1 複雑な例

```
@startyaml
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



12.2 特定のキー（記号と Unicode の使用）

```
@startyaml
@fruit: Apple
$size: Large
&color: Red
: Heart
%: Per mille
@endyaml
```

@fruit	Apple
\$size	Large
&color	Red
♥	Heart
%	Per mille

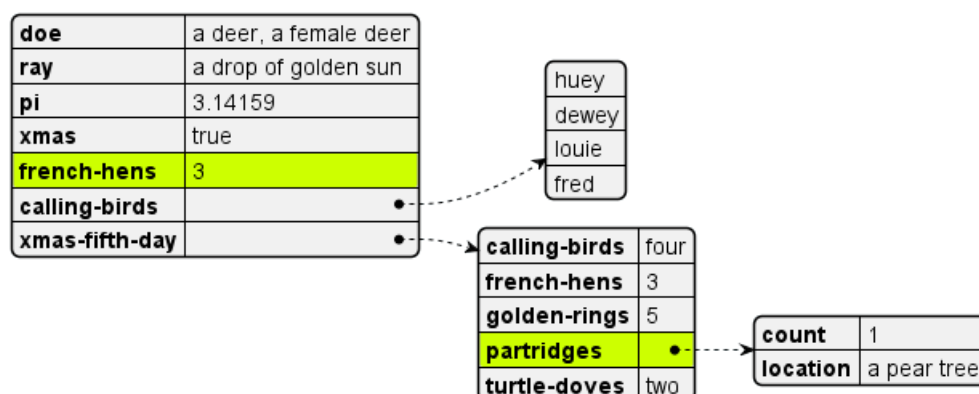
[Ref. QA-13376]

12.3 一部をハイライトする

12.3.1 通常スタイル

```
@startyaml
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"
```

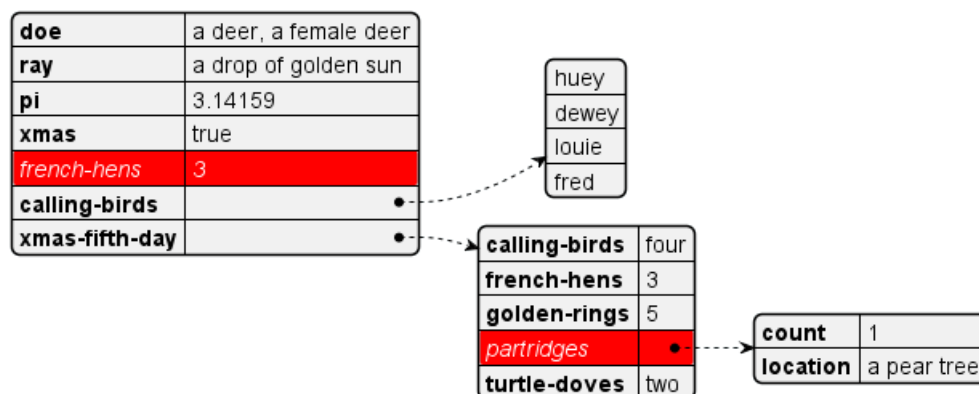
```
doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



12.3.2 カスタムスタイル

```
@startyaml
<style>
yamlDiagram {
  highlight {
    BackGroundColor red
    FontColor white
    FontStyle italic
  }
}
</style>
#highlight "french-hens"
#highlight "xmas-fifth-day" / "partridges"

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml
```



[Ref. QA-13288]

12.4 Using different styles for highlight

It is possible to have different styles for different highlights.

```
@startyaml
<style>
.h1 {
  BackGroundColor green
  FontColor white
}
```

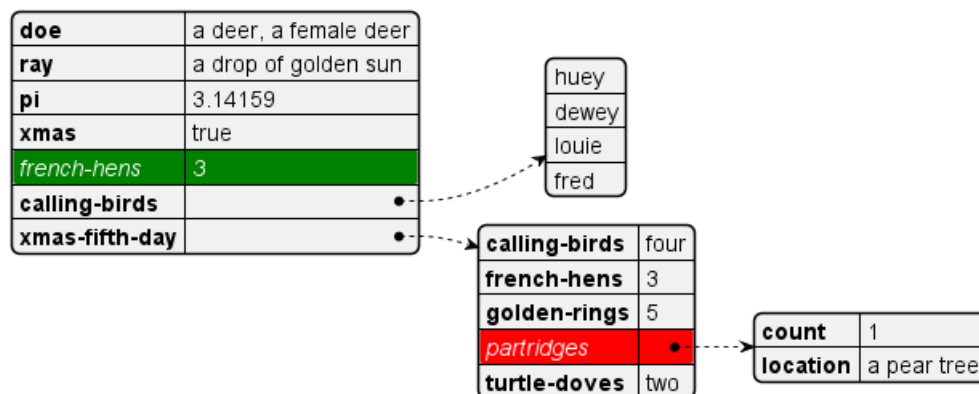


```

    FontStyle italic
  }
  .h2 {
    BackGroundColor red
    FontColor white
    FontStyle italic
  }
</style>
#highlight "french-hens" <<h1>>
#highlight "xmas-fifth-day" / "partridges" <<h2>>

doe: "a deer, a female deer"
ray: "a drop of golden sun"
pi: 3.14159
xmas: true
french-hens: 3
calling-birds:
- huey
- dewey
- louie
- fred
xmas-fifth-day:
calling-birds: four
french-hens: 3
golden-rings: 5
partridges:
count: 1
location: "a pear tree"
turtle-doves: two
@endyaml

```



[Ref. QA-15756, GH-1393]

12.5 グローバルなスタイル指定

12.5.1 スタイル無し (デフォルト)

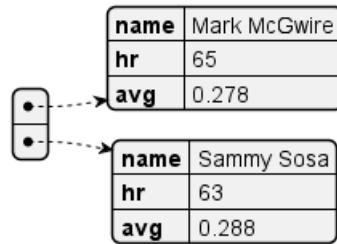
```

@startyaml
-
  name: Mark McGwire
  hr: 65
  avg: 0.278
-
  name: Sammy Sosa
  hr: 63

```



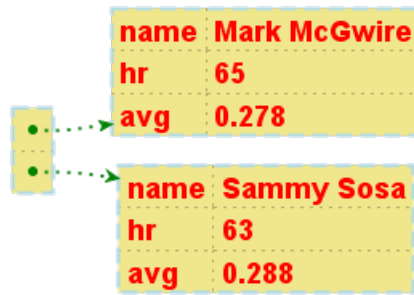
```
    avg: 0.288
@endyaml
```



12.5.2 スタイル有り

スタイルを使って、要素の描画方法を変更することができます。

```
@startyaml
<style>
yamlDiagram {
  node {
    BackGroundColor lightblue
    LineColor lightblue
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor Khaki
    RoundCorner 0
    LineThickness 2
    LineStyle 10-5
    separator {
      LineThickness 0.5
      LineColor black
      LineStyle 1-5
    }
  }
  arrow {
    BackGroundColor lightblue
    LineColor green
    LineThickness 2
    LineStyle 2-5
  }
}
</style>
-
  name: Mark McGwire
  hr: 65
  avg: 0.278
-
  name: Sammy Sosa
  hr: 63
  avg: 0.288
@endyaml
```



[Ref. QA-13123]

13 ネットワーク図 (nwdiag)

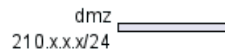
nwdiag は、小宮健 (Takeshi KOMIYA) によって作られた、ネットワーク図を生成するためのツールです。

その文法はわかりやすくシンプルなので、PlantUML にも組み込まれました。以下に、公式サイトサンプルと同等の図を載せています。

13.1 シンプルなネットワーク図

13.1.1 ネットワークの定義

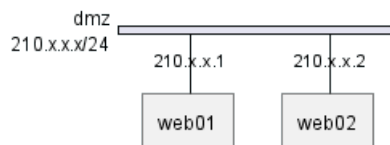
```
@startuml
nwdiag {
  network dmz {
    address = "210.x.x.x/24"
  }
}
@enduml
```



13.1.2 ネットワーク上にいくつかの要素またはサーバーを定義する

```
@startuml
nwdiag {
  network dmz {
    address = "210.x.x.x/24"

    web01 [address = "210.x.x.1"];
    web02 [address = "210.x.x.2"];
  }
}
@enduml
```



13.1.3 完全な例

```
@startuml
nwdiag {
  network dmz {
    address = "210.x.x.x/24"

    web01 [address = "210.x.x.1"];
    web02 [address = "210.x.x.2"];
  }
  network internal {
    address = "172.x.x.x/24";

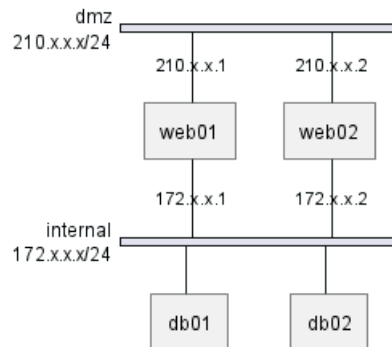
    web01 [address = "172.x.x.1"];
    web02 [address = "172.x.x.2"];
  }
}
```



```

    db01;
    db02;
  }
}
@enduml

```



13.2 複数のアドレスを定義するケース

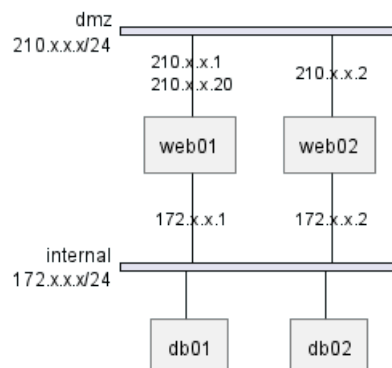
```

@startuml
nwdiag {
  network dmz {
    address = "210.x.x.x/24"

    // set multiple addresses (using comma)
    web01 [address = "210.x.x.1, 210.x.x.20"];
    web02 [address = "210.x.x.2"];
  }
  network internal {
    address = "172.x.x.x/24";

    web01 [address = "172.x.x.1"];
    web02 [address = "172.x.x.2"];
    db01;
    db02;
  }
}
@enduml

```



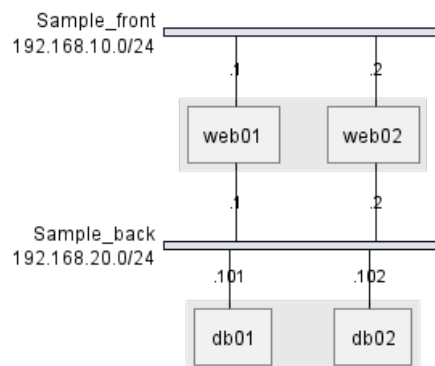
13.3 ノードのグルーピング

13.3.1 ネットワーク定義の中でグループを定義

```
@startuml
nwdiag {
  network Sample_front {
    address = "192.168.10.0/24";

    // define group
    group web {
      web01 [address = ".1"];
      web02 [address = ".2"];
    }
  }
  network Sample_back {
    address = "192.168.20.0/24";
    web01 [address = ".1"];
    web02 [address = ".2"];
    db01 [address = ".101"];
    db02 [address = ".102"];

    // define network using defined nodes
    group db {
      db01;
      db02;
    }
  }
}
@enduml
```



13.3.2 ネットワーク定義の外でグループを定義

```
@startuml
nwdiag {
  // define group outside of network definitions
  group {
    color = "#FFAAAA";

    web01;
    web02;
    db01;
  }

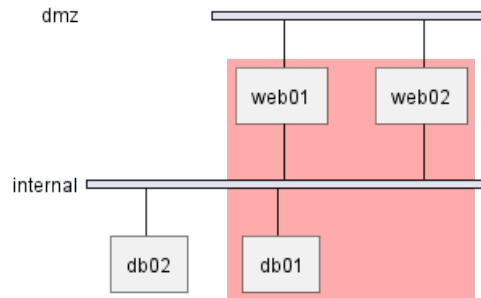
  network dmz {
```



```

    web01;
    web02;
}
network internal {
    web01;
    web02;
    db01;
    db02;
}
}
@enduml

```



13.3.3 一つのネットワークに複数のグループを定義

13.3.4 グループが2つの例

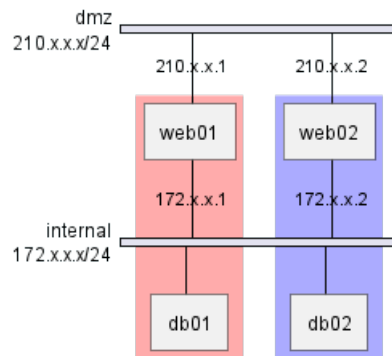
```

@startuml
nwdiag {
    group {
        color = "#FFaaaa";
        web01;
        db01;
    }
    group {
        color = "#aaaaFF";
        web02;
        db02;
    }
    network dmz {
        address = "210.x.x.x/24"

        web01 [address = "210.x.x.1"];
        web02 [address = "210.x.x.2"];
    }
    network internal {
        address = "172.x.x.x/24";

        web01 [address = "172.x.x.1"];
        web02 [address = "172.x.x.2"];
        db01 ;
        db02 ;
    }
}
@enduml

```



[Ref. QA-12663]

13.3.5 グループが3つの例

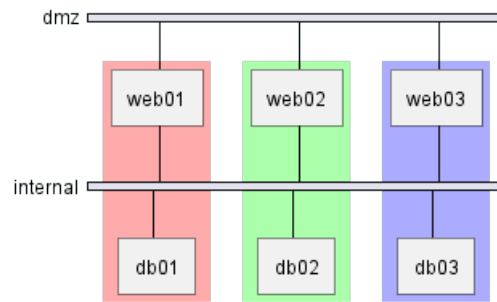
```

@startuml
nwdiag {
  group {
    color = "#FFaaaa";
    web01;
    db01;
  }
  group {
    color = "#aaFFaa";
    web02;
    db02;
  }
  group {
    color = "#aaaaFF";
    web03;
    db03;
  }
}

network dmz {
  web01;
  web02;
  web03;
}

network internal {
  web01;
  db01 ;
  web02;
  db02 ;
  web03;
  db03;
}
}
@enduml

```



[Ref. QA-13138]

13.4 ネットワークとグループに対する拡張文法

13.4.1 ネットワーク

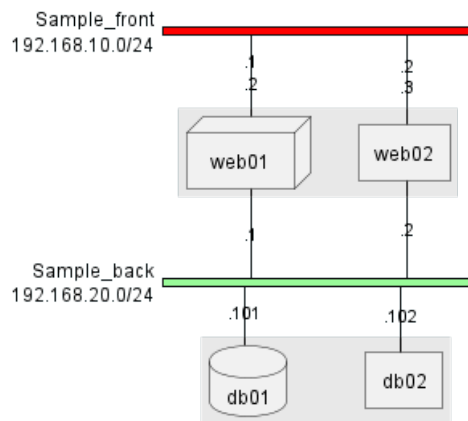
ネットワーク、ネットワーク要素に対して、次の項目が設定可能です：

- アドレス (コンマ, 区切り)
- 色
- 説明
- 形状

```
@startuml
nwdiag {
  network Sample_front {
    address = "192.168.10.0/24"
    color = "red"

    // define group
    group web {
      web01 [address = ".1, .2", shape = "node"]
      web02 [address = ".2, .3"]
    }
  }
  network Sample_back {
    address = "192.168.20.0/24"
    color = "palegreen"
    web01 [address = ".1"]
    web02 [address = ".2"]
    db01 [address = ".101", shape = database ]
    db02 [address = ".102"]

    // define network using defined nodes
    group db {
      db01;
      db02;
    }
  }
}
@enduml
```



13.4.2 グループ

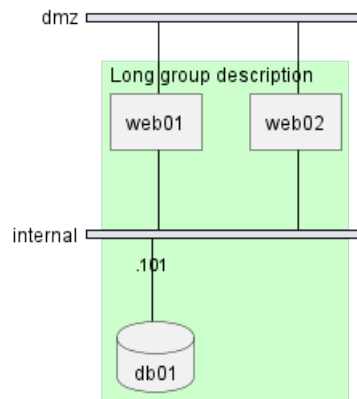
グループに対して、次の項目が設定可能です：

- 色
- 説明

```
@startuml
nwdiag {
  group {
    color = "#CCFFCC";
    description = "Long group description";

    web01;
    web02;
    db01;
  }

  network dmz {
    web01;
    web02;
  }
  network internal {
    web01;
    web02;
    db01 [address = ".101", shape = database];
  }
}
@enduml
```



[Ref. QA-12056]

13.5 スプライトの使用

標準ライブラリやその他のライブラリに含まれる、あらゆるスプライト（アイコン）を使用できます。スプライトは `<$sprite>` の記法を使います。で改行できます。また、その他の Creole 記法も使用できます。

```

@startuml
!include <office/Servers/application_server>
!include <office/Servers/database_server>

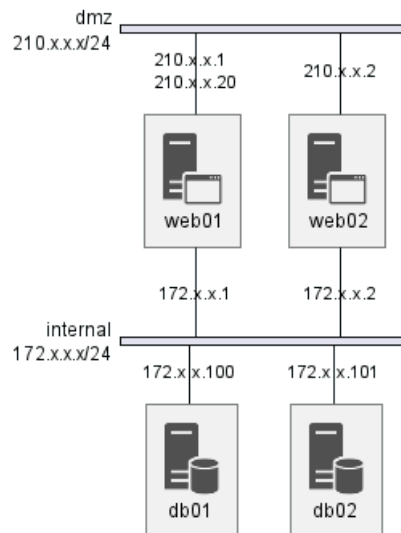
nwdiag {
  network dmz {
    address = "210.x.x.x/24"

    // set multiple addresses (using comma)
    web01 [address = "210.x.x.1, 210.x.x.20", description = "<$application_server>\n web01"]
    web02 [address = "210.x.x.2", description = "<$application_server>\n web02"];
  }
  network internal {
    address = "172.x.x.x/24";

    web01 [address = "172.x.x.1"];
    web02 [address = "172.x.x.2"];
    db01 [address = "172.x.x.100", description = "<$database_server>\n db01"];
    db02 [address = "172.x.x.101", description = "<$database_server>\n db02"];
  }
}
@enduml

```





[Ref. QA-11862]

13.6 OpenIconic の使用

ネットワークまたはノードの説明で、OpenIconic のアイコンを使用することもできます。

<&icon> の記法でアイコンを表示します。<&icon*n> でサイズを n 倍にします。で改行できます：

@startuml

```

nwdiag {
  group nightly {
    color = "#FFAAAA";
    description = "<&clock> Restarted nightly <&clock>";
    web02;
    db01;
  }
  network dmz {
    address = "210.x.x.x/24"

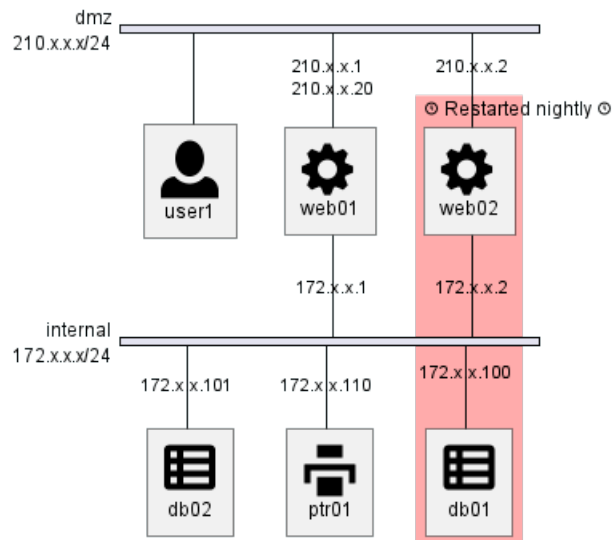
    user [description = "<&person*4.5>\n user1"];
    // set multiple addresses (using comma)
    web01 [address = "210.x.x.1, 210.x.x.20", description = "<&cog*4>\nweb01"];
    web02 [address = "210.x.x.2", description = "<&cog*4>\nweb02"];

  }
  network internal {
    address = "172.x.x.x/24";

    web01 [address = "172.x.x.1"];
    web02 [address = "172.x.x.2"];
    db01 [address = "172.x.x.100", description = "<&spreadsheet*4>\n db01"];
    db02 [address = "172.x.x.101", description = "<&spreadsheet*4>\n db02"];
    ptr [address = "172.x.x.110", description = "<&print*4>\n ptr01"];
  }
}
@enduml

```





13.7 複数のネットワークに一つのノードを定義

異なる二つ以上のネットワークに同一のノードを使用することができます。この場合、*nwdiag* ではネットワークの上をジャンプする線が描画されます。

```

@startuml
nwdiag {
    // define group at outside network definitions
    group {
        color = "#7777FF";

        web01;
        web02;
        db01;
    }

    network dmz {
        color = "pink"

        web01;
        web02;
    }

    network internal {
        web01;
        web02;
        db01 [shape = database ];
    }

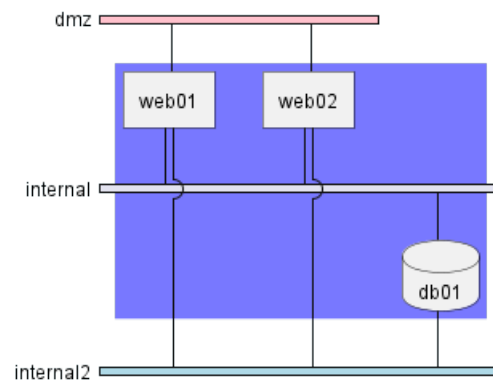
    network internal2 {
        color = "LightBlue";

        web01;
        web02;
        db01;
    }
}

```



@enduml



13.8 ピア接続

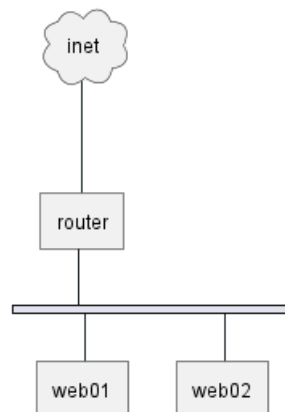
ピア接続は、二つのノード間を単純につなぐものです。この場合は、横長の「バス線」ネットワークは使用されません。

```

@startuml
nwdiag {
    inet [shape = cloud];
    inet -- router;

    network {
        router;
        web01;
        web02;
    }
}
@enduml

```



13.9 ピア接続とグループ

13.9.1 グループ無し

```

@startuml
nwdiag {
    internet [ shape = cloud];

```

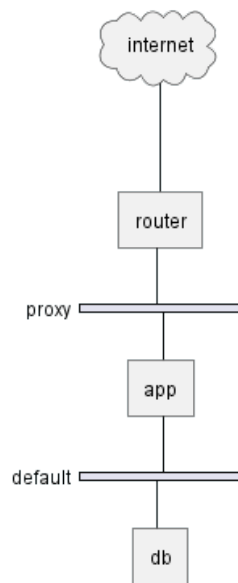


```

internet -- router;

network proxy {
    router;
    app;
}
network default {
    app;
    db;
}
}
@enduml

```



13.9.2 1 番目にグループを記述

```

@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    group {
        color = "pink";
        app;
        db;
    }

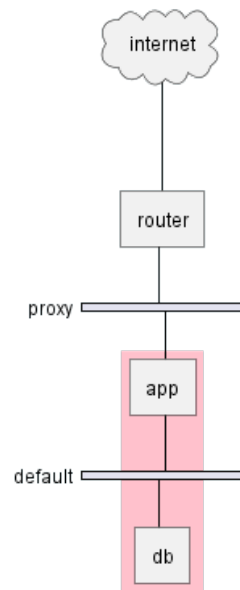
    network proxy {
        router;
        app;
    }

    network default {
        app;
        db;
    }
}

```



@enduml



13.9.3 2 番目にグループを記述

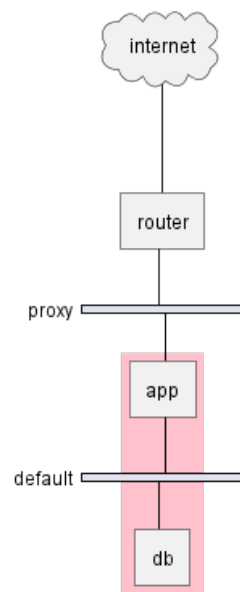
```

@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }

    group {
        color = "pink";
        app;
        db;
    }

    network default {
        app;
        db;
    }
}
@enduml
  
```

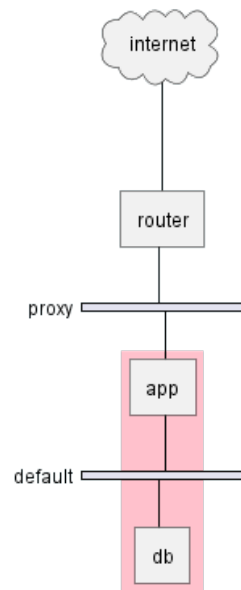


TODO: FIXME proxy から'db' へ線が引かれてしまいます。('db' は'default network' にしかつながつていないはずです。) [グループ無しの例を参照]

13.9.4 3 番目にグループを記述

```
@startuml
nwdiag {
    internet [ shape = cloud];
    internet -- router;

    network proxy {
        router;
        app;
    }
    network default {
        app;
        db;
    }
    group {
        color = "pink";
        app;
        db;
    }
}
@enduml
```



TODO: FIXME [Ref. Issue#408 and QA-12655] **TODO:** Not totally fixed

13.10 ネットワーク図にタイトル、ヘッダ、フッタ、キャプション、凡例を追加する

```
@startuml
```

```
header some header
```

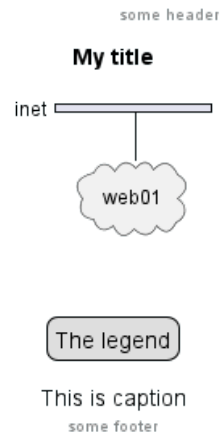
```
footer some footer
```

```
title My title
```

```
nwdiag {
  network inet {
    web01 [shape = cloud]
  }
}
```

```
legend
The legend
end legend
```

```
caption This is caption
@enduml
```

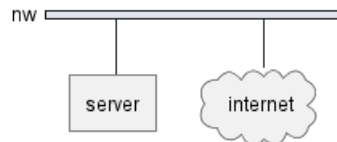


[Ref. QA-11303 and Common commands]

13.11 影の有無

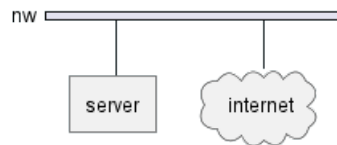
13.11.1 影有り (デフォルト)

```
@startuml
nwdiag {
  network nw {
    server;
    internet;
  }
  internet [shape = cloud];
}
@enduml
```



13.11.2 影無し

```
@startuml
<style>
root {
  shadowing 0
}
</style>
nwdiag {
  network nw {
    server;
    internet;
  }
  internet [shape = cloud];
}
@enduml
```



[Ref. QA-14516]

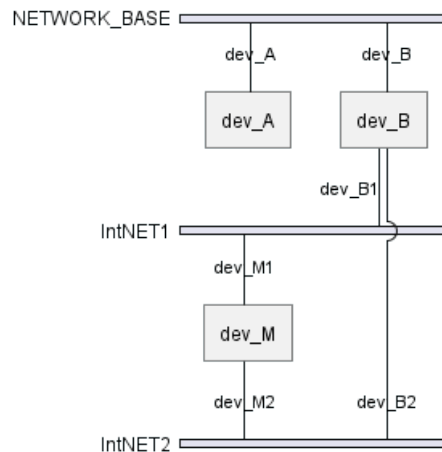
13.12 ネットワークの幅の変更

ネットワークの幅を変更することができます。一部の（もしくはすべての）ネットワークの幅を full（最大幅）に設定して揃えることができます。

可能な組合せの例を示します：

- 無し

```
@startuml
nwdiag {
  network NETWORK_BASE {
    dev_A [address = "dev_A" ]
    dev_B [address = "dev_B" ]
  }
  network IntNET1 {
    dev_B [address = "dev_B1" ]
    dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
    dev_B [address = "dev_B2" ]
    dev_M [address = "dev_M2" ]
  }
}
@enduml
```



- 1 番目のみ

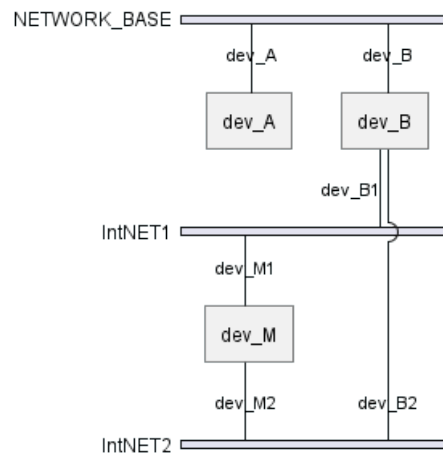
```
@startuml
nwdiag {
  network NETWORK_BASE {
    width = full
    dev_A [address = "dev_A" ]
    dev_B [address = "dev_B" ]
  }
}
```



```

}
network IntNET1 {
  dev_B [address = "dev_B1" ]
  dev_M [address = "dev_M1" ]
}
network IntNET2 {
  dev_B [address = "dev_B2" ]
  dev_M [address = "dev_M2" ]
}
}
}
@enduml

```

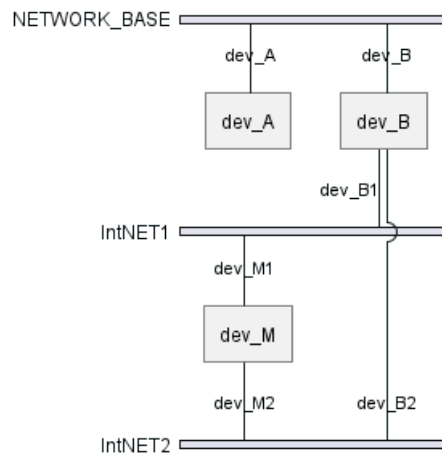


- 1 番目と 2 番目

```

@startuml
nwdiag {
  network NETWORK_BASE {
    width = full
    dev_A [address = "dev_A" ]
    dev_B [address = "dev_B" ]
  }
  network IntNET1 {
    width = full
    dev_B [address = "dev_B1" ]
    dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
    dev_B [address = "dev_B2" ]
    dev_M [address = "dev_M2" ]
  }
}
}
@enduml

```

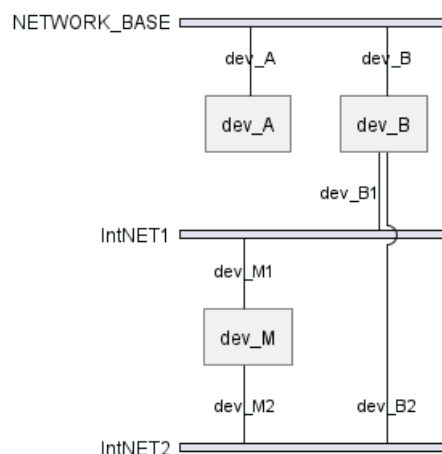


- すべてに最大幅を指定

```

@startuml
nwdiag {
  network NETWORK_BASE {
    width = full
    dev_A [address = "dev_A" ]
    dev_B [address = "dev_B" ]
  }
  network IntNET1 {
    width = full
    dev_B [address = "dev_B1" ]
    dev_M [address = "dev_M1" ]
  }
  network IntNET2 {
    width = full
    dev_B [address = "dev_B2" ]
    dev_M [address = "dev_M2" ]
  }
}
@enduml

```

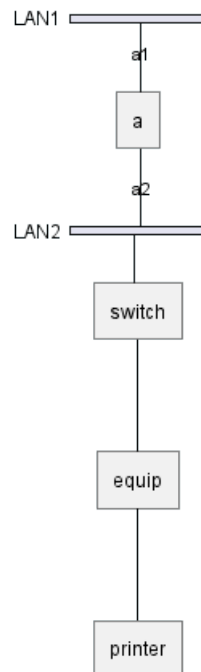


13.13 その他の内部ネットワーク

その他の内部ネットワーク (TCP/IP、USB、SERIAL...) を定義することもできます。

- アドレスまたは種類の指定無し

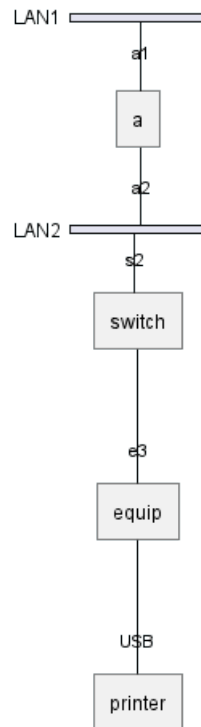
```
@startuml
nwdiag {
  network LAN1 {
    a [address = "a1"];
  }
  network LAN2 {
    a [address = "a2"];
    switch;
  }
  switch -- equip;
  equip -- printer;
}
@enduml
```



- アドレスまたは種類の指定有り

```
@startuml
nwdiag {
  network LAN1 {
    a [address = "a1"];
  }
  network LAN2 {
    a [address = "a2"];
    switch [address = "s2"];
  }
  switch -- equip;
  equip [address = "e3"];
  equip -- printer;
  printer [address = "USB"];
}
@enduml
```





[Ref. QA-12824]

13.14 グローバルスタイルの使用

13.14.1 スタイル無し (デフォルト)

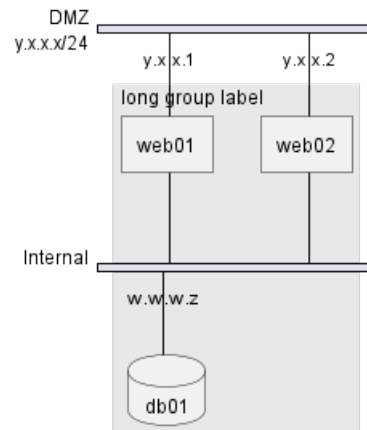
```

@startuml
nwdiag {
  network DMZ {
    address = "y.x.x.x/24"
    web01 [address = "y.x.x.x.1"];
    web02 [address = "y.x.x.x.2"];
  }

  network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
  }

  group {
    description = "long group label";
    web01;
    web02;
    db01;
  }
}
@enduml

```



13.14.2 スタイル有り

スタイルを使用して要素の見た目を変更することができます。

```

@startuml
<style>
nwdiagDiagram {
  network {
    BackGroundColor green
    LineColor red
    LineThickness 1.0
    FontSize 18
    FontColor navy
  }
  server {
    BackGroundColor pink
    LineColor yellow
    LineThickness 1.0
    ' FontXXX only for description or label
    FontSize 18
    FontColor #blue
  }
  arrow {
    ' FontXXX only for address
    FontSize 17
    FontColor #red
    FontName Monospaced
    LineColor black
  }
  group {
    BackGroundColor cadetblue
    LineColor black
    LineThickness 2.0
    FontSize 11
    FontStyle bold
    Margin 5
    Padding 5
  }
}
</style>
nwdiag {

```



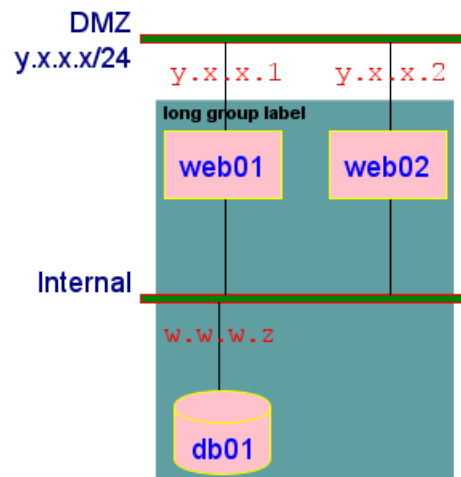
```

network DMZ {
    address = "y.x.x.x/24"
    web01 [address = "y.x.x.1"];
    web02 [address = "y.x.x.2"];
}

network Internal {
    web01;
    web02;
    db01 [address = "w.w.w.z", shape = database];
}

group {
    description = "long group label";
    web01;
    web02;
    db01;
}
}
@enduml

```



[Ref. QA-14479]

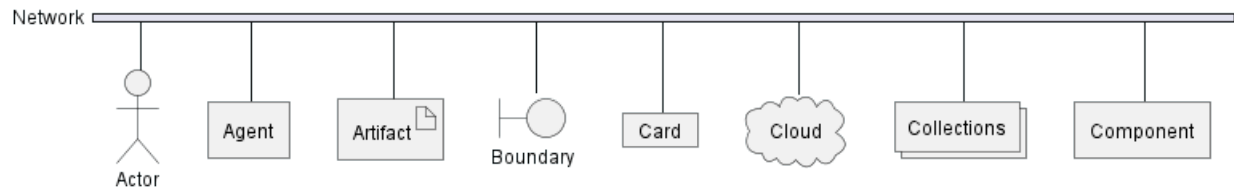
13.15 付録: ネットワーク図 (nwdiag) 上のすべての形状のテスト

```

@startuml
nwdiag {
    network Network {
        Actor      [shape = actor]
        Agent      [shape = agent]
        Artifact    [shape = artifact]
        Boundary    [shape = boundary]
        Card        [shape = card]
        Cloud       [shape = cloud]
        Collections [shape = collections]
        Component   [shape = component]
    }
}
@enduml

```

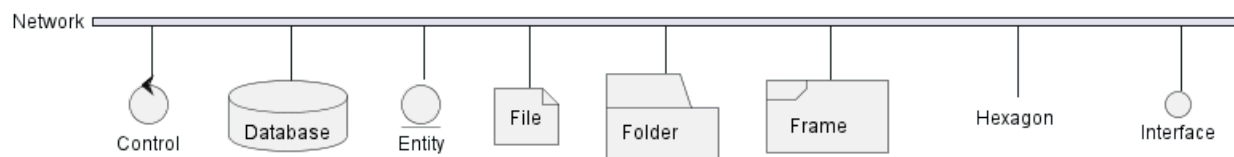




```

@startuml
nwdiag {
  network Network {
    Control      [shape = control]
    Database      [shape = database]
    Entity        [shape = entity]
    File          [shape = file]
    Folder        [shape = folder]
    Frame         [shape = frame]
    Hexagon       [shape = hexagon]
    Interface     [shape = interface]
  }
}
@enduml

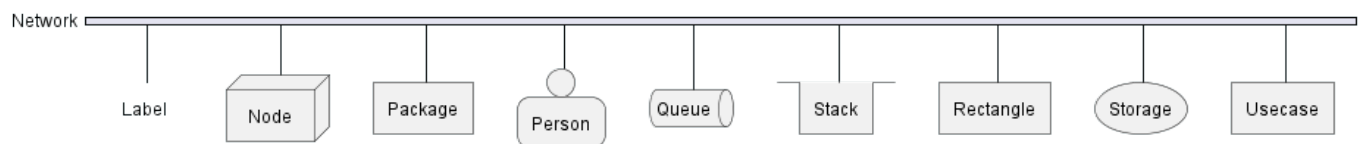
```



```

@startuml
nwdiag {
  network Network {
    Label        [shape = label]
    Node         [shape = node]
    Package      [shape = package]
    Person       [shape = person]
    Queue        [shape = queue]
    Stack        [shape = stack]
    Rectangle    [shape = rectangle]
    Storage      [shape = storage]
    Usecase      [shape = usecase]
  }
}
@enduml

```



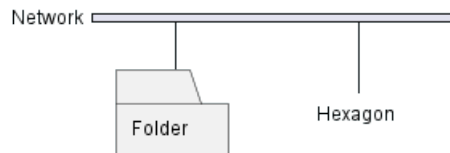
TODO: FIXME olli olli level 0 フォルダのラベルが重なって表示される olli olli level 0 六角形が表示されない olli olli



```

@startuml
nwdiag {
network Network {
Folder [shape = folder]
Hexagon [shape = hexagon]
}
}
@enduml

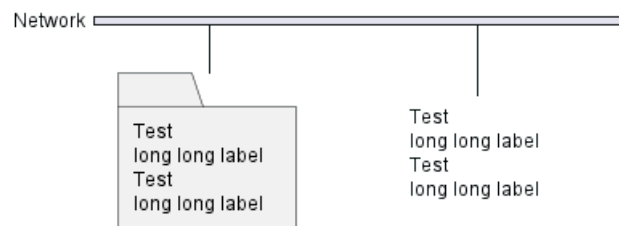
```



```

@startuml
nwdiag {
network Network {
Folder [shape = folder, description = "Test, long long label\nTest, long long label"]
Hexagon [shape = hexagon, description = "Test, long long label\nTest, long long label"]
}
}
@enduml

```



TODO: FIXME

14 Salt (ワイヤフレームによる GUI 設計ツール)

Salt はグラフィカルインタフェースの設計 (ウェブサイトワイヤフレーム、ページの大まかな設計) を助ける PlantUML のサブプロジェクトです。

このツールの用途は、簡単なサンプルのウィンドウについて議論することです。

キーワード `@startsalt`、または、`@startuml` と次の行に続くキーワード `salt` の、いずれかを使用することができます。

14.1 基本のウィジェット

ウィンドウは中括弧で始めて中括弧で閉じなければなりません。

次のように定義できます。

- ボタンは [と] で括ります。
- ラジオボタンは (と) で括ります。
- チェックボックスは [と] で括ります。
- テキスト領域は " で括ります。
- ドロップリストは ^ で括ります。

```
@startsalt
{
  Just plain text
  [This is my button]
  () Unchecked radio
  (X) Checked radio
  [] Unchecked box
  [X] Checked box
  "Enter text here  "
  ^This is a droplist^
}
@endsalt
```

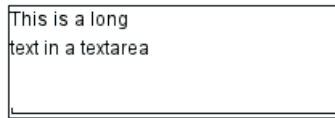


14.2 テキストエリア

次のように、テキストエリアを表現することができます:

```
@startsalt
{+
  This is a long
  text in a textarea
  .
  "
}
@endsalt
```





注意

- ドット (.) によって、縦方向のスペースを埋めています
- 最後の行のスペース (" ") によって、領域の幅を広げています

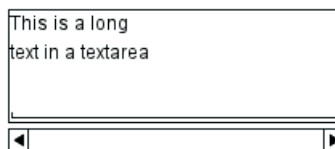
[Ref. QA-14765]

これに、スクロールバーを追加できます:

```
@startsalt
{SI
  This is a long
  text in a textarea
  .
  "
}
@endsalt
```



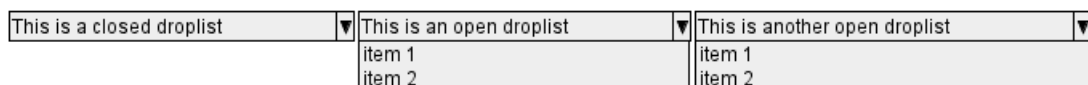
```
@startsalt
{S-
  This is a long
  text in a textarea
  .
  "
}
@endsalt
```



14.3 ドロップリストの開閉

~ で挟まれた値を追加することで、ドロップリストを開いた状態にすることができます:

```
@startsalt
{
  ^This is a closed droplist^ |
  ^This is an open droplist^^ item 1^^ item 2^ |
  ^This is another open droplist^ item 1^ item 2^
}
@endsalt
```



[Ref. QA-4184]



14.4 罫線の使用 [|、#、!、-、+]

表は括弧 { で開始すれば自動的に作成されます。そして、列を分割するには | を使う必要があります。

例：

```
@startsalt
{
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt
```

行や列の罫線を表示したいときは、括弧で開始した直後に、以下のように定義された 1 文字を使用してください。：

文字	結果
#	全ての縦横の罫線を表示する
!	全ての縦線を表示する
-	全ての横線を表示する
+	外枠を表示する

```
@startsalt
{+
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt
```

14.5 グループボックス [^]

```
@startsalt
{^"My group box"
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt
```

[Ref. QA-5840]

14.6 セパレータの使用 [..、==、~~、-]

いくつかの横線をセパレータとして使用することができます。

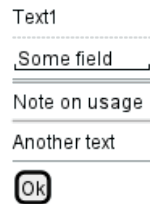
```
@startsalt
{
```



```

Text1
..
"Some field"
==
Note on usage
~~
Another text
--
[Ok]
}
@endsalt

```



14.7 木構造ウィジェット [T]

木構造を作るには、{T で開始して階層を示すために + を使用する必要があります。

```

@startsalt
{
{T
+ World
++ America
+++ Canada
+++ USA
++++ New York
++++ Boston
+++ Mexico
++ Europe
+++ Italy
+++ Germany
++++ Berlin
++ Africa
}
}
@endsalt

```



14.8 木構造と表 [T]

木構造と表を組み合わせることができます。

```

@startsalt
{
{T

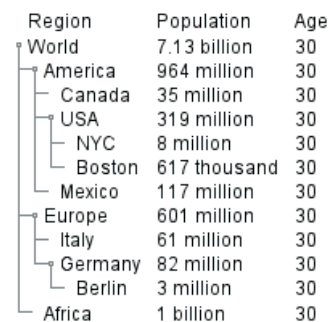
```



```

+Region      | Population  | Age
+ World      | 7.13 billion | 30
++ America   | 964 million  | 30
+++ Canada   | 35 million   | 30
+++ USA      | 319 million  | 30
++++ NYC     | 8 million    | 30
++++ Boston  | 617 thousand | 30
+++ Mexico   | 117 million  | 30
++ Europe    | 601 million  | 30
+++ Italy    | 61 million   | 30
+++ Germany  | 82 million   | 30
++++ Berlin  | 3 million    | 30
++ Africa    | 1 billion    | 30
}
}
@endsalt

```



罫線を追加すると次のようになります。

```

@startsalt
{
  ..
  == with T!
  {T!
  +Region      | Population  | Age
  + World      | 7.13 billion | 30
  ++ America   | 964 million  | 30
  }
  ..
  == with T-
  {T-
  +Region      | Population  | Age
  + World      | 7.13 billion | 30
  ++ America   | 964 million  | 30
  }
  ..
  == with T+
  {T+
  +Region      | Population  | Age
  + World      | 7.13 billion | 30
  ++ America   | 964 million  | 30
  }
  ..
  == with T#
  {T#
  +Region      | Population  | Age
  + World      | 7.13 billion | 30
  ++ America   | 964 million  | 30
  }
  ..
}

```



```

}
..
}
@endsalt

```

with T!		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T-		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T+		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

with T#		
Region	Population	Age
World	7.13 billion	30
America	964 million	30

[Ref. QA-1265]

14.9 括弧で括る [{、}]

定義中に、新しい括弧で括ることによりサブ要素を定義することができます。

```

@startsalt
{
Name          | "          "
Modifiers:    | { (X) public | () default | () private | () protected
              | [] abstract | [] final   | [] static }
Superclass:   | { "java.lang.Object " | [Browse...] }
}
@endsalt

```

Name	
Modifiers:	☒ public ☐ default ☐ private ☐ protected ☐ abstract ☐ final ☐ static
Superclass:	java.lang.Object Browse...

14.10 タブの追加 [/]

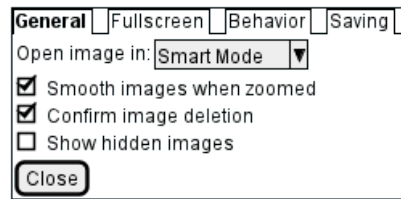
{/ 表記を使用してタブを追加することができます。HTML コードを使用してテキストを太字にできることに注意してください。

```

@startsalt
{+
{/ <b>General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt

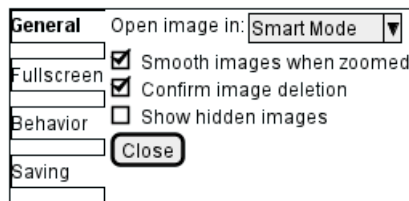
```





タブは垂直方向にも配向できます :

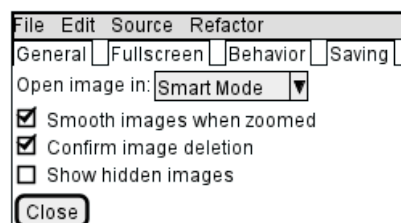
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



14.11 メニューの使用 [*]

{* 表記でメニューを追加することができます。

```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```

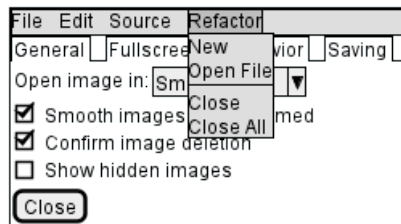


メニューを開くことも可能です :

```

@startsalt
{+
{* File | Edit | Source | Refactor
  Refactor | New | Open File | - | Close | Close All }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt

```

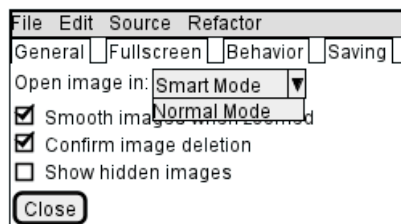


ドロップリストを開くことも可能です：

```

@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^^Normal Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt

```



[Ref. QA-4184]

14.12 テーブル (上級)

テーブルのための 2 つの特別な表記を使用することができます。

- * は左のセルとの結合となります
- . は空のセルとなります

```

@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2

```



```

Row header 2 | A long cell | *
}
@endsalt

```

	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	

14.13 スクロールバー [S、SI、S-]

次の例のように、{S でスクロールバーを使用できます：

- {S：水平方向、垂直方向のスクロールバー

```

@startsalt
{S
Message
.
.
.
.
}
@endsalt

```



- {SI：垂直方向のスクロールバーのみ

```

@startsalt
{SI
Message
.
.
.
.
}
@endsalt

```



- {S-：水平方向のスクロールバーのみ

```

@startsalt
{S-
Message
.
.
.
.
}
@endsalt

```



14.14 色

ウィジェットのテキストの色を変えることができます。

```
@startsalt
{
  <color:Blue>Just plain text
  [This is my default button]
  [<color:green>This is my green button]
  [<color:#9a9a9a>This is my disabled button]
  [] <color:red>Unchecked box
  [X] <color:green>Checked box
  "Enter text here  "
  ^This is a droplist^
  ^<color:#9a9a9a>This is a disabled droplist^
  ^<color:red>This is a red droplist^
}
@endsalt
```



[Ref. QA-12177]

14.15 Salt での Creole の使用

Creole または HTML Creole 記法を使用することができます:

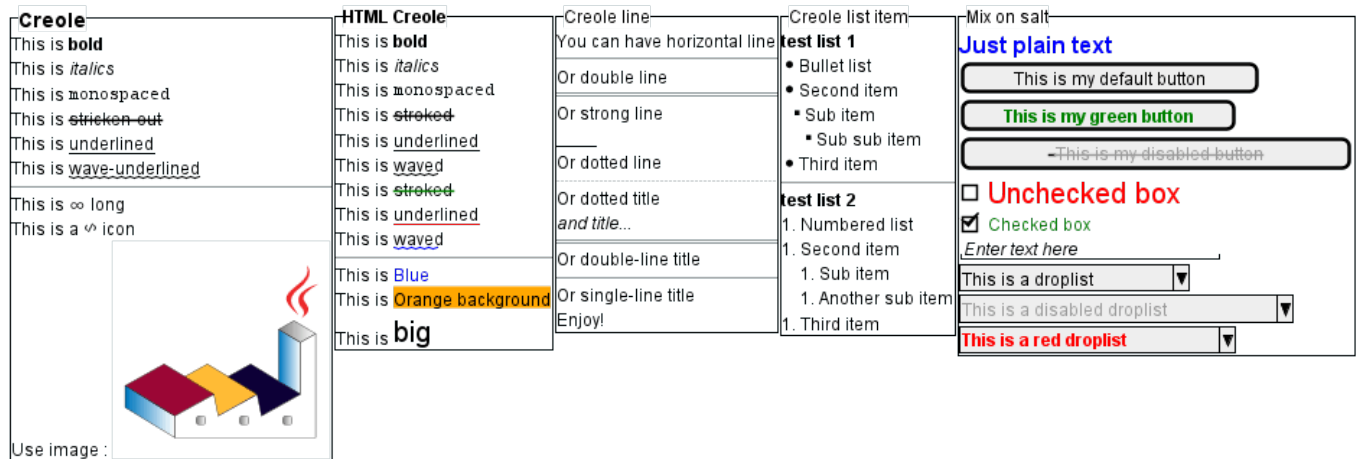
```
@startsalt
{{^==Creole
  This is bold
  This is italics
  This is "monospaced"
  This is stricken-out
  This is underlined
  This is ~wave-underlined~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
}|
{^<b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
```



```

This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
}|
{^Creole line
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}|
{^Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}|
{^Mix on salt
==<color:Blue>Just plain text
[This is my default button]
[<b><color:green>This is my green button]
[ ---<color:#9a9a9a>This is my disabled button-- ]
[] <size:20><color:red>Unchecked box
[X] <color:green>Checked box
"//Enter text here// "
^This is a droplist^
^<color:#9a9a9a>This is a disabled droplist^
^<b><color:red>This is a red droplist^
}}
@endsalt

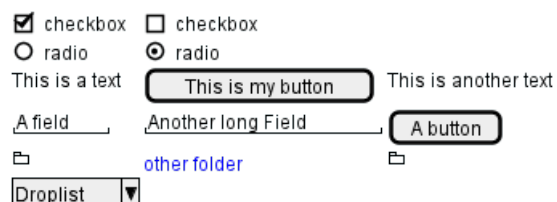
```



14.16 疑似スプライト [《、》]

<< と >> を使って疑似スプライト（スプライトのような画像）を定義し、それ以降で画像を使用することができます。

```
@startsalt
{
  [X] checkbox|[] checkbox
  () radio | (X) radio
  This is a text|[This is my button]|This is another text
  "A field"|"Another long Field"| [A button]
  <<folder
  .....
  .XXXXX.....
  .X...X.....
  .XXXXXXXXXX.
  .X.....X.
  .X.....X.
  .X.....X.
  .X.....X.
  .XXXXXX....
  .....
  >>|<color:blue>other folder|<<folder>>
  ^Droplist^
}
@endsalt
```



[Ref. QA-5849]

14.17 OpenIconic

OpenIconic はとても素晴らしいオープンソースのアイコンセットです。これらのアイコンは、creole parser に統合されていますので、簡単に使うことができます。このような構文で使用できます。:

<&ICON_NAME>

```
@startsalt
{
```



```

Login<&person> | "MyName   "
Password<&key> | "****     "
[Cancel <&circle-x>] | [OK <&account-login>]
}
@endsalt

```



全アイコンのリストは OpenIconic Website にあります。もしくは、次の特別なダイアグラムを使用してください：

```

@startuml
listopeniconic
@enduml

```

List Open Iconic

Credit to
<https://useiconic.com/open>

↗ account-login
 ↗ account-logout
 ↗ action-redo
 ↗ action-undo
 ≡ align-center
 ≡ align-left
 ≡ align-right
 ☆ aperture
 ↓ arrow-bottom
 ○ arrow-circle-bottom
 ○ arrow-circle-left
 ○ arrow-circle-right
 ○ arrow-circle-top
 ← arrow-left
 → arrow-right
 ↓ arrow-thick-bottom
 ← arrow-thick-left
 → arrow-thick-right
 ↑ arrow-thick-top
 ↑ arrow-top
 📶 audio-spectrum
 🎧 audio
 † badge
 🚫 ban
 📊 bar-chart
 🛒 basket
 🔋 battery-empty
 🔋 battery-full
 🧴 beaker

🔔 bell
 📶 bluetooth
 B bold
 ⚙ bolt
 📖 book
 📌 bookmark
 📦 box
 📁 briefcase
 £ british-pound
 🌐 browser
 🖌 brush
 🐛 bug
 📣 bullhorn
 📊 calculator
 📅 calendar
 📷 camera-slr
 ▼ caret-bottom
 ◀ caret-left
 ▶ caret-right
 ▲ caret-top
 🗑 cart
 💬 chat
 ✓ check
 ▼ chevron-bottom
 ◀ chevron-left
 ▶ chevron-right
 ▲ chevron-top
 🔍 circle-check
 ✖ circle-x
 📋 clipboard
 🕒 clock
 ☁ cloud-download
 ☁ cloud-upload

☁ cloud
 ☁ cloudy
 📄 code
 🧠 cog
 ☐ collapse-down
 ◀ collapse-left
 ▶ collapse-right
 ☐ collapse-up
 ⚙ compass
 ⚙ command
 📄 comment-square
 ⚙ compass
 ⚙ contrast
 ≡ copywriting
 📄 credit-card
 📄 crop
 📄 dashboard
 ⬇ data-transfer-download
 ⬆ data-transfer-upload
 🗑 delete
 📞 dial
 📄 document
 \$ dollar
 " double-quote-sans-left
 " double-quote-sans-right
 " double-quote-serif-left
 " double-quote-serif-right
 💧 droplet
 🚪 eject
 ⬆ elevator
 "" ellipses
 📁 envelope-closed
 📁 envelope-open
 € euro

≡ excerpt
 ☐ expand-down
 ◀ expand-left
 ▶ expand-right
 ☐ expand-up
 🔗 external-link
 👁 eye
 🐼 eyedropper
 📄 file
 🔥 fire
 🚩 flag
 ⚡ flash
 📁 folder
 🍴 fork
 🖥 fullscreen-enter
 🖥 fullscreen-exit
 🌐 globe
 ⚡ graph
 📊 grid-four-up
 📊 grid-three-up
 📊 grid-two-up
 📁 hard-drive
 📄 header
 🎧 headphones
 ❤ heart
 🏠 home
 🖼 image
 📁 inbox
 ∞ infinity
 ⓘ info
 I italic
 ≡ justify-center
 ≡ justify-left

≡ justify-right
 🔑 key
 📄 laptop
 📄 layers
 💡 lightbulb
 🔗 link-broken
 🔗 link-intact
 🔗 link-rich
 ≡ list
 ↗ location
 🔒 lock-locked
 🔒 lock-unlocked
 🔄 loop-circular
 ☐ loop-square
 ☐ loop
 🔍 magnifying-glass
 📍 map-marker
 🗺 map
 ⏸ media-pause
 ▶ media-play
 🎵 media-record
 ⏮ media-skip-backward
 ▶ media-skip-forward
 ⏮ media-step-backward
 ▶ media-step-forward
 📄 media-stop
 🏥 medical-cross
 ≡ menu
 🎤 microphone
 ➖ minus
 📄 monitor
 🌙 moon
 + move

🎵 musical-note
 📄 paperclip
 🖋 pencil
 👤 person
 📱 phone
 📊 pie-chart
 📌 pin
 🎮 play-circle
 + plus
 ⏻ power-standby
 🖨 print
 📁 project
 📄 puzzle-piece
 ? question-mark
 🌧 rain
 ✖ random
 🔄 reload
 ↕ resize-both
 ⬆ resize-height
 ⬆ resize-width
 📡 rss-alt
 📄 rss
 📄 script
 📄 share-boxed
 ➦ share
 🛡 shield
 📶 signal
 📶 signpost
 ⬆ sort-ascending
 ⬆ sort-descending
 📊 spreadsheet

★ star
 ☀ sun
 📱 tablet
 🏷 tag
 🏷 tags
 🎯 target
 📄 task
 📄 terminal
 T text
 🖱 thumb-down
 👍 thumb-up
 ⌚ timer
 ➡ transfer
 🗑 trash
 U underline
 📄 vertical-align-bottom
 📄 vertical-align-center
 📄 vertical-align-top
 📄 video
 🔊 volume-high
 🔊 volume-low
 🔊 volume-off
 ⚠ warning
 📶 wifi
 🛠 wrench
 ✖ x
 🏠 yen
 🔍 zoom-in
 🔍 zoom-out

14.18 タイトル、ヘッダ、フッタ、キャプション、凡例を追加する

```

@startsalt
title My title
header some header
footer some footer
caption This is caption
legend
The legend
end legend

```

```

{+
  Login | "MyName   "
  Password | "****     "
  [Cancel] | [ OK ]
}

```



@endsalt

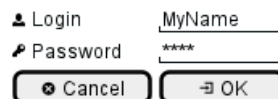


(参照: 共通コマンド)

14.19 拡大、DPI

14.19.1 拡大なし (デフォルト)

```
@startsalt
{
  <&person> Login   | "MyName   "
  <&key> Password  | "****   "
  [<&circle-x> Cancel ] | [ <&account-login> OK   ]
}
@endsalt
```



14.19.2 scale

scale コマンドを使って、生成する画像を拡大できます。

小数または分数で拡大率を指定できます。ピクセル単位で width (幅) または height (高さ) を指定することもできます。幅と高さの両方を指定することもできます。この場合は、指定したサイズの内側に収まるように調整されます。

```
@startsalt
scale 2
{
  <&person> Login   | "MyName   "
  <&key> Password  | "****   "
  [<&circle-x> Cancel ] | [ <&account-login> OK   ]
}
@endsalt
```



(参照: 共通コマンドの拡大)

14.19.3 DPI

skinparam dpi コマンドを使って、生成する画像を拡大することもできます。

```

@startsalt
skinparam dpi 200
{
    <&person> Login | "MyName  "
    <&key> Password | "****  "
    [<&circle-x> Cancel ] | [ <&account-login> OK  ]
}
@endsalt

```



14.20 Salt をアクティビティ図の上に表示する

こちらの説明を参照してください。

```

@startuml
(*) --> "
{{
    salt
    {+
    <b>an example
    choose one option
    ()one
    ()two
    [ok]
    }
}}
" as choose

choose -right-> "
{{
    salt
    {+
    <b>please wait
    operation in progress
    <&clock>
    [cancel]
    }
}}
" as wait
wait -right-> "
{{
    salt
    {+
    <b>success
    congratulations!
    [ok]
    }
}}
" as success

wait -down-> "
{{

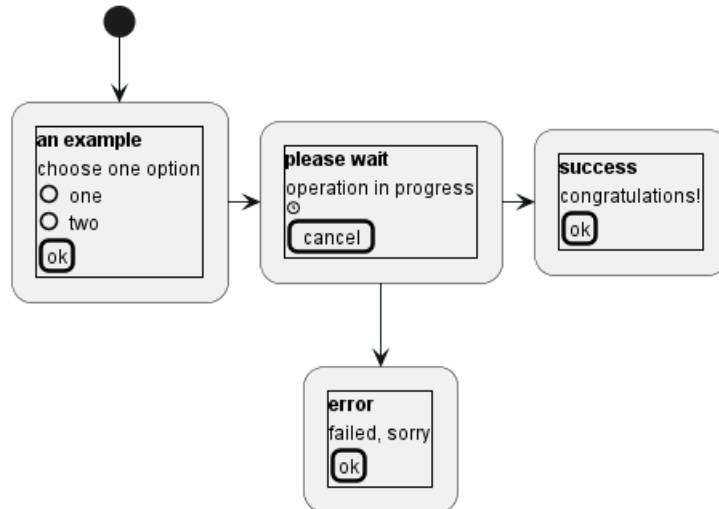
```



```

salt
{+
<b>error
failed, sorry
[ok]
}
}}
"
@enduml

```



マクロ定義と組み合わせることもできます。

```

@startuml
!unquoted procedure SALT($x)
"{{
salt
%invoke_procedure("_"+"$x)
}}" as $x
!endprocedure

!procedure _choose()
{+
<b>an example
choose one option
()one
()two
[ok]
}
!endprocedure

!procedure _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endprocedure

!procedure _success()
{+
<b>success

```

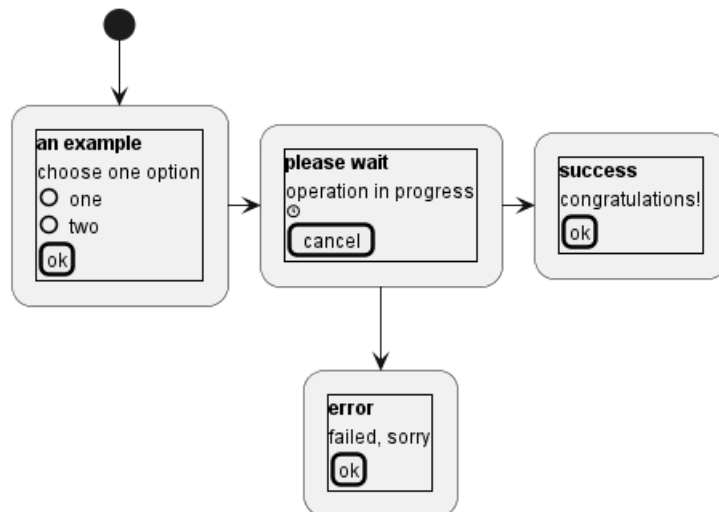
```

congratulations!
[ok]
}
!endprocedure

!procedure _error()
{+
<b>error
failed, sorry
[ok]
}
!endprocedure

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)
wait -down-> SALT(error)
@enduml

```



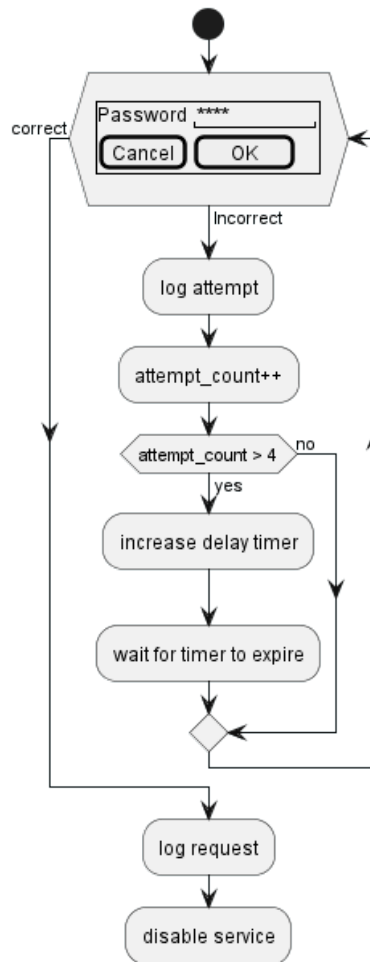
14.21 Salt をアクティビティ図の繰り返し条件 (前判定) の上に表示する

salt をアクティビティ図の繰り返し条件 (前判定) の上に表示することができます。

```

@startuml
start
while (\n{\nsalt\n{+\nPassword | "****      "\n[Cancel] | [ OK  ]}\n}}\n) is (Incorrect)
    :log attempt;
    :attempt_count++;
    if (attempt_count > 4) then (yes)
        :increase delay timer;
        :wait for timer to expire;
    else (no)
    endif
endwhile (correct)
:log request;
:disable service;
@enduml

```



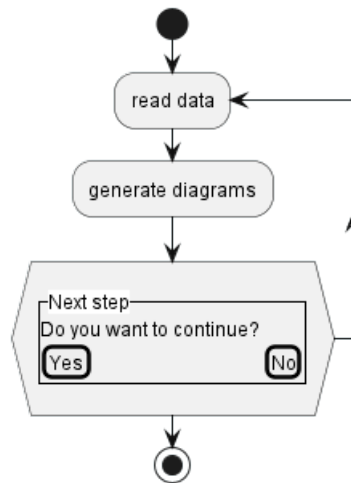
[Ref. QA-8547]

14.22 Salt をアクティビティ図の繰り返し条件 (後判定) の上に表示する

salt をアクティビティ図の繰り返し条件 (後判定) の上に表示することができます。

```

@startuml
start
repeat :read data;
    :generate diagrams;
repeat while (\n{\nsalt\n{"Next step"\n  Do you want to continue? \n[Yes]|[No]\n}\n})
stop
@enduml
  
```



[Ref. QA-14287]

14.23 skinparam

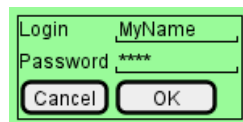
一部の skinparam コマンドを使用して、見た目を変更することができます。

例:

```

@startsalt
skinparam Backgroundcolor palegreen
{+
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt

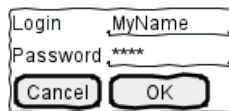
```



```

@startsalt
skinparam handwritten true
{+
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt

```



TODO: FIXME skinparam には、salt では使えないものもあります:

```

@startsalt
skinparam defaultFontName monospaced
{+
  Login    | "MyName  "
  Password | "****   "
  [Cancel] | [ OK   ]
}
@endsalt

```



14.24 スタイル

一部のスタイルコマンドを使用して、見た目を変更することができます。

例:

```
@startsalt
<style>
saltDiagram {
  BackgroundColor palegreen
}
</style>
{+
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
```

TODO: FIXME スタイルには、salt では使えないものもあります:

```
@startsalt
<style>
saltDiagram {
  Fontname Monospaced
  FontSize 10
  FontStyle italic
  LineThickness 0.5
  LineColor red
}
</style>
{+
  Login      | "MyName  "
  Password   | "****    "
  [Cancel]   | [ OK    ]
}
@endsalt
```

[Ref. QA-13460]

15 アーキテクチャ図

現在、このダイアグラムは提案段階です。将来的に変更されるかもしれません。

新しいシンタックス案の提案を歓迎します！よりよい形を模索するのに、あなたからの意見や提案が役に立ちます！！

15.1 アーキテクチャの要素

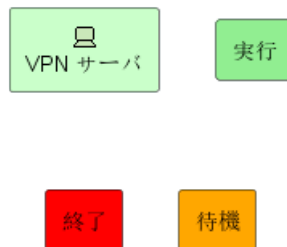
各要素は `archimate` で定義します。

ステレオタイプとして、アイコンを使うことができます。使用できるアイコンの一覧は、[[# 使用できるアイコン一覧](#) | [こちらを参照してください](#)。]

HTML のカラーネームを使って、色の変更ができます。また、いくつかのキーワード (`Business`, `Application`, `Motivation`, `Strategy`, `Technology`, `Physical`, `Implementation`) を使うこともできます。

```
@startuml
archimate #Technology "VPN サーバ" as vpnServerA <<technology-device>>

rectangle 実行 #lightgreen
rectangle 終了 #red
rectangle 待機 #orange
@enduml
```



15.2 ジャンクション

プリプロセス機能を使って `circle` を定義し、使用してください。

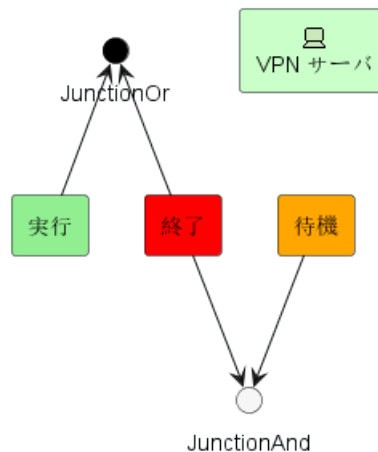
```
@startuml
!define Junction_Or circle #black
!define Junction_And circle #whitesmoke

Junction_And JunctionAnd
Junction_Or JunctionOr

archimate #Technology "VPN サーバ" as vpnServerA <<technology-device>>

rectangle 実行 #lightgreen
rectangle 終了 #red
rectangle 待機 #orange
実行 -up-> JunctionOr
終了 -up-> JunctionOr
終了 -down-> JunctionAnd
待機 -down-> JunctionAnd
@enduml
```





15.3 例 1

```

@startuml
skinparam rectangle<<behavior>> {
  roundCorner 25
}
sprite $bProcess jar:archimate/business-process
sprite $aService jar:archimate/application-service
sprite $aComponent jar:archimate/application-component

rectangle "Handle claim" as HC <<$bProcess>><<behavior>> #Business
rectangle "Capture Information" as CI <<$bProcess>><<behavior>> #Business
rectangle "Notify\nAdditional Stakeholders" as NAS <<$bProcess>><<behavior>> #Business
rectangle "Validate" as V <<$bProcess>><<behavior>> #Business
rectangle "Investigate" as I <<$bProcess>><<behavior>> #Business
rectangle "Pay" as P <<$bProcess>><<behavior>> #Business

HC *-down- CI
HC *-down- NAS
HC *-down- V
HC *-down- I
HC *-down- P

CI -right->> NAS
NAS -right->> V
V -right->> I
I -right->> P

rectangle "Scanning" as scanning <<$aService>><<behavior>> #Application
rectangle "Customer administration" as customerAdministration <<$aService>><<behavior>> #Application
rectangle "Claims administration" as claimsAdministration <<$aService>><<behavior>> #Application
rectangle Printing <<$aService>><<behavior>> #Application
rectangle Payment <<$aService>><<behavior>> #Application

scanning -up-> CI
customerAdministration -up-> CI
claimsAdministration -up-> NAS
claimsAdministration -up-> V
claimsAdministration -up-> I
Payment -up-> P

Printing -up-> V
Printing -up-> P
  
```



```

rectangle "Document\nManagement\nSystem" as DMS <<$aComponent>> #Application
rectangle "General\nCRM\nSystem" as CRM <<$aComponent>> #Application
rectangle "Home & Away\nPolicy\nAdministration" as HAPA <<$aComponent>> #Application
rectangle "Home & Away\nFinancial\nAdministration" as HFPA <<$aComponent>> #Application

```

```

DMS .up.|> scanning
DMS .up.|> Printing
CRM .up.|> customerAdministration
HAPA .up.|> claimsAdministration
HFPA .up.|> Payment

```

```

legend left

```

```

Example from the "Archisurance case study" (OpenGroup).

```

```

See

```

```

====

```

```

<$bProcess> :business process

```

```

====

```

```

<$aService> : application service

```

```

====

```

```

<$aComponent> : application component

```

```

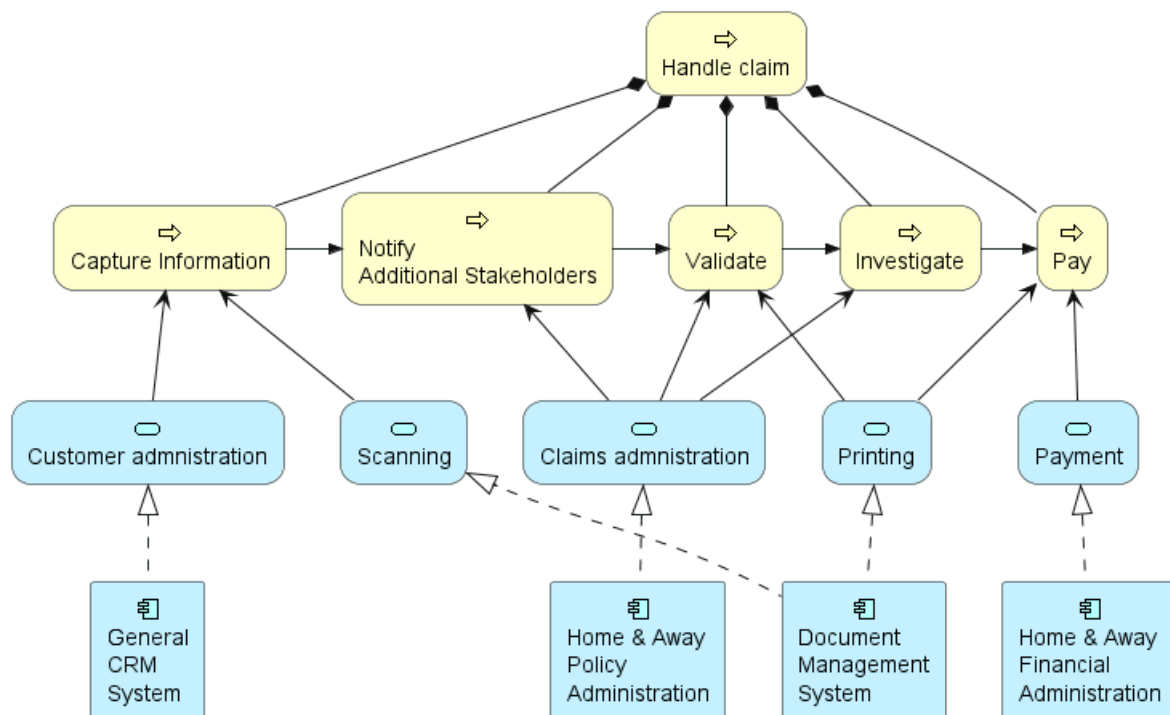
endlegend

```

```

@enduml

```



Example from the "Archisurance case study" (OpenGroup).

See

⇒ :business process

○ : application service

☐ : application component

15.4 例 2

```

@startuml

```



```
skinparam roundcorner 25
```

```
rectangle "Capture Information" as CI <<$archimate/business-process>> #Business
@enduml
```

⇒
Capture Information

15.5 使用できるアイコン一覧

アーキテクチャ図で使用できるアイコンの一覧は、次のコードで表示することができます。

```
@startuml
listsprite
@enduml
```

List Current Sprites

Credit to
<http://www.archimate4tool.com>

archimate :

access
activity
actor
aggregation
application-collaboration
application-component
application-data-object
application-event
application-function
application-interface
application-process
application-service
assessment-filled
assessment
assignment
association-unidirect
association
business-activity
business-actor
business-collaboration
business-contract
business-event
business-function
business-interaction
business-interface
business-location
business-meaning

business-object
business-process
business-product
business-representation
business-role
business-service
business-value
collaboration
communication-path
component
composition
constraint-filled
constraint
contract
deliverable-filled
deliverable
device
driver-filled
driver
event
flow
function
gap-filled
gap
goal-filled
goal
implementation-deliverable
implementation-event
implementation-gap
implementation-plateau
implementation-workpackage
influence
interaction
interface-required

interface-symmetric
interface
junction-and
junction-or
junction
location
meaning
motivation-assessment
motivation-constraint
motivation-driver
motivation-goal
motivation-meaning
motivation-outcome
motivation-principle
motivation-requirement
motivation-stakeholder
motivation-value
network
node
object
physical-distribution-network
physical-equipment
physical-facility
physical-material
plateau
principle-filled
principle
process
product
realisation
representation
requirement-filled
requirement
role

service
serving
specialisation
specialization
stakeholder-filled
strategy-capability
strategy-course-of-action
strategy-resource
strategy-value-stream
system-software
technology-artifact
technology-collaboration
technology-communication-network
technology-communication-path
technology-device
technology-event
technology-function
technology-infra-interface
technology-infra-service
technology-interface
technology-interaction
technology-network
technology-node
technology-path
technology-process
technology-service
technology-system-software
triggering
used-by
value
workpackage-filled

15.6 ArchiMate マクロ

15.6.1 Archimate マクロとライブラリ

Archimate マクロの一覧は Archimate-PlantUML で定義されています。このマクロはアーキテクチャ図の作成を簡単にしてくれます。Archimate は PlantUML の標準ライブラリにネイティブに存在します。

15.6.2 Archimate 要素

マクロを使用した ArchiMate 要素の生成は次のように行います：Category_ElementName(nameOfTheElement, "description")

例：

- Motivation カテゴリに含まれる「ステークホルダー」要素を定義する場合、次のように記述します：Motivation_Stakeholder(StakeholderElement, "Stakeholder Description"):

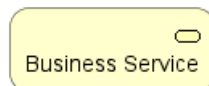


```
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
@enduml
```



- 「ビジネスサービス」要素を定義する場合は、`Business_Service(BService, "Business Service")` :

```
@startuml
!include <archimate/Archimate>
Business_Service(BService, "Business Service")
@enduml
```



15.6.3 Archimate の関係 (relationship)

Archimate の関係は、次のように定義します：`Rel_RelationType(fromElement, toElement, "description")`
 また、次のように、2つの要素の方向を定義します：`Rel_RelationType_Direction(fromElement, toElement, "description")`

次の `RelationTypes` がサポートされています：

- Access
- Aggregation
- Assignment
- Association
- Composition
- Flow
- Influence
- Realization
- Serving
- Specialization
- Triggering

次の `Directions` がサポートされています：

- Up
- Down
- Left
- Right

例：

- 上で定義した「ステークホルダー」と「ビジネスサービス」に対して、コンポジション関係を定義する場合、次のように記述します

```
Rel_Composition(StakeholderElement, BService, "Description for the relationship")

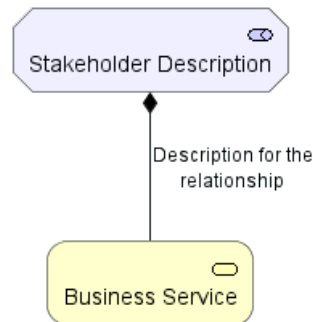
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
```



```

Business_Service(BService, "Business Service")
Rel_Composition(StakeholderElement, BService, "Description for the relationship")
@enduml

```

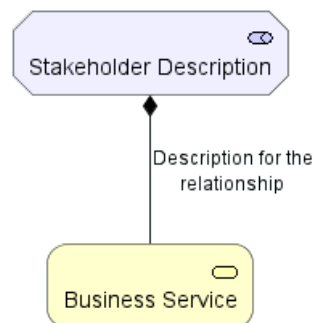


- Unordered List ItemTo orient the two elements in top - down position, the syntax will be

```

Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@startuml
!include <archimate/Archimate>
Motivation_Stakeholder(StakeholderElement, "Stakeholder Description")
Business_Service(BService, "Business Service")
Rel_Composition_Down(StakeholderElement, BService, "Description for the relationship")
@enduml

```



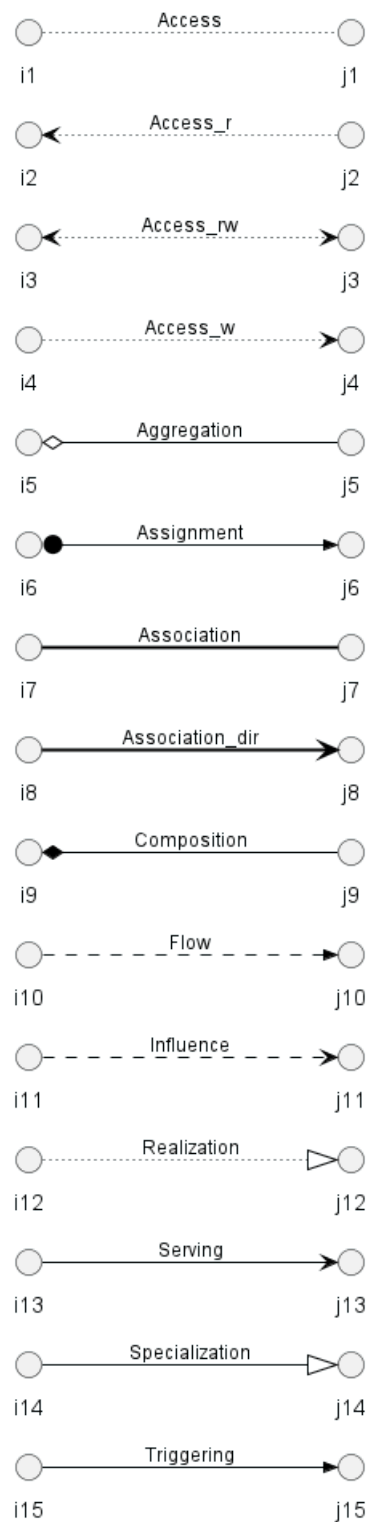
15.6.4 付録：すべての Archimate RelationType の例

```

@startuml
left to right direction
skinparam nodesep 4
!include <archimate/Archimate>
Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i8, j8, Association_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i3, j3, Access_rw)
Rel_Access_r(i2, j2, Access_r)
Rel_Access(i1, j1, Access)
@enduml

```





```

@startuml
title ArchiMate Relationships Overview
skinparam nodesep 5
<style>
interface {
    shadowing 0
    backgroundcolor transparent
    linecolor transparent
    FontColor transparent

```

```

}
</style>
!include <archimate/Archimate>
left to right direction

rectangle Other {
() i14
() j14
}

rectangle Dynamic {
() i10
() j10
() i15
() j15
}

rectangle Dependency {
() i13
() j13
() i4
() j4
() i11
() j11
() i7
() j7
}

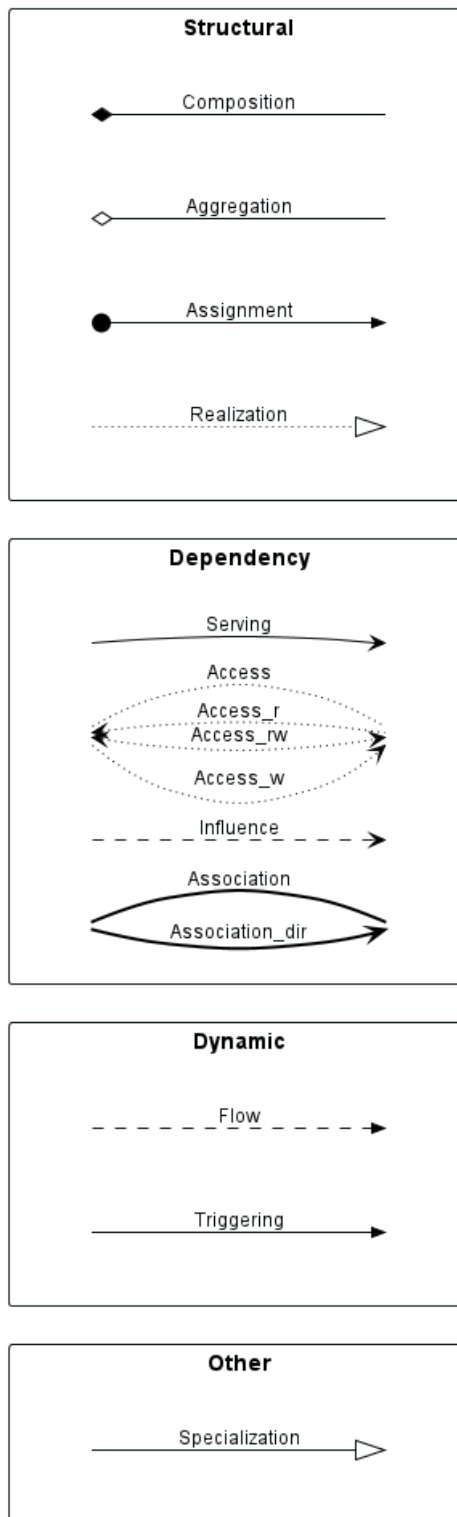
rectangle Structural {
() i9
() j9
() i5
() j5
() i6
() j6
() i12
() j12
}

Rel_Triggering(i15, j15, Triggering)
Rel_Specialization(i14, j14, Specialization)
Rel_Serving(i13, j13, Serving)
Rel_Realization(i12, j12, Realization)
Rel_Influence(i11, j11, Influence)
Rel_Flow(i10, j10, Flow)
Rel_Composition(i9, j9, Composition)
Rel_Association_dir(i7, j7, \nAssociation_dir)
Rel_Association(i7, j7, Association)
Rel_Assignment(i6, j6, Assignment)
Rel_Aggregation(i5, j5, Aggregation)
Rel_Access_w(i4, j4, Access_w)
Rel_Access_rw(i4, j4, Access_rw)
Rel_Access_r(i4, j4, Access_r)
Rel_Access(i4, j4, Access)
@enduml

```



ArchiMate Relationships Overview



[Adapted from Archimate PR#25]

16 ガントチャート

ガントチャートは、SVC 文型 (主語-動詞-補語) の単純な文によって、自然な英語を書くように記述できます。

16.1 タスクの定義

タスクは角カッコで定義します。

16.1.1 期間

タスクの期間は、動詞 **lasts** で指定します。

```
@startgantt
[プロトタイプを設計] lasts 15 days
[プロトタイプをテスト] lasts 10 days
-- すべての例 --
[Task 1 (1日)] lasts 1 day
[T2 (5日)] lasts 5 days
[T3 (1週間)] lasts 1 week
[T4 (1週間と4日)] lasts 1 week and 4 days
[T5 (2週間)] lasts 2 weeks
@endgantt
```



1 週間 (week) は休業日 (closed) を除く日数を表します。土曜と日曜を休業日に設定した場合、week は 5 日を表すようになります。

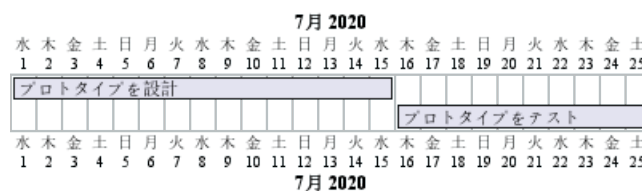
16.1.2 開始

タスクの開始は、動詞 **start** で指定します。

```
@startgantt
language ja

[プロトタイプを設計] lasts 15 days
[プロトタイプをテスト] lasts 10 days

Project starts 2020-07-01
[プロトタイプを設計] starts 2020-07-01
[プロトタイプをテスト] starts 2020-07-16
@endgantt
```



16.1.3 終了

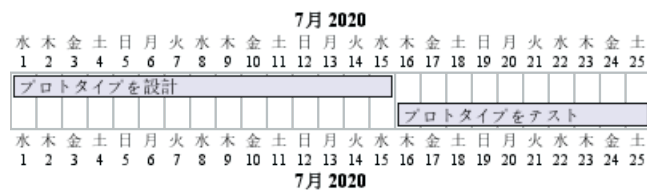
タスクの終了は、動詞 `end` で指定します。

```
@startgantt
language ja
```

```
[プロトタイプを設計] lasts 15 days
[プロトタイプをテスト] lasts 10 days
```

```
Project starts 2020-07-01
[プロトタイプを設計] ends 2020-07-15
[プロトタイプをテスト] ends 2020-07-25
```

```
@endgantt
```



16.1.4 開始と終了

開始と終了の両方を日付で指定することも可能です。

```
@startgantt
language ja
```

```
Project starts 2020-07-01
[プロトタイプを設計] starts 2020-07-01
[プロトタイプをテスト] starts 2020-07-16
[プロトタイプを設計] ends 2020-07-15
[プロトタイプをテスト] ends 2020-07-25
```

```
@endgantt
```



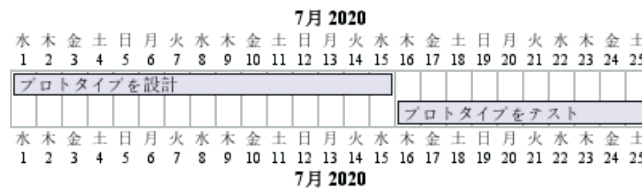
16.2 AND 接続詞を使った 1 行宣言

`and` でつなぐことで、宣言を 1 行にできます。

```
@startgantt
language ja
```

```
Project starts 2020-07-01
[プロトタイプを設計] starts 2020-07-01 and ends 2020-07-15
[プロトタイプをテスト] starts 2020-07-16 and lasts 10 days
@endgantt
```





16.3 依存関係

タスク間の依存関係を定義することができます。

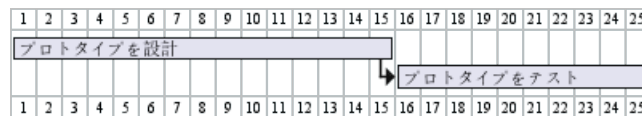
```
@startgantt
```

```
[プロトタイプを設計] lasts 15 days
```

```
[プロトタイプをテスト] lasts 10 days
```

```
[プロトタイプをテスト] starts at [プロトタイプを設計]'s end
```

```
@endgantt
```



```
@startgantt
```

```
[プロトタイプを設計] lasts 10 days
```

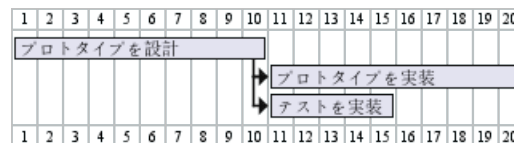
```
[プロトタイプを実装] lasts 10 days
```

```
[テストを実装] lasts 5 days
```

```
[プロトタイプを実装] starts at [プロトタイプを設計]'s end
```

```
[テストを実装] starts at [プロトタイプを実装]'s start
```

```
@endgantt
```



16.4 短い名前

as キーワードを使用して、タスクに短い名前を定義できます。

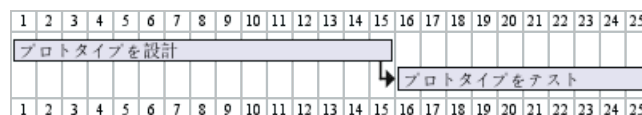
```
@startgantt
```

```
[プロトタイプを設計] as [D] lasts 15 days
```

```
[プロトタイプをテスト] as [T] lasts 10 days
```

```
[T] starts at [D]'s end
```

```
@endgantt
```



16.5 色のカスタマイズ

is colored in で、色のカスタマイズができます。

```
@startgantt
```

```
[プロトタイプを設計] lasts 13 days
```

```
[プロトタイプをテスト] lasts 4 days
```

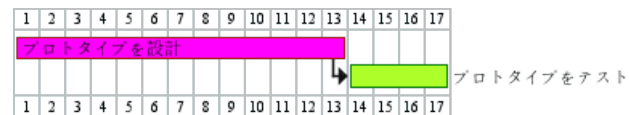
```
[プロトタイプをテスト] starts at [プロトタイプを設計]'s end
```

```
[プロトタイプを設計] is colored in Fuchsia/FireBrick
```

```
[プロトタイプをテスト] is colored in GreenYellow/Green
```

```
@endgantt
```





16.6 進捗状況

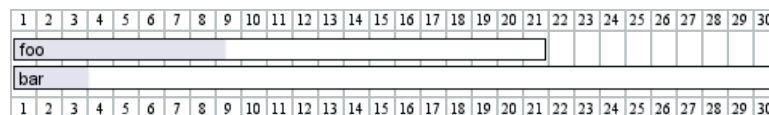
タスクの進捗状況を示すことができます。

16.6.1 完了率による追加

タスクの完了状況を、コマンドで設定することができます。

- is xx% completed
- is xx% complete

```
@startgantt
[foo] lasts 21 days
[foo] is 40% completed
[bar] lasts 30 days and is 10% complete
@endgantt
```



16.6.2 完了の色を変更する (スタイル別)

```
@startgantt
```

```
<style>
ganttdiagram {
  task {
    BackGroundColor GreenYellow
    LineColor Green
    unstarted {
      BackGroundColor Fuchsia
      LineColor FireBrick
    }
  }
}
</style>
```

```
[Prototype design] lasts 7 days
[Test prototype 0] lasts 4 days
[Test prototype 10] lasts 4 days
[Test prototype 20] lasts 4 days
[Test prototype 30] lasts 4 days
[Test prototype 40] lasts 4 days
[Test prototype 50] lasts 4 days
[Test prototype 60] lasts 4 days
[Test prototype 70] lasts 4 days
[Test prototype 80] lasts 4 days
[Test prototype 90] lasts 4 days
[Test prototype 100] lasts 4 days
```

```
[Test prototype 0] starts at [Prototype design]'s end
[Test prototype 10] starts at [Prototype design]'s end
[Test prototype 20] starts at [Prototype design]'s end
[Test prototype 30] starts at [Prototype design]'s end
```



```

[Test prototype 40] starts at [Prototype design]'s end
[Test prototype 50] starts at [Prototype design]'s end
[Test prototype 60] starts at [Prototype design]'s end
[Test prototype 70] starts at [Prototype design]'s end
[Test prototype 80] starts at [Prototype design]'s end
[Test prototype 90] starts at [Prototype design]'s end
[Test prototype 100] starts at [Prototype design]'s end

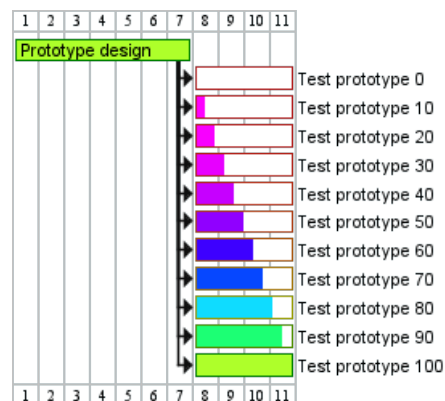
```

```

[Test prototype 0] is 0% complete
[Test prototype 10] is 10% complete
[Test prototype 20] is 20% complete
[Test prototype 30] is 30% complete
[Test prototype 40] is 40% complete
[Test prototype 50] is 50% complete
[Test prototype 60] is 60% complete
[Test prototype 70] is 70% complete
[Test prototype 80] is 80% complete
[Test prototype 90] is 90% complete
[Test prototype 100] is 100% complete

```

```
@endgantt
```



[参考 : QA-8297]

16.7 マイルストーン

happens でマイルストーンを定義できます。

16.7.1 依存関係に基づくマイルストーン

```
@startgantt
```

```

[プロトタイプをテスト] lasts 10 days
[プロトタイプが完成] happens at [プロトタイプをテスト]'s end
[製造ラインの準備] lasts 12 days
[製造ラインの準備] starts at [プロトタイプをテスト]'s end

```

```
@endgantt
```



16.7.2 日付指定のマイルストーン

```
@startgantt
```

```
language ja
```



```

Project starts 2020-07-01
[プロトタイプをテスト] lasts 10 days
[プロトタイプが完成] happens 2020-07-10
[製造ラインの準備] lasts 12 days
[製造ラインの準備] starts at [プロトタイプをテスト]'s end
@endgantt

```



16.7.3 全タスク完了のマイルストーン

```

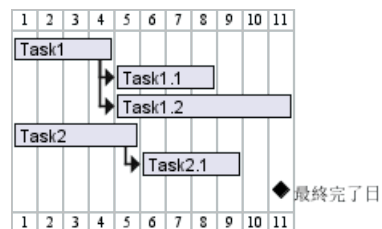
@startgantt
[Task1] lasts 4 days
then [Task1.1] lasts 4 days
[Task1.2] starts at [Task1]'s end and lasts 7 days

[Task2] lasts 5 days
then [Task2.1] lasts 4 days

[最終完了日] happens at [Task1.1]'s end
[最終完了日] happens at [Task1.2]'s end
[最終完了日] happens at [Task2.1]'s end

@endgantt

```



[Ref. QA-10764]

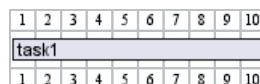
16.8 ハイパーリンク

タスクにハイパーリンクを追加することもできます。

```

@startgantt
[task1] lasts 10 days
[task1] links to [[http://plantuml.com]]
@endgantt

```



16.9 日付の表示

プロジェクトの開始日時に、特定の日付を指定できます。デフォルトでは、一番最初のタスクが指定した日付から開始します。

```

@startgantt
language ja

```



```

Project starts the 20th of september 2017
[プロトタイプを設計] as [タスク1] lasts 13 days
[タスク1] is colored in Lavender/LightBlue
@endgantt

```



16.10 日付の色

特定の日に色を付けることができます。

```

@startgantt
language ja

```

```

Project starts the 2020/09/01

```

```

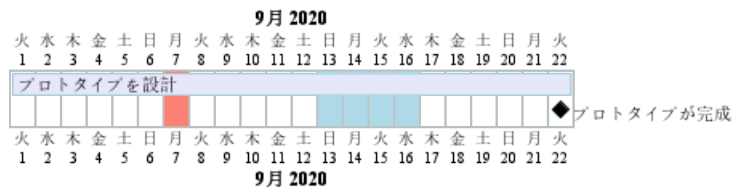
2020/09/07 is colored in salmon
2020/09/13 to 2020/09/16 are colored in lightblue

```

```

[プロトタイプを設計] as [TASK1] lasts 22 days
[TASK1] is colored in Lavender/LightBlue
[プロトタイプが完成] happens at [TASK1]'s end
@endgantt

```



16.11 スケールの変更

長期にわたるプロジェクトの場合、次のいずれかのパラメータを使用してスケールを変更することができます。

- printscale
- gantt scale
- project scale

次のいずれかの値を使用します。

- daily - 日 (デフォルト)
- weekly - 週
- monthly - 月
- quarterly - 四半期
- yearly - 年

(See QA-11272, QA-9041 and QA-10948)

16.11.1 daily (デフォルト)

```

@startgantt
saturday are closed

```



sunday are closed

Project starts the 1st of january 2021
 [プロトタイプ設計完了] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [テスト] lasts 14 days
 [TASK1]->[テスト]

2021-01-18 to 2021-01-22 are named [完了審査会]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.11.2 weekly

@startgantt
 printscale weekly
 saturday are closed
 sunday are closed

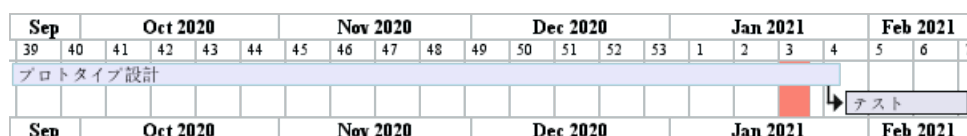
Project starts the 1st of january 2021
 [プロトタイプ設計完了] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [テスト] lasts 14 days
 [TASK1]->[テスト]

2021-01-18 to 2021-01-22 are named [完了審査会]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



@startgantt
 printscale weekly
 Project starts the 20th of september 2020
 [プロトタイプ設計] as [TASK1] lasts 130 days
 [TASK1] is colored in Lavender/LightBlue
 [テスト] lasts 20 days
 [TASK1]->[テスト]

2021-01-18 to 2021-01-22 are named [完了審査会]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



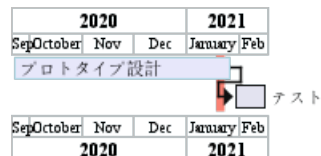
16.11.3 monthly

```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[プロトタイプ設計] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[テスト] lasts 20 days
[TASK1]->[テスト]
```

2021-01-18 to 2021-01-22 are named [完了審査会]

2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```



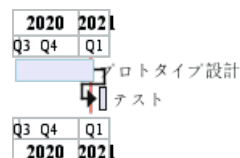
16.11.4 quarterly

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[プロトタイプ設計] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[テスト] lasts 20 days
[TASK1]->[テスト]
```

2021-01-18 to 2021-01-22 are named [完了審査会]

2021-01-18 to 2021-01-22 are colored in salmon

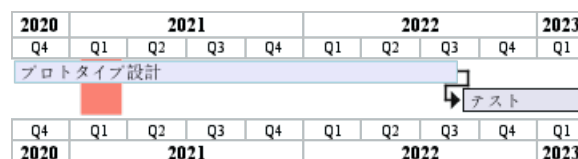
```
@endgantt
```



```
@startgantt
projectscale quarterly
Project starts the 1st of october 2020
[プロトタイプ設計] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[テスト] lasts 200 days
[TASK1]->[テスト]
```

2021-01-18 to 2021-03-22 are colored in salmon

```
@endgantt
```

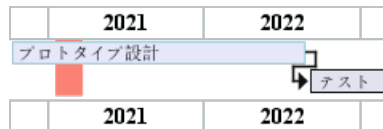


16.11.5 yearly

```
@startgantt
projectscale yearly
```

Project starts the 1st of october 2020
 [プロトタイプ設計] as [TASK1] lasts 700 days
 [TASK1] is colored in Lavender/LightBlue
 [テスト] lasts 200 days
 [TASK1]->[テスト]

2021-01-18 to 2021-03-22 are colored in salmon
 @endgantt



16.12 ズーム（全スケールの例）

パラメータを使って、ズームを変更できます。

- zoom <integer>

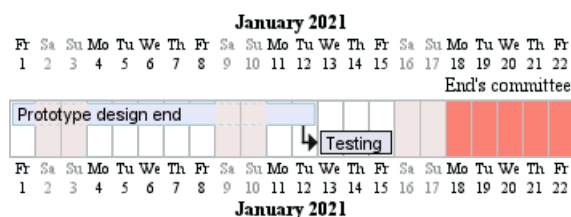
16.12.1 日単位スケールでのズーム

16.12.2 ズームなし

```
@startgantt
printscale daily
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 8 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 3 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.3 ズームあり

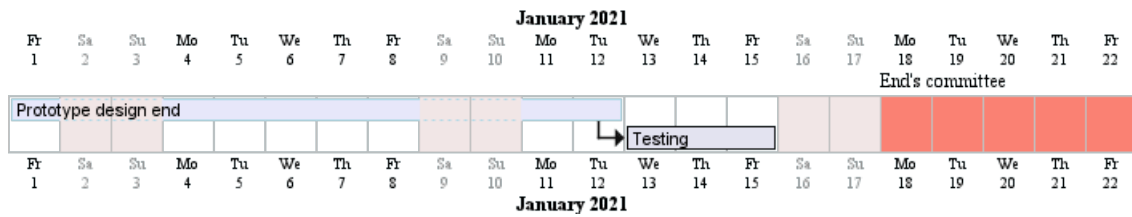
```
@startgantt
printscale daily zoom 2
saturday are closed
sunday are closed
```

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 8 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 3 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]



2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



[Ref.QA-13725]

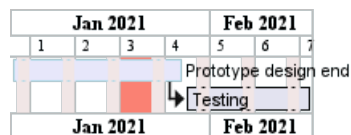
16.12.4 週単位スケールでのズーム

16.12.5 ズームなし

@startgantt
 printscale weekly
 saturday are closed
 sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt

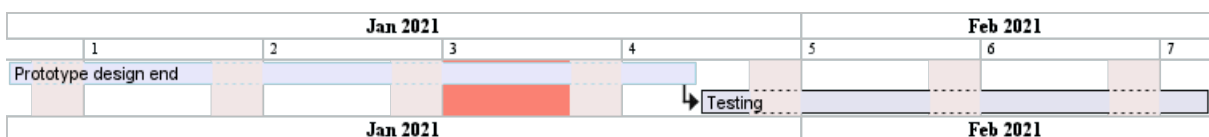


16.12.6 ズームあり

@startgantt
 printscale weekly zoom 4
 saturday are closed
 sunday are closed

Project starts the 1st of january 2021
 [Prototype design end] as [TASK1] lasts 19 days
 [TASK1] is colored in Lavender/LightBlue
 [Testing] lasts 14 days
 [TASK1]->[Testing]

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt

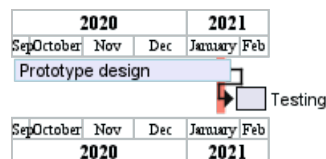


16.12.7 月単位スケールでのズーム

16.12.8 ズームなし

```
@startgantt
projectscale monthly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

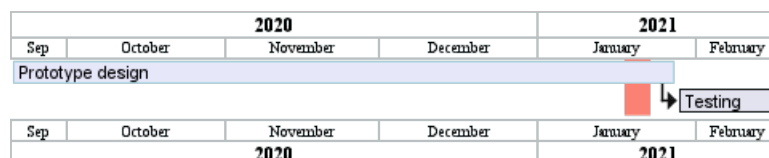
2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



16.12.9 ズームあり

```
@startgantt
projectscale monthly zoom 3
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt



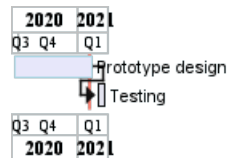
16.12.10 四半期単位スケールでのズーム

16.12.11 ズームなし

```
@startgantt
projectscale quarterly
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]
 2021-01-18 to 2021-01-22 are colored in salmon
 @endgantt





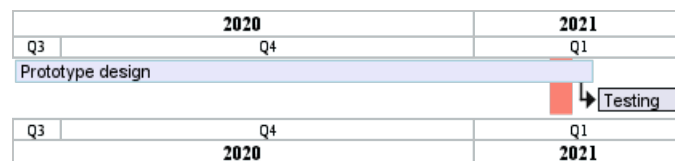
16.12.12 ズームあり

```
@startgantt
projectscale quarterly zoom 7
Project starts the 20th of september 2020
[Prototype design] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 20 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-01-22 are named [End's committee]

2021-01-18 to 2021-01-22 are colored in salmon

```
@endgantt
```



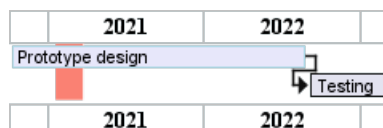
16.12.13 年単位スケールでのズーム

16.12.14 ズームなし

```
@startgantt
projectscale yearly
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon

```
@endgantt
```



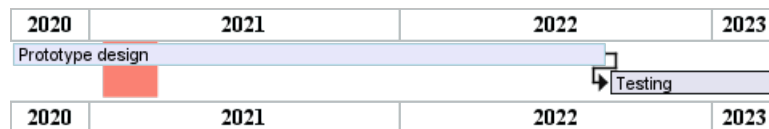
16.12.15 ズームあり

```
@startgantt
projectscale yearly zoom 2
Project starts the 1st of october 2020
[Prototype design] as [TASK1] lasts 700 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 200 days
[TASK1]->[Testing]
```

2021-01-18 to 2021-03-22 are colored in salmon

```
@endgantt
```

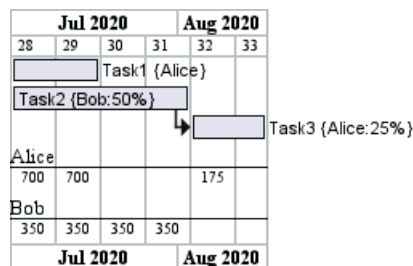




16.13 Weekscale with Weeknumbers or Calendar Date

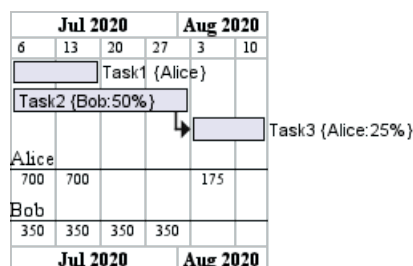
16.13.1 With Weeknumbers (*by default*)

```
@startgantt
printscale weekly
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```



16.13.2 With Calendar Date

```
@startgantt
printscale weekly with calendar date
Project starts the 6th of July 2020
[Task1] on {Alice} lasts 2 weeks
[Task2] on {Bob:50%} lasts 2 weeks
then [Task3] on {Alice:25%} lasts 3 days
@endgantt
```



[Ref. QA-11630]

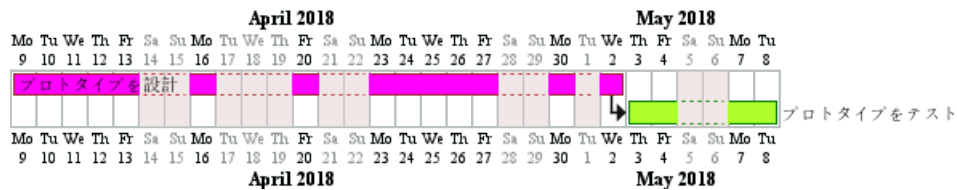
16.14 休業日

特定の曜日・日付を休業日に指定できます。

```
@startgantt
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[プロトタイプを設計] lasts 14 days
[プロトタイプをテスト] lasts 4 days
[プロトタイプをテスト] starts at [プロトタイプを設計]'s end
```



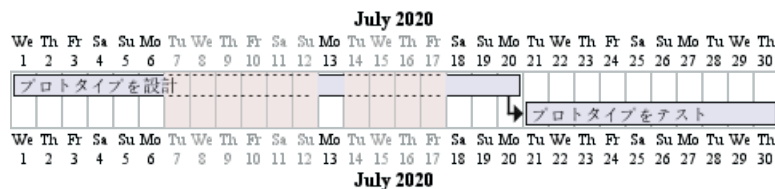
```
[プロトタイプを設計] is colored in Fuchsia/FireBrick
[プロトタイプをテスト] is colored in GreenYellow/Green
@endganttt
```



さらに、休業期間中の特定の日だけを休業日にすることができます。

```
@startganttt
2020-07-07 to 2020-07-17 is closed
2020-07-13 is open

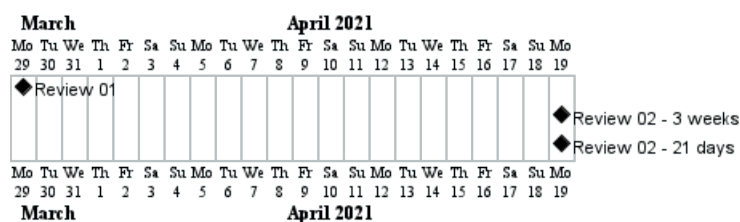
Project starts the 2020-07-01
[プロトタイプを設計] lasts 10 days
Then [プロトタイプをテスト] lasts 10 days
@endganttt
```



16.15 休業日に基づく 1 週間 (week) の定義

1 週間 (week) は、週に含まれる休業日を除く日数を表します:

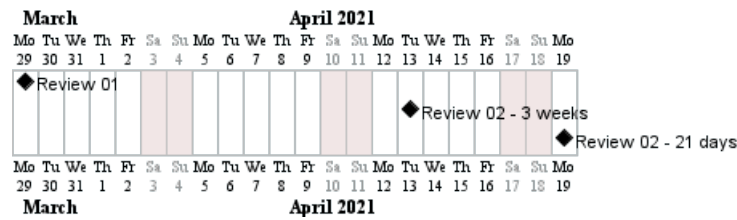
```
@startganttt
Project starts 2021-03-29
[Review 01] happens at 2021-03-29
[Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
[Review 02 - 21 days] happens on 21 days after [Review 01]'s end
@endganttt
```



土曜日と日曜日を休業日に指定した場合、1 週間 (week) は 5 日に相当するため、次のようになります:

```
@startganttt
Project starts 2021-03-29
saturday are closed
sunday are closed
[Review 01] happens at 2021-03-29
[Review 02 - 3 weeks] happens on 3 weeks after [Review 01]'s end
[Review 02 - 21 days] happens on 21 days after [Review 01]'s end
@endganttt
```





[Ref. QA-13434]

16.16 Working days

It is possible to manage working days.

```
@startgantt
```

```
saturday are closed
```

```
sunday are closed
```

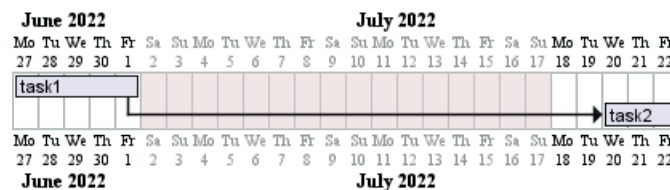
```
2022-07-04 to 2022-07-15 is closed
```

```
Project starts 2022-06-27
```

```
[task1] starts at 2022-06-27 and lasts 1 week
```

```
[task2] starts 2 working days after [task1]'s end and lasts 3 days
```

```
@endgantt
```



[Ref. QA-16188]

16.17 簡単なタスク継承

then を使えば、連続したタスクを表すことができます。

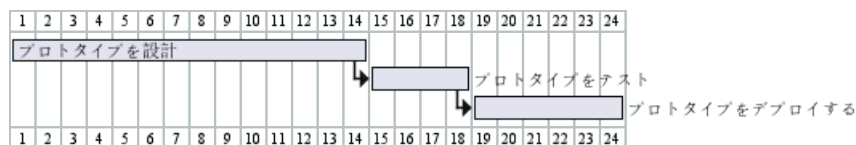
```
@startgantt
```

```
 [プロトタイプを設計] lasts 14 days
```

```
then [プロトタイプをテスト] lasts 4 days
```

```
then [プロトタイプをデプロイする] lasts 6 days
```

```
@enduml
```



矢印-> を使っても表せます。

```
@startgantt
```

```
 [プロトタイプを設計] lasts 14 days
```

```
 [プロトタイプをビルド] lasts 4 days
```

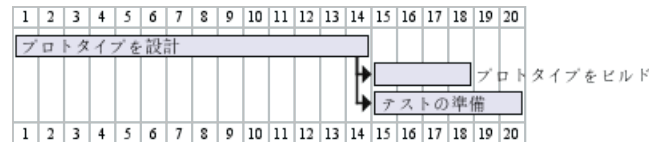
```
 [テストの準備] lasts 6 days
```

```
 [プロトタイプを設計] -> [プロトタイプをビルド]
```

```
 [プロトタイプを設計] -> [テストの準備]
```

```
@endgantt
```

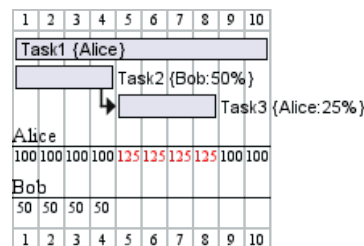




16.18 リソースを使用する

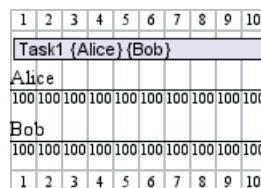
on キーワードとリソース名を波括弧に入れて記述することで、リソースを使用するタスクを表すことができます。

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt
```



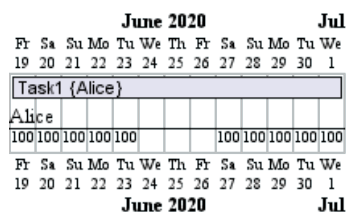
タスクには複数のリソースを割り当てることができます。

```
@startgantt
[Task1] on {Alice} {Bob} lasts 20 days
@endgantt
```



リソースを特定の日だけ割り当てないことができます。

```
@startgantt
project starts on 2020-06-19
[Task1] on {Alice} lasts 10 days
{Alice} is off on 2020-06-24 to 2020-06-26
@endgantt
```



16.19 Hide resources

16.19.1 Without any hiding (by default)

```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
```



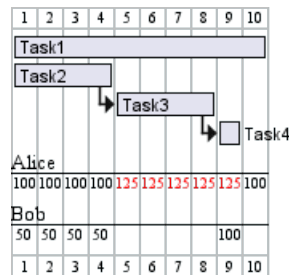
```
@endgantt
```



16.19.2 Hide resources names

You can hide resources names and percentage, on tasks, using the `hide ressources names` keywords.

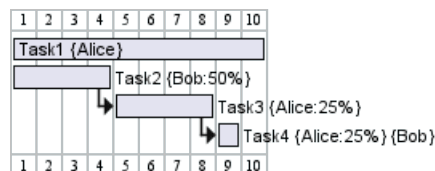
```
@startgantt
hide ressources names
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```



16.19.3 Hide resources footbox

You can also hide resources names on bottom of the diagram using the `hide ressources footbox` keywords.

```
@startgantt
hide ressources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt
```



16.19.4 Hide the both (resources names and resources footbox)

You can also hide the both.

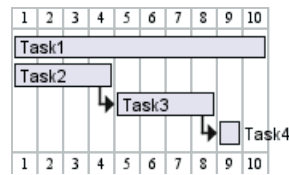
```
@startgantt
hide ressources names
hide ressources footbox
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
```



```

then [Task3] on {Alice:25%} lasts 1 days
then [Task4] on {Alice:25%} {Bob} lasts 1 days
@endgantt

```



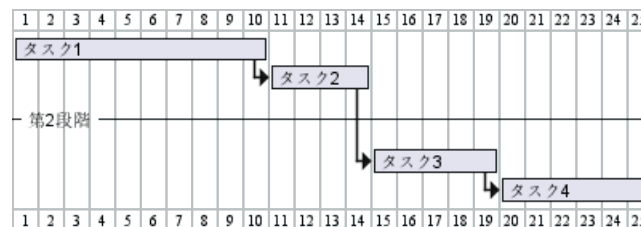
16.20 区切り線

--を使って、タスクを区切ることができます。

```

@startgantt
[タスク1] lasts 10 days
then [タスク2] lasts 4 days
-- 第2段階 --
then [タスク3] lasts 5 days
then [タスク4] lasts 6 days
@endgantt

```



16.21 Vertical Separator

You can add Vertical Separators with the syntax: Separator just [at].

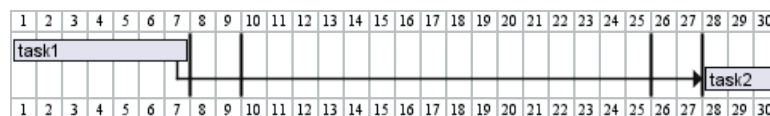
```

@startgantt
[task1] lasts 1 week
[task2] starts 20 days after [task1]'s end and lasts 3 days

Separator just at [task1]'s end
Separator just 2 days after [task1]'s end

Separator just at [task2]'s start
Separator just 2 days before [task2]'s start
@endgantt

```



[Ref. QA-16247]

16.22 複雑な例

1 つのタスクに対して、and で同時に複数の設定を行えます。

依存関係に遅延日数を加えることもできます。

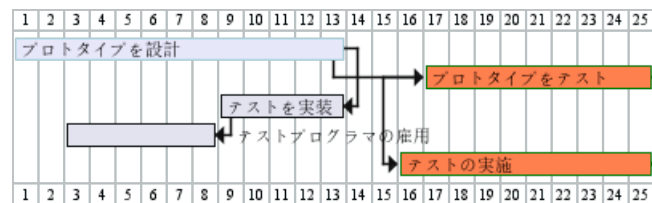
```

@startgantt
[プロトタイプを設計] lasts 13 days and is colored in Lavender/LightBlue
[プロトタイプをテスト] lasts 9 days and is colored in Coral/Green and starts 3 days after [プロトタイプを設計]
[テストを実装] lasts 5 days and ends at [プロトタイプを設計]'s end

```



[テストプログラムの雇用] lasts 6 days and ends at [テストを実装]'s start
 [テストの実施] is colored in Coral/Green
 [テストの実施] starts 1 day before [プロトタイプをテスト]'s start and ends at [プロトタイプをテスト]
 @endgantt



16.23 コメント

共通コマンドのページに書かれている通り、`blockquote` Everything that starts with `simple quote` ' is a comment.

You can also put comments on several lines using `/'` to start and `/'` to end. `blockquote` (つまり、コメント行は、(空白文字を除くと) シングルクォーテーション' で始まっている必要があります。)

```
@startgantt
' これはコメントです
```

```
[T1] lasts 3 days
```

```
/' このコメントは
複数行にわたっています '/'
```

```
[T2] starts at [T1]'s end and lasts 1 day
@endgantt
```

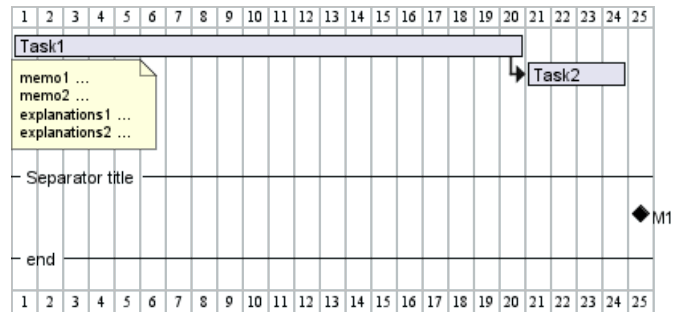


16.24 スタイルの使用

16.24.1 スタイルなし (デフォルト)

```
@startgantt
[Task1] lasts 20 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt
```





16.24.2 スタイルあり

スタイルを使って要素のレンダリングを変更できます。

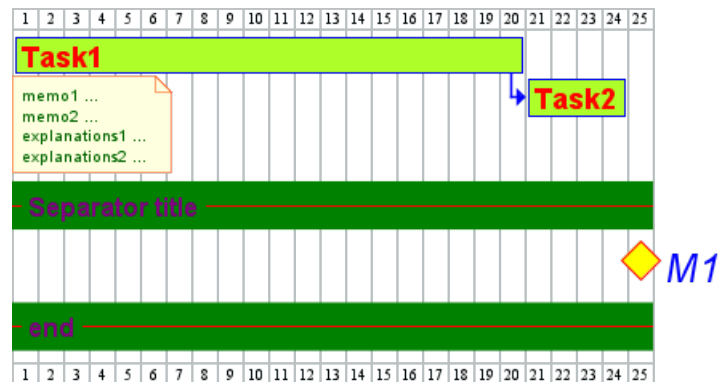
```
@startgantt
<style>
ganttDiagram {
task {
  FontName Helvetica
  FontColor red
  FontSize 18
  FontStyle bold
  BackGroundColor GreenYellow
  LineColor blue
}
milestone {
  FontColor blue
  FontSize 25
  FontStyle italic
  BackGroundColor yellow
  LineColor red
}
note {
  FontColor DarkGreen
  FontSize 10
  LineColor OrangeRed
}
arrow {
  FontName Helvetica
  FontColor red
  FontSize 18
  FontStyle bold
  BackGroundColor GreenYellow
  LineColor blue
}
separator {
  LineColor red
  BackGroundColor green
  FontSize 16
  FontStyle bold
  FontColor purple
}
}
</style>
[Task1] lasts 20 days
note bottom
  memo1 ...
  memo2 ...
```



```

    explanations1 ...
    explanations2 ...
end note
[Task2] lasts 4 days
[Task1] -> [Task2]
-- Separator title --
[M1] happens on 5 days after [Task1]'s end
-- end --
@endgantt

```



[Ref. QA-10835, QA-12045, QA-11877 and PR-438]

16.24.3 スタイルあり（完全な例）

```

@startgantt
<style>
ganttDiagram {
task {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
}
milestone {
    FontColor blue
    FontSize 25
    FontStyle italic
    BackGroundColor yellow
    LineColor red
}
note {
    FontColor DarkGreen
    FontSize 10
    LineColor OrangeRed
}
arrow {
    FontName Helvetica
    FontColor red
    FontSize 18
    FontStyle bold
    BackGroundColor GreenYellow
    LineColor blue
    LineStyle 8.0;13.0
    LineThickness 3.0
}

```



```

separator {
  BackgroundColor lightGreen
  LineStyle 8.0;3.0
  LineColor red
  LineThickness 1.0
  FontSize 16
  FontStyle bold
  FontColor purple
  Margin 5
  Padding 20
}
timeline {
  BackgroundColor Bisque
}
closed {
  BackgroundColor pink
  FontColor red
}
}
</style>
Project starts the 2020-12-01

[Task1] lasts 10 days
sunday are closed

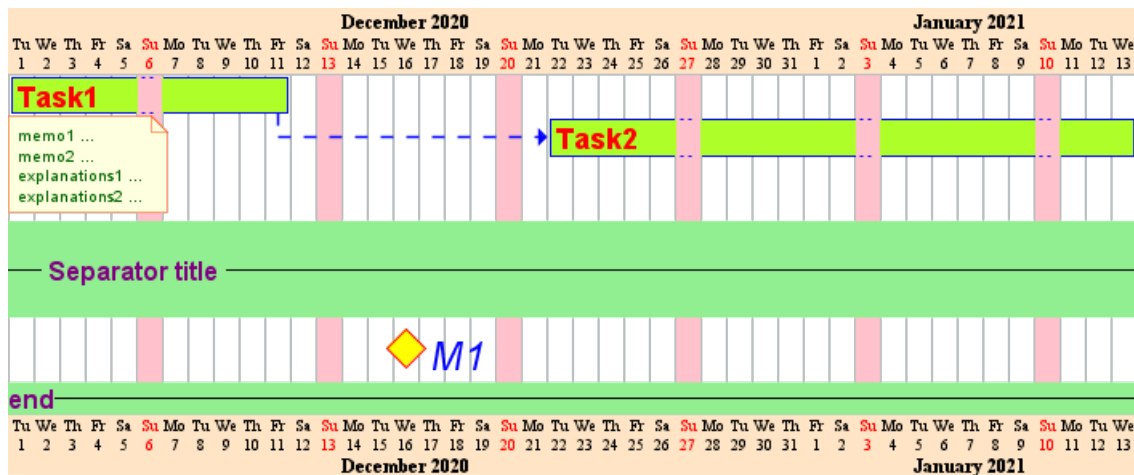
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

[Task2] lasts 20 days
[Task2] starts 10 days after [Task1]'s end
-- Separator title --
[M1] happens on 5 days after [Task1]'s end

<style>
separator {
  LineColor black
  Margin 0
  Padding 0
}
</style>

-- end --
@endgantt

```



[Ref. QA-13570, QA-13672]

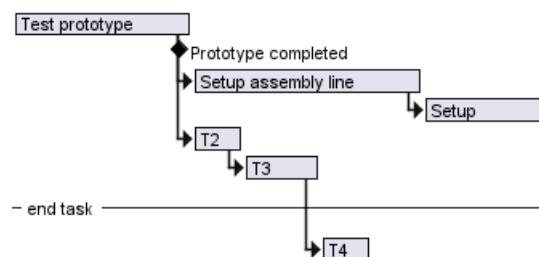
TODO: DONE 区切り線のスタイルと矢印のすべてのスタイル (太さ等) に感謝します

16.24.4 簡潔なスタイル

スタイルを使用して、ガントチャートを簡潔な表示にすることができます。タスク、依存関係、関連した期間のみを表示しますが、実際の開始日や実際のスケールは表示しません:

```
@startgantt
<style>
ganttDiagram {
  timeline {
    LineColor transparent
    FontColor transparent
  }
}
</style>
```

```
hide footbox
[Test prototype] lasts 7 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 9 days
[Setup assembly line] starts at [Test prototype]'s end
then [Setup] lasts 5 days
[T2] lasts 2 days and starts at [Test prototype]'s end
then [T3] lasts 3 days
-- end task --
then [T4] lasts 2 days
@endgantt
```



[Ref. QA-13971]

別の例:

```
@startgantt
```

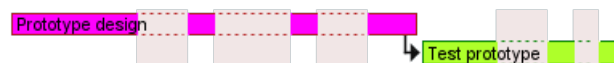


```

<style>
ganttDiagram {
  timeline {
    LineColor transparent
    FontColor transparent
  }
  closed {
    FontColor transparent
  }
}
</style>

hide footbox
project starts the 2018/04/09
saturday are closed
sunday are closed
2018/05/01 is closed
2018/04/17 to 2018/04/19 is closed
[Prototype design] lasts 9 days
[Test prototype] lasts 5 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt

```



[Ref. QA-13464]

16.25 ノートの追加

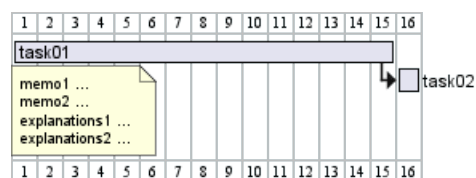
```

@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...
  explanations1 ...
  explanations2 ...
end note

```

[task01] -> [task02]

```
@endgantt
```



期間の重なりがある場合の例。

```

@startgantt
[task01] lasts 15 days
note bottom
  memo1 ...
  memo2 ...

```



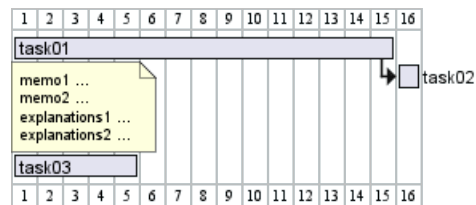
```

    explanations1 ...
    explanations2 ...
end note

[task01] -> [task02]
[task03] lasts 5 days

@endgantt

```



```

@startgantt

-- test01 --

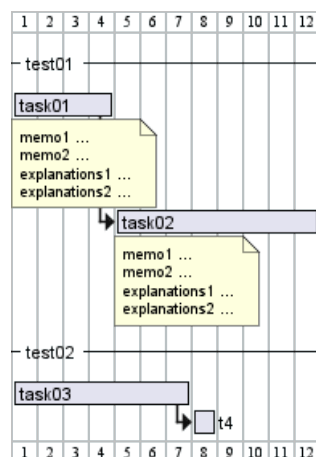
[task01] lasts 4 days
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note

[task02] lasts 8 days
[task01] -> [task02]
note bottom
'note left
memo1 ...
memo2 ...
explanations1 ...
explanations2 ...
end note

-- test02 --

[task03] as [t3] lasts 7 days
[t3] -> [t4]
@endgantt

```



TODO: DONE *Thanks for correction (of #386 on v1.2020.18) when overlapping*

@startgantt

Project starts 2020-09-01

[taskA] starts 2020-09-01 and lasts 3 days

[taskB] starts 2020-09-10 and lasts 3 days

[taskB] displays on same row as [taskA]

[task01] starts 2020-09-05 and lasts 4 days

then [task02] lasts 8 days

note bottom

note for task02

more notes

end note

then [task03] lasts 7 days

note bottom

note for task03

more notes

end note

-- separator --

[taskC] starts 2020-09-02 and lasts 5 days

[taskD] starts 2020-09-09 and lasts 5 days

[taskD] displays on same row as [taskC]

[task 10] starts 2020-09-05 and lasts 5 days

then [task 11] lasts 5 days

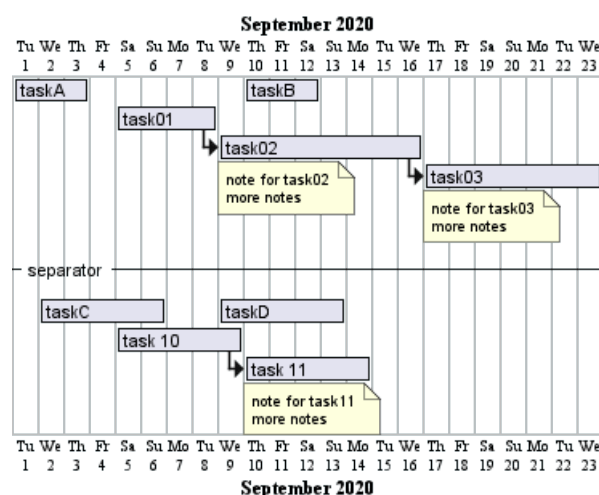
note bottom

note for task11

more notes

end note

@endgantt



16.26 タスクの中断

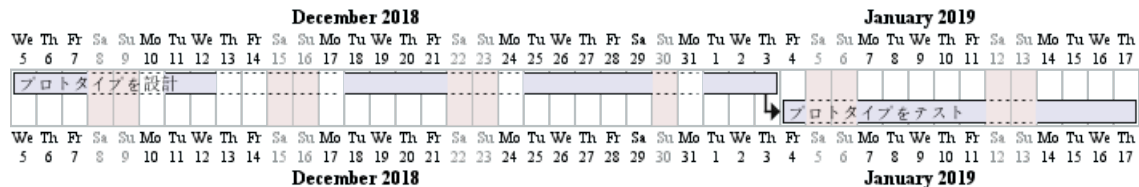
@startgantt

Project starts the 5th of december 2018

saturday are closed



```
sunday are closed
2018/12/29 is opened
[プロトタイプを設計] lasts 17 days
[プロトタイプを設計] pauses on 2018/12/13
[プロトタイプを設計] pauses on 2018/12/14
[プロトタイプを設計] pauses on monday
[プロトタイプをテスト] starts at [プロトタイプを設計]'s end and lasts 2 weeks
@endgantt
```

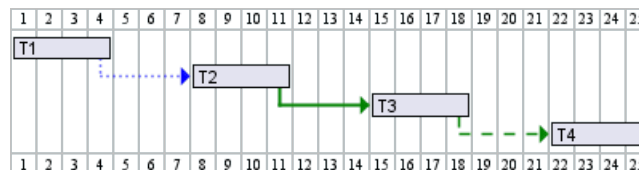


16.27 リンクの色の変更

次の構文でリンクの色を変更できます:

- with `<color> <style> link`

```
@startgantt
[T1] lasts 4 days
[T2] lasts 4 days and starts 3 days after [T1]'s end with blue dotted link
[T3] lasts 4 days and starts 3 days after [T2]'s end with green bold link
[T4] lasts 4 days and starts 3 days after [T3]'s end with green dashed link
@endgantt
```



- もしくは、矢印のスタイルを直接指定します

```
@startgantt
<style>
ganttDiagram {
  arrow {
    LineColor blue
  }
}
</style>
[プロトタイプを設計] lasts 7 days
[プロトタイプを実装] lasts 4 days
[テストの準備] lasts 6 days
[プロトタイプを設計] -[#FF00FF]-> [プロトタイプを実装]
[プロトタイプを設計] -[dotted]-> [テストの準備]
Then [テスト実施] lasts 4 days
@endgantt
```



[Ref. QA-13693]

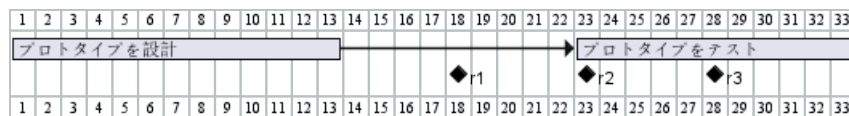


16.28 タスクやマイルストーンを同じ行に表示する

次の記法でタスクやマイルストーンを同じ行に表示することができます:

- [T|M] displays on same row as [T|M]

```
@startgantt
[プロトタイプを設計] lasts 13 days
[プロトタイプをテスト] lasts 4 days and 1 week
[プロトタイプをテスト] starts 1 week and 2 days after [プロトタイプを設計]'s end
[プロトタイプをテスト] displays on same row as [プロトタイプを設計]
[r1] happens on 5 days after [プロトタイプを設計]'s end
[r2] happens on 5 days after [r1]'s end
[r3] happens on 5 days after [r2]'s end
[r2] displays on same row as [r1]
[r3] displays on same row as [r1]
@endgantt
```

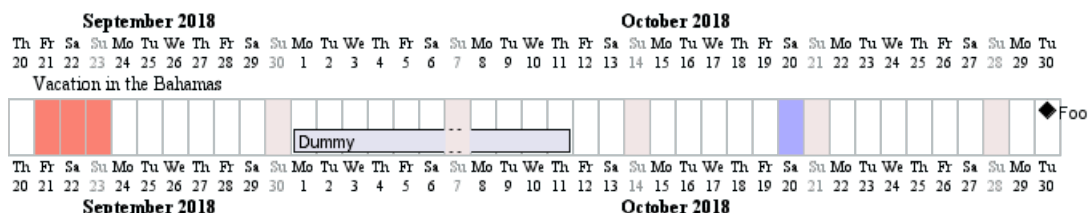


16.29 今日に色を付ける

```
@startgantt
Project starts the 20th of september 2018
sunday are close
2018/09/21 to 2018/09/23 are colored in salmon
2018/09/21 to 2018/09/30 are named [Vacation in the Bahamas]
```

```
today is 30 days after start and is colored in #AAF
[Foo] happens 40 days after start
[Dummy] lasts 10 days and starts 10 days after start
```

```
@endgantt
```



16.30 2つのマイルストーンに挟まれたタスク

```
@startgantt
project starts on 2020-07-01
[開始] happens 2020-07-03
[終了] happens 2020-07-13
[プロトタイプを設計] occurs from [開始] to [終了]
@endgantt
```



16.31 Grammar and verbal form

Verbal form	Example
[T] starts	
[M] happens	

16.32 タイトル、ヘッダ、フッタ、キャプション、凡例を追加する

```
@startgantt
```

```
header これはヘッダです
```

```
footer これはフッタです
```

```
title これはタイトルです
```

```
[プロトタイプ設計] lasts 13 days
```

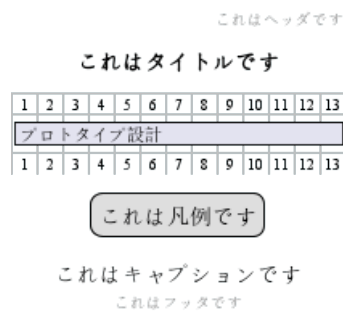
```
legend
```

```
これは凡例です
```

```
end legend
```

```
caption これはキャプションです
```

```
@endgantt
```



(参考：共通コマンド)

16.33 下部のボックスの除去（すべてのスケールでの例）

hide footbox キーワードを使用して、ガントチャート下部のボックスを取り除くことができます。（シーケンス図と同様）

例：

- 日単位（プロジェクト開始日の指定無し）

```
@startgantt
```

```
hide footbox
```

```
title Foot Box removed
```

```
[プロトタイプ設計] lasts 15 days
```

```
[プロトタイプをテスト] lasts 10 days
```

```
@endgantt
```

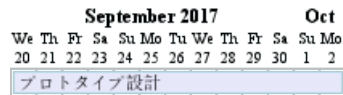


- 日単位

```
@startgantt
```

```
Project starts the 20th of september 2017
[プロトタイプ設計] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
```

```
hide footbox
@endgantt
```



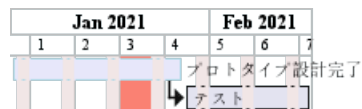
- 週単位

```
@startgantt
hide footbox
```

```
printscale weekly
saturday are closed
sunday are closed
```

```
Project starts the 1st of january 2021
[プロトタイプ設計完了] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[テスト] lasts 14 days
[TASK1]->[テスト]
```

```
2021-01-18 to 2021-01-22 are named [完了審査会]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



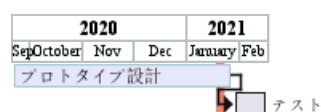
- 月単位

```
@startgantt
```

```
hide footbox
```

```
projectscale monthly
Project starts the 20th of september 2020
[プロトタイプ設計] as [TASK1] lasts 130 days
[TASK1] is colored in Lavender/LightBlue
[テスト] lasts 20 days
[TASK1]->[テスト]
```

```
2021-01-18 to 2021-01-22 are named [完了審査会]
2021-01-18 to 2021-01-22 are colored in salmon
@endgantt
```



- 四半期単位

```
@startgantt
```



```
hide footbox
```

```
projectscale quarterly
```

```
Project starts the 1st of october 2020
```

```
[プロトタイプ設計] as [TASK1] lasts 700 days
```

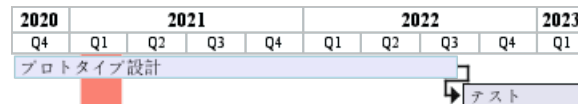
```
[TASK1] is colored in Lavender/LightBlue
```

```
[テスト] lasts 200 days
```

```
[TASK1]->[テスト]
```

```
2021-01-18 to 2021-03-22 are colored in salmon
```

```
@endgantt
```



- 年単位

```
@startgantt
```

```
hide footbox
```

```
projectscale yearly
```

```
Project starts the 1st of october 2020
```

```
[プロトタイプ設計] as [TASK1] lasts 700 days
```

```
[TASK1] is colored in Lavender/LightBlue
```

```
[テスト] lasts 200 days
```

```
[TASK1]->[テスト]
```

```
2021-01-18 to 2021-03-22 are colored in salmon
```

```
@endgantt
```



16.34 カレンダーの言語

ガントチャートのカレンダーの言語を変更するには、`language <xx>` コマンドを使用します。<xx> には ISO 639 言語コードが入ります。

16.34.1 英語 (*en*、デフォルト)

```
@startgantt
```

```
saturday are closed
```

```
sunday are closed
```

```
Project starts 2021-01-01
```

```
[Prototype design end] as [TASK1] lasts 19 days
```

```
[TASK1] is colored in Lavender/LightBlue
```

```
[Testing] lasts 14 days
```

```
[TASK1]->[Testing]
```

```
2021-01-18 to 2021-01-22 are colored in salmon
```

```
@endgantt
```




```

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

```

```

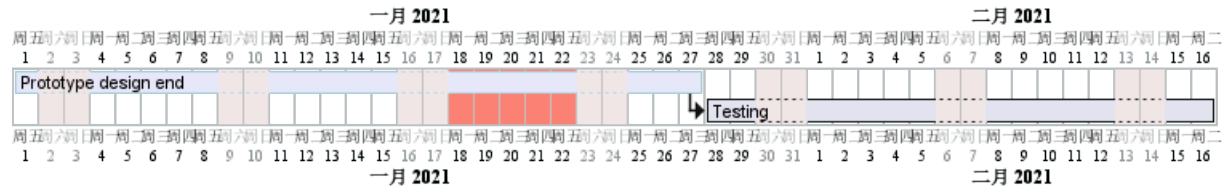
2021-01-18 to 2021-01-22 are colored in salmon

```

```

@endgantt

```



16.34.5 韓国語 (ko)

```

@startgantt
language ko
saturday are closed
sunday are closed

```

```

Project starts 2021-01-01
[Prototype design end] as [TASK1] lasts 19 days
[TASK1] is colored in Lavender/LightBlue
[Testing] lasts 14 days
[TASK1]->[Testing]

```

```

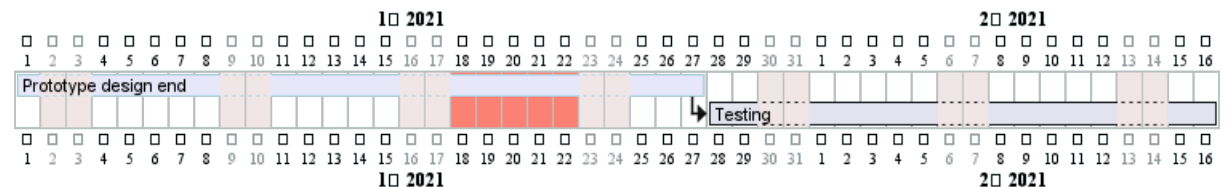
2021-01-18 to 2021-01-22 are colored in salmon

```

```

@endgantt

```



16.35 タスクやマイルストーンの削除

いくつかの「タスク」や「マイルストーン」を「通常完了」ではなく「deleted」とマークして、廃棄や延期などの可能性のあるタスクを区別することができます。

```

@startgantt
[Prototype design] lasts 1 weeks
then [Prototype completed] lasts 4 days
[End Prototype completed] happens at [Prototype completed]'s end
then [Test prototype] lasts 5 days
[End Test prototype] happens at [Test prototype]'s end

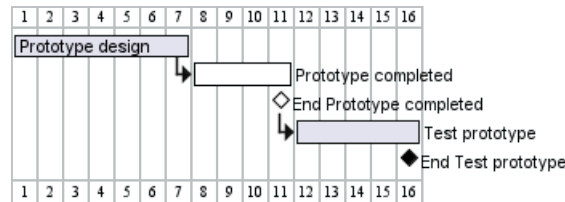
```

```

[Prototype completed] is deleted
[End Prototype completed] is deleted
@endgantt

```



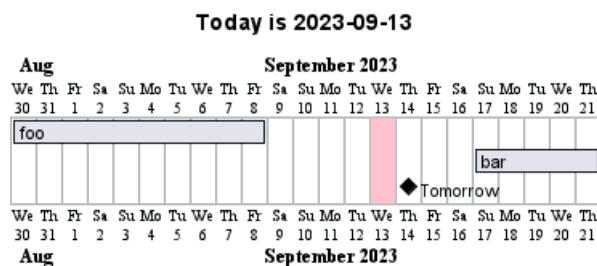


[Ref. QA-9129]

16.36 Start a project, a task or a milestone a number of days before or after today

You can start a project, a task or a milestone a number of days before or after today, using the builtin functions `%now` and `%date`:

```
@startgantt
title Today is %date("YYYY-MM-dd")
!$now = %now()
!$past = %date("YYYY-MM-dd", $now - 14*24*3600)
Project starts $past
today is colored in pink
[foo] lasts 10 days
[bar] lasts 5 days and starts %date("YYYY-MM-dd", $now + 4*24*3600)
[Tomorrow] happens %date("YYYY-MM-dd", $now + 1*24*3600)
@endgantt
```



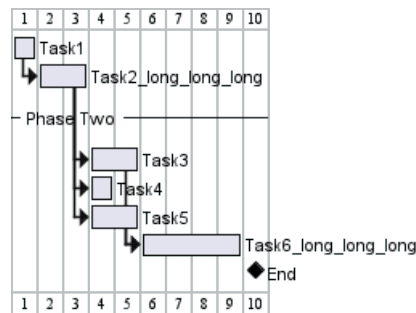
[Ref. QA-16285]

16.37 Change Label position

16.37.1 The labels are near elements (*by default*)

```
@startgantt
[Task1] lasts 1 days
then [Task2_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



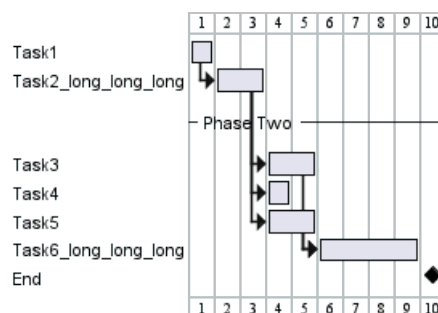


To change the label position, you can use the command `label`:

16.37.2 Label on first column

- Left aligned

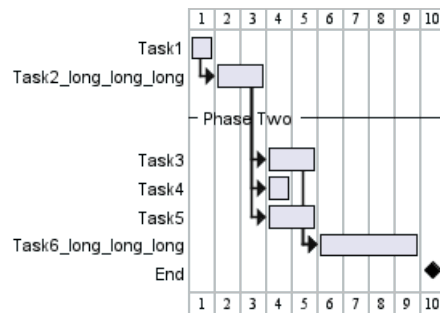
```
@startgantt
Label on first column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

```
@startgantt
Label on first column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```

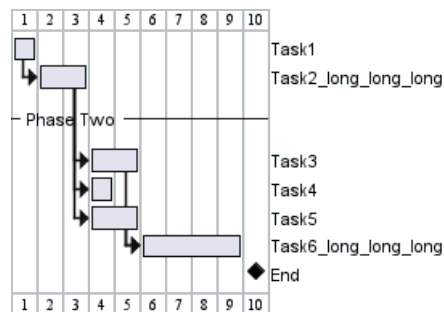




16.37.3 Label on last column

- Left aligned

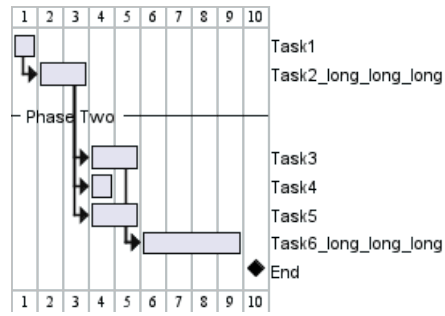
```
@startgantt
Label on last column and left aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```



- Right aligned

```
@startgantt
Label on last column and right aligned
[Task1] lasts 1 days
then [Task2_long_long_long] as [T2] lasts 2 days
-- Phase Two --
then [Task3] as [T3] lasts 2 days
[Task4] as [T4] lasts 1 day
[Task5] as [T5] lasts 2 days
[T2] -> [T4]
[T2] -> [T5]
[Task6_long_long_long] as [T6] lasts 4 days
[T3] -> [T6]
[T5] -> [T6]
[End] happens 1 day after [T6]'s end
@endgantt
```





[Ref. QA-12433]

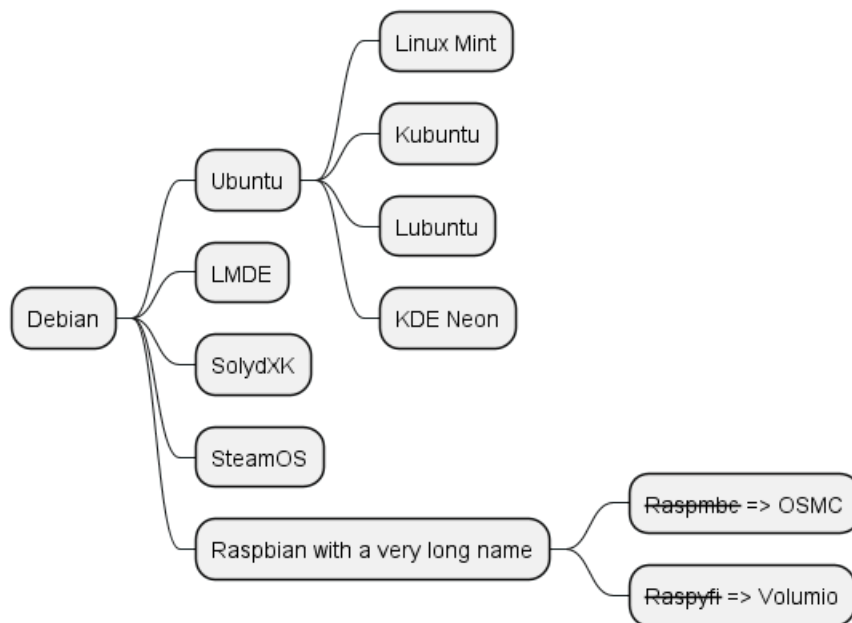
17 マインドマップ

マインドマップはまだベータ版です。文法は予告なく変更するかもしれません。

17.1 OrgMode の文法

OrgMode と互換性のある文法です。

```
@startmindmap
* Debian
** Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** LMDE
** SolydXK
** SteamOS
** Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio
@endmindmap
```

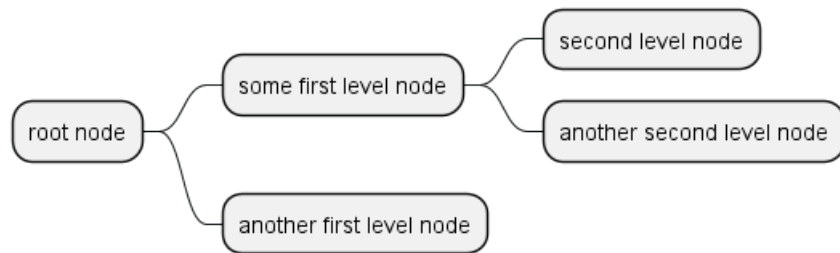


17.2 Markdown 構文

この構文は、Markdown と互換性があります。

```
@startmindmap
* root node
* some first level node
* second level node
* another second level node
* another first level node
@endmindmap
```



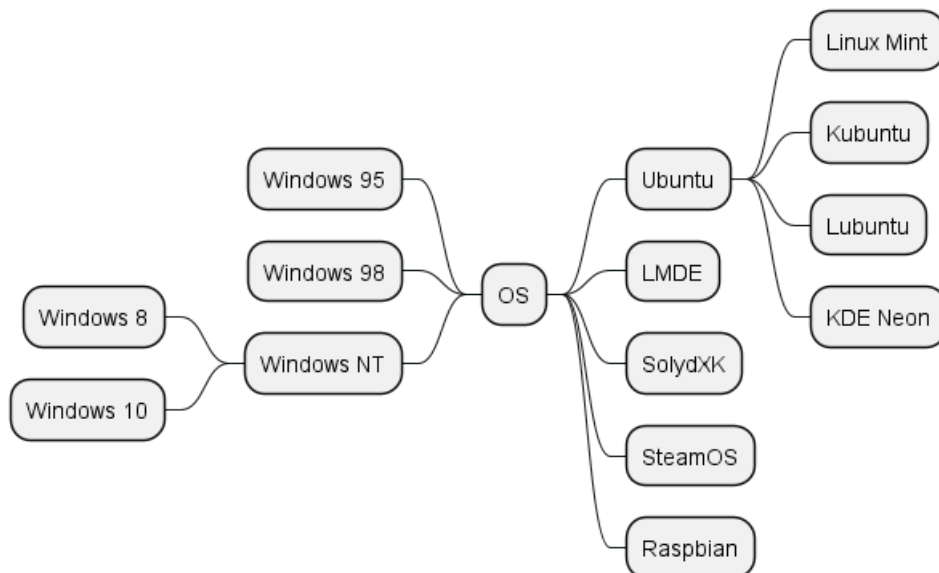


17.3 算術記号による表記

枝を左右どちらの側に伸ばすかを、以下のように算術記号で指定できます。

```

@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
--- Windows 10
@endmindmap
  
```



17.4 複数行

: と; を使って、複数行の箱を作ることができます。

```

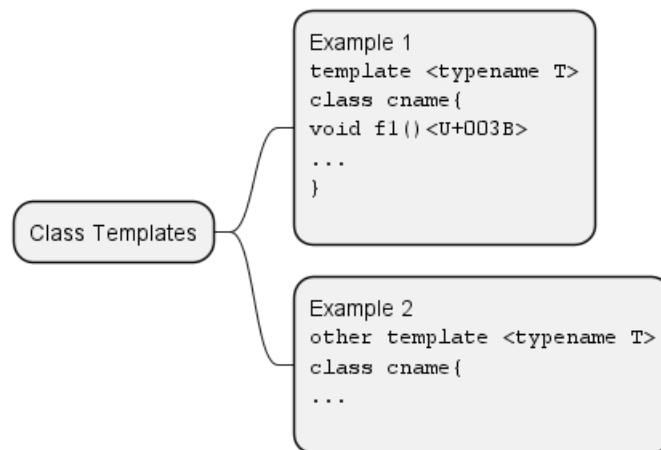
@startmindmap
* Class Templates
  
```



```

**:Example 1
<code>
template <typename T>
class cname{
void f1()<U+003B>
...
}
</code>
;
**:Example 2
<code>
other template <typename T>
class cname{
...
</code>
;
@endmindmap

```



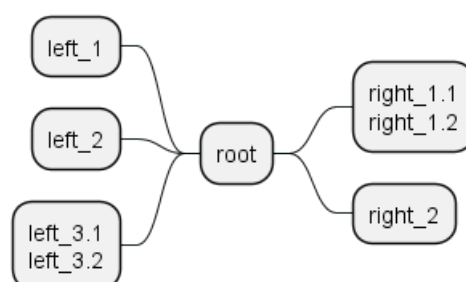
```

@startmindmap
+ root
**:right_1.1
right_1.2;
++ right_2

left side

-- left_1
-- left_2
**:left_3.1
left_3.2;
@endmindmap

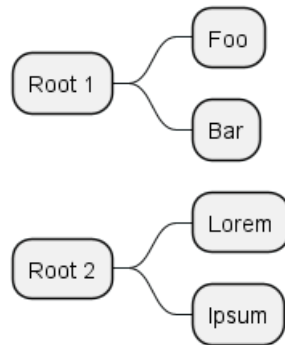
```



17.5 Multiroot Mindmap

You can create multiroot mindmap, as:

```
@startmindmap
* Root 1
** Foo
** Bar
* Root 2
** Lorem
** Ipsum
@endmindmap
```



[Ref. QH-773]

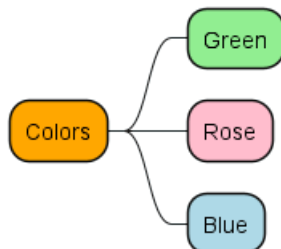
17.6 色

ノードの色を変えることができます。

17.6.1 インラインの色指定

- OrgMode の文法

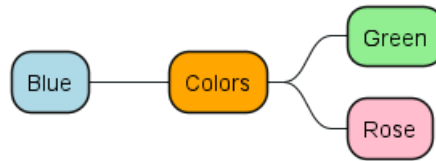
```
@startmindmap
* [#Orange] Colors
** [#lightgreen] Green
** [#FFBCC] Rose
** [#lightblue] Blue
@endmindmap
```



- 算術記号による表記

```
@startmindmap
+ [#Orange] Colors
++ [#lightgreen] Green
++ [#FFBCC] Rose
-- [#lightblue] Blue
@endmindmap
```

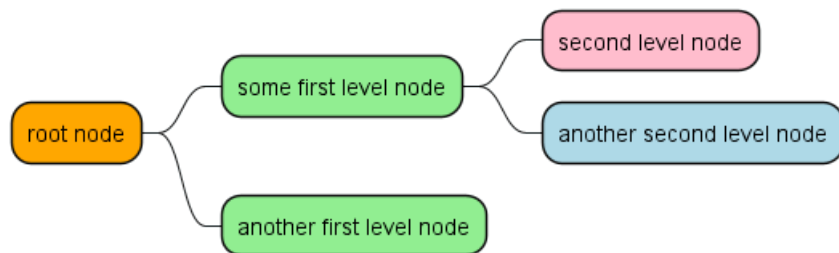




- Markdown の文法

```

@startmindmap
* [#Orange] root node
  * [#lightgreen] some first level node
    * [#FFBBCC] second level node
    * [#lightblue] another second level node
  * [#lightgreen] another first level node
@endmindmap
  
```



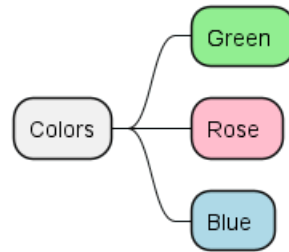
17.6.2 スタイルによる色指定

- OrgMode の文法

```

@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
* Colors
** Green <<green>>
** Rose <<rose>>
** Blue <<your_style_name>>
@endmindmap
  
```

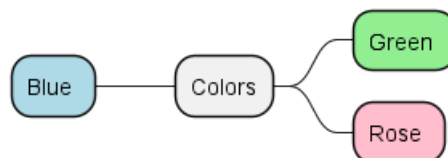




- 算術記号による表記

```

@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
+ Colors
++ Green <<green>>
++ Rose <<rose>>
-- Blue <<your_style_name>>
@endmindmap
  
```



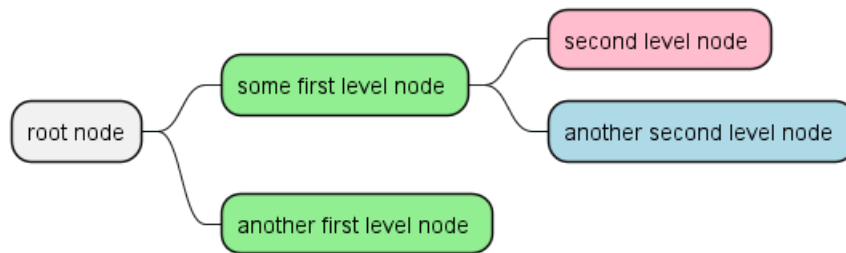
- Markdown の文法

```

@startmindmap
<style>
mindmapDiagram {
  .green {
    BackgroundColor lightgreen
  }
  .rose {
    BackgroundColor #FFBBCC
  }
  .your_style_name {
    BackgroundColor lightblue
  }
}
</style>
* root node
* some first level node <<green>>
* second level node <<rose>>
* another second level node <<your_style_name>>
  
```



```
* another first level node <<green>>
@endmindmap
```

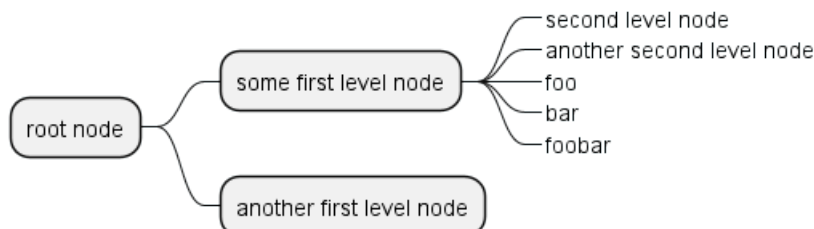


[Ref. GA-920]

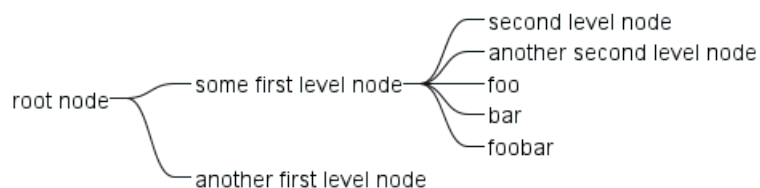
17.7 箱を消す

アンダースコア `_` を使うことで、箱を消すことができます。

```
@startmindmap
* root node
** some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
** another first level node
@endmindmap
```



```
@startmindmap
*_ root node
**_ some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
**_ another first level node
@endmindmap
```



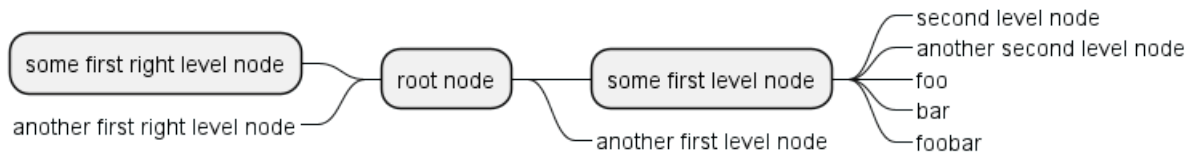
```
@startmindmap
+ root node
```



```

++ some first level node
+++_ second level node
+++_ another second level node
+++_ foo
+++_ bar
+++_ foobar
++_ another first level node
-- some first right level node
--_ another first right level node
@endmindmap

```



17.8 図の方向の変更

図の両側を使うことができます。

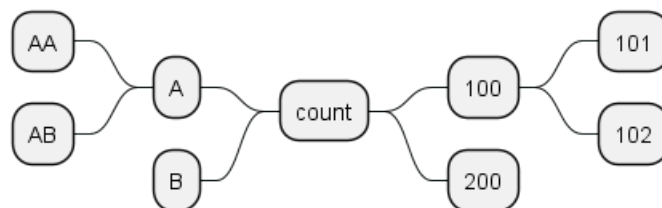
```

@startmindmap
* count
** 100
*** 101
*** 102
** 200

left side

** A
*** AA
*** AB
** B
@endmindmap

```



17.9 完全な例

```

@startmindmap
caption figure 1
title My super title

* <&flag>Debian
** <&globe>Ubuntu
*** Linux Mint
*** Kubuntu
*** Lubuntu
*** KDE Neon
** <&graph>LMDE

```



```

** <&pulse>SolydXK
** <&people>SteamOS
** <&star>Raspbian with a very long name
*** <s>Raspmbc</s> => OSMC
*** <s>Raspyfi</s> => Volumio

```

```

header
My super header
endheader

center footer My super footer

```

```

legend right
  Short
  legend
endlegend
@endmindmap

```

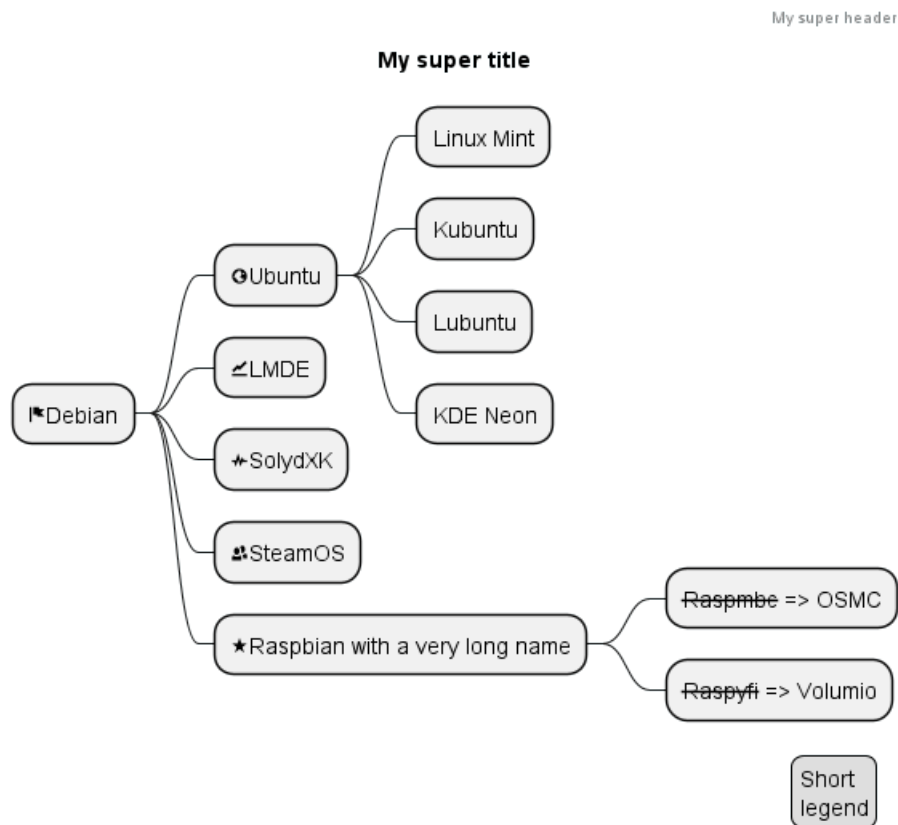


figure 1
My super footer

17.10 スタイル変更

17.10.1 ノード (node)、深さ (depth)

```

@startmindmap
<style>
mindmapDiagram {
  node {
    BackgroundColor lightGreen
  }
  :depth(1) {

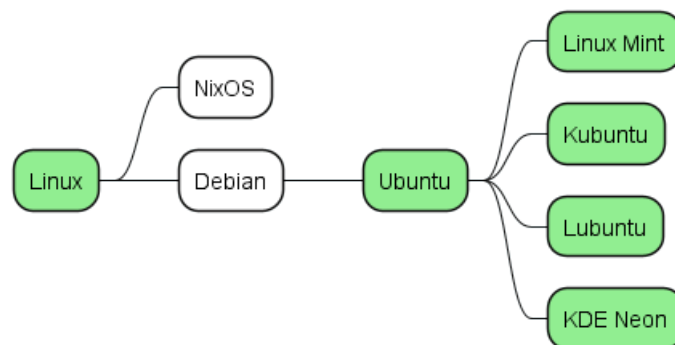
```



```

        BackGroundColor white
    }
}
</style>
* Linux
** NixOS
** Debian
*** Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap

```

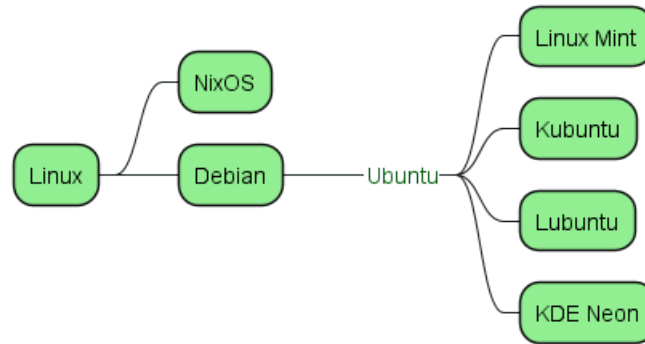


17.10.2 枠無し (boxless)

```

@startmindmap
<style>
mindmapDiagram {
    node {
        BackGroundColor lightGreen
    }
    boxless {
        FontColor darkgreen
    }
}
</style>
* Linux
** NixOS
** Debian
*** _ Ubuntu
**** Linux Mint
**** Kubuntu
**** Lubuntu
**** KDE Neon
@endmindmap

```



17.11 単語の折り返し

`MaximumWidth` で、最大幅を設定すると、自動的に単語を折り返すことができます。使用する単位はピクセルです。

@startmindmap

```

<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

leafNode {
    LineColor gold
    RoundCorner 0
    Padding 3
}

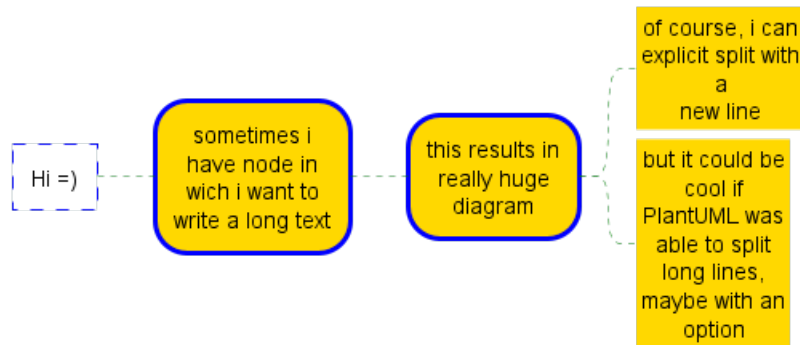
arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in wich i want to write a long text
*** this results in really huge diagram
  
```



```
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option

@endmindmap
```



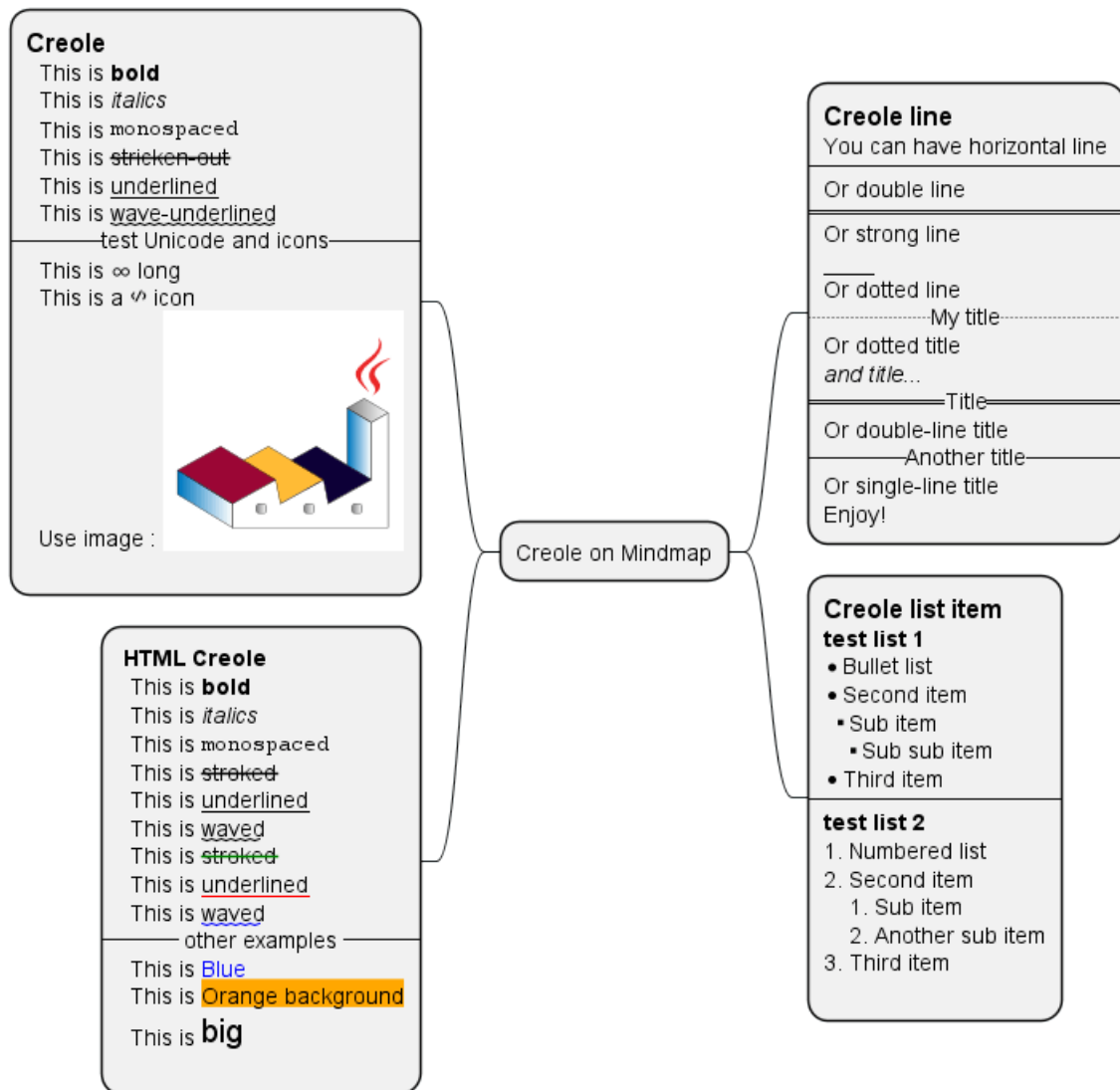
17.12 Creole on Mindmap diagram

You can use Creole or HTML Creole on Mindmap:

```
@startmindmap
* Creole on Mindmap
left side
**::=Creole
  This is bold
  This is italics
  This is "monospaced"
  This is stricken-out
  This is underlined
  This is ~wave-underlined~
--test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
;
**:: <b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
-- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
;
right side
**::=Creole line
You can have horizontal line
----
Or double line
=====
Or strong line
```



```
----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**:=Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endmindmap
```



[Ref. QA-17838]

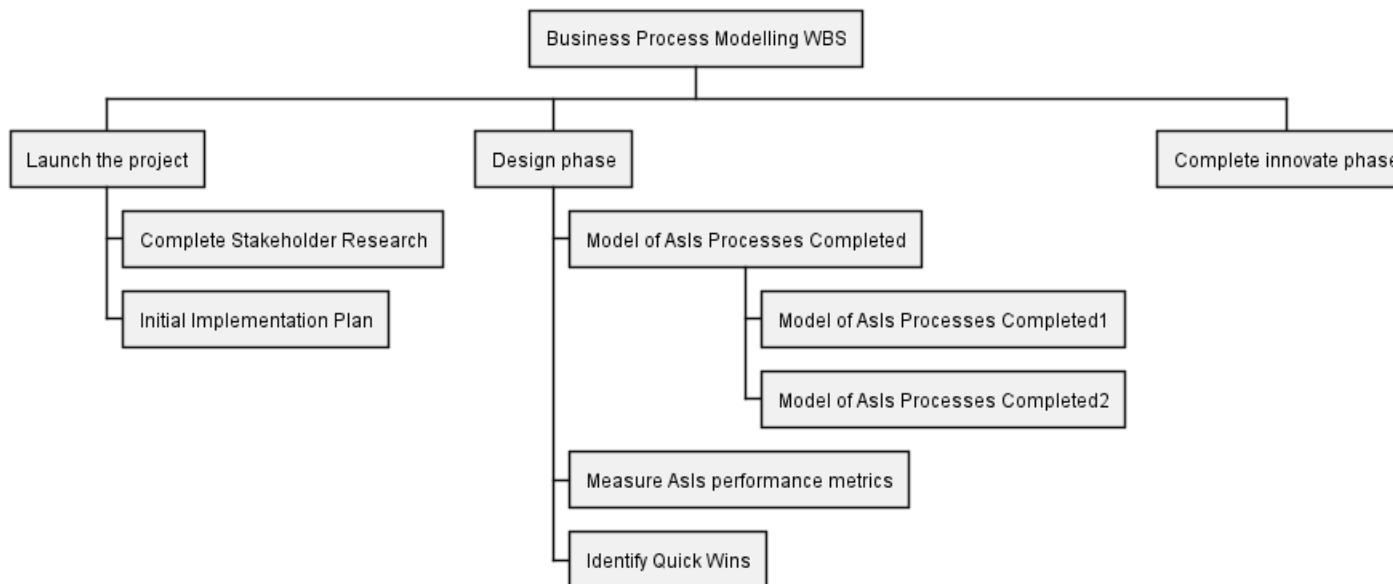
18 Work Breakdown Structure

WBS 図はまだベータ版です。文法は予告なく変更するかもしれません。

18.1 OrgMode の文法

OrgMode と互換性のある文法です。

```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
**** Model of AsIs Processes Completed1
**** Model of AsIs Processes Completed2
*** Measure AsIs performance metrics
*** Identify Quick Wins
** Complete innovate phase
@endwbs
```

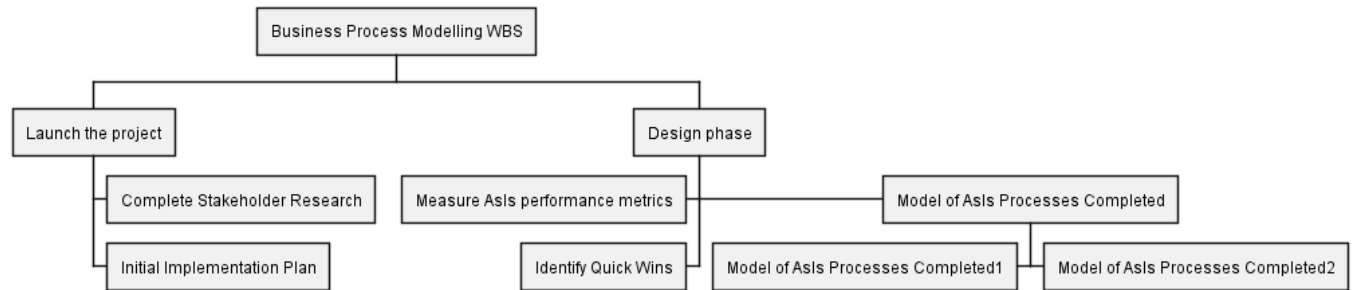


18.2 方向の変更

< と > を使うことで、方向を変更できます。

```
@startwbs
* Business Process Modelling WBS
** Launch the project
*** Complete Stakeholder Research
*** Initial Implementation Plan
** Design phase
*** Model of AsIs Processes Completed
****< Model of AsIs Processes Completed1
****> Model of AsIs Processes Completed2
***< Measure AsIs performance metrics
***< Identify Quick Wins
@endwbs
```



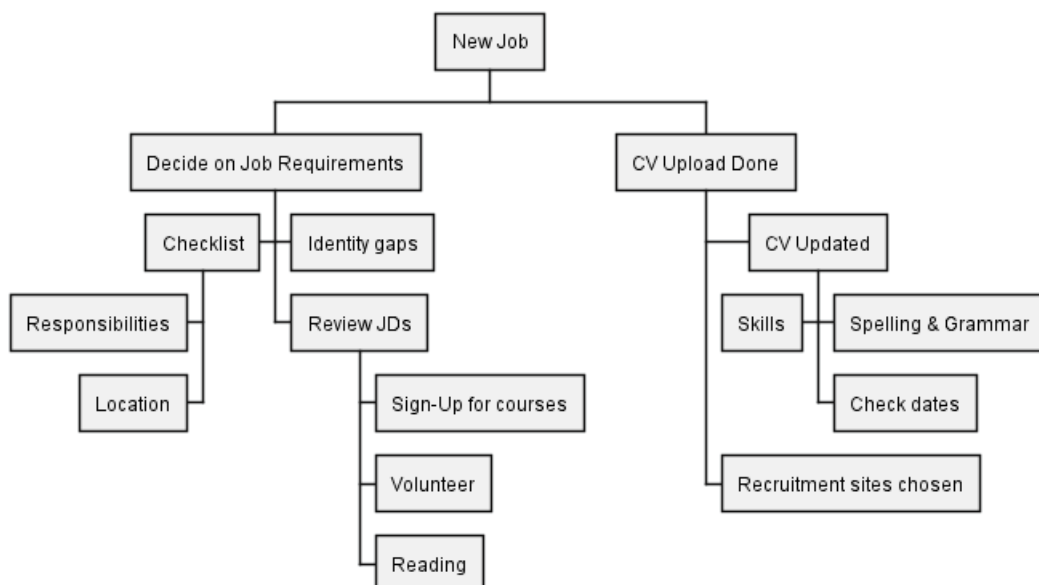


18.3 算術記号による表記

左右どちらの側に配置するかを、以下のように算術記号で指定できます。

```

@startwbs
+ New Job
++ Decide on Job Requirements
+++ Identity gaps
+++ Review JDs
++++ Sign-Up for courses
++++ Volunteer
++++ Reading
++- Checklist
++-- Responsibilities
++-- Location
++ CV Upload Done
+++ CV Updated
++++ Spelling & Grammar
++++ Check dates
---- Skills
+++ Recruitment sites chosen
@endwbs
  
```



18.4 複数行

: と; を使って、マインドマップと同様に、複数行テキストの箱を作ることができます。

```
@startwbs
```

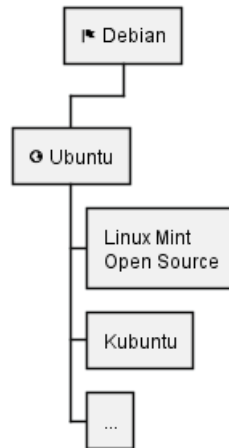
```

* <&flag> Debian
** <&globe> Ubuntu

***:Linux Mint
Open Source;

*** Kubuntu
*** ...
@endwbs

```



[Ref. QA-13945]

18.5 箱を消す

アンダースコア `_` を使って箱を非表示にすることができます。

18.5.1 算術記号を使う場合

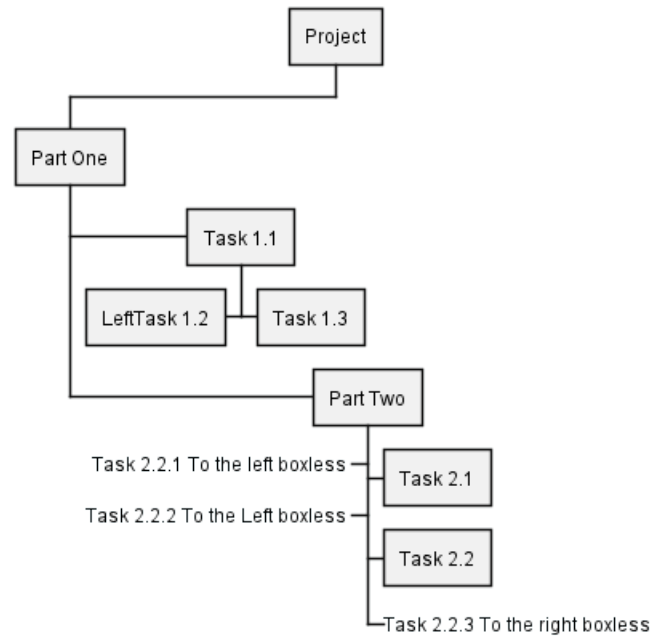
18.5.2 一部の箱を非表示にする

```

@startwbs
+ Project
+ Part One
+ Task 1.1
- LeftTask 1.2
+ Task 1.3
+ Part Two
+ Task 2.1
+ Task 2.2
- _ Task 2.2.1 To the left boxless
- _ Task 2.2.2 To the Left boxless
+ _ Task 2.2.3 To the right boxless
@endwbs

```

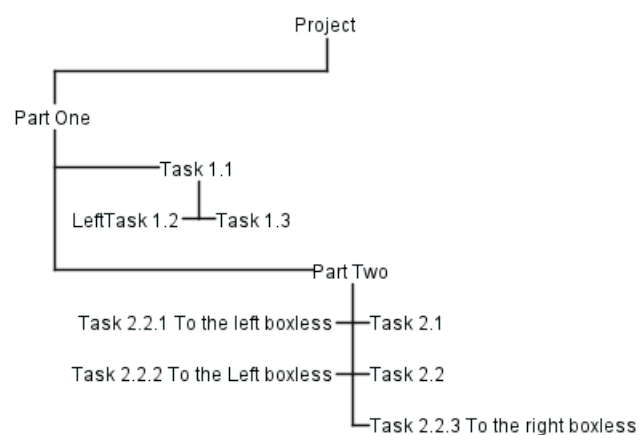




18.5.3 すべての箱を非表示にする

```

@startwbs
+_ Project
+_ Part One
+_ Task 1.1
- LeftTask 1.2
+_ Task 1.3
+_ Part Two
+_ Task 2.1
+_ Task 2.2
- Task 2.2.1 To the left boxless
- Task 2.2.2 To the Left boxless
+_ Task 2.2.3 To the right boxless
@endwbs
  
```



18.5.4 OrgMode の文法の場合

18.5.5 一部の箱を非表示にする

```

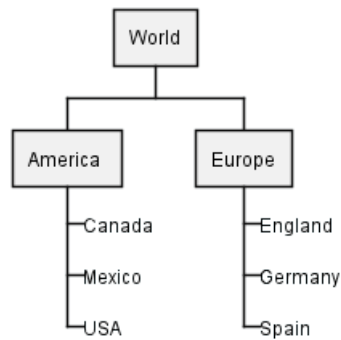
@startwbs
* World
** America
  
```



```

***_ Canada
***_ Mexico
***_ USA
** Europe
***_ England
***_ Germany
***_ Spain
@endwbs

```



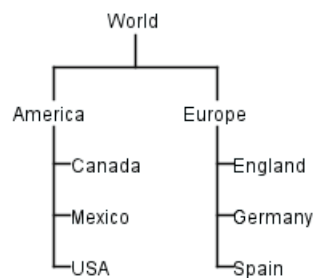
[Ref. QA-13297]

18.5.6 すべての箱を非表示にする

```

@startwbs
*_ World
**_ America
***_ Canada
***_ Mexico
***_ USA
**_ Europe
***_ England
***_ Germany
***_ Spain
@endwbs

```



[Ref. QA-13355]

18.6 色（インライン指定とスタイルの色）

ノードの色を変更できます：

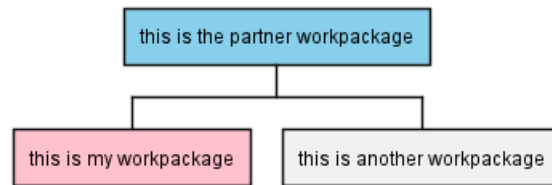
- インラインの色指定

```

@startwbs
*[#SkyBlue] this is the partner workpackage
**[#pink] this is my workpackage
** this is another workpackage
@endwbs

```

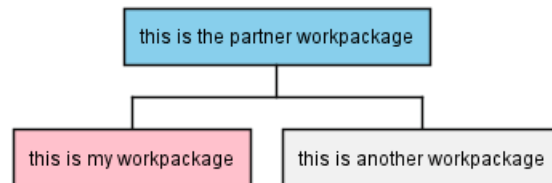




```

@startwbs
+[#SkyBlue] this is the partner workpackage
++[#pink] this is my workpackage
++ this is another workpackage
@endwbs

```



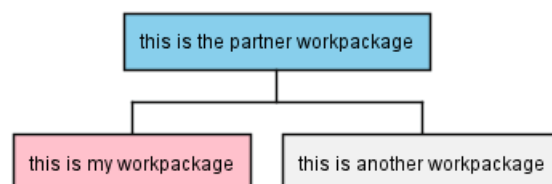
[Ref. QA-12374, only from v1.2020.20]

- スタイルの色指定

```

@startwbs
<style>
wbsDiagram {
  .pink {
    BackgroundColor pink
  }
  .your_style_name {
    BackgroundColor SkyBlue
  }
}
</style>
* this is the partner workpackage <<your_style_name>>
** this is my workpackage <<pink>>
** this is another workpackage
@endwbs

```



```

@startwbs
<style>
wbsDiagram {
  .pink {
    BackgroundColor pink
  }
  .your_style_name {
    BackgroundColor SkyBlue
  }
}
</style>
+ this is the partner workpackage <<your_style_name>>
++ this is my workpackage <<pink>>

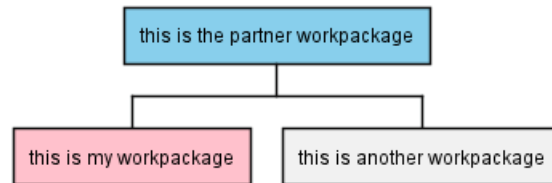
```



```

++ this is another workpackage
@endwbs

```



18.7 スタイルを適用する

ダイアグラムのスタイルを変更することができます。

```

@startwbs
<style>
wbsDiagram {
  // all lines (meaning connector and borders, there are no other lines in WBS) are black by default
  Linecolor black
  arrow {
    // note that connector are actually "arrow" even if they don't look like as arrow
    // This is to be consistent with other UML diagrams. Not 100% sure that it's a good idea
    // So now connector are green
    LineColor green
  }
  :depth(0) {
    // will target root node
    BackgroundColor White
    RoundCorner 10
    LineColor red
    // Because we are targetting depth(0) for everything, border and connector for level 0 will be
  }
  arrow {
    :depth(2) {
      // Targetting only connector between Mexico-Chihuahua and USA-Texas
      LineColor blue
      LineStyle 4
      LineThickness .5
    }
  }
  node {
    :depth(2) {
      LineStyle 2
      LineThickness 2.5
    }
  }
  boxless {
    // will target boxless node with '_'
    FontColor darkgreen
  }
}
</style>
* World
** America
*** Canada
*** Mexico
**** Chihuahua
*** USA
**** Texas

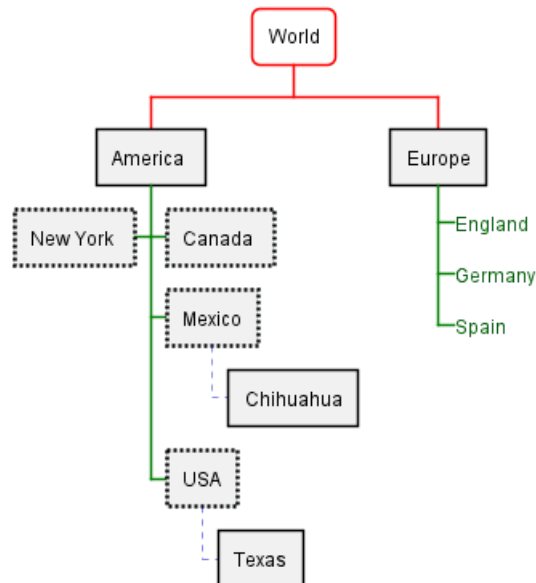
```



```

***< New York
** Europe
***_ England
***_ Germany
***_ Spain
@endwbs

```



18.8 単語の折り返し

`MaximumWidth` で、最大幅を設定すると、自動的に単語を折り返すことができます。使用する単位はピクセルです。

```
@startwbs
```

```

<style>
node {
    Padding 12
    Margin 3
    HorizontalAlignment center
    LineColor blue
    LineThickness 3.0
    BackgroundColor gold
    RoundCorner 40
    MaximumWidth 100
}

rootNode {
    LineStyle 8.0;3.0
    LineColor red
    BackgroundColor white
    LineThickness 1.0
    RoundCorner 0
    Shadowing 0.0
}

leafNode {
    LineColor gold
    RoundCorner 0

```



```

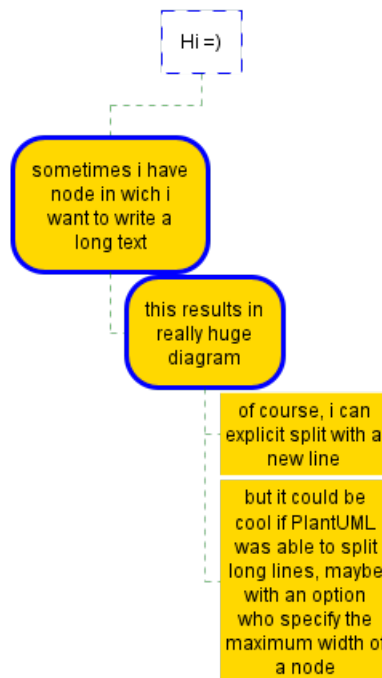
    Padding 3
}

arrow {
    LineStyle 4
    LineThickness 0.5
    LineColor green
}
</style>

* Hi =)
** sometimes i have node in wich i want to write a long text
*** this results in really huge diagram
**** of course, i can explicit split with a\nnew line
**** but it could be cool if PlantUML was able to split long lines, maybe with an option who specify

@endwbs

```



18.9 Add arrows between WBS elements

You can add arrows between WBS elements.

Using alias with as:

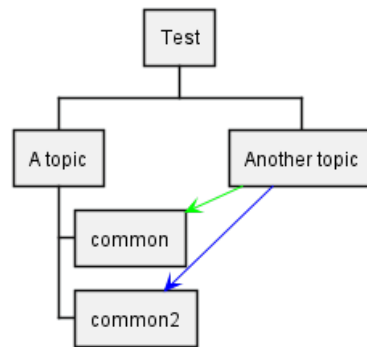
```

@startwbs
<style>
.foo {
    LineColor #00FF00;
}
</style>
* Test
** A topic
*** "common" as c1
*** "common2" as c2
** "Another topic" as t2
t2 -> c1 <<foo>>
t2 ..> c2 #blue

```



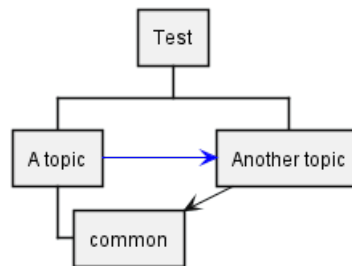
@endwbs



Using alias in parentheses:

```

@startwbs
* Test
**(b) A topic
***(c1) common
**(t2) Another topic
t2 --> c1
b -> t2 #blue
@endwbs
  
```



[Ref. QA-16251]

18.10 Creole on WBS diagram

You can use Creole or HTML Creole on WBS:

```

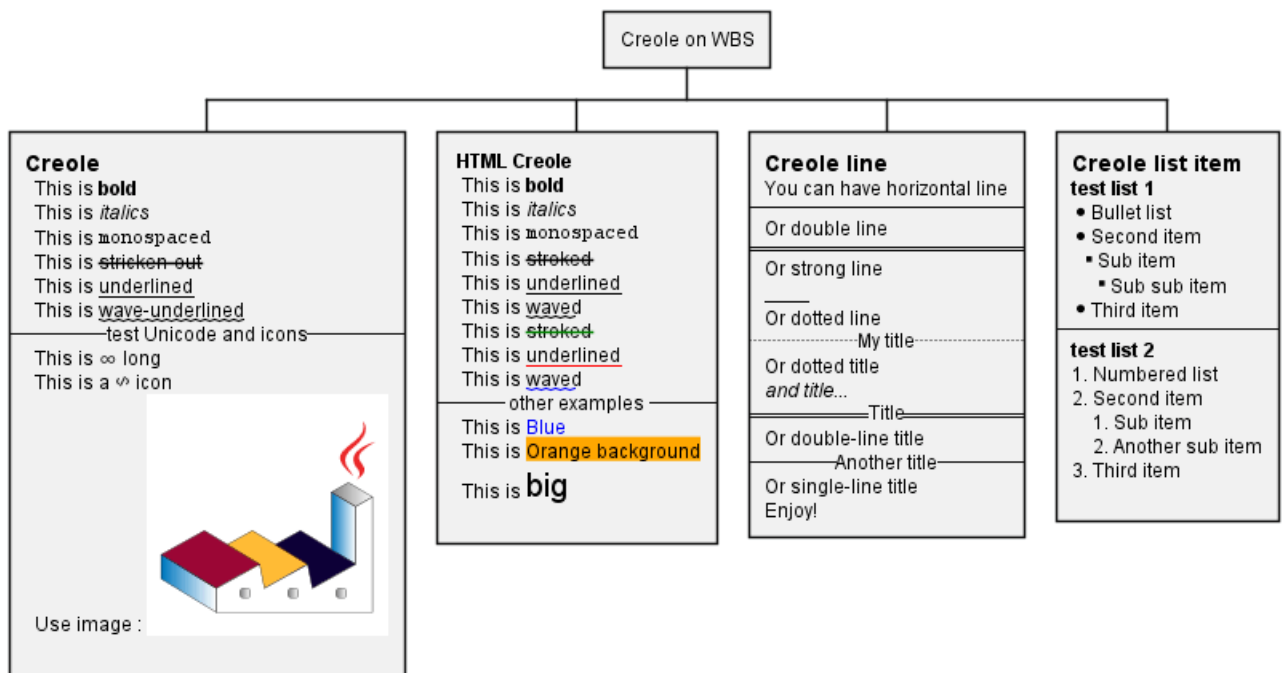
@startwbs
* Creole on WBS
**::=Creole
  This is bold
  This is italics
  This is "monospaced"
  This is stricken-out
  This is underlined
  This is ~wave-underlined~
--test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
  Use image : <img:http://plantuml.com/logo3.png>
;
**: <b>HTML Creole
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  
```



```

This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
;
**::=Creole line
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
**::=Creole list item
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
;
@endwbs

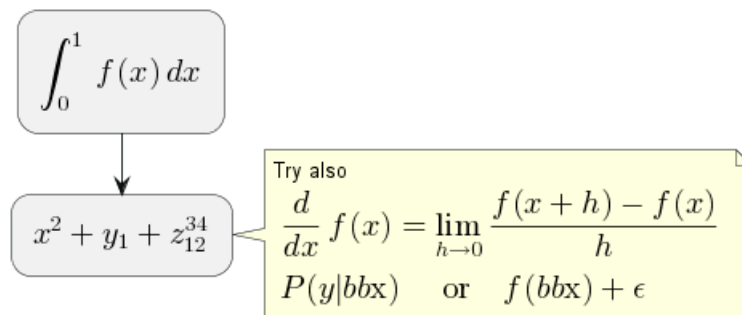
```



19 数式

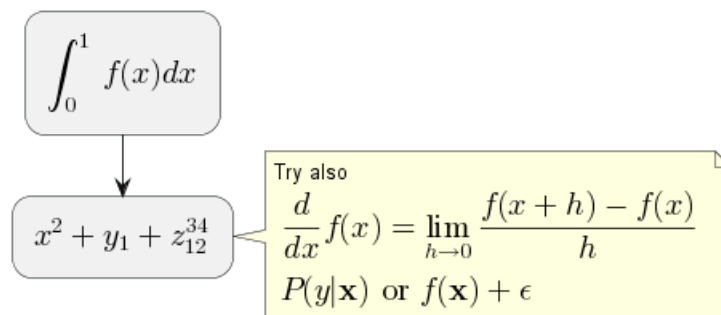
PlantUML では、AsciiMath の構文が使用できます。

```
@startuml
: <math>\int_0^1 f(x) dx</math>;
: <math>x^2 + y_1 + z_{12}^{34}</math>;
note right
Try also
<math>\frac{d}{dx} f(x) = \lim_{h \rightarrow 0} (f(x+h) - f(x)) / h</math>
<math>P(y | \mathbf{x}) \text{ or } f(\mathbf{x}) + \epsilon</math>
end note
@enduml
```



また、JLaTeXMath の構文も使用できます:

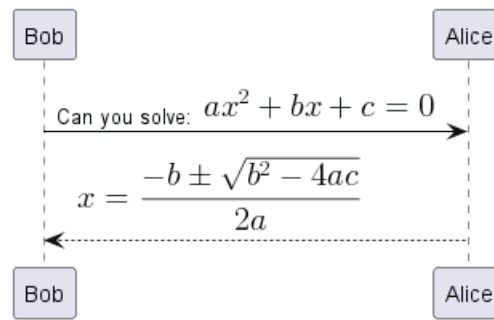
```
@startuml
: <latex>\int_0^1 f(x) dx</latex>;
: <latex>x^2 + y_1 + z_{12}^{34}</latex>;
note right
Try also
<latex>\frac{d}{dx} f(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}</latex>
<latex>P(y | \mathbf{x}) \text{ or } f(\mathbf{x}) + \epsilon</latex>
end note
@enduml
```



他の例:

```
@startuml
Bob -> Alice : Can you solve: <math>ax^2 + bx + c = 0</math>
Alice --> Bob: <math>x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}</math>
@enduml
```





19.1 単体で使用する場合

AsciiMath で記述した式を単体で使いたい場合は、`@startmath` と `@endmath` を使用します。

```

@startmath
f(t)=(a_0)/2 + \sum_{n=1}^{\infty} a_n \cos((n\pi t)/L) + \sum_{n=1}^{\infty} b_n \sin((n\pi t)/L)
@endmath

```

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi t}{L}\right)$$

また、JLaTeXMath の場合は、`@startlatex` と `@endlatex` を使用します。

```

@startlatex
\sum_{i=0}^{n-1} (a_i + b_i^2)
@endlatex

```

$$\sum_{i=0}^{n-1} (a_i + b_i^2)$$

19.2 どのように処理しているのか

これらの数式を表示するのに、PlantUML では 2 つのオープンソースプロジェクトを使用します。

- AsciiMath : AsciiMath を LaTeX へ変換します。
- JLatexMath LaTeX で書かれた式を表示します。JLaTeXMath は LaTeX のコードを表示するのに最適な Java のライブラリです。

ASCIIMathTeXImg.js は十分に小さいので、PlantUML の標準ディストリビューションに組み込まれています。

JLatexMath は比較的大きいです。ダウンロードページからダウンロードし、4 つの jar ファイル (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* and *jlm_greek.jar*) を PlantUML.jar と同じディレクトリに置く必要があります。



20 ER 図

インフォメーションエンジニアリングの表記法をベースにしています。

すでに存在している Class Diagram の拡張になります。

拡張内容:

- インフォメーションエンジニアリング用の関係線の追加
- **entity** を、クラス図の **class** と読み替え
- 必須属性を表すものとして、***** の表示修飾子を追加

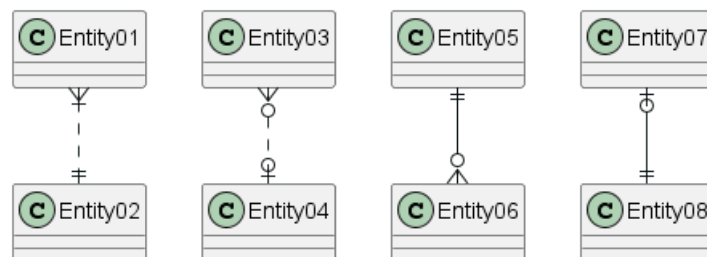
また、クラス図と同じ文法です。クラス図の機能は全て使うことができます。

20.1 インフォメーションエンジニアリングの関係線

Type	記号
0 か 1	o--
1 のみ	--
0 以上	}o--
1 以上	} --

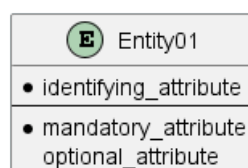
例:

```
@startuml
Entity01 }|...|| Entity02
Entity03 }o...o| Entity04
Entity05 ||--o{ Entity06
Entity07 |o--|| Entity08
@enduml
```



20.2 エンティティ

```
@startuml
entity Entity01 {
    * identifying_attribute
    --
    * mandatory_attribute
    optional_attribute
}
@enduml
```

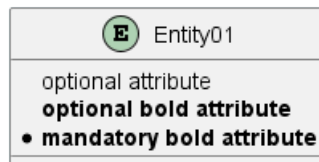


もう一度いいますが、普通のクラス図の文法です。(class の代わりに **entity** を使うのはさておいて)。クラス図でできることは、ER 図でもできます。



* 表示修飾子は必須属性を表します。空白を 1 文字後ろに入れることで、強調と解釈されることを防ぐと良いでしょう:

```
@startuml
entity Entity01 {
    optional attribute
    **optional bold attribute**
    * **mandatory bold attribute**
}
@enduml
```



20.3 完全な例

```
@startuml

' hide the spot
hide circle

' avoid problems with angled crows feet
skinparam linetype ortho

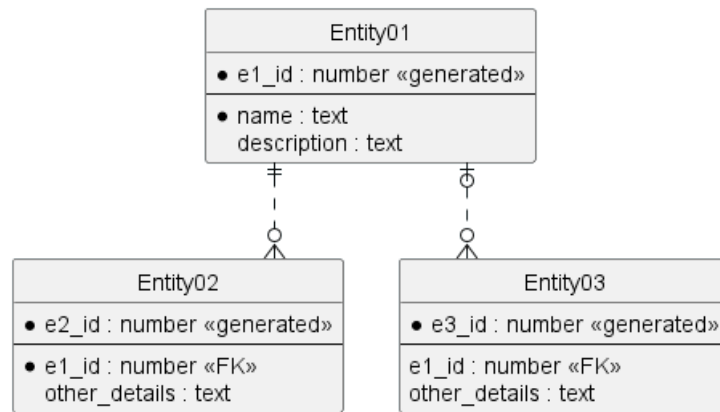
entity "Entity01" as e01 {
    *e1_id : number <<generated>>
    --
    *name : text
    description : text
}

entity "Entity02" as e02 {
    *e2_id : number <<generated>>
    --
    *e1_id : number <<FK>>
    other_details : text
}

entity "Entity03" as e03 {
    *e3_id : number <<generated>>
    --
    e1_id : number <<FK>>
    other_details : text
}

e01 ||..o{ e02
e01 |o..o{ e03

@enduml
```



現状では、エンティティに角度付きで関係線を引こうとすると、見た目がよく有りません。(訳者注:45度でつながってしまいます)

`linetype ortho` `skinparam` を使うことで回避できます。

21 共通コマンド

Explore these commands to create diagrams that are both functional and aesthetically pleasing, tailoring each element to your exact specifications.

21.1 コメント

21.1.1 単一行コメント

シングルクォート'で始まる行はコメントです。

```
@startuml
'Line comments use a single apostrophe
@enduml
```

21.1.2 ブロックコメント

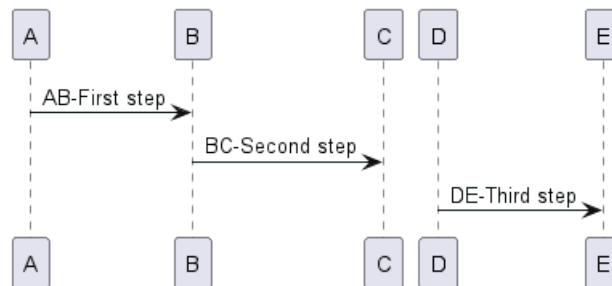
ブロックコメントは C 言語と同様のスタイルですが、* の代わりにシングルクォート'を使用します。/'で開始して'/で終了する範囲に、複数行のコメントを記述することができます。

```
@startuml
/'
many lines comments
here
'/
@enduml
```

[Ref. QA-1353]

一つの行内でブロックコメントを使用することもできます:

```
@startuml
/' case 1 '/ A -> B : AB-First step
           B -> C : BC-Second step
/' case 2 '/ D -> E : DE-Third step
@enduml
```



[Ref. QA-3906 and QA-3910]

[Ref. GH-214]

21.2 拡大

scale コマンドを使って、生成する画像を拡大できます。

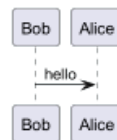
小数または分数で拡大率を指定できます。ピクセル単位で width (幅) または height (高さ) を指定することもできます。幅と高さの両方を指定することもできます。この場合は、指定したサイズの内側に収まるように調整されます。

- scale 1.5
- scale 2/3
- scale 200 width
- scale 200 height



- scale 200*100
- scale max 300*200
- scale max 1024 width
- scale max 800 height

```
@startuml
scale 180*90
Bob->Alice : hello
@enduml
```



21.3 タイトル

title キーワードを使用してタイトルを入れることができます。タイトルではを使用して改行することができます。

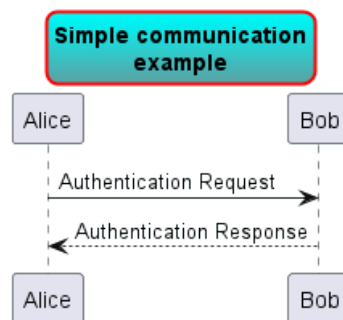
skinparam 設定を使用してタイトルに枠線を付けることができます。

```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue
```

```
title Simple communication\nexample
```

```
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
```

```
@enduml
```



タイトル中で creole 書式を使用することもできます。

また、**title** と **end title** キーワードを使用することで、複数行にわたってタイトルを記述することもできます。

```
@startuml

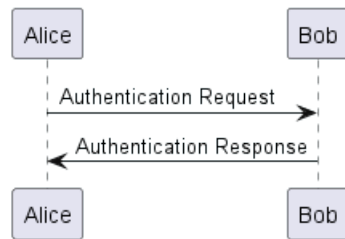
title
  <u>Simple</u> communication example
  on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title
```

```
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response
```



```
@enduml
```

**Simple communication example
on several lines and using creole tags**



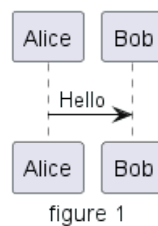
21.4 キャプション

`caption` キーワードを使用して図の下部にキャプションを入れることができます。

```
@startuml
```

```
caption figure 1
Alice -> Bob: Hello
```

```
@enduml
```



21.5 フッタとヘッダ

`footer`、`header` のコマンドを使って、生成された図にフッタとヘッダを追加することができます。

`center`、`left`、`right` を使ってフッタ、ヘッダの表示位置を指定することもできます。

タイトルと同様に、複数行にわたってフッタまたはヘッダを定義することができます。

また、フッタとヘッダでは HTML タグを使用することもできます。

```
@startuml
```

```
Alice -> Bob: Authentication Request
```

```
header
```

```
<font color=red>Warning:</font>
```

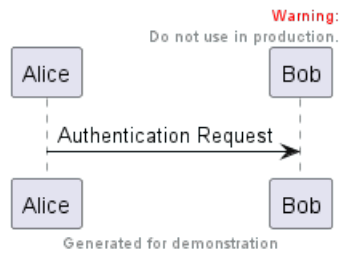
```
Do not use in production.
```

```
endheader
```

```
center footer Generated for demonstration
```

```
@enduml
```





21.6 図の凡例

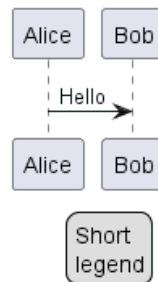
`legend` と `end legend` を使って凡例を追加できます。

`left`、`right`、`top`、`bottom`、`center` を使って、凡例の位置を指定することもできます。

```

@startuml
Alice -> Bob : Hello
legend right
    Short
    legend
endlegend
@enduml

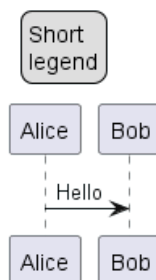
```



```

@startuml
Alice -> Bob : Hello
legend top left
    Short
    legend
endlegend
@enduml

```



21.7 付録：すべての図の例

21.7.1 アクティビティ図

```

@startuml
header some header

footer some footer

```



```

title My title

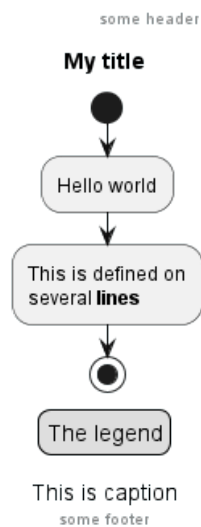
caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml

```



21.7.2 アーキテクチャ図

```

@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

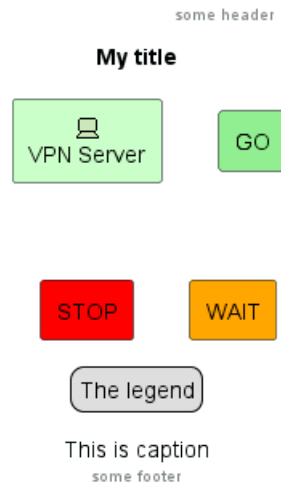
archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml

```





21.7.3 クラス図

```
@startuml
header some header

footer some footer

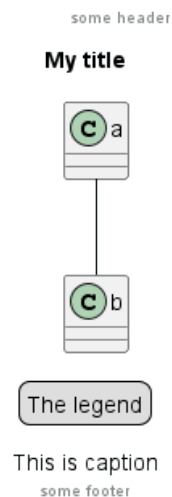
title My title

caption This is caption

legend
The legend
end legend

a -- b

@enduml
```



21.7.4 コンポーネント図、配置図、ユースケース図

```
@startuml
header some header

footer some footer
```



```

title My title

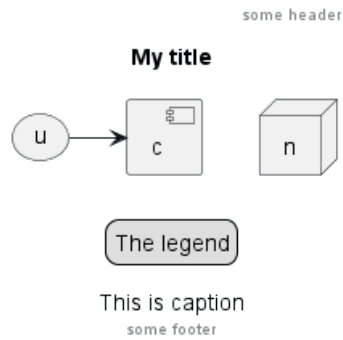
caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml

```



21.7.5 ガントチャート

```

@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[t] lasts 5 days

@enduml

```

PlantUML 1.2023.11

[From string (line 15)]

@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[t] lasts 5 days
Syntax Error?

TODO: DONE [(Header, footer) corrected on V1.2020.18]

21.7.6 オブジェクト図

```
@startuml
header some header

footer some footer

title My title

caption This is caption

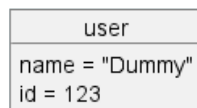
legend
The legend
end legend

object user {
    name = "Dummy"
    id = 123
}

@enduml
```

some header

My title



The legend

This is caption

some footer

21.7.7 マインドマップ

```
@startmindmap
header some header
```



```

footer some footer

title My title

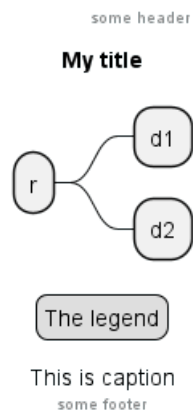
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap

```



21.7.8 ネットワーク図 (nwdiag)

```

@startuml
header some header

footer some footer

title My title

caption This is caption

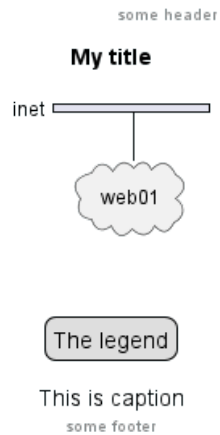
legend
The legend
end legend

nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}

@enduml

```





21.7.9 シーケンス図

```
@startuml
header some header

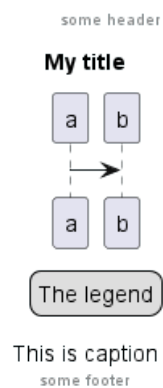
footer some footer

title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml
```



21.7.10 ステート図

```
@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
enduml
```



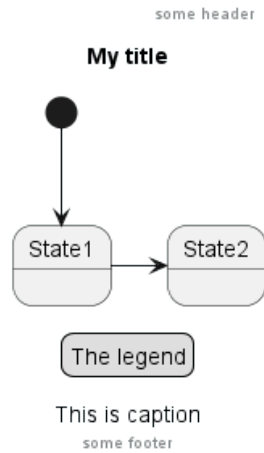
```

end legend

[*] --> State1
State1 -> State2

@enduml

```



21.7.11 タイミング図

```

@startuml
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

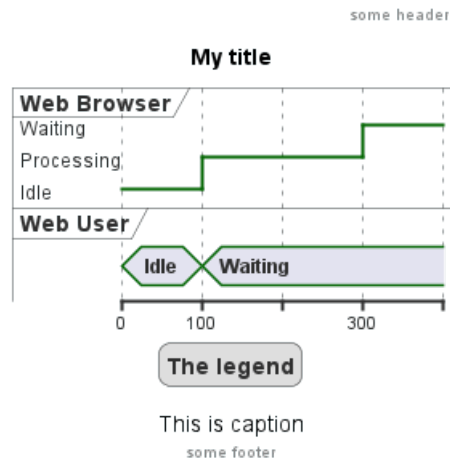
@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml

```





21.7.12 Work Breakdown Structure (WBS)

```
@startwbs
header some header

footer some footer

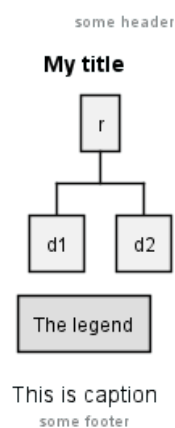
title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endwbs
```



TODO: DONE *[Corrected on V1.2020.17]*

21.7.13 Wireframe (SALT)

```
@startsalt
header some header
```



```

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
  Login    | "MyName  "
  Password | "****    "
  [Cancel] | [ OK    ]
}
@endsalt

```



TODO: DONE [*Corrected on V1.2020.18*]

21.8 付録：すべての図でスタイルを指定した例

TODO: DONE

FYI:

- すべてが正常に使えるのはシーケンス図のみです。
- **title**、**caption**、**legend** は、**salt diagram** 以外のすべての図で正常に使えます。

TODO: FIXME

- 現状 (*test on 1.2020.18-19*)、**header**、**footer** はシーケンス図を除くすべての図で正常に動作しません。

To be fix; Thanks

TODO: FIXME

ここでは、すべての図を使って **title**、**header**、**footer**、**caption**、**legend** を、デバッグ形式でテストしています：

```

<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple

```



```

}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>

```

21.8.1 アクティビティ図

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

```

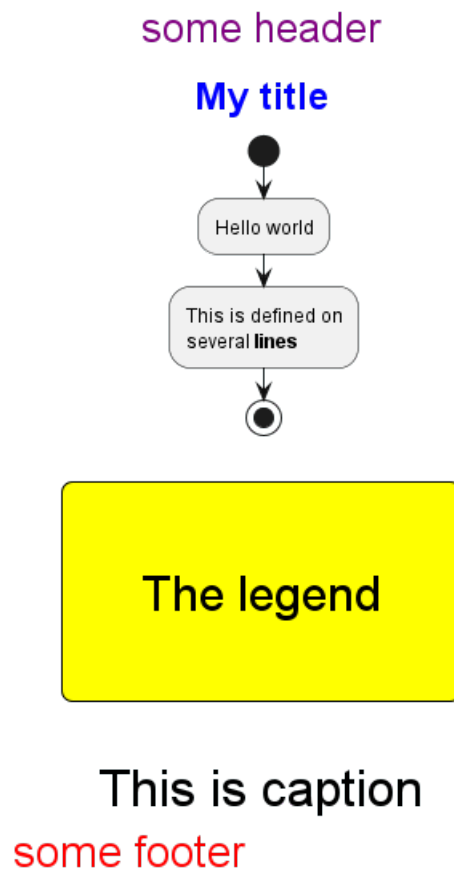


```
caption This is caption
```

```
legend
The legend
end legend
```

```
start
:Hello world;
:This is defined on
several **lines**;
stop
```

```
@enduml
```



21.8.2 アーキテクチャ図

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}
```



```
footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

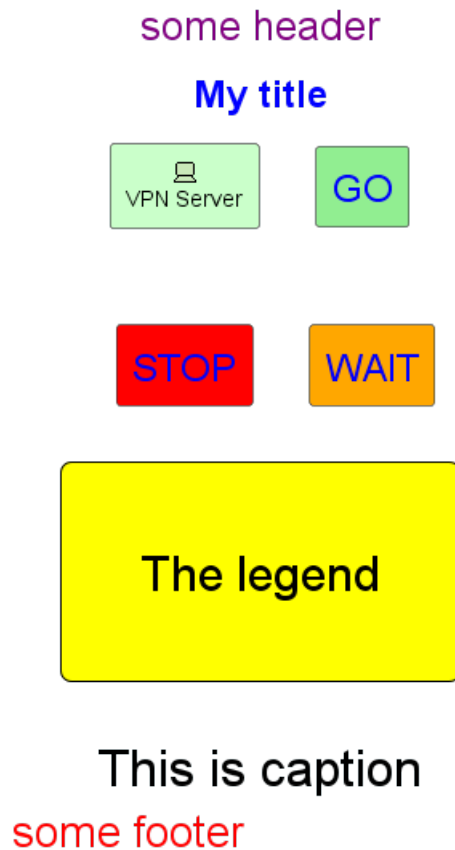
caption This is caption

legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml
```



21.8.3 クラス図

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  
```

```

    FontSize 32
}
</style>
header some header

footer some footer

title My title

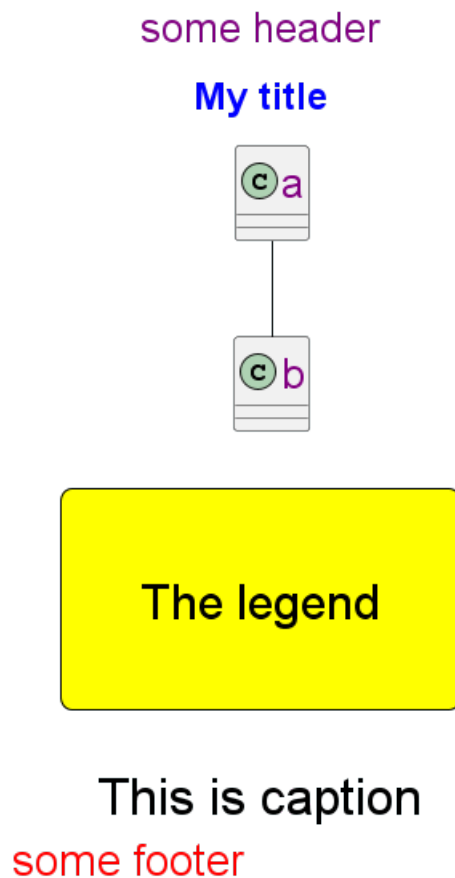
caption This is caption

legend
The legend
end legend

a -- b

@enduml

```



21.8.4 コンポーネント図、配置図、ユースケース図

```

@startuml
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}
header {

```



```
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}
```

```
footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}
```

```
legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}
```

```
caption {
    FontSize 32
}
```

</style>

header some header

footer some footer

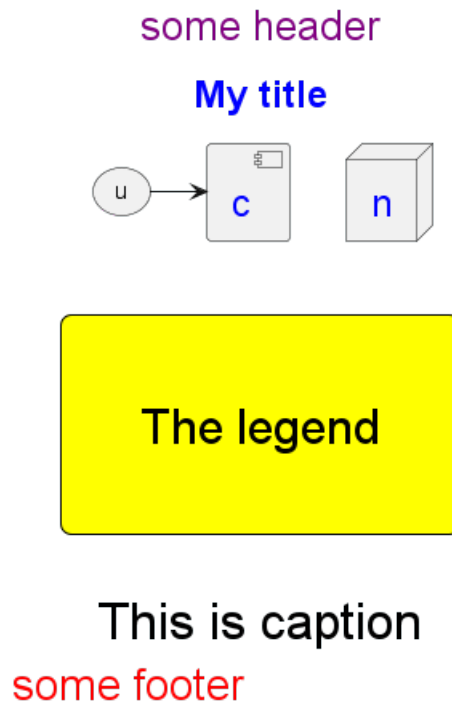
title My title

caption This is caption

```
legend
The legend
end legend
```

```
node n
(u) -> [c]
```

@enduml



21.8.5 ガントチャート

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

```



```

footer some footer

title My title

caption This is caption

legend
The legend
end legend

[t] lasts 5 days

@enduml

```

PlantUML 1.2023.11

[From string (line 45)]

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
...
... ( skipping 20 lines )
...
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

```

[t] lasts 5 days
Syntax Error?

21.8.6 オブジェクト図

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple

```



```
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

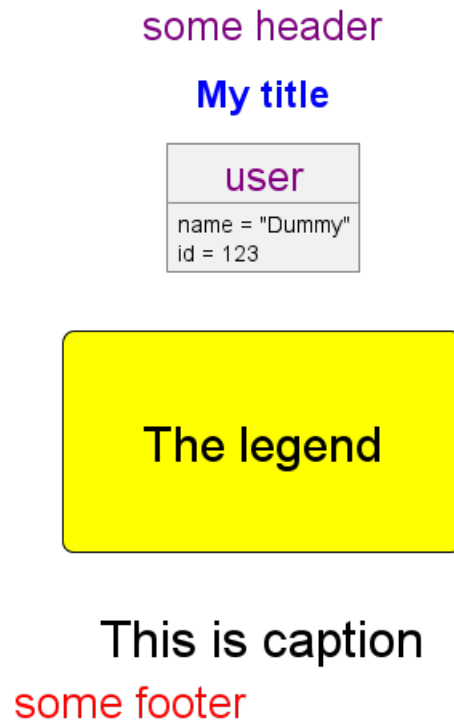
title My title

caption This is caption

legend
The legend
end legend

object user {
  name = "Dummy"
  id = 123
}

@enduml
```



21.8.7 マインドマップ

```

@startmindmap
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

```



```

footer some footer

title My title

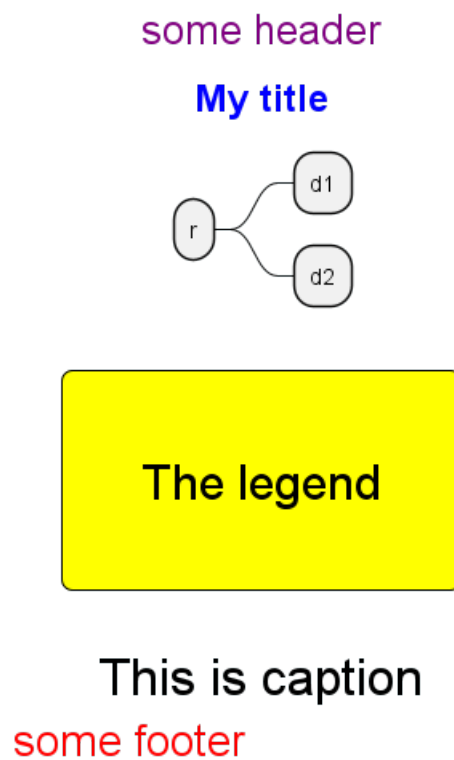
caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap

```



21.8.8 ネットワーク図 (nwdiag)

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

```



```
footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

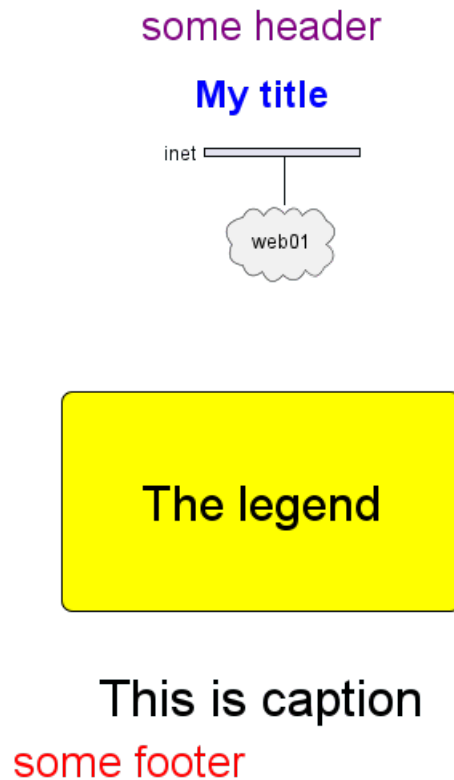
title My title

caption This is caption

legend
The legend
end legend

nwdiag {
  network inet {
    web01 [shape = cloud]
  }
}

@enduml
```



21.8.9 シーケンス図

```
@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
```



```

header some header

footer some footer

title My title

caption This is caption

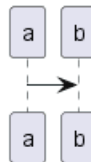
legend
The legend
end legend

a->b
@enduml

```

some header

My title



The legend

This is caption

some footer

21.8.10 ステート図

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28

```



```

    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

caption {
    FontSize 32
}
</style>
header some header

footer some footer

title My title

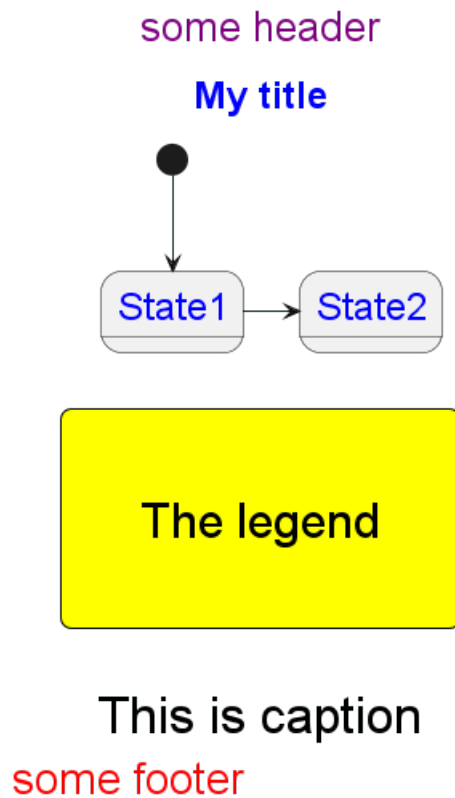
caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml

```



21.8.11 タイミング図

```

@startuml
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

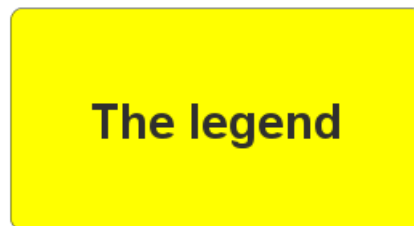
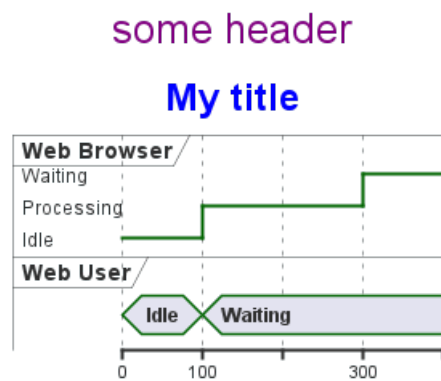
@100
WU is Waiting
WB is Processing

@300
WB is Waiting

```



@enduml



This is caption

some footer

21.8.12 Work Breakdown Structure (WBS)

```

@startwbs
<style>
title {
    HorizontalAlignment right
    FontSize 24
    FontColor blue
}

header {
    HorizontalAlignment center
    FontSize 26
    FontColor purple
}

footer {
    HorizontalAlignment left
    FontSize 28
    FontColor red
}

legend {
    FontSize 30
    BackGroundColor yellow
    Margin 30
    Padding 50
}

```



```

caption {
  FontSize 32
}
</style>
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

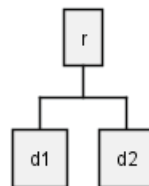
* r
** d1
** d2

@endwbs

```

some header

My title



The legend

This is caption
some footer

21.8.13 Wireframe (SALT)

TODO: FIXME Fix all (title, caption, legend, header, footer) for salt. **TODO:** FIXME

```

@startsalt
<style>
title {
  HorizontalAlignment right
  FontSize 24
  FontColor blue
}

```



```

header {
  HorizontalAlignment center
  FontSize 26
  FontColor purple
}

footer {
  HorizontalAlignment left
  FontSize 28
  FontColor red
}

legend {
  FontSize 30
  BackGroundColor yellow
  Margin 30
  Padding 50
}

caption {
  FontSize 32
}
</style>
@startsalt
header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

{+
  Login      | "MyName   "
  Password   | "****     "
  [Cancel]   | [ OK      ]
}
@endsalt

```



21.9 メインフレーム

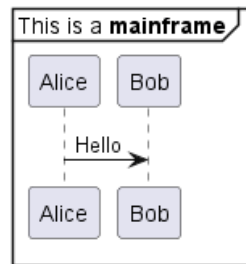
```

@startuml
mainframe This is a **mainframe**

```



```
Alice->Bob : Hello
@enduml
```



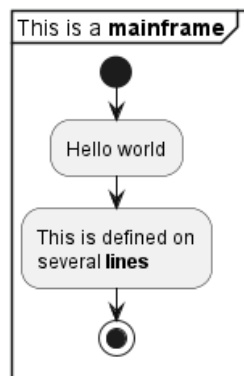
[Ref. QA-4019 and Issue#148]

21.10 付録: すべての図に対するメインフレームの例

21.10.1 アクティビティ図

```
@startuml
mainframe This is a **mainframe**

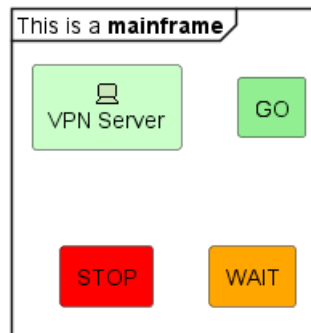
start
:Hello world;
:This is defined on
several **lines**;;
stop
@enduml
```



21.10.2 アーキテクチャ図

```
@startuml
mainframe This is a **mainframe**

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>
rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange
@enduml
```

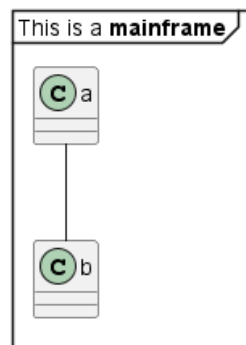


TODO: FIXME Cropped on the top and on the left **TODO: FIXME**

21.10.3 クラス図

```
@startuml
mainframe This is a **mainframe**

a -- b
@enduml
```

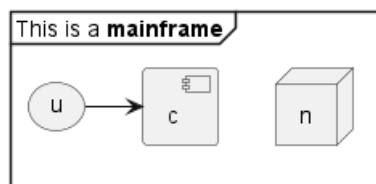


TODO: FIXME Cropped on the top and on the left **TODO: FIXME**

21.10.4 コンポーネント図、配置図、ユースケース図

```
@startuml
mainframe This is a **mainframe**

node n
(u) -> [c]
@enduml
```



TODO: FIXME Cropped on the top and on the left **TODO: FIXME**

21.10.5 ガントチャート

```
@startuml
mainframe This is a **mainframe**

[t] lasts 5 days
@enduml
```



PlantUML 1.2023.11

[From string (line 4)]

```
@startuml
mainframe This is a **mainframe**
```

```
[t] lasts 5 days
```

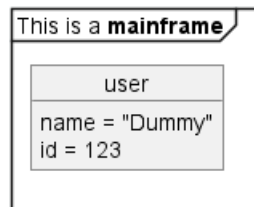
Syntax Error?

TODO: FIXME Cropped on the top and on the left **TODO: FIXME**

21.10.6 オブジェクト図

```
@startuml
mainframe This is a **mainframe**

object user {
    name = "Dummy"
    id = 123
}
@enduml
```

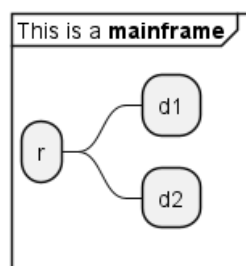


TODO: FIXME Cropped on the top! **TODO: FIXME**

21.10.7 マインドマップ

```
@startmindmap
mainframe This is a **mainframe**

* r
** d1
** d2
@endmindmap
```



21.10.8 ネットワーク図 (nwdiag)

```
@startuml
mainframe This is a **mainframe**

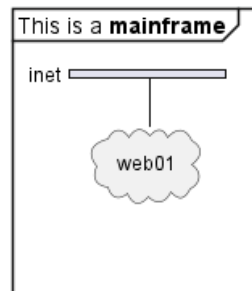
nwdiag {
    network inet {
        web01 [shape = cloud]
    }
}
```



```

}
}
@enduml

```



TODO: FIXME Cropped on the top! **TODO: FIXME**

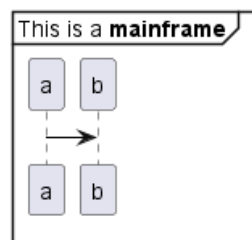
21.10.9 シーケンス図

```

@startuml
mainframe This is a **mainframe**

a->b
@enduml

```



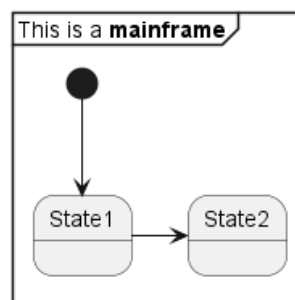
21.10.10 ステート図

```

@startuml
mainframe This is a **mainframe**

[*] --> State1
State1 -> State2
@enduml

```



TODO: FIXME Cropped on the top and on the left **TODO: FIXME**

21.10.11 タイミング図

```

@startuml
mainframe This is a **mainframe**

robust "Web Browser" as WB

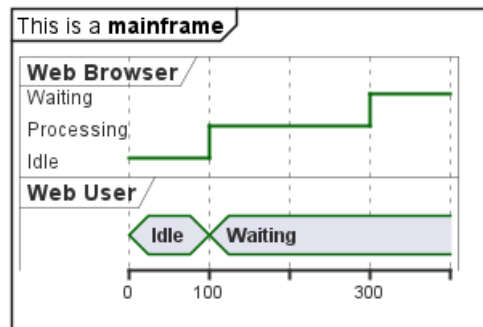
```



```

concise "Web User" as WU
@0
WU is Idle
WB is Idle
@100
WU is Waiting
WB is Processing
@300
WB is Waiting
@enduml

```

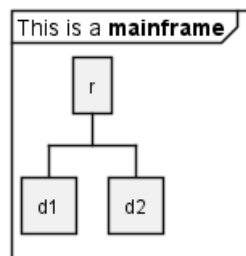


21.10.12 Work Breakdown Structure (WBS)

```

@startwbs
mainframe This is a **mainframe**
* r
** d1
** d2
@endwbs

```



21.10.13 ワイヤフレーム (SALT)

```

@startsalt
mainframe This is a **mainframe**
{+
    Login      | "MyName"  |
    Password   | "****"   |
    [Cancel]   | [ OK ]   |
}
@sendsalt

```



21.11 付録: すべての図に対するタイトル、ヘッダ、フッタ、キャプション、凡例、メインフレームの例

21.11.1 アクティビティ図

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

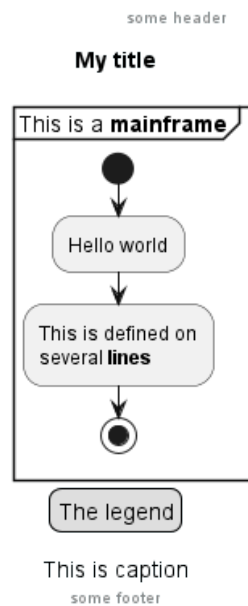
title My title

caption This is caption

legend
The legend
end legend

start
:Hello world;
:This is defined on
several **lines**;
stop

@enduml
```



21.11.2 アーキテクチャ図

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption
```

```

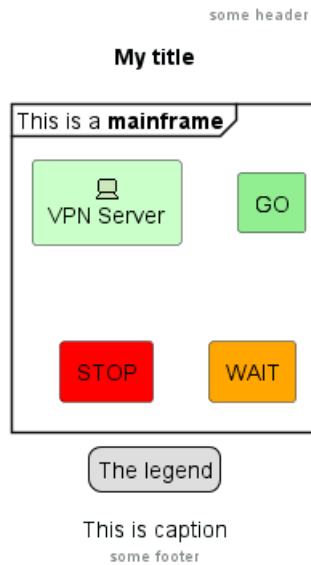
legend
The legend
end legend

archimate #Technology "VPN Server" as vpnServerA <<technology-device>>

rectangle GO #lightgreen
rectangle STOP #red
rectangle WAIT #orange

@enduml

```



21.11.3 クラス図

```

@startuml
mainframe This is a mainframe
header some header

footer some footer

title My title

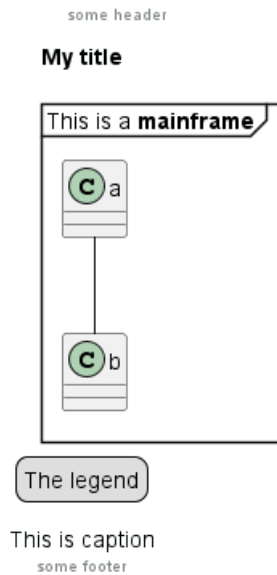
caption This is caption

legend
The legend
end legend

a -- b

@enduml

```



21.11.4 コンポーネント図、配置図、ユースケース図

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

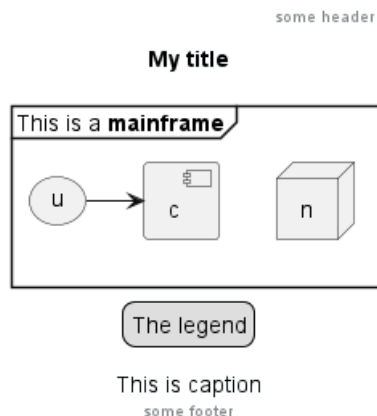
title My title

caption This is caption

legend
The legend
end legend

node n
(u) -> [c]

@enduml
```



21.11.5 ガントチャート

```
@startuml
mainframe This is a **mainframe**
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
The legend
end legend
```

```
[t] lasts 5 days
```

```
@enduml
```

PlantUML 1.2023.11

[From string (line 16)]

```
@startuml
mainframe This is a **mainframe**
header some header
```

```
footer some footer
```

```
title My title
```

```
caption This is caption
```

```
legend
The legend
end legend
```

```
[t] lasts 5 days
Syntax Error?
```

21.11.6 オブジェクト図

```
@startuml
mainframe This is a **mainframe**
header some header
```

```
footer some footer
```

```
title My title
```

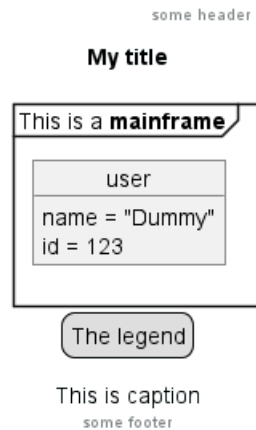
```
caption This is caption
```

```
legend
The legend
end legend
```

```
object user {
    name = "Dummy"
    id = 123
}
```

```
@enduml
```





21.11.7 マインドマップ

```
@startmindmap
mainframe This is a **mainframe**
header some header

footer some footer

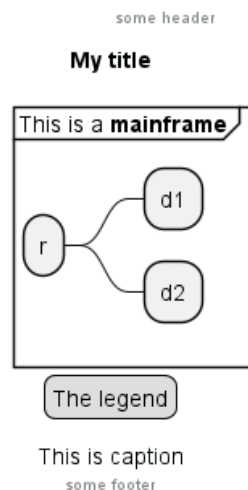
title My title

caption This is caption

legend
The legend
end legend

* r
** d1
** d2

@endmindmap
```



21.11.8 ネットワーク図 (nwdiag)

```
@startuml
mainframe This is a **mainframe**
header some header
```



```

footer some footer

title My title

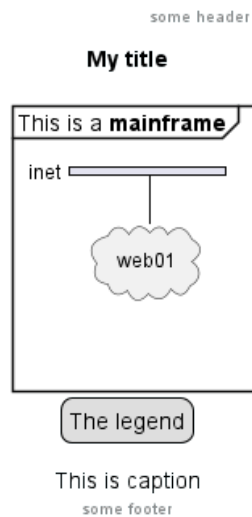
caption This is caption

legend
The legend
end legend

nwdiag {
  network inet {
    web01 [shape = cloud]
  }
}

@enduml

```



21.11.9 シーケンス図

```

@startuml
mainframe This is a **mainframe**
header some header

footer some footer

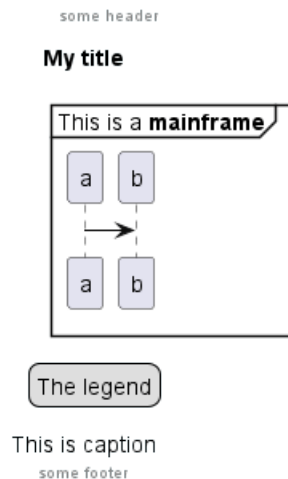
title My title

caption This is caption

legend
The legend
end legend

a->b
@enduml

```



21.11.10 ステート図

```
@startuml
mainframe This is a **mainframe**
header some header

footer some footer

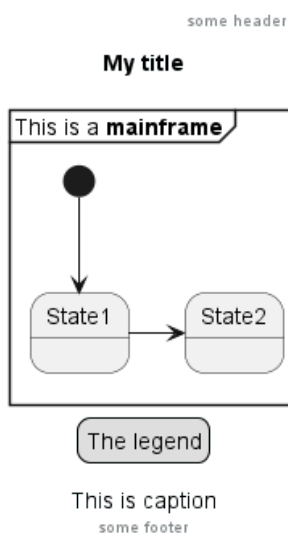
title My title

caption This is caption

legend
The legend
end legend

[*] --> State1
State1 -> State2

@enduml
```



21.11.11 タイミング図

```
@startuml
mainframe This is a **mainframe**
```



```

header some header

footer some footer

title My title

caption This is caption

legend
The legend
end legend

robust "Web Browser" as WB
concise "Web User" as WU

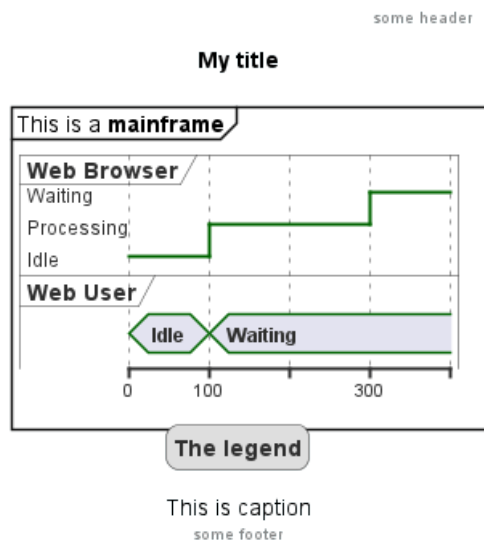
@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting

@enduml

```



21.11.12 Work Breakdown Structure (WBS)

```

@startwbs
mainframe This is a **mainframe**
header some header

footer some footer

title My title

caption This is caption

```



```

legend
The legend
end legend

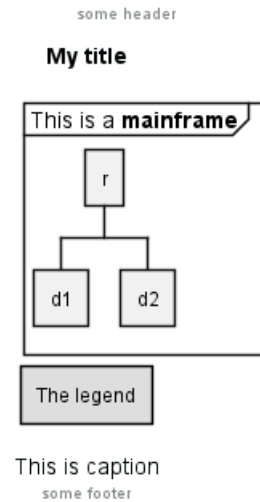
```

```

* r
** d1
** d2

```

```
@endwbs
```



21.11.13 ワイヤフレーム (SALT)

```

@startsalt
mainframe This is a **mainframe**
header some header

footer some footer

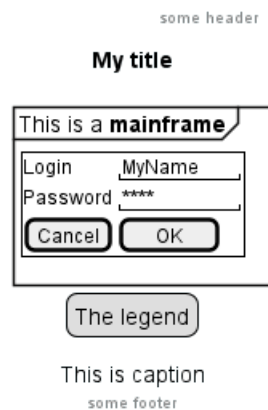
title My title

caption This is caption

legend
The legend
end legend

{+
  Login    | "MyName  "
  Password | "****    "
  [Cancel] | [ OK    ]
}
@endsalt

```



22 Creole

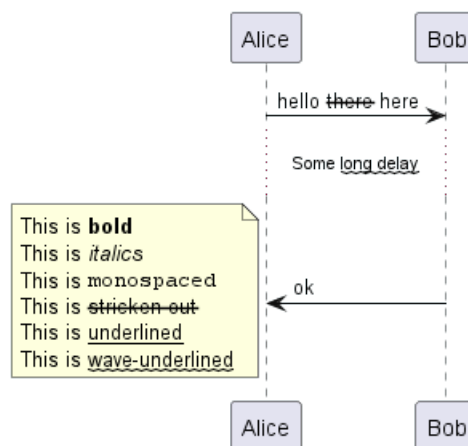
Creole は、さまざまな wiki で使われる、軽量の共通マークアップ言語です。PlantUML には、軽量の Creole エンジンが内蔵されており、テキストのスタイルを統一的な方法で定義することができます。

この構文はすべてのダイアグラムでサポートされています。

後方互換性のため、HTML 構文も残されていることにご注意ください。

22.1 テキストの強調

```
@startuml
Alice -> Bob : hello --there-- here
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
  This is bold
  This is //italics//
  This is "monospaced"
  This is --stricken-out--
  This is underlined
  This is ~wave-underlined~
end note
@enduml
```



22.2 リスト

ノードのテキストや注釈の中で、番号付きリストと箇条書きリストを使用できます。

TODO: FIXME リストとそのサブリストの中で番号付きリストと箇条書きリストを混ぜて使うとうまくいきません。

```
@startuml
object demo {
  * 箇条書きリスト
  * 二つ目の項目
}
note left
  * 箇条書きリスト
  * 二つ目の項目
  ** サブ項目
end note

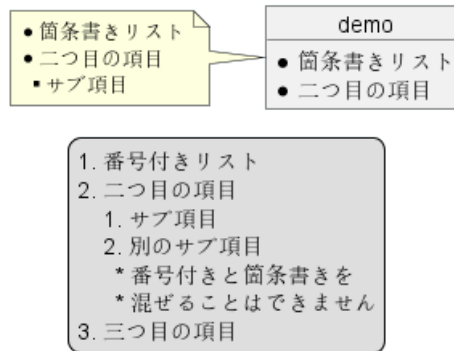
legend
  # 番号付きリスト
  # 二つ目の項目
```



```

## サブ項目
## 別のサブ項目
    * 番号付きと箇条書きを
    * 混ぜることはできません
# 三つ目の項目
end legend
@enduml

```



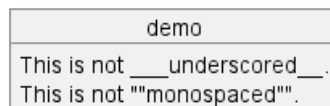
22.3 エスケープ文字

チルダ ~ を使用して、creole の特殊文字をエスケープすることができます。

```

@startuml
object demo {
    This is not ~___underscored___.
    This is not ~""monospaced"".
}
@enduml

```

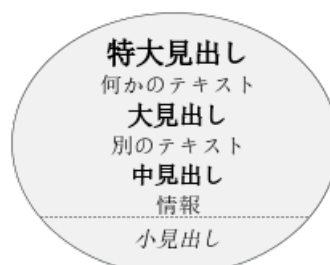


22.4 見出し

```

@startuml
usecase UC1 as "
= 特大見出し
何かのテキスト
== 大見出し
別のテキスト
=== 中見出し
情報
....
==== 小見出し"
@enduml

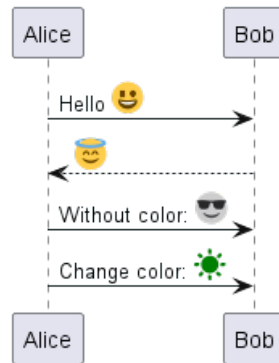
```



22.5 絵文字

以下の構文で、Twemoji の絵文字を利用できます:

```
@startuml
Alice -> Bob : Hello <:1f600:>
return <:innocent:>
Alice -> Bob : Without color: <#0:sunglasses:>
Alice -> Bob : Change color: <#green:sunny:>
@enduml
```



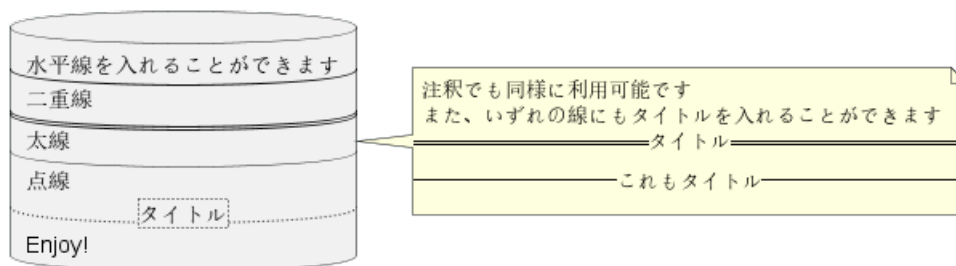
次の Unicode ブロックが利用できます:

- Unicode block 26
- Unicode block 1F3
- Unicode block 1F4
- Unicode block 1F5
- Unicode block 1F6
- Unicode block 1F9

22.6 水平線

```
@startuml
database DB1 as "
水平線を入れることができます
----
二重線
====
太線
----
点線
..タイトル..
Enjoy!
"
note right
  注釈でも同様に利用可能です
  また、いずれの線にもタイトルを入れることができます
  ==タイトル==
  --これもタイトル--
end note
@enduml
```





22.7 リンク

URL とリンクを使用することもできます。

単純なリンクは二重の角括弧を使用します（クラス図のフィールドとメソッドでは三重の角括弧を使用します）

例:

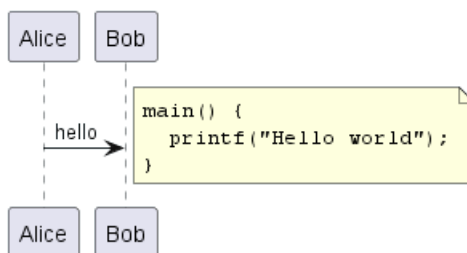
- `[[http://plantuml.com]]`
- `[[http://plantuml.com This label is printed]]`
- `[[http://plantuml.com{Optional tooltip} This label is printed]]`

認証が必要な URL も利用できます。

22.8 コード

`<code>` を使用して、ダイアグラム中にプログラミングのコードを記述できます（シンタックスハイライトは未対応です）。

```
@startuml
Alice -> Bob : hello
note right
<code>
main() {
    printf("Hello world");
}
</code>
end note
@enduml
```



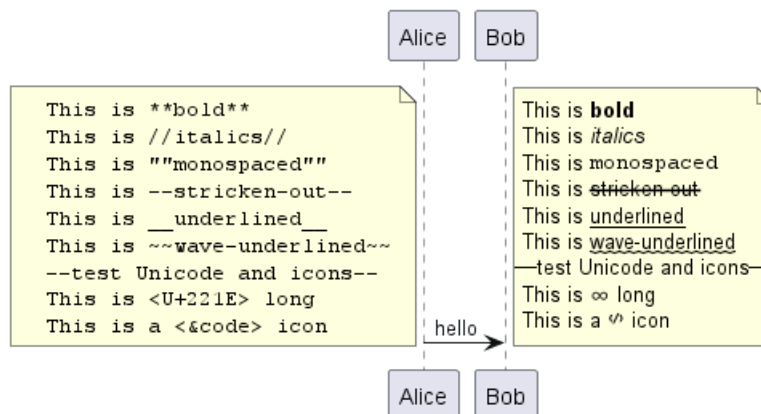
これは、PlantUML のコードとその出力結果を表示する場合に、特に便利です：

```
@startuml
Alice -> Bob : hello
note left
<code>
This is **bold**
This is //italics//
This is "monospaced"
This is --stricken-out--
This is __underlined__
</code>
end note
@enduml
```

```

This is ~~wave-underlined~~
--test Unicode and icons--
This is <U+221E> long
This is a <&code> icon
</code>
end note
note right
  This is bold
  This is //italics//
  This is "monospaced"
  This is --stricken-out--
  This is __underlined__
  This is ~~wave-underlined~~
  --test Unicode and icons--
  This is <U+221E> long
  This is a <&code> icon
end note
@enduml

```



22.9 テーブル

22.9.1 テーブルの作成

| で区切ることで、テーブルを作成できます。

```

@startuml
skinparam titleFontSize 14
title
  シンプルなテーブルの例
  |= |= table |= header |
  | a | table | row |
  | b | table | row |
end title
[*] --> State1
@enduml

```



シンプルなテーブルの例

	table	header
a	table	row
b	table	row

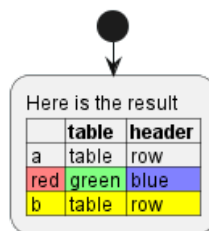


22.9.2 行とセルの色

行とセルに背景色を設定できます。

```

@startuml
start
:Here is the result
|= |= table |= header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |;
@enduml
  
```



22.9.3 枠線とテキストの色

枠線とテキストの色を設定することもできます。

```

@startuml
title
<#lightblue,#red>|= Step |= Date |= Name |= Status |= Link |
<#lightgreen>| 1.1 | TBD | plantuml news |<#Navy><color:OrangeRed><b> Unknown | [[https://plantuml
end title
@enduml
  
```

Step	Date	Name	Status	Link
1.1	TBD	plantuml news	Unknown	plantuml news

[Ref. QA-7184]

22.9.4 枠線無し（背景と同色の枠線）

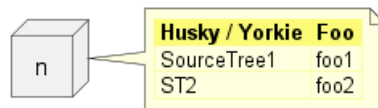
枠線の色を背景色と同じ色に設定することができます。

```

@startuml
node n
note right of n
  <#FBFB77,#FBFB77>|= Husky / Yorkie |= Foo |
  | SourceTree1 | foo1 |
  | ST2 | foo2 |
end
  
```



```
end note
@enduml
```



[Ref. QA-12448]

22.9.5 ヘッダを太字にするかどうか

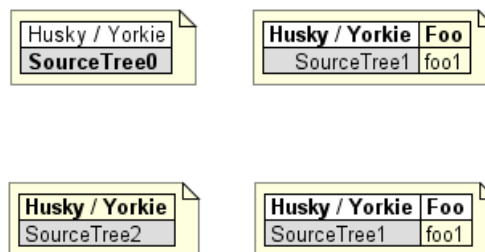
セルの最初の文字を = にすると、太字にすることができます（通常はヘッダを表すために使用します）。

```
@startuml
note as deepCSS0
  |<#white> Husky / Yorkie |
  |=<#gainsboro> SourceTree0 |
endnote

note as deepCSS1
  |= <#white> Husky / Yorkie |= Foo |
  |<#gainsboro><r> SourceTree1 | foo1 |
endnote

note as deepCSS2
  |= Husky / Yorkie |
  |<#gainsboro> SourceTree2 |
endnote

note as deepCSS3
  <#white>|= Husky / Yorkie |= Foo |
  |<#gainsboro> SourceTree1 | foo1 |
endnote
@enduml
```



[Ref. QA-10923]

22.10 ツリー

|_ の文字列を使ってツリーを作ることができます。

title のような共通のコマンドに対して：

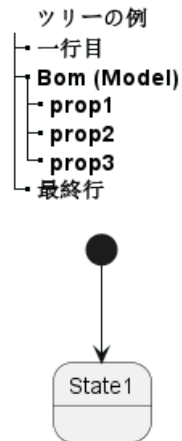
```
@startuml
skinparam titleFontSize 14
title
  ツリーの例
  |_ 一行目
  |_ **Bom (Model)**
    |_ prop1
    |_ prop2
```



```

    |_ prop3
    |_ 最終行
end title
[*] --> State1
@enduml

```



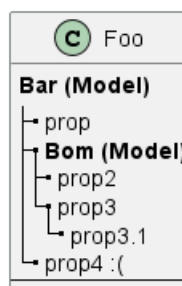
クラス図に対して。

(セパレータを使って、空の 2 つ目の区画を作る必要があることに注意してください。そうしないと、**(Model)** に含まれる括弧によって、テキストが別の区画に移動されてしまいます):

```

@startuml
class Foo {
**Bar (Model)**
|_ prop
|_ **Bom (Model)**
|_   |_ prop2
|_   |_ prop3
|_     |_ prop3.1
|_ prop4 :(
--
}
@enduml

```



[Ref. QA-3448]

コンポーネント図、配置図に対して：

```

@startuml
[A] as A
rectangle "Box B" {
    component B [
        Level 1
        |_ Level 2a
        |_   |_ Level 3a
        |_   |_ Level 3b
    ]
}

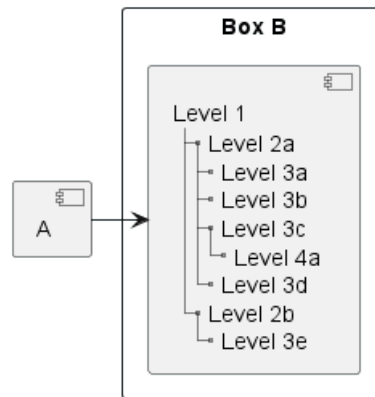
```



```

    |_ Level 3c
      |_ Level 4a
    |_ Level 3d
  |_ Level 2b
    |_ Level 3e
]
}
A -> B
@enduml

```



[Ref. QA-11365]

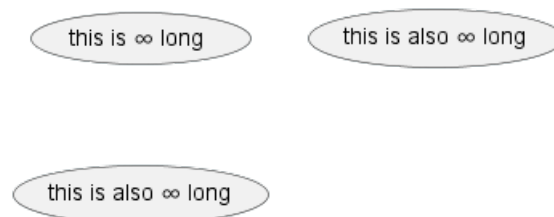
22.11 特殊文字

&#XXXX または、<U+XXXX> の構文、または直接記述することで、任意の Unicode 文字を使うことができます。

```

@startuml
usecase direct as "this is ∞ long"
usecase ampHash as "this is also ∞ long"
usecase angleBrackets as "this is also <U+221E> long"
@enduml

```



Please note that not all Unicode chars appear correctly, depending on what fonts are installed (on your local system or the PlantUML server, depending on which one you use). For characters that are emoji, it's better to use the Emoji notation. See Issue 72 for more details.

22.12 レガシー HTML

Creole と同時に、次の HTML タグも利用可能です：

- で太字
- <u> または <u:#AAAAAA> または <u:[[color|colorName]]> で下線
- <i> でイタリック
- <s> または <s:#AAAAAA> または <s:[[color|colorName]]> で打消し線
- <w> または <w:#AAAAAA> または <w:[[color|colorName]]> で波下線
- <plain> でプレーンテキスト



- `<color:#AAAAAA>` または `<color:[[color|colorName]]>` で文字色
- `<back:#AAAAAA>` または `<back:[[color|colorName]]>` で背景色
- `<size:nn>` でフォントサイズの変更
- `<img:file>` : ファイルシステム中にアクセス可能なファイルを指定
- `<img:http://plantuml.com/logo3.png>` : インターネット上で利用可能な URL を指定

```
@startuml
:* <color:red>文字色</color>の変更
* <back:cadetblue>背景色</back>の変更
* <size:18>フォントサイズ</size>の変更
* <u>legacy</u> <b>HTML <i>tag</i></b>
* <u:red>HTML</u><s:green>タグ中での</s><w:#0000FF>色の使用</w>
----
* 画像 : <img:http://plantuml.com/logo3.png>
;
@enduml
```



22.12.1 一般的な HTML 要素

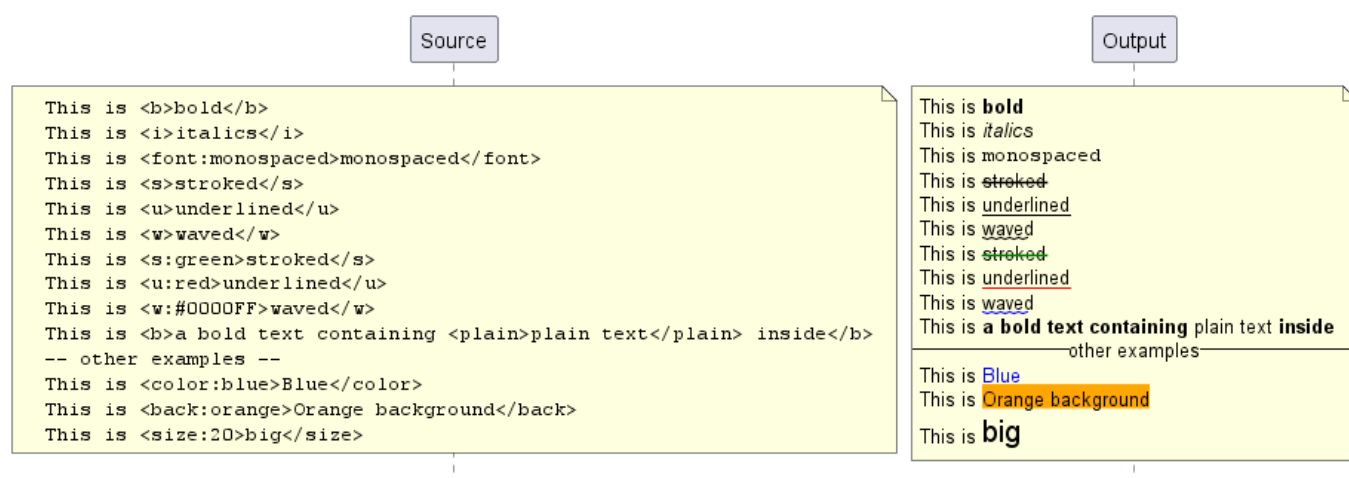
```
@startuml
hide footbox
note over Source
<code>
  This is <b>bold</b>
  This is <i>italics</i>
  This is <font:monospaced>monospaced</font>
  This is <s>stroked</s>
  This is <u>underlined</u>
  This is <w>waved</w>
  This is <s:green>stroked</s>
  This is <u:red>underlined</u>
  This is <w:#0000FF>waved</w>
  This is <b>a bold text containing <plain>plain text</plain> inside</b>
  -- other examples --
  This is <color:blue>Blue</color>
  This is <back:orange>Orange background</back>
  This is <size:20>big</size>
</code>
end note
/note over Output
  This is <b>bold</b>
```



```

This is <i>italics</i>
This is <font:monospaced>monospaced</font>
This is <s>stroked</s>
This is <u>underlined</u>
This is <w>waved</w>
This is <s:green>stroked</s>
This is <u:red>underlined</u>
This is <w:#0000FF>waved</w>
This is <b>a bold text containing <plain>plain text</plain> inside</b>
-- other examples --
This is <color:blue>Blue</color>
This is <back:orange>Orange background</back>
This is <size:20>big</size>
end note
@enduml

```



[Ref. QA-5254 for *plain*]

22.12.2 上付き文字、下付き文字 [sub, sup]

```

@startuml
:<code>
これは「カフェイン」の分子式です: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
</code>
これは「カフェイン」の分子式です: C<sub>8</sub>H<sub>10</sub>N<sub>4</sub>O<sub>2</sub>
----
<code>
これはピタゴラスの定理です: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>
</code>
これはピタゴラスの定理です: a<sup>2</sup> + b<sup>2</sup> = c<sup>2</sup>;
@enduml

```

これは「カフェイン」の分子式です: C₈H₁₀N₄O₂

これは「カフェイン」の分子式です: C₈H₁₀N₄O₂

これはピタゴラスの定理です: a² + b² = c²

これはピタゴラスの定理です: a² + b² = c²

22.13 OpenIconic

OpenIconic はオープンソースの素晴らしいアイコンセットです。これらのアイコンは creole パーサーに組み込まれているので、簡単に使用することができます。



次の構文を使用します：<&ICON_NAME>.

```
@startuml
title: <size:20><&heart>OpenIconicを使用<&heart></size>
class Wifi
note left
    Click on <&wifi>
end note
@enduml
```

♥OpenIconicを使用♥



利用可能なアイコンの一覧は、OpenIconic Website にあります。もしくは、次の特別なダイアグラムを使用します：

```
@startuml
listopeniconic
@enduml
```

List Open Iconic

Credit to
<https://useiconic.com/open>

account-login	bell	cloud	excerpt	justify-right	musical-note	star
account-logout	bluetooth	cloudy	expand-down	key	paperclip	sun
action-redo	bold	code	expand-left	laptop	pencil	tablet
action-undo	bolt	cog	expand-right	layers	people	tag
align-center	book	collapse-down	expand-up	lightbulb	person	tags
align-left	bookmark	collapse-left	external-link	link-broken	phone	target
align-right	box	collapse-right	eye	link-intact	pie-chart	task
aperture	briefcase	collapse-up	eyedropper	list-rich	pin	terminal
arrow-bottom	british-pound	command	file	list	play-circle	text
arrow-circle-bottom	browser	comment-square	fire	location	plus	thumb-down
arrow-circle-left	brush	compass	flag	lock-locked	power-standby	thumb-up
arrow-circle-right	bug	contrast	flash	lock-unlocked	print	timer
arrow-circle-top	bullhorn	copywriting	folder	loop-circular	project	transfer
arrow-left	calculator	credit-card	fork	loop-square	pulse	trash
arrow-right	calendar	crop	fullscreen-enter	loop	puzzle-piece	underline
arrow-thick-bottom	camera-slr	dashboard	fullscreen-exit	magnifying-glass	question-mark	vertical-align-bottom
arrow-thick-left	caret-bottom	data-transfer-download	globe	map-marker	rain	vertical-align-center
arrow-thick-right	caret-left	data-transfer-upload	graph	map	random	vertical-align-top
arrow-thick-top	caret-right	delete	grid-four-up	media-pause	reload	video
arrow-top	caret-top	dial	grid-three-up	media-play	resize-both	volume-high
audio-spectrum	cart	document	grid-two-up	media-record	resize-height	volume-low
audio	chat	dollar	hard-drive	media-skip-backward	resize-width	volume-off
badge	check	double-quote-sans-left	header	media-skip-forward	rss-alt	warning
ban	chevron-bottom	double-quote-sans-right	headphones	media-step-backward	rss	wifi
bar-chart	chevron-left	double-quote-serif-left	heart	media-step-forward	script	wrench
basket	chevron-right	double-quote-serif-right	home	media-stop	share-boxed	x
battery-empty	chevron-top	droplet	image	medical-cross	share	yen
battery-full	circle-check	eject	inbox	menu	shield	zoom-in
beaker	circle-x	elevator	infinity	microphone	signal	zoom-out
	clipboard	ellipses	info	minus	signpost	
	clock	envelope-closed	italic	monitor	sort-ascending	
	cloud-download	envelope-open	justify-center	moon	sort-descending	
	cloud-upload	euro	justify-left	move	spreadsheet	

22.14 Appendix: Examples of "Creole List" on all diagrams

22.14.1 Activity

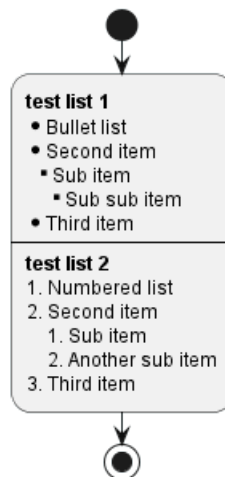
```
@startuml
start
:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
```



```

**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
stop
@enduml

```



22.14.2 Class

TODO: FIXME

- *Sub item*
- *Sub sub item*

TODO: FIXME

```

@startuml

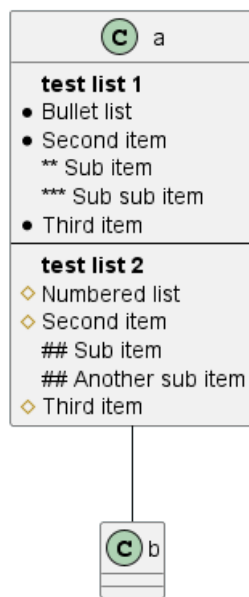
class a {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

a -- b

@enduml

```



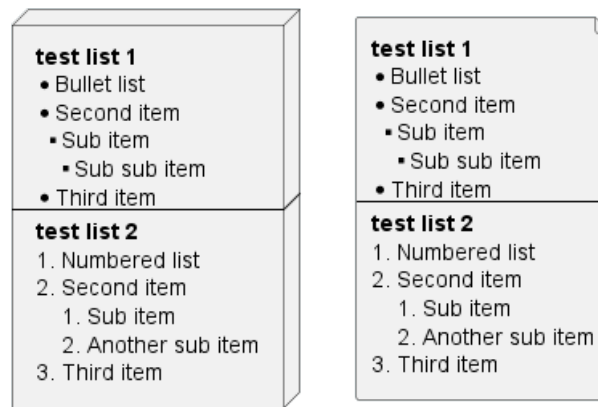


22.14.3 Component, Deployment, Use-Case

```
@startuml
node n [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

file f as "
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
"
@enduml
```





TODO: DONE [Corrected in V1.2020.18]

22.14.4 Gantt project planning

N/A

22.14.5 Object

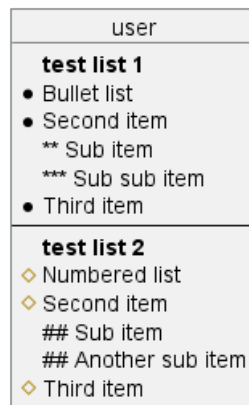
TODO: FIXME

- *Sub item*
- *Sub sub item*

TODO: FIXME

```
@startuml
object user {
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
}

@enduml
```



22.14.6 MindMap

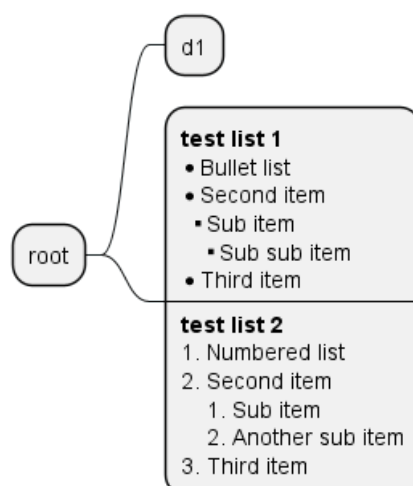
@startmindmap

```

* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;

```

@endmindmap



22.14.7 Network (nwdiag)

```

@startuml
nwdiag {

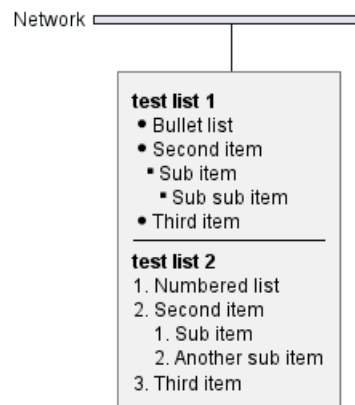
```



```

network Network {
    Server [description="**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub i
}
@enduml

```

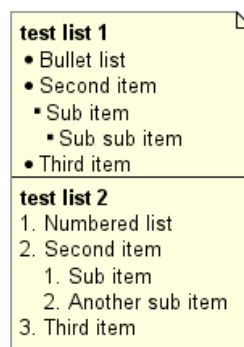


22.14.8 Note

```

@startuml
note as n
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
end note
@enduml

```



22.14.9 Sequence

```

@startuml
<style>
participant {HorizontalAlignment left}
</style>

```



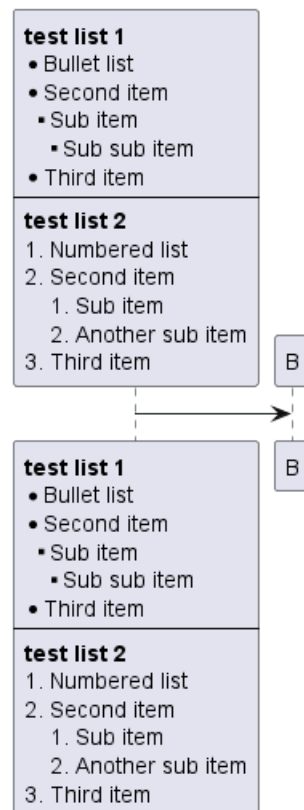
```

participant Participant [
**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
]

participant B

Participant -> B
@enduml

```



[Ref. QA-15232]

22.14.10 State

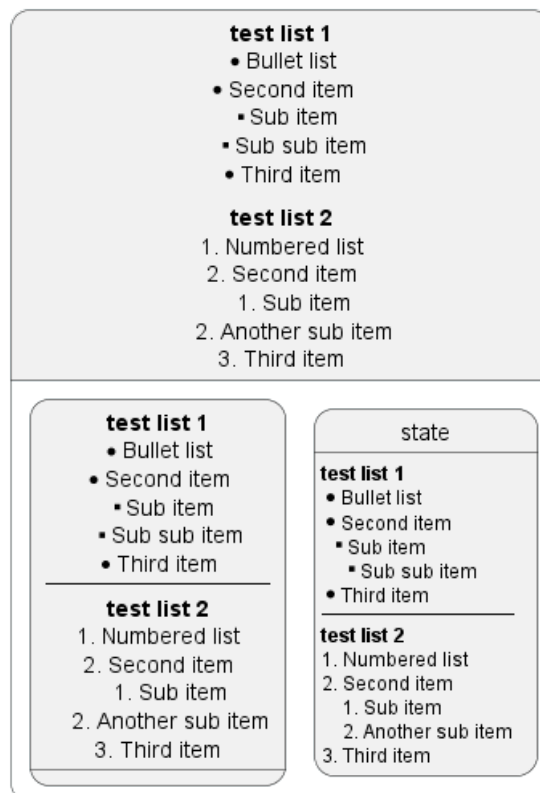
```

@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n-
state "**test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n-

```



```
state : **test list 1**\n* Bullet list\n* Second item\n** Sub item\n*** Sub sub item\n* Third item\n\n}\n@enduml
```



[Ref. QA-16978]

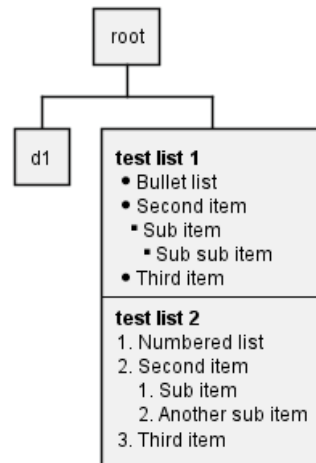
22.14.11 WBS

```
@startwbs
```

```
* root
** d1
**:**test list 1**
* Bullet list
* Second item
** Sub item
*** Sub sub item
* Third item
----
**test list 2**
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item;
```

```
@endwbs
```





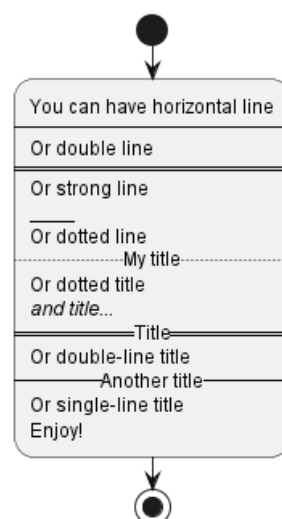
22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams

22.15.1 Activity

TODO: FIXME strong line ____ TODO: FIXME

```

@startuml
start
:You can have horizontal line
----
Or double line
====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!;
stop
@enduml
  
```



22.15.2 Class

```

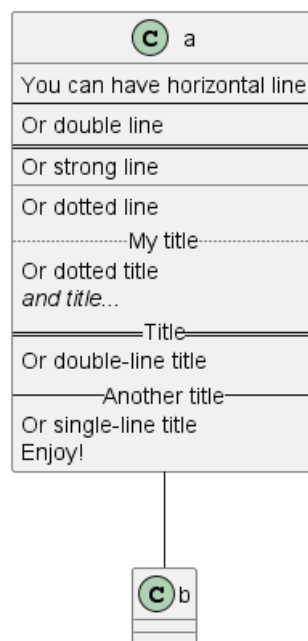
@startuml

class a {
You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..
Or dotted title
//and title... //
==Title==
Or double-line title
--Another title--
Or single-line title
Enjoy!
}

a -- b

@enduml

```



22.15.3 Component, Deployment, Use-Case

```

@startuml
node n [
You can have horizontal line
----
Or double line
=====
Or strong line
-----
Or dotted line
..My title..

```



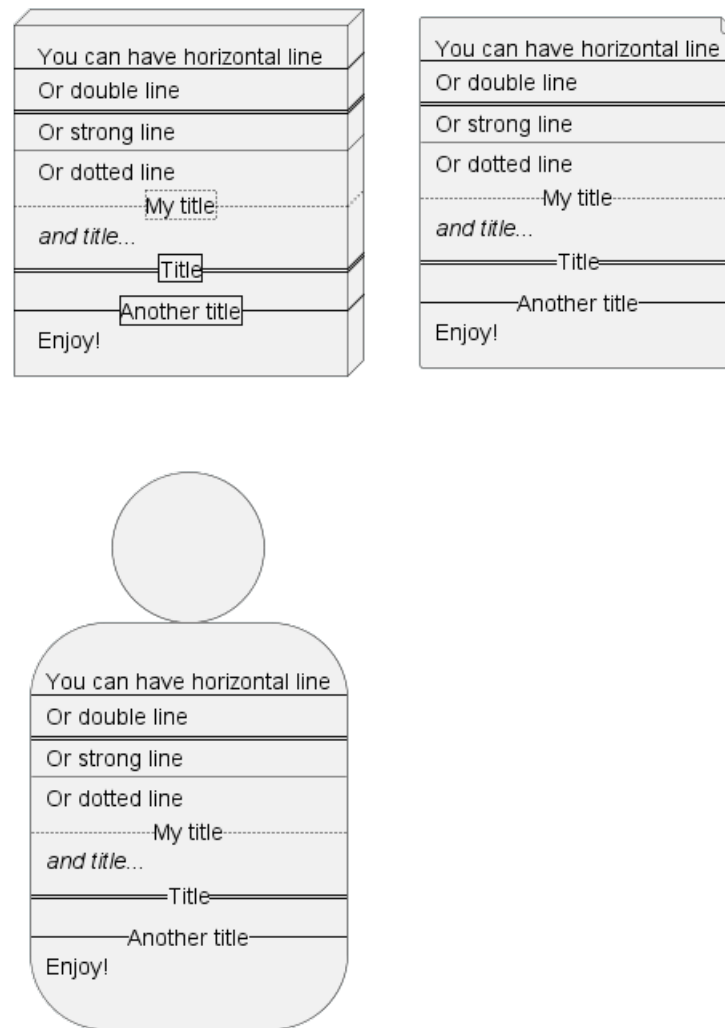
```
//and title... //
==Title==
--Another title--
Enjoy!
]

file f as "
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
"

person p [

You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!

]
@enduml
```



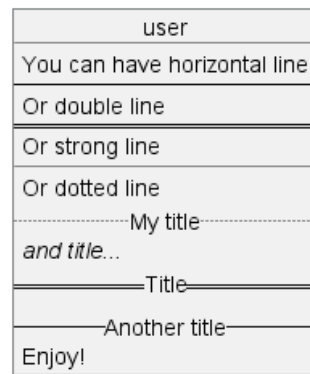
22.15.4 Gantt project planning

N/A

22.15.5 Object

```
@startuml
object user {
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
}

@enduml
```



TODO: DONE [Corrected on V1.2020.18]

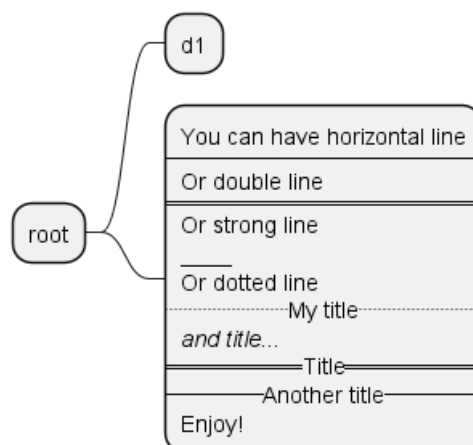
22.15.6 MindMap

TODO: FIXME strong line ____ **TODO:** FIXME

@startmindmap

```
* root
** d1
** :You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;
```

@endmindmap



22.15.7 Network (nwdiag)

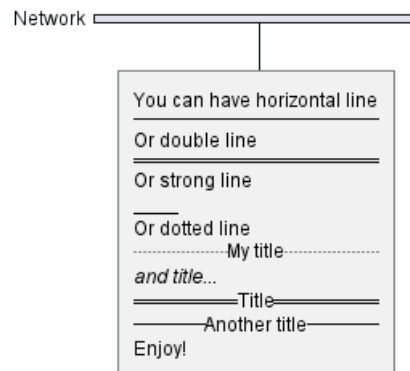
```
@startuml
nwdiag {
  network Network {
```



```

    Server [description="You can have horizontal line\n----\nOr double line\n===\nOr strong line\n\n"]
}
@enduml

```

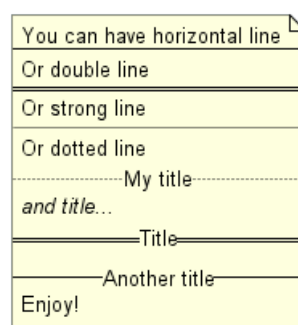


22.15.8 Note

```

@startuml
note as n
You can have horizontal line
----
Or double line
===
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
end note
@enduml

```



22.15.9 Sequence

```

@startuml
<style>
participant {HorizontalAlignment left}
</style>
participant Participant [
You can have horizontal line
----
Or double line

```



```

====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!
]

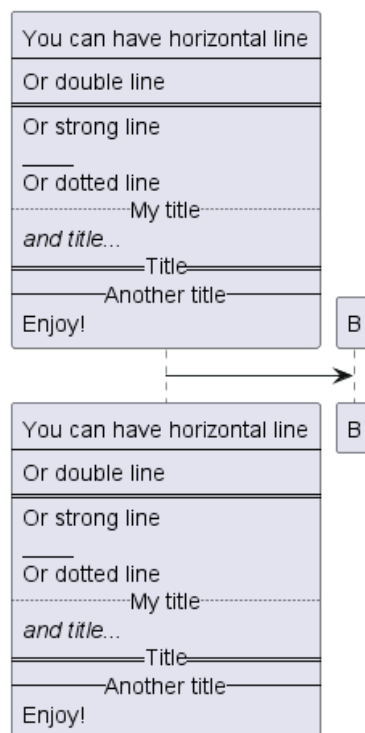
```

participant B

```

Participant -> B
@enduml

```



[Ref. QA-15232]

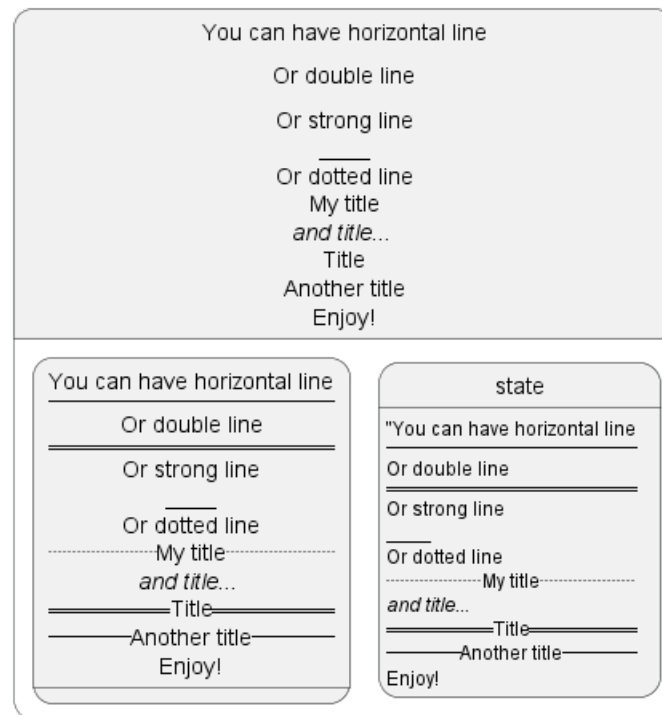
22.15.10 State

```

@startuml
<style>
stateDiagram {
title {HorizontalAlignment left}
}
</style>
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted line\n....\nMy title\n..and title...\n//Title\n--Another title\n--\nEnjoy!"
state "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted line\n....\nMy title\n..and title...\n//Title\n--Another title\n--\nEnjoy!"
state : "You can have horizontal line\n----\nOr double line\n====\nOr strong line\n____\nOr dotted line\n....\nMy title\n..and title...\n//Title\n--Another title\n--\nEnjoy!"
}
@enduml

```





[Ref. QA-16978]

22.15.11 WBS

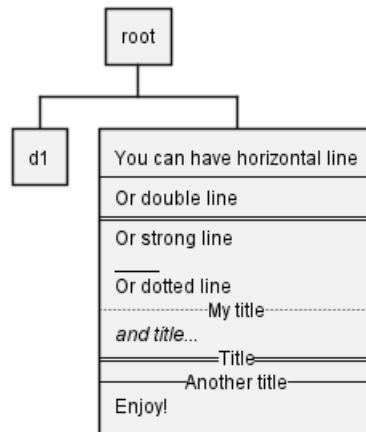
TODO: FIXME strong line ____ **TODO:** FIXME

@startwbs

```
* root
** d1
** :You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
//and title... //
==Title==
--Another title--
Enjoy!;
```

@endwbs





22.16 スタイル対応表 (Creole と HTML)

スタイル	Creole	Legacy HTML like
bold	This is bold	This is bold
<i>italics</i>	This is <i>italics</i>	This is <i>italics</i>
monospaced	This is "monospaced"	This is <font:monospaced>monospaced
stroked	This is --stroked--	This is <s>stroked</s>
underlined	This is __underlined__	This is <u>underlined</u>
waved	This is ~~~	This is <w>waved</w>

@startmindmap

* CreoleとHTMLで\n同一のスタイルを設定

:Creole**

<#silver>|= code|= output|

```

| \n This is "~~~bold~~~"\n | \n This is bold |
| \n This is "~~//italics~~"\n | \n This is italics |
| \n This is "~~"monospaced~~" "\n | \n This is "monospaced" |
| \n This is "~~--stroked~~"\n | \n This is --stroked-- |
| \n This is "~~__underlined__"\n | \n This is __underlined__ |
| \n This is "<U+007E><U+007E>waved<U+007E><U+007E>"\n | \n This is ~~~waved~~ |;
**:<b>Legacy HTML like

```

<#silver>|= code|= output|

```

| \n This is "~~<b>bold</b>~~"\n | \n This is <b>bold</b> |
| \n This is "~~<i>italics</i>~~"\n | \n This is <i>italics</i> |
| \n This is "~~<font:monospaced>monospaced</font>~~"\n | \n This is <font:monospaced>monospaced</font> |
| \n This is "~~<s>stroked</s>~~"\n | \n This is <s>stroked</s> |
| \n This is "~~<u>underlined</u>~~"\n | \n This is <u>underlined</u> |
| \n This is "~~<w>waved</w>~~"\n | \n This is <w>waved</w> |

```

And color as a bonus...

<#silver>|= code|= output|

```

| \n This is "~~<s>""<color:green>"green"</color>""</s>~~"\n | \n This is <s:green>stroked</s> |
| \n This is "~~<u>""<color:red>"red"</color>""</u>~~"\n | \n This is <u:red>underlined</u> |
| \n This is "~~<w>""<color:#0000FF>"#0000FF"</color>""</w>~~"\n | \n This is <w:#0000FF>waved</w> |

```

@endmindmap



CreoleとHTMLで
同一のスタイルを設定

Creole	
code	output
This is **bold**	This is bold
This is <i>//italics//</i>	This is <i>italics</i>
This is <code>""monospaced""</code>	This is monospaced
This is --stroked--	This is stroked
This is <u>__underlined__</u>	This is <u>underlined</u>
This is <u>~~waved~~</u>	This is <u>waved</u>

Legacy HTML like	
code	output
This is bold	This is bold
This is <i>italics</i>	This is <i>italics</i>
This is <font:monospaced>monospaced	This is monospaced
This is <s>stroked</s>	This is stroked
This is <u>underlined</u>	This is <u>underlined</u>
This is <w>waved</w>	This is <u>waved</u>
And color as a bonus...	
code	output
This is <s:green>stroked</s>	This is stroked
This is <u:red>underlined</u>	This is <u>underlined</u>
This is <w:#0000FF>waved</w>	This is <u>waved</u>



23 スプライトの定義と使用

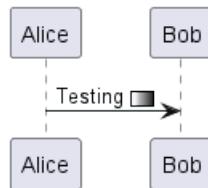
スプライトとは、図の中で使用できる小さい画像のことです。

PlantUML では、モノクロで 4、8、16 段階のグレースケールが使えます。

スプライトの定義は、ピクセルごとに 0~F の 16 進数を使用します。

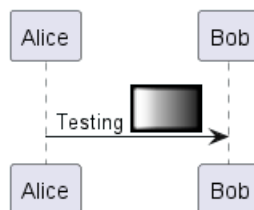
その後、<\$XXX> の形式でスプライトが使用できます。XXX はスプライトの名前が入ります。where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1>
@enduml
```



スプライトの倍率を変えることができます。

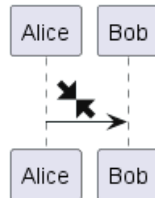
```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



23.1 Inline SVG sprite

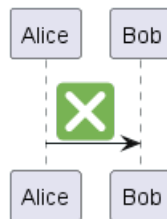
You can also use inlined SVG for sprites. Only a tiny subset of SVG directives is possible, so you probably have to compress existing SVG files using <https://vecta.io/nano>.

```
@startuml
sprite foo1 <svg width="8" height="8" viewBox="0 0 8 8"><path d="M1 0l-1 1 1.5 1.5-1.5 1.5h4v-4l-1.5
Alice->Bob : <$foo1*3>
@enduml
```



Another example:

```
@startuml
sprite foo1 <svg viewBox="0 0 36 36">
<path fill="#77B255" d="M36 32c0 2.209-1.791 4-4 4H4c-2.209 0-4-1.791-4-4V4c0-2.209 1.791-4 4-4h28c2
<path fill="#FFF" d="M21.529 18.006l8.238-8.238c.977-.976.977-2.559 0-3.535-.977-.977-2.559-.977-3.5
</svg>
Alice->Bob : <$foo1>
@enduml
```

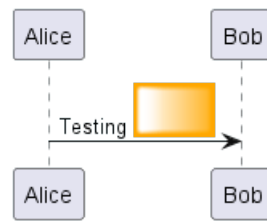


23.2 色の変更

スプライトはモノクロですが、色を変更することが可能です。

```
@startuml
sprite $foo1 {
FFFFFFFFFFFFFFFFF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
F0123456789ABCF
FFFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1,scale=3.4,color=orange>
@enduml
```





23.3 スプライトへの変換

次のコマンドで、スプライトをエンコードできます：

```
java -jar plantuml.jar -encodesprite 16z foo.png
```

foo.png は使用したい画像です（自動的にグレースケールに変換されます）

-encodesprite に続けて、形式を指定します。4、8、16、4z、8z、16z が使用可能です。

数値はグレースケールの段階数を表します。z を付けるとスプライトの定義を圧縮することができます。

23.4 スプライトをインポートする

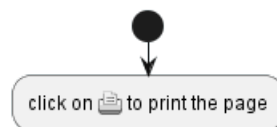
GUI を起動して、既存の画像からスプライトを生成することもできます。

メニューバーの **File/Open Sprite Window** をクリックします。

クリップボードに画像をコピーすると、それに対応したスプライトの定義がいくつか表示されます。その中から必要なものを選択してください。

23.5 例

```
@startuml
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEqB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrr
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwv
sprite $disk {
  444445566677881
  436000000009991
  43600000000ACA1
  53700000001A7A1
  53700000012B8A1
  53800000123B8A1
  63800001233C9A1
  634999AABBC99B1
  744566778899AB1
  7456AAAAA99AAB1
  8566AFC228AABB1
  8567AC8118BBBB1
  867BD4433BBBBB1
  39AAAAAABBBBBBC1
}
}
```



```

title Use of sprites (<$printer>, <$bug>...)

class Example {
  Can have some bug : <$bug>
  Click on <$disk> to save
}

note left : The printer <$printer> is available

@enduml

```



23.6 StdLib

PlantUML StdLib には、アーキテクチャ、クラウドサービス、ロゴなどの様々な IT 関連のアイコンが含まれています。そこには AWS、Azure、Kubernetes、C4、プロダクトロゴなどが含まれます。これらのライブラリを検索するには：

- PlantUML StdLib の Github フォルダを参照する。
- 興味のある StdLib コレクションのソースリポジトリを参照する。例えば、ロゴに興味がある場合、これは `gilbarbara-plantuml-sprites` から来ており、スプライトの一覧をすぐに見つけることができます。（次のセクションで示すスプライトの一覧表示は、残念ながらグレースケールとなりますが、この一覧では色が着いています。）
- Hitchhiker's Guide to PlantUML の詳細を読む。Standard Library Sprites や PlantUML Stdlib Overview のセクションなど。

23.7 スプライトを一覧表示する

`listsprites` コマンドを使って、スプライトの一覧を表示できます：

- これ単体で使用すると、ArchiMate スプライトの一覧を表示します。
- 図中に何らかのスプライトライブラリをインクルードしている場合、このコマンドはそれらのスプライトの一覧を表示します。View all the icons with `listsprites` に説明があります。

(Hitchhikers Guide to PlantUML からの例)

```

@startuml

!define osaPuml https://raw.githubusercontent.com/Crashedmind/PlantUML-opensecurityarchitecture2-icons
!include osaPuml/Common.puml
!include osaPuml/User/all.puml

listsprites
@enduml

```





多くのコレクションは **all** というファイルを含んでいて、コレクション全体を一度に見ることができます。もしくは、必要なスプライトを一つずつ探してインクルードします。残念ながら、StdLib に含まれるコレクションのバージョンには **all** ファイルが含まれていないものも多いので、上で見たように StdLib からではなく、github からインクルードしています。

スプライトはすべてグレースケールですが、ほとんどのコレクションには適切な色を含んだマクロが定義されています。

24 skinparam コマンド

skinparam コマンドを使用すると、描画される色やフォントを変更することができます。

例:

```
skinparam backgroundColor transparent
```

24.1 使用法

このコマンドは、

- 他のコマンドと同様に、図の定義中
- インクルードしたファイル中
- コマンドラインや Ant タスクで指定された設定ファイル中

で使用することができます。

24.2 入れ子

繰り返しを避けるため、定義を入れ子にすることができます。例えば、

```
skinparam xxxxParam1 value1
skinparam xxxxParam2 value2
skinparam xxxxParam3 value3
skinparam xxxxParam4 value4
```

この定義と以下の定義は全く同じ意味になります:

```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

24.3 白黒

skinparam monochrome true コマンドを使用すると、出力結果を白黒にすることができます。

```
@startuml
```

```
skinparam monochrome true
```

```
actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C
```

```
User -> A: DoWork
activate A
```

```
A -> B: Create Request
activate B
```

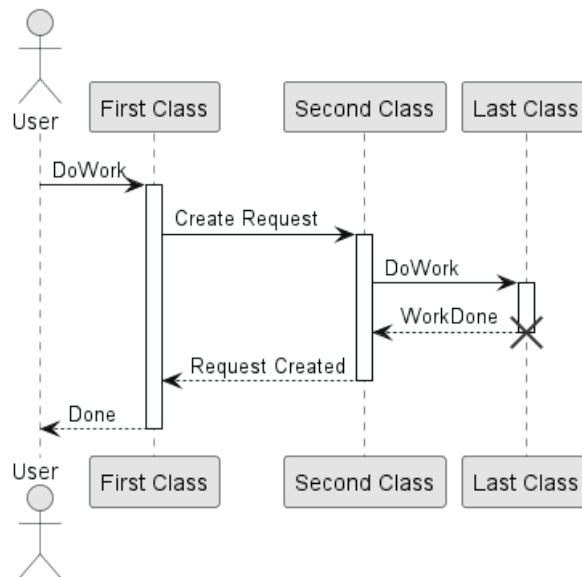
```
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C
```

```
B --> A: Request Created
deactivate B
```



```
A --> User: Done
deactivate A
```

```
@enduml
```



24.4 影の有無

skinparam shadowing false を使用すると、影の描画を無くすことができます。

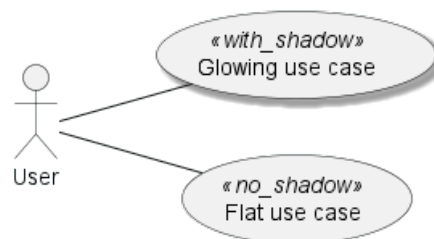
```
@startuml
```

```
left to right direction
```

```
skinparam shadowing<<no_shadow>> false
skinparam shadowing<<with_shadow>> true
```

```
actor User
(Glowing use case) <<with_shadow>> as guc
(Flat use case) <<no_shadow>> as fuc
User -- guc
User -- fuc
```

```
@enduml
```



24.5 色の反転

skinparam monochrome reverse コマンドを使用すると、出力結果が色の反転した白黒となります。これは、背景色が黒の環境の場合に便利です。

```
@startuml
```



```

skinparam monochrome reverse

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

A -> B: Create Request
activate B

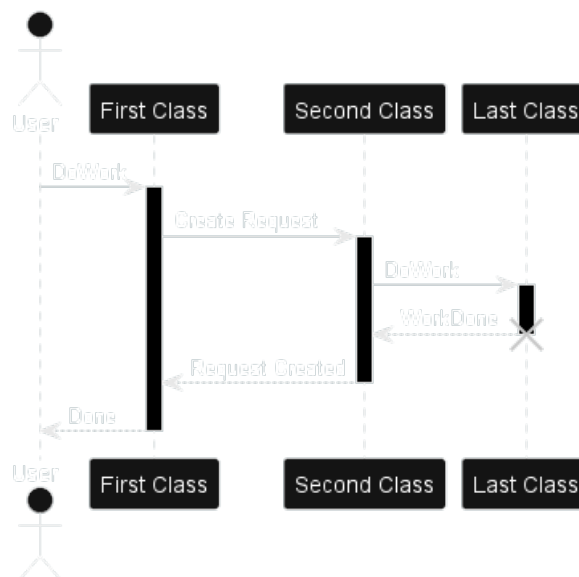
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



24.6 色

標準色名や RGB コードを使用することができます。

```

@startuml
colors
@enduml

```

APPLICATION	Crimson	DeepPink	Indigo	LightYellow	Navy	RoyalBlue	Turquoise
AliceBlue	Cyan	DeepSkyBlue	Ivory	Lime	OldLace	STRATEGY	Violet
AntiqueWhite	DarkBlue	DimGray	Khaki	LimeGreen	Olive	SaddleBrown	Wheat
Aqua	DarkCyan	DimGrey	Lavender	Linen	OliveDrab	Salmon	White
Aquamarine	DarkGoldenRod	DodgerBlue	LavenderBlush	MOTIVATION	Orange	SandyBrown	WhiteSmoke
Azure	DarkGray	FireBrick	LawnGreen	Magenta	OrangeRed	SeaGreen	Yellow
BUSINESS	DarkGreen	FloralWhite	LemonChiffon	Maroon	Orchid	SeaShell	YellowGreen
Beige	DarkGrey	ForestGreen	LightBlue	MediumAquaMarine	PHYSICAL	Sienna	
Bisque	DarkKhaki	Fuchsia	LightCoral	MediumBlue	PaleGoldenRod	Silver	
Black	DarkMagenta	Gainsboro	LightCyan	MediumOrchid	PaleGreen	SkyBlue	
BlanchedAlmond	DarkOliveGreen	GhostWhite	LightGoldenRodYellow	MediumPurple	PaleTurquoise	SlateBlue	
Blue	DarkOrchid	Gold	LightGray	MediumSeaGreen	PaleVioletRed	SlateGray	
BlueViolet	DarkRed	GoldenRod	LightGreen	MediumSlateBlue	PapayaWhip	SlateGrey	
Brown	DarkSalmon	Gray	LightGrey	MediumSpringGreen	PeachPuff	Snow	
BurlyWood	DarkSeaGreen	Green	LightPink	MediumTurquoise	Peru	SpringGreen	
CadetBlue	DarkSlateBlue	GreenYellow	LightSalmon	MediumVioletRed	Pink	SteelBlue	
Chartreuse	DarkSlateGray	Grey	LightSeaGreen	MidnightBlue	Plum	TECHNOLOGY	
Chocolate	DarkSlateGrey	HoneyDew	LightSkyBlue	MintCream	PowderBlue	Tan	
Coral	DarkTurquoise	HotPink	LightSlateGray	MistyRose	Purple	Teal	
CornflowerBlue	DarkViolet	IMPLEMENTATION	LightSlateGrey	Moccasin	Red	Thistle	
Cornsilk	Darkorange	IndianRed	LightSteelBlue	NavajoWhite	RosyBrown	Tomato	

transparent は、画像の背景色にのみ使用可能です。

24.7 文字色、フォント名、フォントサイズ

xxxFontColor、xxxFontSize、xxxFontName のパラメータを使用して、文字の描画を変更することができます。

例:

```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Aapex
```

skinparam defaultFontName を使用すると、すべての文字描画に対してデフォルト値を設定することもできます。

例:

```
skinparam defaultFontName Aapex
```

フォント名はシステムに大きく依存していることにご注意ください。ポータビリティを考慮するなら、最小限の使用にとどめてください。Helvetica と Courier であれば、どのようなシステムでも使用できるでしょう。

多くのパラメータがありますが、次のコマンドで一覧を出力することができます:

```
java -jar plantuml.jar -language
```

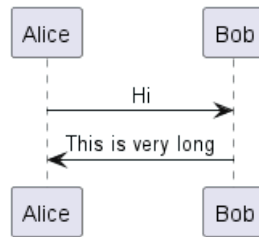
24.8 テキストの配置

skinparam sequenceMessageAlign を使用すると、テキストの配置を left、right、center に設定することができます。direction または reverseDirection を使用すると、矢印の方向に応じてテキストの位置が決定されます。

パラメタ名	デフォルト値	備考
sequenceMessageAlign	left	シーケンス図のメッセージに使用
sequenceReferenceAlign	center	シーケンス図の ref over に使用

```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml
```

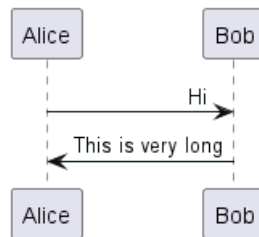




```

@startuml
skinparam sequenceMessageAlign right
Alice -> Bob : Hi
Bob -> Alice : This is very long
@enduml

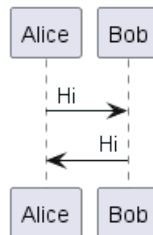
```



```

@startuml
skinparam sequenceMessageAlign direction
Alice -> Bob : Hi
Bob -> Alice: Hi
@enduml

```



24.9 例

```

@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

skinparam sequence {
  ArrowColor DeepSkyBlue
  ActorBorderColor DeepSkyBlue
  LifeLineBorderColor blue
  LifeLineBackgroundColor #A9DCDF

  ParticipantBorderColor DeepSkyBlue
  ParticipantBackgroundColor DodgerBlue
  ParticipantFontName Impact
  ParticipantFontSize 17
  ParticipantFontColor #A9DCDF

  ActorBackgroundColor aqua
  ActorFontColor DeepSkyBlue
  ActorFontSize 17
  ActorFontName Aapex
}

```



```

}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C

User -> A: DoWork
activate A

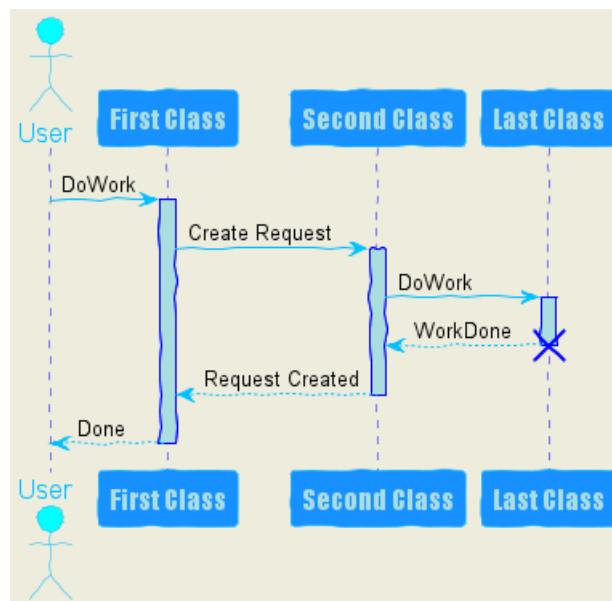
A -> B: Create Request
activate B

B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A
@enduml

```



```

@startuml
skinparam handwritten true

skinparam actor {
  BorderColor black
  FontName Courier
  BackgroundColor<< Human >> Gold
}

skinparam usecase {
  BackgroundColor DarkSeaGreen
  BorderColor DarkSlateGray
}

BackgroundColor<< Main >> YellowGreen

```

```

BorderColor<< Main >> YellowGreen

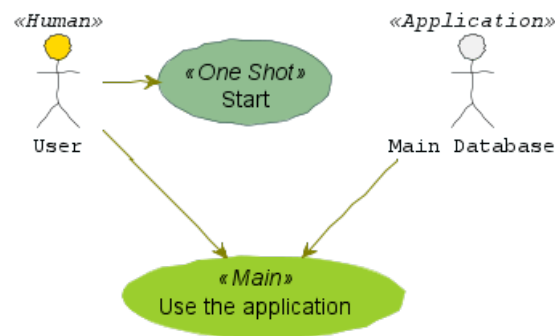
ArrowColor Olive
}

User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

User -> (Start)
User --> (Use)

MySql --> (Use)
@enduml

```



```

@startuml
skinparam roundcorner 20
skinparam class {
  BackgroundColor PaleGreen
  ArrowColor SeaGreen
  BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

```

```

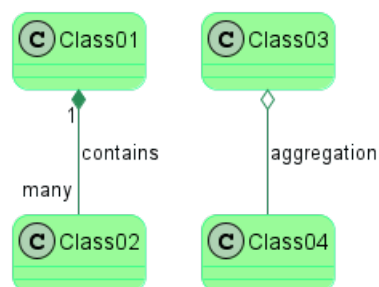
Class01 "1" *-- "many" Class02 : contains

```

```

Class03 o-- Class04 : aggregation
@enduml

```



```

@startuml
skinparam interface {
  backgroundColor RosyBrown
  borderColor orange
}

```

```

skinparam component {
  FontSize 13
}

```



```

BackgroundColor<<Apache>> LightCoral
BorderColor<<Apache>> #FF6655
FontName Courier
BorderColor black
BackgroundColor gold
ArrowFontName Impact
ArrowColor #FF6655
ArrowFontColor #777777
}

```

```

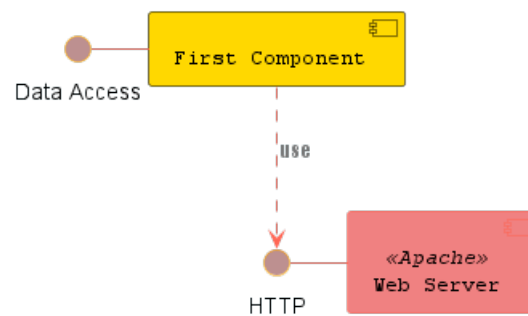
() "Data Access" as DA
[Web Server] << Apache >>

```

```

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server]
@enduml

```



```

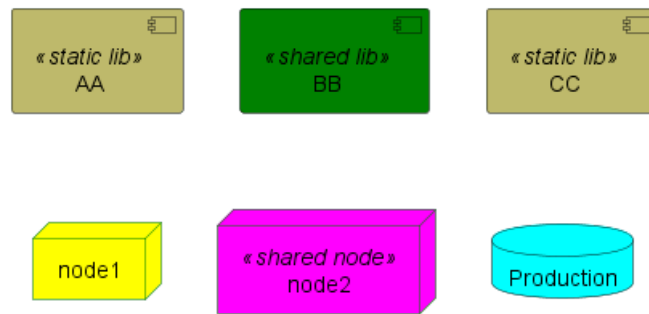
@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

node node1
node node2 <<shared node>>
database Production

skinparam component {
    backgroundColor<<static lib>> DarkKhaki
    backgroundColor<<shared lib>> Green
}

skinparam node {
    borderColor Green
    backgroundColor Yellow
    backgroundColor<<shared node>> Magenta
}
skinparam databaseBackgroundColor Aqua
@enduml

```



24.10 すべての skinparam パラメータの一覧

コマンドラインで `-language` を使用するか、次のコマンドを使用して、skinparam パラメータの一覧を表示する図を生成することができます:

- `help skinparams`
- `skinparameters`

24.10.1 コマンドライン: `-language` コマンド

ドキュメンテーションは常に最新とは限らないため、次のコマンドを使用して全パラメータの一覧を確認することができます:

```
java -jar plantuml.jar -language
```

24.10.2 コマンド: `help skinparams`

以下の結果が得られます。このコマンドのソースコードは `CommandHelpSkinparam.java` にあります。

```
@startuml
help skinparams
@enduml
```

Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>

(Details by typing `license keyword`)



PlantUML 1.2023.11

[From string (line 2)]

```
@startuml
help skinparams
Syntax Error?
```

24.10.3 コマンド: `skinparameters`

```
@startuml
skinparameters
@enduml
```



ActivityBackgroundColor	ClassFontStyle	FolderStereotypeFontSize	NoteFontStyle	SequenceDelayFontName
ActivityBorderColor	ClassStereotypeFontColor	FolderStereotypeFontStyle	NoteShadowing	SequenceDelayFontSize
ActivityBorderThickness	ClassStereotypeFontName	FooterFontColor	NoteTextAlignment	SequenceDelayFontStyle
ActivityDiamondFontColor	ClassStereotypeFontSize	FooterFontName	ObjectAttributeFontColor	SequenceDividerBorderThickness
ActivityDiamondFontName	ClassStereotypeFontStyle	FooterFontSize	ObjectAttributeFontName	SequenceDividerFontColor
ActivityDiamondFontSize	CloudFontColor	FooterFontStyle	ObjectAttributeFontSize	SequenceDividerFontName
ActivityDiamondFontStyle	CloudFontName	FrameFontColor	ObjectAttributeFontStyle	SequenceDividerFontSize
ActivityFontColor	CloudFontSize	FrameFontName	ObjectBorderThickness	SequenceDividerFontStyle
ActivityFontName	CloudFontStyle	FrameFontSize	ObjectFontColor	SequenceGroupBodyBackgroundColor
ActivityFontSize	CloudStereotypeFontColor	FrameFontStyle	ObjectFontName	SequenceGroupBorderThickness
ActivityFontStyle	CloudStereotypeFontName	FrameStereotypeFontColor	ObjectFontSize	SequenceGroupFontColor
ActorBackgroundColor	CloudStereotypeFontStyle	FrameStereotypeFontName	ObjectFontStyle	SequenceGroupFontName
ActorBorderColor	ColorArrowSeparationSpace	FrameStereotypeFontSize	ObjectStereotypeFontColor	SequenceGroupFontSize
ActorFontColor	ComponentBorderThickness	FrameStereotypeFontStyle	ObjectStereotypeFontName	SequenceGroupFontStyle
ActorFontName	ComponentFontColor	GenericDisplay	ObjectStereotypeFontSize	SequenceGroupHeaderFontColor
ActorFontSize	ComponentFontName	Guillemet	ObjectStereotypeFontStyle	SequenceGroupHeaderFontName
ActorFontStyle	ComponentFontSize	Handwritten	PackageBorderThickness	SequenceGroupHeaderFontSize
ActorStereotypeFontColor	ComponentFontStyle	HeaderFontColor	PackageFontColor	SequenceGroupHeaderFontStyle
ActorStereotypeFontName	ComponentFontStyle	HeaderFontName	PackageFontName	SequenceLifeLineBorderColor
ActorStereotypeFontSize	ComponentStereotypeFontColor	HeaderFontSize	PackageFontSize	SequenceLifeLineBorderThickness
ActorStereotypeFontStyle	ComponentStereotypeFontName	HeaderFontStyle	PackageFontStyle	SequenceMessageAlignment
AgentBorderThickness	ComponentStereotypeFontSize	HexagonBorderThickness	PackageStereotypeFontColor	SequenceMessageTextAlignment
AgentFontColor	ComponentStereotypeFontStyle	HexagonFontColor	PackageStereotypeFontName	SequenceNewpageSeparatorColor
AgentFontName	ComponentStyle	HexagonFontName	PackageStereotypeFontSize	SequenceParticipant
AgentFontSize	ConditionEndStyle	HexagonFontSize	PackageStereotypeFontStyle	SequenceParticipantBorderThickness
AgentFontStyle	ConditionStyle	HexagonFontStyle	PackageStyle	SequenceReferenceAlignment
AgentStereotypeFontColor	ControlFontColor	HexagonStereotypeFontColor	PackageTitleAlignment	SequenceReferenceBackgroundColor
AgentStereotypeFontName	ControlFontName	HexagonStereotypeFontName	Padding	SequenceReferenceBorderThickness
AgentStereotypeFontSize	ControlFontSize	HexagonStereotypeFontSize	PageBorderColor	SequenceReferenceFontColor
AgentStereotypeFontStyle	ControlFontStyle	HexagonStereotypeFontStyle	PageExternalColor	SequenceReferenceFontName
ArchimateBorderThickness	ControlStereotypeFontColor	HyperlinkColor	PageMargin	SequenceReferenceFontSize
ArchimateFontColor	ControlStereotypeFontName	HyperlinkUnderline	ParticipantFontColor	SequenceReferenceFontStyle
ArchimateFontName	ControlStereotypeFontSize	IconEMandatoryColor	ParticipantFontName	SequenceReferenceHeaderBackgroundColor
ArchimateFontSize	ControlStereotypeFontStyle	IconPackageBackgroundColor	ParticipantFontSize	SequenceReferenceHeaderFontColor
ArchimateFontStyle	DatabaseFontColor	IconPackageColor	ParticipantFontStyle	SequenceStereotypeFontColor
ArchimateStereotypeFontColor	DatabaseFontName	IconPrivateBackgroundColor	ParticipantPadding	SequenceStereotypeFontSize
ArchimateStereotypeFontName	DatabaseFontSize	IconPrivateColor	ParticipantStereotypeFontColor	SequenceStereotypeFontStyle
ArchimateStereotypeFontSize	DatabaseFontStyle	IconProtectedBackgroundColor	ParticipantStereotypeFontName	Shadowing
ArchimateStereotypeFontStyle	DatabaseStereotypeFontColor	IconProtectedColor	ParticipantStereotypeFontSize	StackFontColor
ArrowFontColor	DatabaseStereotypeFontName	IconPublicBackgroundColor	ParticipantStereotypeFontStyle	StackFontName
ArrowFontName	DatabaseStereotypeFontSize	IconPublicColor	PartitionBorderThickness	StackFontSize
ArrowFontSize	DatabaseStereotypeFontStyle	InterfaceFontColor	PartitionFontColor	StackFontStyle
ArrowFontStyle	DefaultFontColor	InterfaceFontName	PartitionFontName	StackStereotypeFontColor
ArrowHeadColor	DefaultFontName	InterfaceFontSize	PartitionFontSize	StackStereotypeFontName
ArrowLollipopColor	DefaultFontSize	InterfaceFontStyle	PartitionFontStyle	StackStereotypeFontSize
ArrowMessageAlignment	DefaultFontStyle	InterfaceStereotypeFontColor	PathHoverColor	StackStereotypeFontStyle
ArrowThickness	DefaultMonospacedFontName	InterfaceStereotypeFontName	PersonBorderThickness	StateAttributeFontColor
ArtifactFontColor	DefaultTextAlignment	InterfaceStereotypeFontSize	PersonFontColor	StateAttributeFontName
ArtifactFontName	DesignedBackgroundColor	InterfaceStereotypeFontStyle	PersonFontName	StateAttributeFontSize
ArtifactFontSize	DesignedBorderColor	LabelFontColor	PersonFontSize	StateAttributeFontStyle
ArtifactFontStyle	DesignedDomainBorderThickness	LabelFontName	PersonFontStyle	StateBorderColor
ArtifactStereotypeFontColor	DesignedDomainFontColor	LabelFontSize	PersonStereotypeFontColor	StateFontColor
ArtifactStereotypeFontName	DesignedDomainFontName	LabelFontStyle	PersonStereotypeFontName	StateFontName
ArtifactStereotypeFontSize	DesignedDomainFontSize	LabelStereotypeFontColor	PersonStereotypeFontSize	StateFontSize
ArtifactStereotypeFontStyle	DesignedDomainFontStyle	LabelStereotypeFontName	PersonStereotypeFontStyle	StateFontStyle
BackgroundColor	DesignedDomainStereotypeFontColor	LabelStereotypeFontSize	QueueBorderThickness	StateMessageAlignment
BiddableBackgroundColor	DesignedDomainStereotypeFontName	LabelStereotypeFontStyle	QueueFontColor	StereotypePosition
BiddableBorderColor	DesignedDomainStereotypeFontSize	LegendBorderThickness	QueueFontName	StorageFontColor
BoundaryFontColor	DesignedDomainStereotypeFontStyle	LegendFontColor	QueueFontSize	StorageFontName
BoundaryFontName	DiagramBorderColor	LegendFontName	QueueFontStyle	StorageFontSize
BoundaryFontSize	DiagramBorderThickness	LegendFontSize	QueueStereotypeFontColor	StorageFontStyle
BoundaryFontStyle	DomainBackgroundColor	LegendFontStyle	QueueStereotypeFontName	StorageStereotypeFontColor
BoundaryStereotypeFontColor	DomainBorderColor	LexicalBackgroundColor	QueueStereotypeFontSize	StorageStereotypeFontName
BoundaryStereotypeFontName	DomainBorderThickness	LexicalBorderColor	QueueStereotypeFontStyle	StorageStereotypeFontSize
BoundaryStereotypeFontSize	DomainFontColor	LifelineStrategy	Ranksep	StorageStereotypeFontStyle
BoundaryStereotypeFontStyle	DomainFontName	Linetype	RectangleBorderThickness	Style
BoxPadding	DomainFontSize	MachineBackgroundColor	RectangleFontColor	SvglinkTarget
CaptionFontColor	DomainFontStyle	MachineBorderColor	RectangleFontName	SwimlaneBorderThickness
CaptionFontName	DomainStereotypeFontColor	MachineBorderThickness	RectangleFontName	SwimlaneTitleFontColor
CaptionFontSize	DomainStereotypeFontName	MachineFontColor	RectangleFontStyle	SwimlaneTitleFontName
CaptionFontStyle	DomainStereotypeFontSize	MachineFontName	RectangleStereotypeFontColor	SwimlaneTitleFontSize
CardBorderThickness	DomainStereotypeFontStyle	MachineFontSize	RectangleStereotypeFontName	SwimlaneTitleFontStyle
CardFontColor	Dpi	MachineFontStyle	RectangleStereotypeFontSize	SwimlaneWidth
CardFontName	EntityFontColor	MachineStereotypeFontColor	RectangleStereotypeFontStyle	SwimlaneWrapTitleWidth
CardFontSize	EntityFontName	MachineStereotypeFontName	RequirementBackgroundColor	TabSize
CardFontStyle	EntityFontSize	MachineStereotypeFontSize	RequirementBorderColor	TimingFontColor
CardStereotypeFontColor	EntityFontStyle	MachineStereotypeFontStyle	RequirementBorderThickness	TimingFontName
CardStereotypeFontName	EntityStereotypeFontColor	MaxAsciiMessageLength	RequirementFontColor	TimingFontSize
CardStereotypeFontSize	EntityStereotypeFontName	MaxMessageSize	RequirementFontName	TimingFontStyle
CardStereotypeFontStyle	EntityStereotypeFontSize	MinClassWidth	RequirementFontSize	TitleBorderRoundCorner
CircledCharacterFontColor	EntityStereotypeFontStyle	Monochrome	RequirementFontStyle	TitleBorderThickness
CircledCharacterFontName	FileFontColor	NodeFontColor	RequirementStereotypeFontColor	TitleFontColor
CircledCharacterFontSize	FileFontName	NodeFontName	RequirementStereotypeFontName	TitleFontName
CircledCharacterFontStyle	FileFontSize	NodeFontSize	RequirementStereotypeFontSize	TitleFontSize
CircledCharacterRadius	FileFontStyle	NodeFontStyle	RequirementStereotypeFontStyle	TitleFontStyle
ClassAttributeFontColor	FileStereotypeFontColor	NodeStereotypeFontColor	ResponseMessageBelowArrow	UsecaseBorderThickness
ClassAttributeFontName	FileStereotypeFontName	NodeStereotypeFontName	RoundCorner	UsecaseFontColor
ClassAttributeFontSize	FileStereotypeFontSize	NodeStereotypeFontStyle	SameClassWidth	UsecaseFontName
ClassAttributeFontStyle	FileStereotypeFontStyle	NodeStereotypeFontStyle	SequenceActorBorderThickness	UsecaseFontSize
ClassAttributeIconSize	FixCircleLabelOverlapping	Nodessep	SequenceArrowThickness	UsecaseFontStyle
ClassBackgroundColor	FolderFontColor	NoteBackgroundColor	SequenceBoxBorderColor	UsecaseStereotypeFontColor
ClassBorderColor	FolderFontName	NoteBorderColor	SequenceBoxFontColor	UsecaseStereotypeFontName
ClassBorderThickness	FolderFontSize	NoteBorderThickness	SequenceBoxFontName	UsecaseStereotypeFontSize
ClassFontColor	FolderFontStyle	NoteFontColor	SequenceBoxFontSize	UsecaseStereotypeFontStyle
ClassFontName	FolderStereotypeFontColor	NoteFontName	SequenceBoxFontStyle	WrapWidth
ClassFontSize	FolderStereotypeFontName	NoteFontSize	SequenceDelayFontColor	



24.10.4 All Skin Parameters on the Ashley's PlantUML Doc

Ashley's PlantUML Doc の All Skin Parameters で、各 *skinparam* パラメータの見た目を確認することができます。

- <https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html>.

25 前処理

PlantUML にはいくつかの前処理機能があり、それはすべてのダイアグラムに対して利用可能です。

これらの機能は、特殊文字 `#` がエクスクラメーションマーク `!` に置き換えられていることを除くと、C 言語のプリプロセッサに非常によく似ています。

25.1 変数定義 [=, ?=]

これは必須ではありませんが、変数名を `$` で始めることを強く推奨します。

データの種類は 3 つあります。

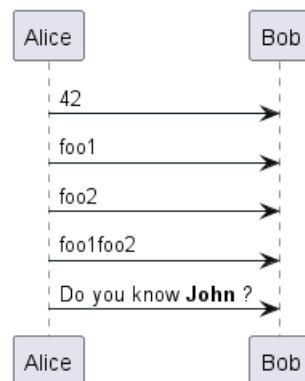
- 整数 (*int*)
- 文字列 (*str*) - シングルクォートもしくはダブルクォートで囲みます
- **JSON** (*JSON*) - 波括弧で囲みます

(JSON 変数の定義と使用方法については、Preprocessing-JSON のページを参照してください)

関数の外側に作られた変数はグローバルであり、関数内を含むどこからでもアクセスすることができます。このことを明示するために、変数定義に `global` というキーワードを付けることもできます。

```
@startuml
!$a = 42
!$ab = "foo1"
!$cd = "foo2"
!$ef = $ab + $cd
!$foo = { "name": "John", "age" : 30 }

Alice -> Bob : $a
Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
Alice -> Bob : Do you know **$foo.name** ?
@enduml
```



次の構文で、変数が未定義の場合のみ代入することができます: `!$a ?= "foo"`

```
@startuml
Alice -> Bob : 1. **$name** should be empty

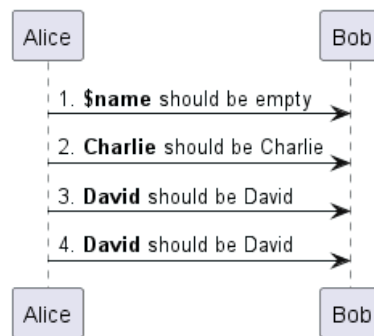
!$name ?= "Charlie"
Alice -> Bob : 2. **$name** should be Charlie

!$name = "David"
Alice -> Bob : 3. **$name** should be David

!$name ?= "Ethan"
Alice -> Bob : 4. **$name** should be David
```



```
@enduml
```



25.2 真偽値

25.2.1 真偽値の表現方法 [0 is false]

本当の意味での boolean 型はありませんが、PlantUML では慣例に従い次のように定義します:

- 整数型の 0 は **false** を意味します
- それ以外の非 null の数値 (1 など) と、すべての文字列 ("1"、あるいは"0" でさえも) は **true** を意味します

[Ref. QA-9702]

25.2.2 真偽値型の演算と演算子 [&&, ||, ()]

真偽値判定では、次の演算子を使用できます:

- 括弧 ()
- and 演算子 &&
- or 演算子 ||

(*if* の判定式の例を参照)

25.2.3 真偽値の組み込み関数 [%false(), %true(), %not(<exp>)]

利便性のため、次の組み込み関数が用意されています:

- %false()
- %true()
- %not(<exp>)

[組み込み関数を参照]

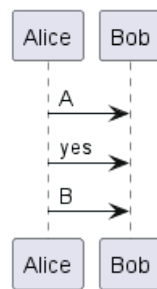
25.3 条件分岐 [!if, !else, !elseif, !endif]

- 条件部に式を記述することができます。
- *else* と *elseif* を使用することもできます。

```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
Alice -> Bob : B
```



@enduml



25.4 While ループ [!while, !endwhile]

!while と!endwhile キーワードを使用して繰り返しを記述できます。

25.4.1 While ループ (アクティビティ図での例)

```

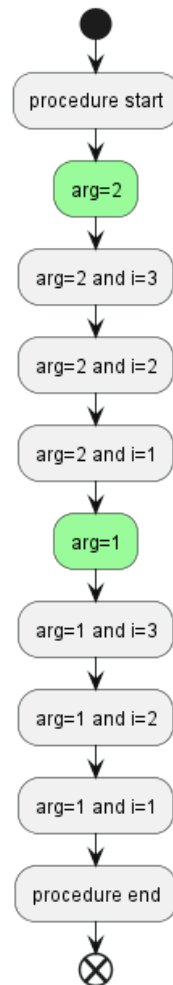
@startuml
!procedure $foo($arg)
  :procedure start;
  !while $arg!=0
    !$i=3
    #palegreen:arg=$arg;
    !while $i!=0
      :arg=$arg and i=$i;
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
  :procedure end;
!endprocedure

```

```

start
$foo(2)
end
@enduml

```



[Adapted from QA-10838]

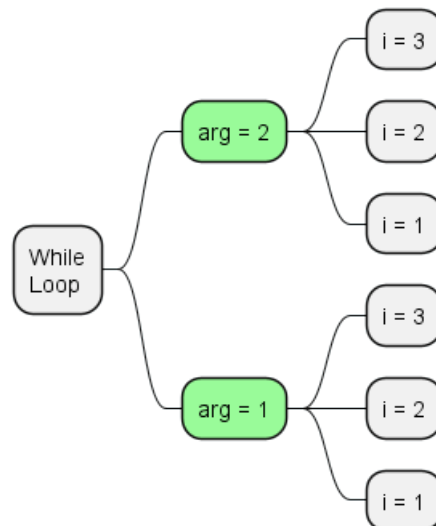
25.4.2 While ループ (マインドマップでの例)

```

@startmindmap
!procedure $foo($arg)
  !while $arg!=0
    !$i=3
    **[#palegreen] arg = $arg
    !while $i!=0
      *** i = $i
      !$i = $i - 1
    !endwhile
    !$arg = $arg - 1
  !endwhile
!endprocedure

*:While
Loop;
$foo(2)
@endmindmap
  
```





25.4.3 While ループ (コンポーネント図/配置図での例)

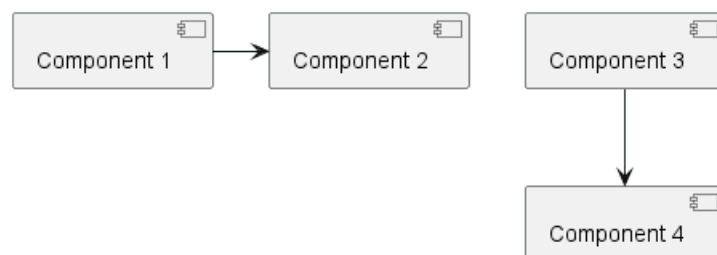
```

@startuml
!procedure $foo($arg)
  !while $arg!=0
    [Component $arg] as $arg
    !$arg = $arg - 1
  !endwhile
!endprocedure

$foo(4)

1->2
3-->4
@enduml

```



[Ref. QA-14088]

25.5 プロシージャ [!procedure, !endprocedure]

- プロシージャ名は \$ で開始する必要があります
- 引数名は \$ で開始する必要があります
- プロシージャから他のプロシージャを呼び出すことができます

例:

```

@startuml
!procedure $msg($source, $destination)
  $source --> $destination
!endprocedure

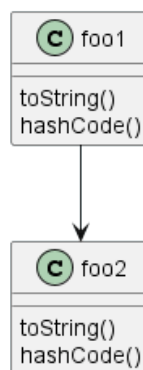
```



```
!procedure $init_class($name)
  class $name {
    $addCommonMethod()
  }
!endprocedure
```

```
!procedure $addCommonMethod()
  toString()
  hashCode()
!endprocedure
```

```
$init_class("foo1")
$init_class("foo2")
$msg("foo1", "foo2")
@enduml
```



プロシージャ内で定義された変数はローカル変数になります。ローカル変数はプロシージャの終了時に破棄されます。

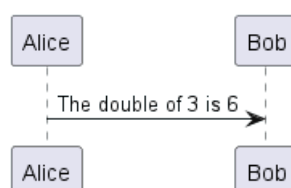
25.6 return 関数 [!function, !endfunction]

return 関数は文字列を出力しません。定義した関数は次の個所で呼び出すことができます:

- 変数定義、ダイアグラムのテキスト中で直接使用する
- 他の return 関数から呼び出す
- プロシージャから呼び出す
- 関数名は \$ で開始する必要があります
- 引数名は \$ で開始する必要があります

```
@startuml
!function $double($a)
!return $a + $a
!endfunction
```

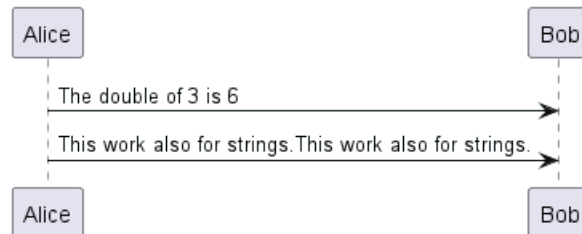
```
Alice -> Bob : The double of 3 is $double(3)
@enduml
```



簡単な関数の定義は、一行で記述することができます:

```
@startuml
!function $double($a) !return $a + $a

Alice -> Bob : The double of 3 is $double(3)
Alice -> Bob : $double("This work also for strings.")
@enduml
```

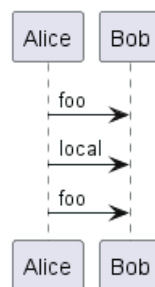


プロシージャと同様に、デフォルトで変数はローカル変数になります（関数の終了時に破棄されます）。しかし、関数からグローバル変数にアクセスすることもできます。`local` キーワードを使用すると、同じ名前のグローバル変数が存在したとしてもローカル変数として定義することができます。

```
@startuml
!function $dummy()
!local $ijk = "local"
!return "Alice -> Bob : " + $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```



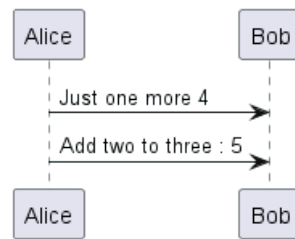
25.7 引数のデフォルト値

プロシージャと `return` 関数では、引数のデフォルト値を定義することができます。

```
@startuml
!function $inc($value, $step=1)
!return $value + $step
!endfunction

Alice -> Bob : Just one more $inc(3)
Alice -> Bob : Add two to three : $inc(3, 2)
@enduml
```





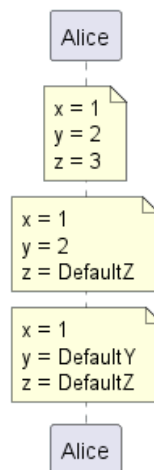
引数リストの最後尾から連続した複数個の引数に、デフォルト値を設定することができます。

```

@startuml
!procedure defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
  x = $x
  y = $y
  z = $z
end note
!endprocedure

defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml

```



25.8 クォーテーション不要プロシージャ/関数 [!unquoted]

デフォルトでは、プロシージャや関数の文字列型引数はクォーテーション記号で囲む必要があります。`unquoted` キーワードを使用すると、プロシージャや関数の引数をクォーテーション記号で囲まずに記述できるようになります。

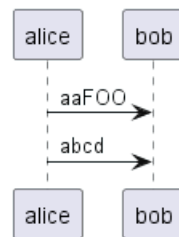
```

@startuml
!unquoted function id($text1, $text2="F00") !return $text1 + $text2

alice -> bob : id(aa)
alice -> bob : id(ab,cd)
@enduml

```





25.9 キーワード引数

Python と同様の、キーワード引数を使用することができます:

```
@startuml
```

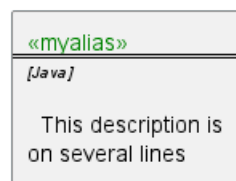
```
!unquoted procedure $element($alias, $description="", $label="", $technology="", $size=12, $colour="")
rectangle $alias as "
<color:$colour><<$alias>></color>
==$label==
//<size:$size>[$technology]</size>//
```

```
    $description"
```

```
!endprocedure
```

```
$element(myalias, "This description is %newline()on several lines", $size=10, $technology="Java")
```

```
@enduml
```



25.10 ファイルや URL のインクルード [!include, !include_many, !include_once]

!include を使用すると、ダイアグラムにファイルをインクルードすることができます。URL を使用してインターネット/イントラネット上のファイルをインクルードすることもできます。インターネット上の認証が必要なリソースにアクセスすることもできます。詳細は URL 認証機能を確認してください。

例えば、複数のダイアグラムに、まったく同じクラスが現れるような状況を考えてください。クラスの定義を何度も記述するのではなく、別のファイルに定義しておくことができます。

```
@startuml
```

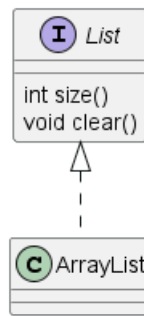
```
interface List
```

```
List : int size()
```

```
List : void clear()
```

```
List <|.. ArrayList
```

```
@enduml
```



File List.iuml

```

interface List
List : int size()
List : void clear()
  
```

List.iuml は、複数のダイアグラムにインクルードすることができます。このファイルに変更があった場合、インクルード先のすべてのダイアグラムに変更が及びます。

インクルードされるファイルに複数の @startuml/@enduml ブロックを記述し、インクルード時に!0 (0 はブロックの番号) を追加すると、特定のブロックをインクルードすることができます。!0 は最初のブロックを表します。

例えば、!include foo.txt!1 とすると、foo.txt に記述された 2 番目の @startuml/@enduml ブロックがインクルードされます。

@startuml(id=MY_OWN_ID) という記法で、@startuml/@enduml ブロックに ID を付与することもできます。このブロックをインクルードするには、インクルード時に!MY_OWN_ID を追加し、!include foo.txt!MY_OWN_ID のようにします。

デフォルトでは、一つのファイルは一度だけインクルードできます。一つのファイルを複数回インクルードしたい場合は、!include の代わりに!include_many を使用します。!include_once ディレクティブを使用すると、ファイルが複数回インクルードされた場合にエラーを発生させることができます。

25.11 部分的なインポート [!startsub, !endsub, !includesub]

!startsub NAME と!endsub を使用してテキストのセクションを作成し、他のファイルで!includesub によってインクルードすることができます。例:

file1.puml:

```

@startuml

A -> A : stuff1
!startsub BASIC
B -> B : stuff2
!endsub
C -> C : stuff3
!startsub BASIC
D -> D : stuff4
!endsub
@enduml
  
```

file1.puml は、次の記述と同じ結果になります:

```

@startuml

A -> A : stuff1
B -> B : stuff2
C -> C : stuff3
D -> D : stuff4
@enduml
  
```



一方で、これを利用して次のような file2.puml を作成ことができます:

file2.puml

```
@startuml
```

```
title this contains only B and D
!includesub file1.puml!BASIC
@enduml
```

このファイルは、次の記述と同じ結果になります:

```
@startuml
```

```
title this contains only B and D
B -> B : stuff2
D -> D : stuff4
@enduml
```

25.12 組み込み関数 [%]

デフォルトで定義された関数があります。関数名は%で始まります。

名称	説明
%chr	与えられた Unicode 値に対応する文字を返します。
%darken	与えられた色を暗くした色を返します。第二引数に暗くする割合を指定します。
%date	現在日付を返します。オプションで日付のフォーマットを指定できます。
%dec2hex	10 進数の数値 (Int) に対する 16 進数文字列 (String) を返します。
%dirpath	現在のディレクトリパスを取得します。
%feature	ある機能が、現在実行中の PlantUML のバージョンで利用できるかどうか確認します。
%false	常に false を返します。
%file_exists	ファイルがローカルファイルシステムに存在するかどうか確認します。
%filename	現在のファイル名を取得します。
%function_exists	関数が存在するかどうか確認します。
%get_variable_value	変数の値を取得します。
%getenv	環境変数の値を取得します。
%hex2dec	16 進数文字列 (String) に対する 10 進数の数値 (Int) を返します。
%hsl_color	HSL 形式の色 (%hsl_color(h, s, l)、%hsl_color(h, s, l, a)) を RGBa 形式に変換します。
%intval	String を Int に変換します。
%is_dark	色が暗い色かどうか判定します。
%is_light	色が明るい色かどうか判定します。
%lighten	与えられた色を明るくした色を返します。第二引数に明るくする割合を指定します。
%loadJSON	ローカルファイルまたは URL から、JSON データを読み込みます。
%lower	文字列を小文字に変換します。
%newline	改行文字列を返します。
%not	真偽値の論理否定を返します。
%reverse_color	RGB を使用して色を反転します。
%reverse_hsluv_color	HSLuv を使用して色を反転します。
%set_variable_value	グローバル変数に値を設定します。
%size	文字列または JSON 構造体のサイズを返します。
%string	文字列に変換します。
%strlen	文字列の長さを返します。
%strpos	文字列を検索し、出現位置を返します。
%substr	部分文字列を取り出します。2 つまたは 3 つの引数を取ります。
%true	常に true を返します。
%upper	文字列を大文字に変換します。
%variable_exists	変数が存在するかどうか確認します。
%version	実行中の PlantUML のバージョンを返します。

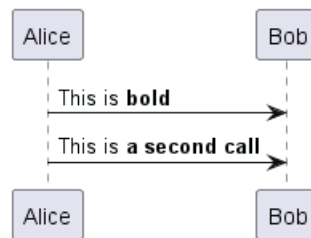


25.13 ログ出力 [!log]

!log を使用すると、ダイアグラム生成時にログを出力することができます。これは、ダイアグラム自体にはまったく影響を与えません。その代わりに、コマンドラインの出力ストリームにログが出力されます。これはデバッグのために有用です。

```
@startuml
!function bold($text)
!$result = "<b>"+ $text + "</b>"
!log Calling bold function with $text. The result is $result
!return $result
!endfunction

Alice -> Bob : This is bold("bold")
Alice -> Bob : This is bold("a second call")
@enduml
```

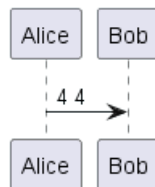


25.14 メモリーダンプ [!dump_memory]

!dump_memory を使用すると、ダイアグラム生成中のメモリの全内容をダンプ出力することができます。!dump_memory の後に任意の文字列を追加することもできます。これは、ダイアグラム自体にはまったく影響を与えませんが、デバッグのために有用です。

```
@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
```



25.15 アサーション [!assert]

ダイアグラムにアサーションを追加できます。

```
@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fails"
@enduml
```



Welcome to PlantUML!

You can start with a simple UML Diagram like:

```
Bob->Alice: Hello
```

Or

```
class Example
```

You will find more information about PlantUML syntax on <https://plantuml.com>

(Details by typing license keyword)



PlantUML 1.2023.11

[From string (line 3)]

@startuml

Alice -> Bob : Hello

!assert %strpos("abcdef", "cd")==3 : "This always fails"

Assertion error : This always fails

25.16 カスタムライブラリの作成 [!import, !include]

複数のインクルードファイルを、一つの.zip または.jar アーカイブにまとめることができます。作成した zip/jar ファイルは、!import ディレクティブを使用してダイアグラムにインポートすることができます。

ライブラリのインポート後、その zip/jar ファイルに含まれるファイルを!include することができます。

例:

```
@startuml
```

```
!import /path/to/customLibrary.zip
```

' この記述で、"customLibrary.zip"が検索パスに追加されます。

```
!include myFolder/myFile.iuml
```

' myFolder/myFile.iuml は、"customLibrary.zip"の中かローカルのファイルシステムに存在していることが期待されます。

...

25.17 検索パス

コマンドラインで、Java プロパティの plantuml.include.path を使って検索パスを指定できます。

例:

```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

この-D オプションは、-jar オプションより前に記述する必要があります。-jar オプションの後に記述した-D オプションは、PlantUML プリプロセッサの定数定義に使用されます。

25.18 引数の文字列結合 [##]

を使用して、マクロ引数に文字列を結合できます。

```
@startuml
```

```
!unquoted procedure COMP_TEXTGENCOMP(name)
```

```
[name] << Comp >>
```

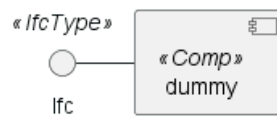
```
interface Ifc << IfcType >> AS name##Ifc
```

```
name##Ifc - [name]
```

```
!endprocedure
```



```
COMP_TEXTGENCOMP(dummy)
@enduml
```



25.19 動的呼び出し [%invoke_procedure(), %call_user_func()]

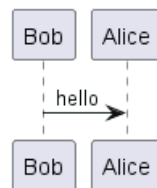
特別な%invoke_procedure() プロシージャを使用して、プロシージャを動的に呼び出すことができます。このプロシージャは、最初の引数に実際に呼び出すプロシージャ名を取ります。それに続く残りの引数は、呼び出されるプロシージャの引数として渡されます。

例:

```
@startuml
!procedure $go()
  Bob -> Alice : hello
!endprocedure

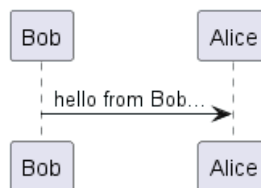
!$wrapper = "$go"

%invoke_procedure($wrapper)
@enduml
```



```
@startuml
!procedure $go($txt)
  Bob -> Alice : $txt
!endprocedure

%invoke_procedure("$go", "hello from Bob...")
@enduml
```

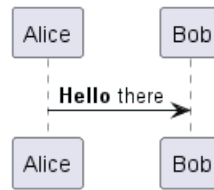


return 関数に対しては、%call_user_func() を使用します:

```
@startuml
!function bold($text)
!return "<b>"+ $text + "</b>"
!endfunction

Alice -> Bob : %call_user_func("bold", "Hello") there
@enduml
```





25.20 データ型に依存した加算演算子の評価 [+]

$\$a + \b の評価は $\$a$ と $\$b$ の型に依存して行われます。

```

@startuml
title
<#LightBlue>|= |= $a |= $b |= <U+0025>string($a + $b)|
<#LightGray>| type | str | str | str (concatenation) |
| example |= "a" |= "b" |= %string("a" + "b") |
<#LightGray>| type | str | int | str (concatenation) |
| ex. |= "a" |= 2 |= %string("a" + 2) |
<#LightGray>| type | str | int | str (concatenation) |
| ex. |= 1 |= "b" |= %string(1 + "b") |
<#LightGray>| type | bool | str | str (concatenation) |
| ex. |= <U+0025>true() |= "b" |= %string(%true() + "b") |
<#LightGray>| type | str | bool | str (concatenation) |
| ex. |= "a" |= <U+0025>false() |= %string("a" + %false()) |
<#LightGray>| type | int | int | int (addition of int) |
| ex. |= 1 |= 2 |= %string(1 + 2) |
<#LightGray>| type | bool | int | int (addition) |
| ex. |= <U+0025>true() |= 2 |= %string(%true() + 2) |
<#LightGray>| type | int | bool | int (addition) |
| ex. |= 1 |= <U+0025>false() |= %string(1 + %false()) |
<#LightGray>| type | int | int | int (addition) |
| ex. |= 1 |= <U+0025>intval("2") |= %string(1 + %intval("2")) |
end title
@enduml
  
```

	$\$a$	$\$b$	$\%string(\$a + \$b)$
type	str	str	str (concatenation)
example	"a"	"b"	ab
type	str	int	str (concatenation)
ex.	"a"	2	a2
type	str	int	str (concatenation)
ex.	1	"b"	1b
type	bool	str	str (concatenation)
ex.	%true()	"b"	1b
type	str	bool	str (concatenation)
ex.	"a"	%false()	a0
type	int	int	int (addition of int)
ex.	1	2	3
type	bool	int	int (addition)
ex.	%true()	2	3
type	int	bool	int (addition)
ex.	1	%false()	1
type	int	int	int (addition)
ex.	1	%intval("2")	3

25.21 JSON 前処理

JSON 前処理機能によって、前処理機能を拡張できます:

- JSON 変数定義



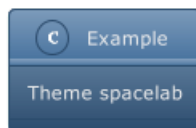
- JSON データへのアクセス
- JSON 配列に対するループ

(詳細は JSON 前処理のページを確認)

25.22 テーマのインクルード [!theme]

!theme ディレクティブを使用するとダイアグラムのデフォルトテーマを変更することができます。

```
@startuml
!theme spacelab
class Example {
    Theme spacelab
}
@enduml
```



詳細はテーマの説明を確認してください。

25.23 古いバージョンからの移行に関する注意事項

現行のプリプロセッサは、レガシーなプリプロセッサからアップデートしたものです。

レガシーな機能の内いくつかは、現行のプリプロセッサでもサポートされていますが、今後はそれらを使わないようにしてください (将来的に削除される可能性があります)。

- **!define** と **!definelong** は使用しないでください。代わりに **!function`**、**!procedure?code??** もしくは変数定義を使用します。
 - **!define** は **return!function** に置き換えてください。
 - **!definelong** は **!procedure** に置き換えてください。
- **!include** で複数のインクルードが可能になったので、**!include_many** を使う必要はありません。
- **!include** は URL を受け取れるようになったので **!includeurl** を使う必要はありません。
- **%date%** などのいくつかの機能は、組み込みの関数 (**%date()** など) に置き換えられました。
- レガシーな **!definelong** マクロを引数無しで呼ぶ場合、必ず括弧を使用する必要があります。 **my_own_definelong()** ではなく **my_own_definelong** のように括弧を省略してしまうと、新しいプリプロセッサでは認識されません。

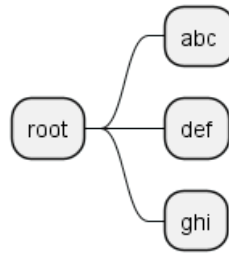
Please contact us if you have any issues.

25.24 %Splitstr builtin function

```
@startmindmap
!$list = %splitstr("abc~def~ghi", "~")

* root
!foreach $item in $list
  ** $item
!endfor
@endmindmap
```





[Ref. QA-15374]

26 Unicode

PlantUML 言語ではアクターやユースケースなどを定義するのに「文字」を使用します。

しかし「文字」は A～Z のアルファベットに限定されません。あらゆる言語のあらゆる文字を使用することができます。

26.1 例

```
@startuml
skinparam handwritten true
skinparam backgroundColor #EEEEBC

actor 使用者
participant "頭等艙" as A
participant "第二類" as B
participant "最後一堂課" as 別的東西
```

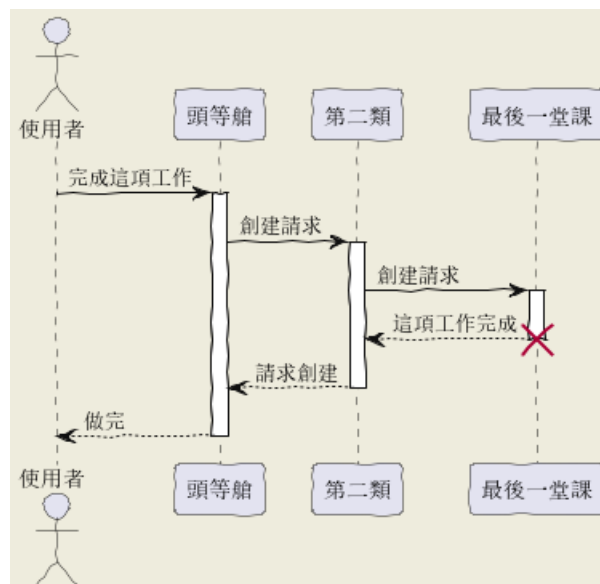
```
使用者 -> A: 完成這項工作
activate A
```

```
A -> B: 創建請求
activate B
```

```
B -> 別的東西: 創建請求
activate 別的東西
別的東西 --> B: 這項工作完成
destroy 別的東西
```

```
B --> A: 請求創建
deactivate B
```

```
A --> 使用者: 做完
deactivate A
@enduml
```



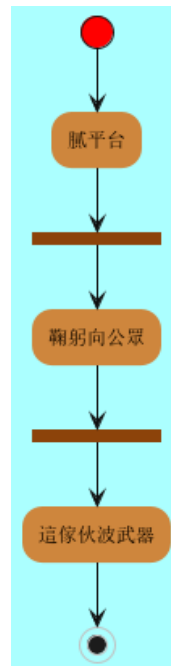
```
@startuml

(*) --> "膩平台"
--> === S1 ===
--> 鞠躬向公眾
```



```
--> === S2 ===
--> 這傢伙波武器
--> (*)
```

```
skinparam backgroundColor #AAFFFF
skinparam activityStartColor red
skinparam activityBarColor SaddleBrown
skinparam activityEndColor Silver
skinparam activityBackgroundColor Peru
skinparam activityBorderColor Peru
@enduml
```



```
@startuml
```

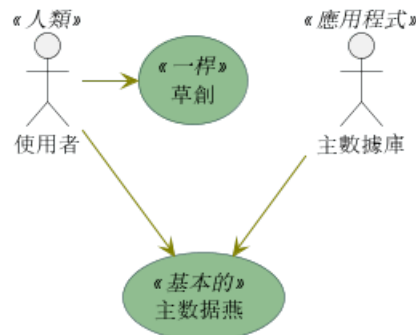
```
skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray
```

```
使用者 << 人類 >>
"主數據庫" as 數據庫 << 應用程式 >>
(草創) << 一桿 >>
"主数据燕" as (贏余) << 基本的 >>
```

```
使用者 -> (草創)
使用者 --> (贏余)
```

```
數據庫 --> (贏余)
@enduml
```

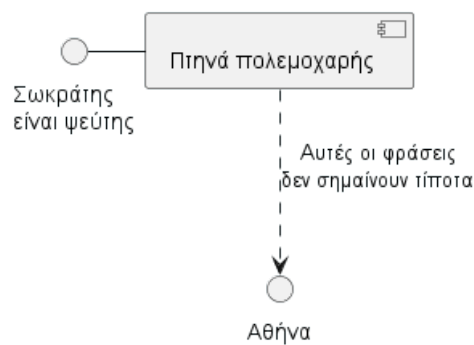




```

@startuml
() "Σ" as Σ
Σ - [Π]
[Π] ..> () A : A
@enduml

```



26.2 文字コード

UML を含むテキストを読み込むために使用するデフォルトの文字コード (charset) は、システムに依存しています。

通常、それで問題は起きないはずですが、他の文字コードを使用したい場合もあるでしょう。例えば、次のようなコマンドを使用します：

```
java -jar plantuml.jar -charset UTF-8 files.txt
```

Ant タスクを使用する場合：

```

<!-- Put images in c:/images directory -->
<target name="main">
<plantuml dir="./src" charset="UTF-8" />

```

インストールした Java 環境によりませんが、次の文字コードが使用可能なはずですが：ISO-8859-1、UTF-8、UTF-16BE、UTF-16LE、UTF-16。

26.3 Using Unicode Character on PlantUML

On PlantUML diagram, you can integrate:

- Special characters using `&#XXXX`; or `<U+XXXX>` form;
- Emoji using `<:XXXXX:>` or `<:NameOfEmoji:>` form.



27 PlantUML 標準ライブラリ

PlantUML の公式標準ライブラリ (`stdlib`) のガイドへようこそ。ここでは、PlantUML のすべての公式リリースに含まれ、より豊かなダイアグラム作成体験を促進する、この不可欠なリソースを掘り下げます。

27.0.1 標準ライブラリの概要

標準ライブラリはファイルやリソースのリポジトリで、PlantUML をより使いやすくするために常に更新されています。

27.0.2 コミュニティからの貢献

ライブラリのコンテンツの大部分は、サードパーティの貢献者によって惜しみなく提供されています。私たちは、ライブラリを充実させる上で極めて重要な役割を果たした、彼らの貴重な貢献に対して、心から感謝の意を表します。

私たちは、標準ライブラリが提供する豊富なリソースを掘り下げ、彼らのダイアグラム作成経験を向上させるだけでなく、この共同作業の努力に貢献し、その一端を担うことができるよう、ユーザーを奨励しています。

27.1 標準ライブラリの一覧

特別なダイアグラムを使用して、標準ライブラリのフォルダー一覧を得ることができます:

```
@startuml
stdlib
@enduml
```

archimate

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/Archimate-PlantUML>**aws**

Version 18.02.22

Delivered by <https://github.com/milo-minderbinder/AWS-PlantUML>**awslib**

Version 14.0.0

Delivered by <https://github.com/aws-labs/aws-icons-for-plantuml>**azure**

Version 2.2.0

Delivered by <https://github.com/plantuml-stdlib/Azure-PlantUML>**c4**

Version 2.8.0

Delivered by <https://github.com/plantuml-stdlib/C4-PlantUML>**classy**

Version 1.0.0

Delivered by <https://github.com/james-gadrow-kr/classy-plantuml>**classy-c4**

Version 1.0.0

Delivered by <https://github.com/james-gadrow-kr/classy-c4>**cloudinsight**

Version 1.0.0

Delivered by <https://github.com/plantuml-stdlib/cicon-plantuml-sprites>**cloudogu**

Version 1.0.2

Delivered by <https://github.com/cloudogu/plantuml-cloudogu-sprites>**domainstory**

Version 0.3

Delivered by <https://github.com/johthor/DomainStory-PlantUML>**elastic**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-Elastic-icons>**kubernetes**

Version 5.3.45

Delivered by <https://github.com/plantuml-stdlib/plantuml-kubernetes-sprites>**logos**

Version 1.1.0

Delivered by <https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites>**material**

Version 0.0.1

Delivered by <https://github.com/Templarian/MaterialDesign>**office**

Version 1.0.0

Delivered by <https://github.com/Roemer/plantuml-office>**osa**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons>**tupadr3**

Version 2.4.0

Delivered by <https://github.com/tupadr3/plantuml-icon-font-sprites>

コマンドラインから `java -jar plantuml.jar -stdlib` を実行することでも同じ一覧を得ることができます。

`java -jar plantuml.jar -extractstdlib` を実行すると、標準ライブラリのすべてのソースを取り出すことができます。すべてのファイルは `stdlib` フォルダに出力されます。



公式の PlantUML リリースのビルドに使用されるソースは、<https://github.com/plantuml/plantuml-stdlib> にホストされています。ライブラリのアップデートや関連するライブラリを追加したい場合は、プルリクエストを作成することができます。

27.2 ArchiMate (archimate)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/archimate
src	https://github.com/ebbypeter/ArchiMate-PlantUML
orig	https://en.wikipedia.org/wiki/ArchiMate

このリポジトリには ArchiMate の PlantUML マクロ、および、ArchiMate の図を簡単に一貫性をもって作成するためのその他のインクルードが含まれています。

```
@startuml
!include <archimate/ArchiMate>

title Archimate Sample - Internet Browser

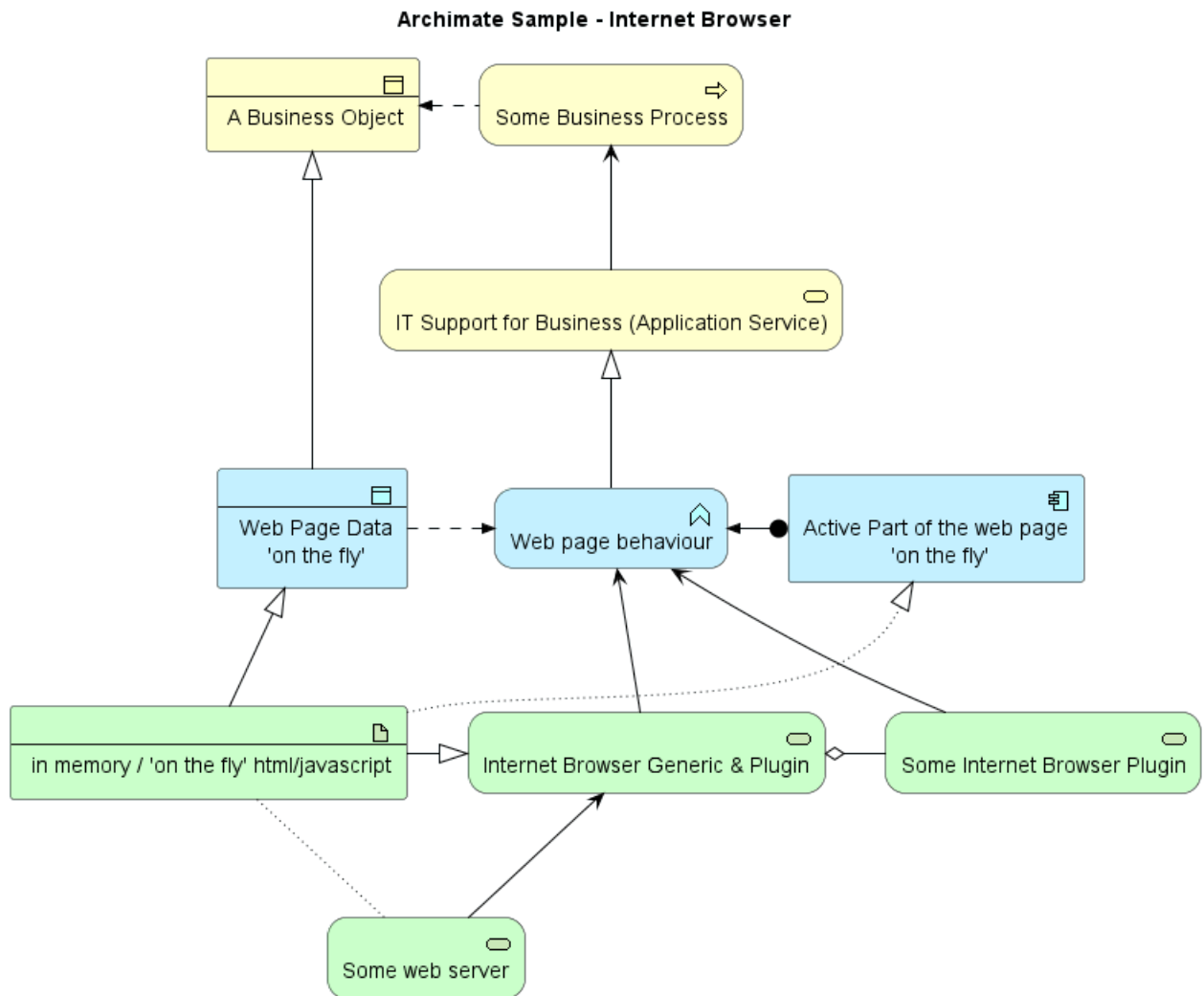
' Elements
Business_Object(businessObject, "A Business Object")
Business_Process(someBusinessProcess, "Some Business Process")
Business_Service(itSupportService, "IT Support for Business (Application Service)")

Application_DataObject(dataObject, "Web Page Data \n 'on the fly'")
Application_Function(webpageBehaviour, "Web page behaviour")
Application_Component(ActivePartWebPage, "Active Part of the web page \n 'on the fly'")

Technology_Artifact(inMemoryItem, "in memory / 'on the fly' html/javascript")
Technology_Service(internetBrowser, "Internet Browser Generic & Plugin")
Technology_Service(internetBrowserPlugin, "Some Internet Browser Plugin")
Technology_Service(webServer, "Some web server")

'Relationships
Rel_Flow_Left(someBusinessProcess, businessObject, "")
Rel_Serving_Up(itSupportService, someBusinessProcess, "")
Rel_Specialization_Up(webpageBehaviour, itSupportService, "")
Rel_Flow_Right(dataObject, webpageBehaviour, "")
Rel_Specialization_Up(dataObject, businessObject, "")
Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")
Rel_Specialization_Up(inMemoryItem, dataObject, "")
Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")
Rel_Specialization_Right(inMemoryItem, internetBrowser, "")
Rel_Serving_Up(internetBrowser, webpageBehaviour, "")
Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")
Rel_Access_Up(webServer, inMemoryItem, "")
Rel_Serving_Up(webServer, internetBrowser, "")
@enduml
```





27.2.1 List possible sprites

You can list all possible sprites for Archimate using the following diagram:

```
@startuml
listsprite
@enduml
```

List Current Sprites

Credit to

<http://www.archimatetool.com>

archimate :

access
 activity
 actor
 aggregation
 application-collaboration
 application-component
 application-data-object
 application-event
 application-function
 application-interaction
 application-interface
 application-process
 application-service
 assessment-filled
 assessment
 assignment
 association-unidirect
 association
 business-activity
 business-actor
 business-collaboration
 business-contract
 business-event
 business-function
 business-interaction
 business-interface
 business-location
 business-meaning

business-object
 business-process
 business-product
 business-representation
 business-role
 business-service
 business-value
 collaboration
 communication-path
 component
 composition
 constraint-filled
 constraint
 contract
 deliverable-filled
 deliverable
 device
 driver-filled
 driver
 event
 flow
 function
 gap-filled
 gap
 goal-filled
 goal
 implementation-deliverable
 implementation-event
 implementation-gap
 implementation-plateau
 implementation-workpackage
 influence
 interaction
 interface-required

interface-symmetric
 interface
 junction-and
 junction-or
 junction
 location
 meaning
 motivation-assessment
 motivation-constraint
 motivation-driver
 motivation-goal
 motivation-meaning
 motivation-outcome
 motivation-principle
 motivation-requirement
 motivation-stakeholder
 motivation-value
 network
 node
 object
 physical-distribution-network
 physical-equipment
 physical-facility
 physical-material
 plateau
 principle-filled
 principle
 process
 product
 realisation
 representation
 requirement-filled
 requirement
 role

service
 serving
 specialisation
 specialization
 stakeholder-filled
 strategy-capability
 strategy-course-of-action
 strategy-resource
 strategy-value-stream
 system-software
 technology-artifact
 technology-collaboration
 technology-communication-network
 technology-communication-path
 technology-device
 technology-event
 technology-function
 technology-infra-interface
 technology-infra-service
 technology-interaction
 technology-interface
 technology-network
 technology-node
 technology-path
 technology-process
 technology-service
 technology-system-software
 triggering
 used-by
 value
 workpackage-filled

27.3 Amazon Labs AWS ライブラリ (awslib)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/awslib
src	https://github.com/aws-labs/aws-icons-for-plantuml
orig	https://aws.amazon.com/jp/architecture/icons/

Amazon Labs AWS ライブラリは、PlantUML のスプライト、マクロ、その他の Amazon Web Services(AWS) 向けサービスとリソースを提供します。

AWS コンポーネントを含む PlantUML ダイアグラムを作成するために使用します。すべての要素は公式の AWS Architecture Icons から生成されていて、PlantUML や C4 model と組み合わせることで、設計、デブロイ、トポロジーをコードとして伝えるための素晴らしい手段となります。

```

@startuml
!include <awslib/AWSCommon>
!include <awslib/InternetOfThings/IoTRule>
!include <awslib/Analytics/KinesisDataStreams>
!include <awslib/ApplicationIntegration/SimpleQueueService>

```

left to right direction

```
agent "Published Event" as event #fff
```

```

IoTRule(iotRule, "Action Error Rule", "error if Kinesis fails")
KinesisDataStreams(eventStream, "IoT Events", "2 shards")
SimpleQueueService(errorQueue, "Rule Error Queue", "failed Rule actions")

```

```

event --> iotRule : JSON message
iotRule --> eventStream : messages
iotRule --> errorQueue : Failed action message
@enduml

```



27.4 アジュールライブラリ [azure]

タイプ	リンク
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/azure
src	https://github.com/RicardoNiepel/Azure-PlantUML/
orig	Microsoft Azure

Azure ライブラリは Microsoft Azure のアイコンで構成されています。

スプライトを含むファイルをインクルードして使用します (例: `!include <azure/Analytics/AzureEventHub>`). インポートすると、通常と同じようにスプライトを使用できます。 `<$sprite_name>`

AzureCommon.puml ファイルをインクルードすることもできます: `!include <azure/AzureCommon>`
 AzureCommon.puml をインポートすると、NAME_OF_SPRITE(parameters...) マクロを使用することができます。

使用例.

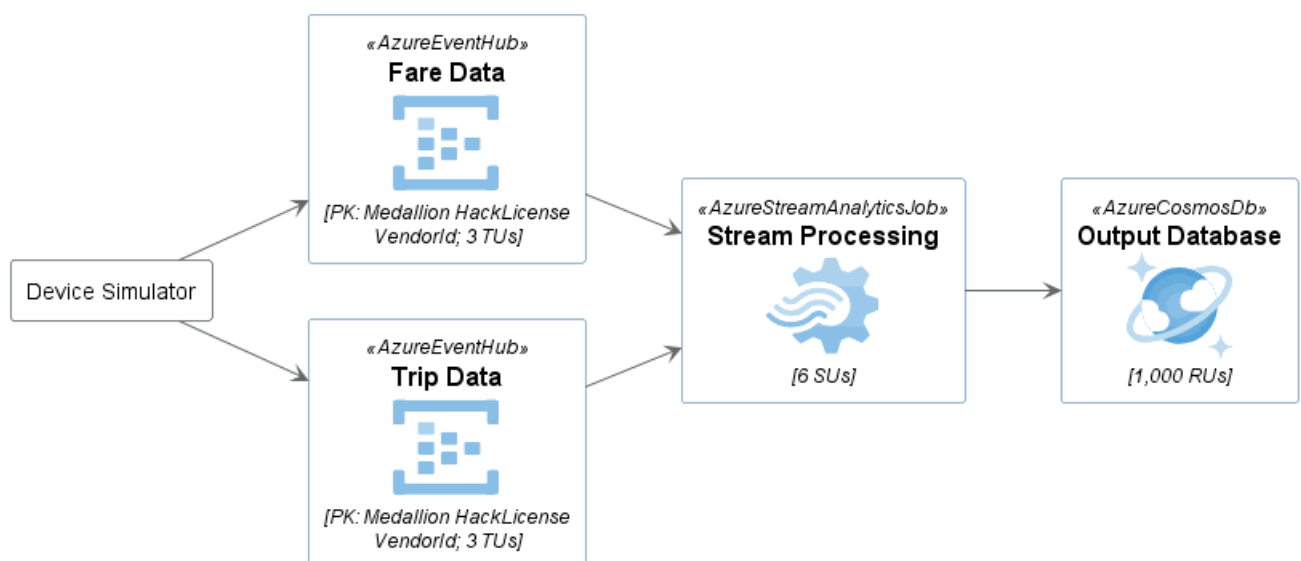
```
@startuml
!include <azure/AzureCommon>
!include <azure/Analytics/AzureEventHub>
!include <azure/Analytics/AzureStreamAnalyticsJob>
!include <azure/Databases/AzureCosmosDb>
```

```
left to right direction
```

```
agent "Device Simulator" as devices #fff
```

```
AzureEventHub(fareDataEventHub, "Fare Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureEventHub(tripDataEventHub, "Trip Data", "PK: Medallion HackLicense VendorId; 3 TUs")
AzureStreamAnalyticsJob(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")
```

```
devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



27.5 C4 ライブラリ (C4)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/C4
src	https://github.com/plantuml-stdlib/C4-PlantUML
orig	https://en.wikipedia.org/wiki/C4_model https://c4model.com

```

@startuml
!include <C4/C4_Container>

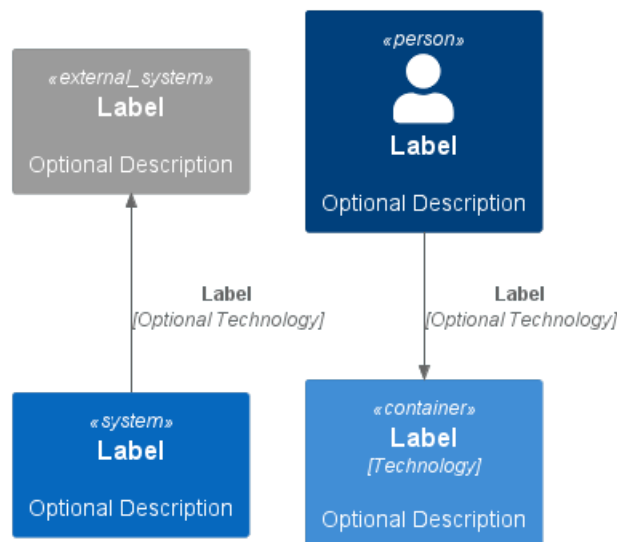
Person(personAlias, "Label", "Optional Description")
Container(containerAlias, "Label", "Technology", "Optional Description")
System(systemAlias, "Label", "Optional Description")

System_Ext(extSystemAlias, "Label", "Optional Description")

Rel(personAlias, containerAlias, "Label", "Optional Technology")

Rel_U(systemAlias, extSystemAlias, "Label", "Optional Technology")
@enduml

```



27.6 Cloud Insight (cloudinsight)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudinsight
src	https://github.com/rabelenda/cicon-plantuml-sprites
orig	Cloudinsight icons

このリポジトリには、Cloudinsight のアイコンから生成した PlantUML スプライトがあります。PlantUML の図の中で簡単に使用することができ、一般的なテクノロジーの美しいビジュアル表現を行うことができます。

```

@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

title Cloudinsight sprites example

skinparam monochrome true

```



```

rectangle "<$tomcat>\nwebapp" as webapp
queue "<$kafka>" as kafka
rectangle "<$java>\ndaemon" as daemon
database "<$cassandra>" as cassandra

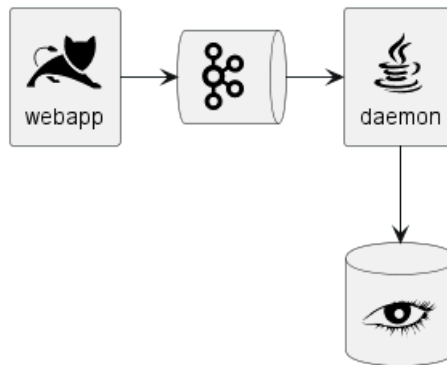
```

```

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml

```

Cloudinsight sprites example



27.7 Clodogu (cloudogu)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/cloudogu
src	https://github.com/cloudogu/plantuml-cloudogu-sprites
orig	https://cloudogu.com

Cloudogu ライブラリは、PlantUML のスプライト、マクロ、その他の Cloudogu 向けサービスとリソースを提供します。

```

@startuml
!include <cloudogu/common>
!include <cloudogu/dogus/jenkins>
!include <cloudogu/dogus/cloudogu>
!include <cloudogu/dogus/scm>
!include <cloudogu/dogus/smeagol>
!include <cloudogu/dogus/nexus>
!include <cloudogu/tools/k8s>

node "Cloudogu Ecosystem" <<$cloudogu>> {
DOGU_JENKINS(jenkins, Jenkins) #ffffff
DOGU_SCM(scm, SCM-Manager) #ffffff
DOGU_SMEAGOL(smeagol, Smeagol) #ffffff
DOGU_NEXUS(nexus, Nexus) #ffffff
}

```

```

TOOL_K8S(k8s, Kubernetes) #ffffff

```

```

actor developer

```

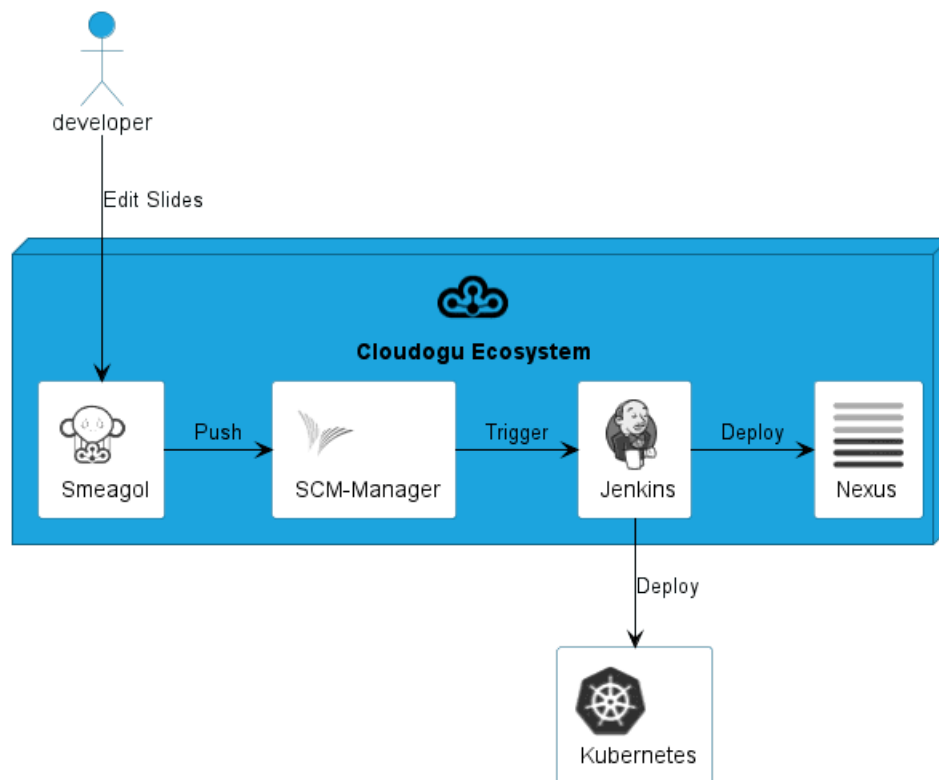
```

developer --> smeagol : "Edit Slides"
smeagol -> scm : Push
scm -> jenkins : Trigger
jenkins -> nexus : Deploy

```



```
jenkins --> k8s : Deploy
@enduml
```



すべての **cloudogu** スプライト

利用可能なすべての cloudogu スプライトは plantuml-cloudogu-sprites を確認してください。

27.8 Elastic ライブラリ (elastic)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/elastic
src	https://github.com/Crashedmind/PlantUML-Elastic-icons
orig	Elastic

Elastic ライブラリには、Elastic のアイコンが含まれています。これは、AWS および Azure ライブラリと類似のものです（同じツールを使用して作られました）。

スプライトの含まれるファイルをインポートして使います。例: `!include elastic/elastic_search/elastic_search` インポートされると、通常のスプライトと同様、`<$sprite_name>` のように使用することができます。

`!include <elastic/common>` のようにして `common.puml` をインクルードすると、そこに定義されたヘルパーマクロを使用することができます。`common.puml` をインポートすると、`NAME_OF_SPRITE(parameters...)` マクロを使用することができます。

使用例:

```
@startuml
!include <elastic/common>
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/logstash/logstash>
!include <elastic/kibana/kibana>
```

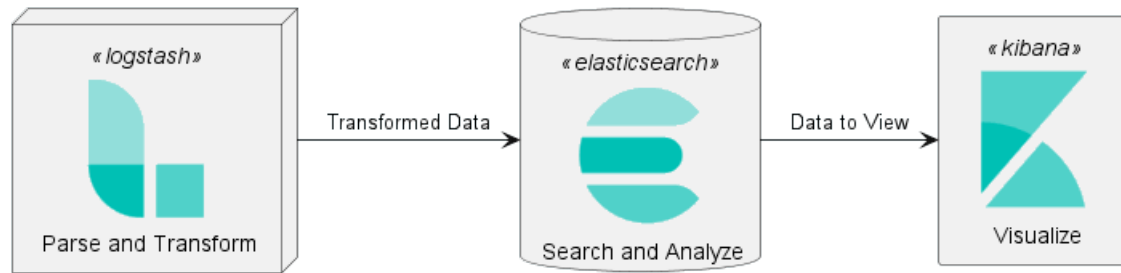
```
ELASTICSEARCH(ElasticSearch, "Search and Analyze", database)
LOGSTASH(Logstash, "Parse and Transform", node)
KIBANA(Kibana, "Visualize", agent)
```



```

Logstash -right-> Elasticsearch: Transformed Data
ElasticSearch -right-> Kibana: Data to View
@enduml

```



すべての **Elastic** スプライト

```

@startuml
'Adapted from https://github.com/Crashedmind/PlantUML-Elastic-icons/blob/master/All.puml

'Elastic stuff here
'=====

!include <elastic/common>
!include <elastic/apm/apm>
!include <elastic/app_search/app_search>
!include <elastic/beats/beats>
!include <elastic/cloud/cloud>
!include <elastic/cloud_in_kubernetes/cloud_in_kubernetes>
!include <elastic/code_search/code_search>
!include <elastic/ece/ece>
!include <elastic/eck/eck>
' Beware of the difference between Crashedmind and plantuml-stdlib version: with '_' usage!
!include <elastic/elasticsearch/elasticsearch>
!include <elastic/endpoint/endpoint>
!include <elastic/enterprise_search/enterprise_search>
!include <elastic/kibana/kibana>
!include <elastic/logging/logging>
!include <elastic/logstash/logstash>
!include <elastic/maps/maps>
!include <elastic/metrics/metrics>
!include <elastic/siem/siem>
!include <elastic/site_search/site_search>
!include <elastic/stack/stack>
!include <elastic/uptime/uptime>

skinparam agentBackgroundColor White

APM(apm)
APP_SEARCH(app_search)
BEATS(beats)
CLOUD(cloud)
CLOUD_IN_KUBERNETES(cloud_in_kubernetes)
CODE_SEARCH(code_search)
ECE(ece)
ECK(eck)
ELASTICSEARCH(elastic_search)
ENDPOINT(endpoint)
ENTERPRISE_SEARCH(enterprise_search)
KIBANA(kibana)

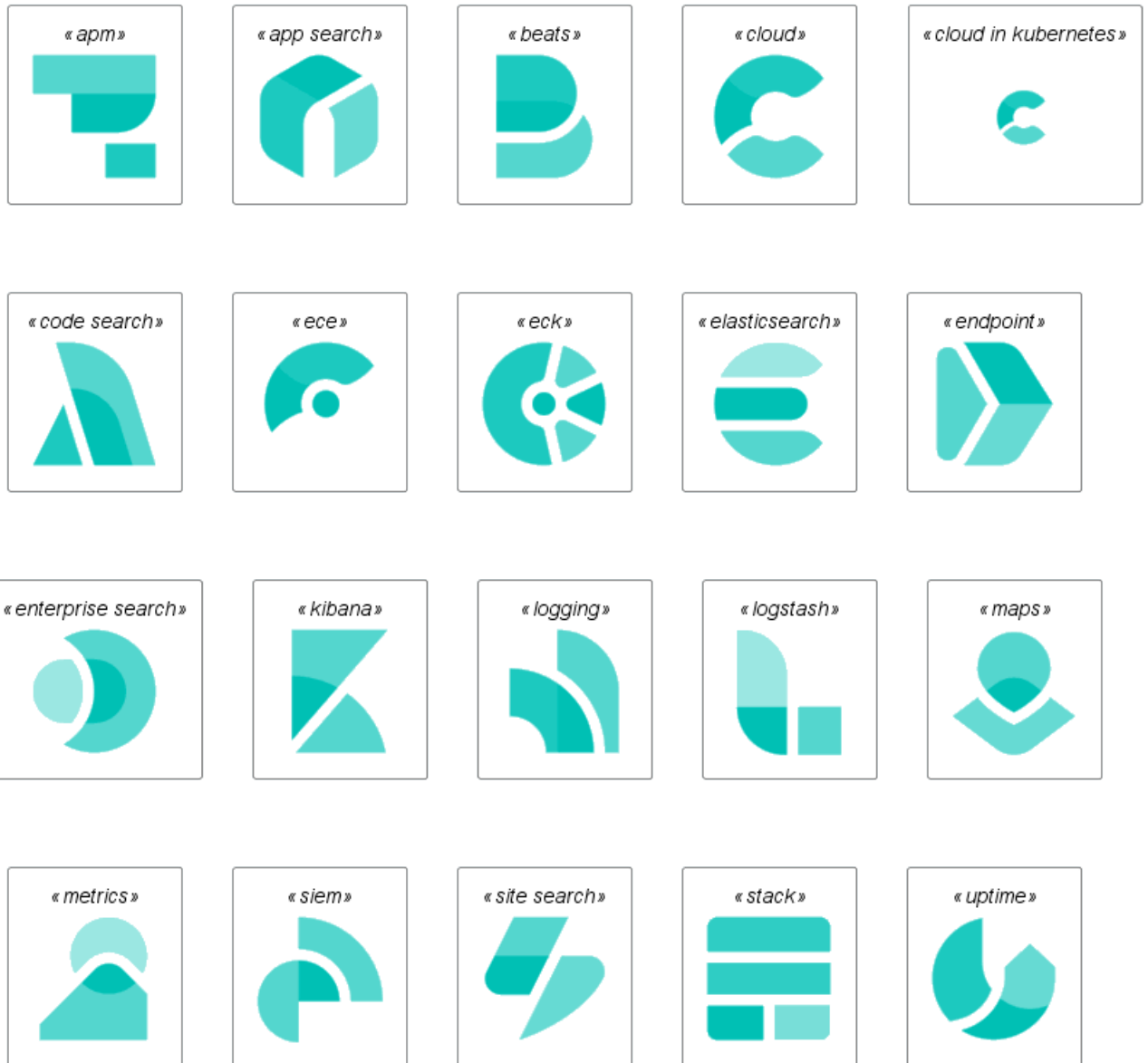
```



```

LOGGING(logging)
LOGSTASH(logstash)
MAPS(maps)
METRICS(metrics)
SIEM(siem)
SITE_SEARCH(site_search)
STACK(stack)
UPTIME(uptime)
@enduml

```



27.9 Google Material Icons (material)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/material
src	https://github.com/Templarian/MaterialDesign
orig	Material Design Icons

このライブラリには、Google やその他の作成者によって作られた、フリーのマテリアルスタイルのアイコンが含まれています。



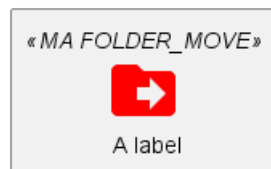
スプライトの含まれるファイルをインポートして使います。例: `!include <material/ma_folder_move>` インポートされると、通常のスプライトと同様、`<$ma_sprite_name>` のように使用することができます。このライブラリでは、他のライブラリの同じ名前のスプライトとの衝突を避けるために、スプライト名にプレフィックス `ma_` を付ける必要があることに注意してください。

`!include <material/common>` のようにして `common.puml` をインクルードすると、そこに定義されたヘルパーマクロを使用することができます。`common.puml` をインポートすると、`MA_NAME_OF_SPRITE(parameters...)` マクロを使用することができます。こちらもプレフィックス `MA_` を付ける必要があることに注意してください。

使用例:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>
```

```
MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label")
@enduml
```



注意：

例えば次のように、クラスとともに `rectangle` を加えようとした場合のように、スプライトマクロをほかの要素と一緒に使う場合にシンタックスエラーが発生することがあります。そのような場合には、マクロの後ろに `{と}` を加えて空の `rectangle` を作るようにしてください。

使用例:

```
@startuml
!include <material/common>
' To import the sprite file you DON'T need to place a prefix!
!include <material/folder_move>
```

```
MA_FOLDER_MOVE(Red, 1, dir, rectangle, "A label") {
}
```

```
class foo {
    bar
}
@enduml
```



27.10 Kubernetes (kubernetes)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/kubernetes
src	https://github.com/michiel/plantuml-kubernetes-sprites
orig	Kubernetes

```
@startuml
```



```

!include <kubernetes/k8s-sprites-unlabeled-25pct>
package "Infrastructure" {
    component "<$master>\nmaster" as master
    component "<$etcd>\netcd" as etcd
    component "<$node>\nnode" as node
}
@enduml

```



27.11 Logos (logos)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/logos
src	https://github.com/plantuml-stdlib/gilbarbara-plantuml-sprites
orig	Gil Barbara's logos

このリポジトリには Gil Barbara's logos から生成された PlantUML のスプライトが含まれています。素晴らしい視覚資料を、PlantUML の図の中で簡単に使用することができます。

```

@startuml
!include <logos/flask>
!include <logos/kafka>
!include <logos/kotlin>
!include <logos/cassandra>

title Gil Barbara's logos example

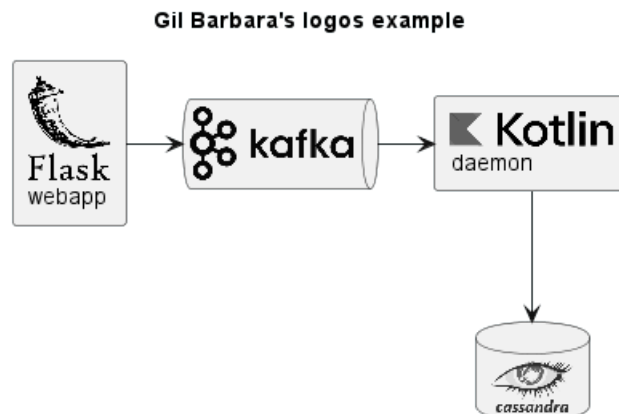
skinparam monochrome true

rectangle "<$flask>\nwebapp" as webapp
queue "<$kafka>" as kafka
rectangle "<$kotlin>\ndaemon" as daemon
database "<$cassandra>" as cassandra

webapp -> kafka
kafka -> daemon
daemon --> cassandra
@enduml

```





```

@startuml
scale 0.7
!include <logos/apple-pay>
!include <logos/dinersclub>
!include <logos/discover>
!include <logos/google-pay>
!include <logos/jcb>
!include <logos/maestro>
!include <logos/mastercard>
!include <logos/paypal>
!include <logos/unionpay>
!include <logos/visaelectron>
!include <logos/visa>
' ...

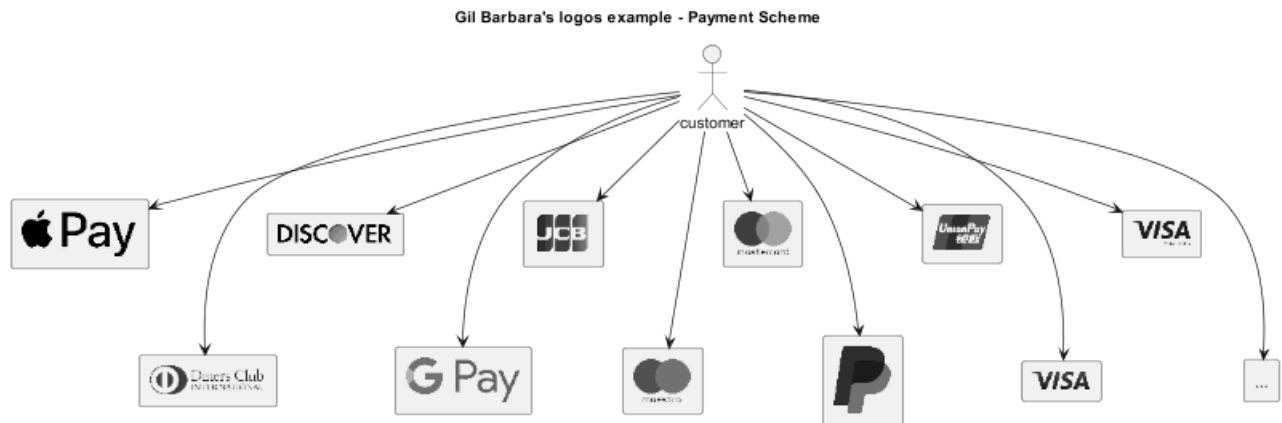
title Gil Barbara's logos example - **Payment Scheme**

actor customer
rectangle "<$apple-pay>" as ap
rectangle "<$dinersclub>" as dc
rectangle "<$discover>" as d
rectangle "<$google-pay>" as gp
rectangle "<$jcb>" as j
rectangle "<$maestro>" as ma
rectangle "<$mastercard>" as m
rectangle "<$paypal>" as p
rectangle "<$unionpay>" as up
rectangle "<$visa>" as v
rectangle "<$visaelectron>" as ve
rectangle "... " as etc

customer --> ap
customer ---> dc
customer --> d
customer ---> gp
customer --> j
customer ---> ma
customer --> m
customer ---> p
customer --> up
customer ---> v
customer --> ve
customer ---> etc
  
```



```
@enduml
```



27.12 Office (office)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/office
src	https://github.com/Roemer/plantuml-office
orig	

スプライト (*.puml) と色付きの PNG アイコンが利用可能です。名前に色名が含まれていたとしてもスプライトはすべてモノクロです (ファイルが自動生成されているためです)。マクロ (以下の例を参照) を使ってスプライトに色を付けるか、フルカラーの PNG を使用することができます。スプライト、PNG、マクロの使い方は以下の例を参照してください。

使用例:

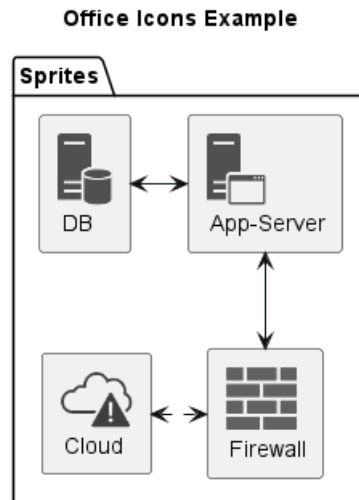
```
@startuml
!include <tupadr3/common>

!include <office/Servers/database_server>
!include <office/Servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

title Office Icons Example

package "Sprites" {
    OFF_DATABASE_SERVER(db,DB)
    OFF_APPLICATION_SERVER(app,App-Server)
    OFF_FIREWALL_ORANGE(fw,Firewall)
    OFF_CLOUD_DISASTER_RED(cloud,Cloud)
    db <-> app
    app <--> fw
    fw <.left.> cloud
}
@enduml
```





```

@startuml
!include <tupadr3/common>

!include <office/servers/database_server>
!include <office/servers/application_server>
!include <office/Concepts/firewall_orange>
!include <office/Clouds/cloud_disaster_red>

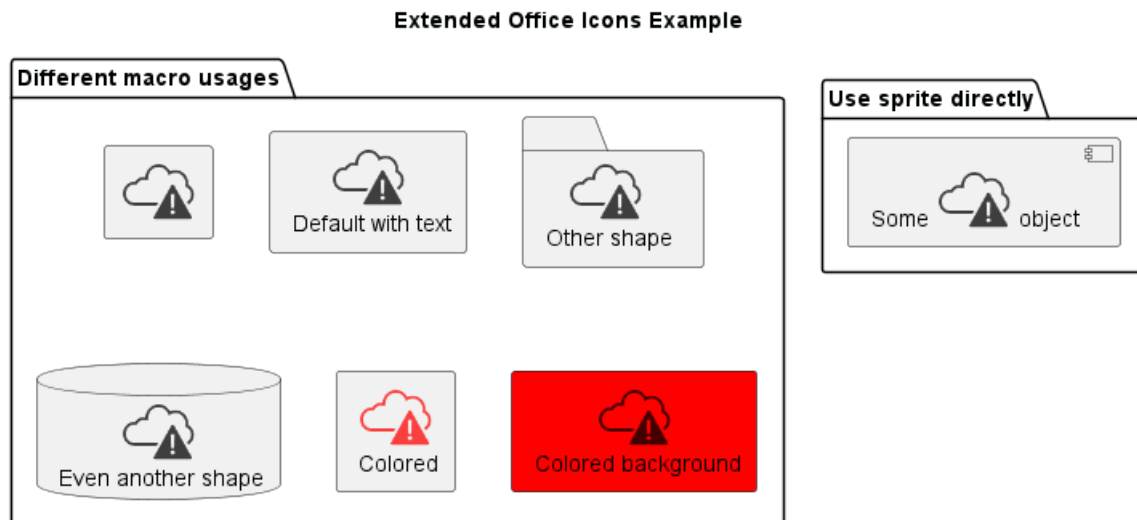
' Used to center the label under the images
skinparam defaultTextAlignment center

title Extended Office Icons Example

package "Use sprite directly" {
    [Some <$cloud_disaster_red> object]
}

package "Different macro usages" {
    OFF_CLOUD_DISASTER_RED(cloud1)
    OFF_CLOUD_DISASTER_RED(cloud2,Default with text)
    OFF_CLOUD_DISASTER_RED(cloud3,Other shape,Folder)
    OFF_CLOUD_DISASTER_RED(cloud4,Even another shape,Database)
    OFF_CLOUD_DISASTER_RED(cloud5,Colored,Rectangle, red)
    OFF_CLOUD_DISASTER_RED(cloud6,Colored background) #red
}
@enduml

```



27.13 Open Security Architecture (OSA) [osa]

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/osa
src	https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all
orig	https://www.opensecurityarchitecture.org

@startuml

'Adapted from <https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons/blob/master/all>

scale .5

!include <osa/arrow/green/left/left>

!include <osa/arrow/yellow/right/right>

!include <osa/awareness/awareness>

!include <osa/contract/contract>

!include <osa/database/database>

!include <osa/desktop/desktop>

!include <osa/desktop/imac/imac>

!include <osa/device_music/device_music>

!include <osa/device_scanner/device_scanner>

!include <osa/device_usb/device_usb>

!include <osa/device_wireless_router/device_wireless_router>

!include <osa/disposal/disposal>

!include <osa/drive_optical/drive_optical>

!include <osa/firewall/firewall>

!include <osa/hub/hub>

!include <osa/ics/drive/drive>

!include <osa/ics/plc/plc>

!include <osa/ics/thermometer/thermometer>

!include <osa/id/card/card>

!include <osa/laptop/laptop>

!include <osa/lifecycle/lifecycle>

!include <osa/lightning/lightning>

!include <osa/media_flash/media_flash>

!include <osa/media_optical/media_optical>

!include <osa/media_tape/media_tape>

!include <osa/mobile/pda/pda>

!include <osa/padlock/padlock>

!include <osa/printer/printer>

!include <osa/site_branch/site_branch>

!include <osa/site_factory/site_factory>

!include <osa/vpn/vpn>

```
!include <osa/wireless/network/network>

rectangle "OSA" {
rectangle "Left:\n <$left>"
rectangle "Right:\n <$right>"
rectangle "Awareness:\n <$awareness>"
rectangle "Contract:\n <$contract>"
rectangle "Database:\n <$database>"
rectangle "Desktop:\n <$desktop>"
rectangle "Imac:\n <$imac>"
rectangle "Device_music:\n <$device_music>"
rectangle "Device_scanner:\n <$device_scanner>"
rectangle "Device_usb:\n <$device_usb>"
rectangle "Device_wireless_router:\n <$device_wireless_router>"
rectangle "Disposal:\n <$disposal>"
rectangle "Drive_optical:\n <$drive_optical>"
rectangle "Firewall:\n <$firewall>"
rectangle "Hub:\n <$hub>"
rectangle "Drive:\n <$drive>"
rectangle "Plc:\n <$plc>"
rectangle "Thermometer:\n <$thermometer>"
rectangle "Card:\n <$card>"
rectangle "Laptop:\n <$laptop>"
rectangle "Lifecycle:\n <$lifecycle>"
rectangle "Lightning:\n <$lightning>"
rectangle "Media_flash:\n <$media_flash>"
rectangle "Media_optical:\n <$media_optical>"
rectangle "Media_tape:\n <$media_tape>"
rectangle "Pda:\n <$pda>"
rectangle "Padlock:\n <$padlock>"
rectangle "Printer:\n <$printer>"
rectangle "Site_branch:\n <$site_branch>"
rectangle "Site_factory:\n <$site_factory>"
rectangle "Vpn:\n <$vpn>"
rectangle "Network:\n <$network>"
}
@enduml
```



```

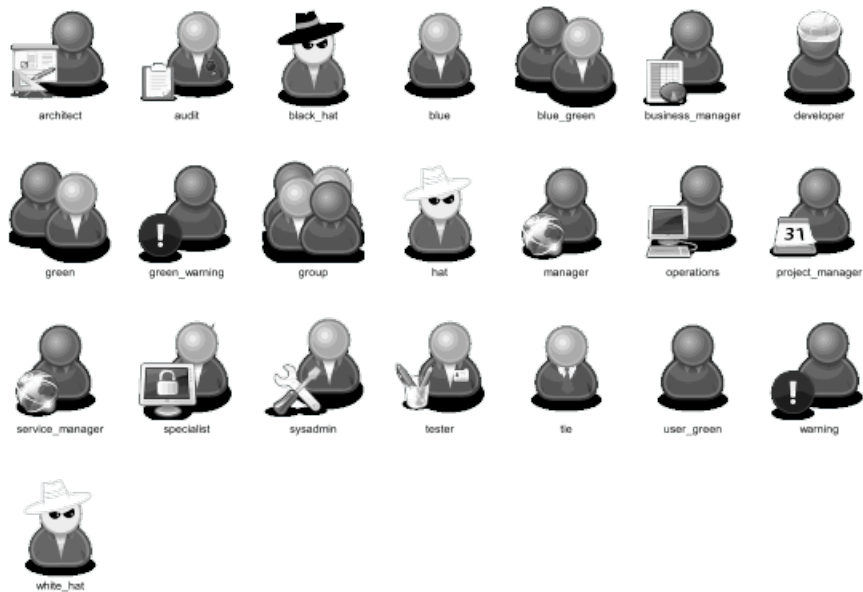
@startuml
scale .5
!include <osa/user/audit/audit>
'beware of 'hat-sprite'
!include <osa/user/black/hat/hat-sprite>
!include <osa/user/blue/blue>
!include <osa/user/blue/security/specialist/specialist>
!include <osa/user/blue/sysadmin/sysadmin>
!include <osa/user/blue/tester/tester>
!include <osa/user/blue/tie/tie>
!include <osa/user/green/architect/architect>
!include <osa/user/green/business/manager/manager>
!include <osa/user/green/developer/developer>
!include <osa/user/green/green>
!include <osa/user/green/operations/operations>
!include <osa/user/green/project/manager/manager>
!include <osa/user/green/service/manager/manager>
!include <osa/user/green/warning/warning>
!include <osa/user/large/group/group>
!include <osa/users/blue/green/green>
!include <osa/user/white/hat/hat>

```

```
listsprites
```



@enduml



27.14 Tupadr3 ライブラリ (tupadr3)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/tupadr3
src	https://github.com/tupadr3/plantuml-icon-font-sprites
orig	https://github.com/tupadr3/plantuml-icon-font-sprites#icon-sets

このライブラリには Devicons や Font Awesome を含む、いくつかのアイコンライブラリが含まれています。

スプライトの含まれるファイルをインポートして使います。例: `!include <font-awesome/align_center>` インポートされると、通常のスプライトと同様、`<$sprite_name>` のように使用することができます。

`!include <font-awesome/common>` のようにして `common.puml` をインクルードすると、そこに定義されたヘルパーマクロを使用することができます。`common.puml` をインポートすると、`NAME_OF_SPRITE(parameters...)` マクロを使用することができます。

使用例:

```
@startuml
!include <tupadr3/common>
!include <tupadr3/font-awesome/server>
!include <tupadr3/font-awesome/database>

title Styling example

FA_SERVER(web1,web1) #Green
FA_SERVER(web2,web2) #Yellow
FA_SERVER(web3,web3) #Blue
FA_SERVER(web4,web4) #YellowGreen

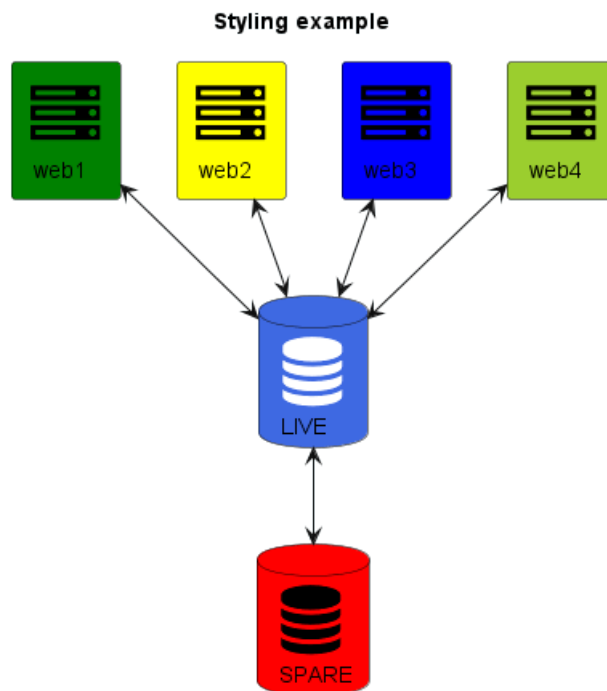
FA_DATABASE(db1,LIVE,database,white) #RoyalBlue
FA_DATABASE(db2,SPARE,database) #Red

db1 <--> db2

web1 <--> db1
web2 <--> db1
web3 <--> db1
```

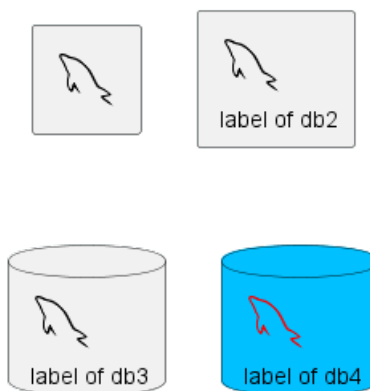


```
web4 <--> db1
@enduml
```



```
@startuml
!include <tupadr3/common>
!include <tupadr3/devicons/mysql>

DEV_MYSQL(db1)
DEV_MYSQL(db2,label of db2)
DEV_MYSQL(db3,label of db3,database)
DEV_MYSQL(db4,label of db4,database,red) #DeepSkyBlue
@enduml
```



27.15 AWS ライブラリ (aws)

Type	Link
stdlib	https://github.com/plantuml/plantuml-stdlib/tree/master/aws
src	https://github.com/milo-minderbinder/AWS-PlantUML
orig	https://aws.amazon.com/en/architecture/icons/

警告: このライブラリは非推奨となる予定です。

代わりに `<awslib>` の使用をお勧めします (上記参照)



hr

AWS ライブラリには Amazon AWS のアイコンが含まれています。それぞれ 2 つの異なるサイズ (normal と large) のアイコンがあります。

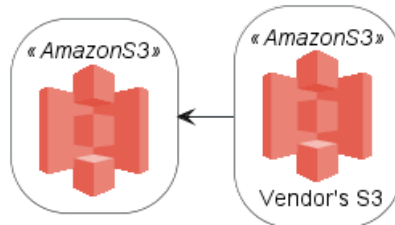
スプライトの含まれるファイルをインポートして使います。例: `!include <aws/Storage/AmazonS3/AmazonS3>`. インポートされると、通常のスプライトと同様、`<$sprite_name>` のように使用することができます。

`!include <aws/common>` のようにして `common.puml` をインクルードすると、そこに定義されたヘルパーマクロを使用することができます。`common.puml` をインポートすると、`NAME_OF_SPRITE(parameters...)` マクロを使用することができます。

使用例:

```
@startuml
!include <aws/common>
!include <aws/Storage/AmazonS3/AmazonS3>
!include <aws/Storage/AmazonS3/bucket/bucket>
```

```
AMAZONS3(s3_internal)
AMAZONS3(s3_partner,"Vendor's S3")
s3_internal <- s3_partner
@enduml
```



Contents

1 シーケンス図	1
1.1 基本的な例	1
1.2 分類子の宣言	2
1.3 複数の行を持つ分類子の宣言	4
1.4 分類子名にアルファベット以外を使う	4
1.5 自分自身へのメッセージ	5
1.6 テキストの位置	5
1.6.1 応答メッセージのテキストを矢印の下に表示する	5
1.7 矢印の見た目を変える	6
1.8 矢印の色を替える	6
1.9 メッセージシーケンスの番号付け	7
1.10 タイトル、ヘッダー、フッター	10
1.11 図の分割	11
1.12 メッセージのグループ化	11
1.13 group の 2 つ目のラベル	12
1.14 メッセージに付けるノート	13
1.15 その他のノート	14
1.16 ノートの形を変える [hnote, rnote]	15
1.17 すべての分類子にまたがるノート [across]	15
1.18 複数のノートを同じレベルに並べる [/]	16
1.19 Creole と HTML	17
1.20 境界線 (区切り線)	18
1.21 リファレンス	18
1.22 遅延	19
1.23 テキストの折り返し	19
1.24 間隔	20
1.25 ライフラインの活性化と破棄	20
1.26 Return	22
1.27 分類子の生成	23
1.28 活性化、非活性化、生成のショートカット記法	23
1.29 インとアウトのメッセージ	25
1.30 インとアウトのメッセージに短い矢印を使う	26
1.31 アンカーと持続時間	27
1.32 ステレオタイプとスポット	28
1.33 タイトルについての詳細	29
1.34 分類子の囲み	30
1.35 フッターの除去	30
1.36 スキンパラメータ	31
1.37 パディングの変更	33
1.38 付録：全種類の矢印の例	34
1.38.1 通常の矢印	34
1.38.2 自分自身への矢印	35
1.38.3 インとアウトのメッセージ ('['、']')	36
1.38.4 インのメッセージ ('[')	36
1.38.5 アウトのメッセージ (']')	38
1.38.6 短いインとアウトのメッセージ ('?')	39
1.38.7 短いインのメッセージ ('?')	39
1.38.8 短いアウトのメッセージ ('?')	40
1.39 特有の skinparam	42
1.39.1 デフォルト	42
1.39.2 ライフラインの設定 (lifelineStrategy)	42
1.39.3 厳密な UML スタイル (style strictuml)	42
1.40 未接続の分類子を表示しない	43
1.41 グループメッセージに色を付ける	43
1.42 メインフレーム	44
1.43 Slanted or odd arrows	44



2 ユースケース図	47
2.1 ユースケース	47
2.2 アクター	47
2.3 アクターのスタイルを変更する	48
2.3.1 棒人間 (デフォルト)	48
2.3.2 Awesome man	48
2.3.3 Hollow man	49
2.4 ユースケースの説明	49
2.5 パッケージ	50
2.6 簡単な例	51
2.7 継承	52
2.8 ノートの使用方法	52
2.9 ステレオタイプ	53
2.10 矢印の方向を変えるには	53
2.11 図を分割する	55
2.12 左から右に描画する	55
2.13 スキン設定 (Skinparam)	56
2.14 完全な例	57
2.15 ビジネスユースケース	57
2.15.1 ビジネスユースケース	57
2.15.2 ビジネスアクター	58
2.16 矢印の色とスタイルを変更する (インラインスタイル)	58
2.17 要素の色とスタイルを変更する (インラインスタイル)	59
2.18 Display JSON Data on Usecase diagram	59
2.18.1 Simple example	59
3 クラス図	60
3.1 宣言する要素	60
3.2 クラス間の関係	61
3.3 関係のラベル	62
3.4 要素名と関係のラベルでの非文字の使用	62
3.4.1 \$ で始まる名前	63
3.5 メソッドの追加	63
3.6 可視性の定義	64
3.7 Abstract と Static	65
3.8 高等なクラス本体	66
3.9 注釈とステレオタイプ	66
3.10 注釈の詳細	67
3.11 フィールド (フィールド、属性、メンバー) またはメソッドへの注釈	68
3.11.1 制限事項	68
3.11.2 フィールドまたはメンバーへの注釈	68
3.11.3 同名のメソッドへの注釈	69
3.12 リンクへの注釈	69
3.13 抽象クラスとインタフェース	70
3.14 属性、メソッド等の非表示	71
3.15 非表示クラス	72
3.16 クラスの削除	72
3.17 タグ付き要素またはワイルドカードの非表示、削除、復元	73
3.18 孤立したクラスを非表示または削除する	74
3.19 ジェネリクスの使用	75
3.20 特殊な目印	76
3.21 パッケージ	76
3.22 パッケージスタイル	76
3.23 名前空間	78
3.24 自動的にパッケージを作成する	78
3.25 ロリポップ (棒付きキャンディー) インタフェース	79
3.26 矢印の向きを変える	79
3.27 関連クラス	80

3.28	同一クラスに複数の関連	81
3.29	化粧をする	82
3.30	ステレオタイプの化粧	82
3.31	色のグラデーション	83
3.32	レイアウトの手助け	84
3.33	大きなファイルの分割	84
3.34	継承 (extends) と実装 (implements)	85
3.35	角括弧を使用した関係 (リンク、矢印) のスタイル	86
3.35.1	線のスタイル	86
3.35.2	線の色	87
3.35.3	線の太さ	87
3.35.4	混合	88
3.36	関係 (リンク、矢印) の色とスタイルを変更する (インラインスタイル)	89
3.37	クラスの色とスタイルを変更する (インラインスタイル)	89
3.38	クラスメンバ間の矢印	90
3.39	継承の矢印のグループ化	91
3.39.1	groupInheritance 1 (グループ化しない)	91
3.39.2	groupInheritance 2 (2 つ以上の場合にグループ化)	92
3.39.3	groupInheritance 3 (3 つ以上の場合にグループ化)	92
3.39.4	groupInheritance 4 (4 つ以上の場合にグループ化)	93
3.40	Display JSON Data on Class or Object diagram	94
3.40.1	Simple example	94
3.41	Packages and Namespaces Enhancement	94
4	オブジェクト図	96
4.1	オブジェクトの定義	96
4.2	オブジェクト間の関係	96
4.3	n-項関連	97
4.4	フィールドの追加	97
4.5	クラス図と共通の機能	98
4.6	マップテーブル (連想配列)	98
4.7	map を利用して PERT (Program Evaluation and Review Technique) 図を作成する	100
4.8	Display JSON Data on Class or Object diagram	101
4.8.1	Simple example	101
5	アクティビティ図 (レガシー版)	103
5.1	単純なアクティビティ	103
5.2	矢印のラベル	103
5.3	矢印の方向を変える	103
5.4	分岐	104
5.5	もっと分岐	105
5.6	同期	106
5.7	長いアクティビティの記述	107
5.8	注釈	107
5.9	パーティション	108
5.10	スキンパラメータ	109
5.11	八角形	110
5.12	完全な例	110
6	アクティビティ図 (新しい構文)	113
6.0.1	新しい構文の利点	113
6.0.2	新シンタックスへの移行	113
6.1	単純なアクティビティ	113
6.2	開始/終了	113
6.3	条件文	114
6.3.1	複数条件 (水平モード)	115
6.3.2	複数条件 (垂直モード)	116
6.4	スイッチとケース [switch, case, endswitch]	117
6.5	アクションの停止を伴う条件文 [kill, detach]	118



6.6	繰り返し (後判定)	119
6.7	repeat 節を中断する [break]	120
6.8	Goto and Label Processing [label, goto]	121
6.9	繰り返し (前判定)	122
6.10	並列処理	123
6.10.1	単純な fork	123
6.10.2	end merge を使った fork	124
6.10.3	end fork のラベル、または join 仕様 (UML joinspec):	125
6.10.4	その他の例	126
6.11	処理の分岐	126
6.11.1	Split	126
6.11.2	入力の分岐 (複数開始)	127
6.11.3	出力の分岐 (複数終了)	128
6.12	注釈	129
6.13	色指定	130
6.14	矢印無しの線	131
6.15	矢印	132
6.16	コネクタ	133
6.17	コネクタの色	133
6.18	グループ化 (パーティション)	134
6.18.1	グループ	134
6.18.2	パーティション	134
6.18.3	グループ、パーティション、パッケージ、四角形、カード	136
6.19	スイムレーン	137
6.20	矢印の除去 (detach, kill)	140
6.21	SDL 図	141
6.22	完全な例	142
6.23	条件のスタイル	144
6.23.1	inside スタイル (デフォルト)	144
6.23.2	diamond スタイル	145
6.23.3	InsideDiamond (または <i>foo1</i>) スタイル	146
6.24	条件終了のスタイル	147
6.24.1	diamond スタイル (デフォルト)	147
6.24.2	水平ライン (hline) スタイル	148
6.25	グローバル (global) スタイルの使用	149
6.25.1	スタイル無し (デフォルト)	149
6.25.2	スタイル有り	149
7	コンポーネント図	152
7.1	コンポーネント	152
7.1.1	名前に関する注意	152
7.2	インタフェース	153
7.3	基本的な例	153
7.4	ノートの使用法	153
7.5	コンポーネントのグループ化	155
7.6	矢印の方向を変える	156
7.7	UML2 表記の使用	158
7.8	UML1 表記の使用	158
7.9	四角形表記の使用 (UML 表記をしない)	159
7.10	長い説明	159
7.11	個々の色	159
7.12	ステレオタイプでスプライトを使用	159
7.13	見かけを変える	160
7.14	特有の skinparam	162
7.14.1	componentStyle	162
7.15	孤立したコンポーネントを非表示または削除する	163
7.16	タグ付け、またはワイルドカードによるコンポーネントの非表示、削除、復元	164
7.17	Display JSON Data on Component diagram	166



7.17.1	Simple example	166
7.18	Port [port, portIn, portOut]	166
7.18.1	Port	166
7.18.2	PortIn	167
7.18.3	PortOut	167
7.18.4	Mixing PortIn & PortOut	168
8	配置図	170
8.1	要素の宣言	170
8.2	要素の宣言 (省略記法)	172
8.2.1	アクター	172
8.2.2	コンポーネント	173
8.2.3	インターフェース	173
8.2.4	ユースケース	173
8.3	リンク、矢印	173
8.4	各括弧を使用した矢印のスタイル	176
8.4.1	線のスタイル	176
8.4.2	線の色	177
8.4.3	線の太さ	177
8.4.4	混合	178
8.5	矢印の色とスタイルを変更する (インラインスタイル)	178
8.6	要素の色とスタイルを変更する (インラインスタイル)	179
8.7	入れ子にできる要素	179
8.8	パッケージと入れ子要素	180
8.8.1	一階層の例	180
8.8.2	他の例	181
8.8.3	すべて入れ子にした例	182
8.9	別名	186
8.9.1	as による単純な別名	186
8.9.2	長い別名の例	187
8.10	角に丸みをつける	189
8.11	特有の skinparam	189
8.11.1	roundCorner	189
8.12	付録: 線の種類の一覧	190
8.13	付録: 矢印の先端と'0' 矢印の一覧	191
8.13.1	矢印の先端	191
8.13.2	丸形の矢印 ('0' 矢印)	192
8.14	付録: すべての要素に対するインラインスタイルのテスト	193
8.14.1	シンプルな要素	193
8.14.2	入れ子の要素	194
8.14.3	サブ要素無し	194
8.14.4	サブ要素有り	195
8.15	付録: すべての要素に対するスタイルのテスト	196
8.15.1	シンプルな要素	196
8.15.2	グローバルスタイル (componentDiagram)	196
8.15.3	エレメント毎のスタイル	197
8.15.4	入れ子要素 (階層無し)	201
8.15.5	グローバルスタイル (componentDiagram)	201
8.15.6	入れ子要素ごとのスタイル	201
8.15.7	入れ子要素 (一階層)	203
8.15.8	グローバルスタイル (componentDiagram)	203
8.15.9	入れ子要素ごとのスタイル	204
8.16	付録: すべての要素にスタイル指定した場合のステレオタイプのテスト	207
8.16.1	単純な要素	207
8.17	Display JSON Data on Deployment diagram	208
8.17.1	Simple example	208
8.18	Mixing Deployment (Usecase, Component, Deployment) element within a Class or Object diagram	209

8.18.1	Mixing all elements	209
8.19	Port [port, portIn, portOut]	211
8.19.1	Port	211
8.19.2	PortIn	212
8.19.3	PortOut	212
8.19.4	Mixing PortIn & PortOut	213
9	ステート図	215
9.1	簡単なステート	215
9.2	ステートの表現を変える	215
9.3	合成状態	216
9.3.1	内部サブ状態	216
9.3.2	サブ状態からサブ状態へ	217
9.4	長い名前	218
9.5	履歴	219
9.6	フォーク (非同期実行)	219
9.7	同時状態	220
9.7.1	水平セパレーター	220
9.7.2	垂直セパレーター 	221
9.8	条件	222
9.9	全ステレオタイプの例 (choice, fork, join, end)	222
9.10	入場点と退場点	223
9.11	入力ピンと出力ピン	224
9.12	展開	225
9.13	矢印の方向	226
9.14	線の色とスタイルを変更する	227
9.15	注釈	227
9.16	リンクへの注釈	228
9.17	その他の注釈	228
9.18	インライン色指定	229
9.19	skinparam	230
9.19.1	すべての状態遷移図特有の skinparam のテスト	231
9.20	スタイル変更	231
9.21	状態の色とスタイルを変更する (インラインスタイル)	232
9.22	別名	234
9.23	Display JSON Data on State diagram	235
9.23.1	Simple example	235
10	タイミング図	236
10.1	要素、ライフラインの定義	236
10.2	バイナリとクロック	237
10.3	メッセージ (相互作用)	237
10.4	相対時間での指定	238
10.5	アンカーポイント	239
10.6	インスタンス指向	239
10.7	スケールの設定	240
10.8	初期状態	240
10.9	複雑な状態	241
10.10	状態の非表示	241
10.11	時間軸を非表示にする	242
10.12	時刻と日付の使用	243
10.13	時間定規 (time constraint) の追加	244
10.14	期間のハイライト	244
10.15	ノートの使用	245
10.16	タイトルなどを追加する	246
10.17	複雑な例	247
10.18	デジタル信号の例	248
10.19	色の追加	249
10.20	グローバルスタイルの使用	250

10.20.1 スタイル無し (デフォルト)	250
10.20.2 スタイル有り	250
10.21 Applying Colors to specific lines	251
10.22 Compact mode	252
10.22.1 By default	252
10.22.2 Global mode with <code>mode compact</code>	253
10.22.3 Local mode with only <code>compact</code> on element	253
11 JSON データを表示する	255
11.1 複雑な例	255
11.2 一部をハイライトする	256
11.3 Using different styles for highlight	256
11.4 JSON の基本要素	257
11.4.1 すべての JSON 基本要素の例	257
11.5 JSON 配列またはテーブル	258
11.5.1 配列型	258
11.5.2 シンプルな配列またはテーブル	259
11.5.3 Number 配列	259
11.5.4 String 配列	259
11.5.5 Boolean 配列	259
11.6 number 型	259
11.7 string 型	260
11.7.1 Unicode	260
11.7.2 2 文字のエスケープシーケンス	260
11.8 最小の JSON の例	261
11.9 Empty table or list	262
11.10 スタイルを使用する	262
11.10.1 スタイル無し (デフォルト)	262
11.10.2 スタイル有り	263
11.11 Display JSON Data on Class or Object diagram	264
11.11.1 Simple example	264
11.11.2 Complex example: with all JSON basic element	264
11.12 Display JSON Data on Deployment (Usecase, Component, Deployment) diagram	265
11.12.1 Simple example	265
11.13 Display JSON Data on State diagram	266
11.13.1 Simple example	266
12 YAML データを表示する	268
12.1 複雑な例	268
12.2 特定のキー (記号と Unicode の使用)	269
12.3 一部をハイライトする	269
12.3.1 通常スタイル	269
12.3.2 カスタムスタイル	270
12.4 Using different styles for highlight	270
12.5 グローバルなスタイル指定	271
12.5.1 スタイル無し (デフォルト)	271
12.5.2 スタイル有り	272
13 ネットワーク図 (nwdiag)	274
13.1 シンプルなネットワーク図	274
13.1.1 ネットワークの定義	274
13.1.2 ネットワーク上にいくつかの要素またはサーバーを定義する	274
13.1.3 完全な例	274
13.2 複数のアドレスを定義するケース	275
13.3 ノードのグルーピング	276
13.3.1 ネットワーク定義の中でグループを定義	276
13.3.2 ネットワーク定義の外でグループを定義	276
13.3.3 一つのネットワークに複数のグループを定義	277
13.3.4 グループが2つの例	277



13.3.5 グループが3つの例	278
13.4 ネットワークとグループに対する拡張文法	279
13.4.1 ネットワーク	279
13.4.2 グループ	280
13.5 スプライトの使用	281
13.6 OpenIconic の使用	282
13.7 複数のネットワークに一つのノードを定義	283
13.8 ピア接続	284
13.9 ピア接続とグループ	284
13.9.1 グループ無し	284
13.9.2 1 番目にグループを記述	285
13.9.3 2 番目にグループを記述	286
13.9.4 3 番目にグループを記述	287
13.10 ネットワーク図にタイトル、ヘッダ、フッタ、キャプション、凡例を追加する	288
13.11 影の有無	289
13.11.1 影有り (デフォルト)	289
13.11.2 影無し	289
13.12 ネットワークの幅の変更	290
13.13 その他の内部ネットワーク	293
13.14 グローバルスタイルの使用	294
13.14.1 スタイル無し (デフォルト)	294
13.14.2 スタイル有り	295
13.15 付録: ネットワーク図 (nwdiag) 上のすべての形状のテスト	296
14 Salt (ワイヤフレームによる GUI 設計ツール)	299
14.1 基本のウィジェット	299
14.2 テキストエリア	299
14.3 ドロップリストの開閉	300
14.4 罫線の使用 [、#、!、-、+]	301
14.5 グループボックス [^]	301
14.6 セパレータの使用 [..、==、~~、-]	301
14.7 木構造ウィジェット [T]	302
14.8 木構造と表 [T]	302
14.9 括弧で括る [{、}]	304
14.10 タブの追加 [/]	304
14.11 メニューの使用 [*]	305
14.12 テーブル (上級)	306
14.13 スクロールバー [S、SI、S-]	307
14.14 色	308
14.15 Salt での Creole の使用	308
14.16 疑似スプライト [«、»]	310
14.17 OpenIconic	310
14.18 タイトル、ヘッダ、フッタ、キャプション、凡例を追加する	311
14.19 拡大、DPI	312
14.19.1 拡大なし (デフォルト)	312
14.19.2 scale	312
14.19.3 DPI	312
14.20 Salt をアクティビティ図の上に表示する	313
14.21 Salt をアクティビティ図の繰り返し条件 (前判定) の上に表示する	315
14.22 Salt をアクティビティ図の繰り返し条件 (後判定) の上に表示する	316
14.23 skinparam	317
14.24 スタイル	318
15 アーキテクチャ図	319
15.1 アーキテクチャの要素	319
15.2 ジャンクション	319
15.3 例 1	320
15.4 例 2	321
15.5 使用できるアイコン一覧	322



15.6 ArchiMate マクロ	322
15.6.1 Archimate マクロとライブラリ	322
15.6.2 Archimate 要素	322
15.6.3 Archimate の関係 (relationship)	323
15.6.4 付録：すべての Archimate RelationType の例	324
16 ガントチャート	328
16.1 タスクの定義	328
16.1.1 期間	328
16.1.2 開始	328
16.1.3 終了	329
16.1.4 開始と終了	329
16.2 AND 接続詞を使った 1 行宣言	329
16.3 依存関係	330
16.4 短い名前	330
16.5 色のカスタマイズ	330
16.6 進捗状況	331
16.6.1 完了率による追加	331
16.6.2 完了の色を変更する (スタイル別)	331
16.7 マイルストーン	332
16.7.1 依存関係に基づくマイルストーン	332
16.7.2 日付指定のマイルストーン	332
16.7.3 全タスク完了のマイルストーン	333
16.8 ハイパーリンク	333
16.9 日付の表示	333
16.10 日付の色	334
16.11 スケールの変更	334
16.11.1 daily (デフォルト)	334
16.11.2 weekly	335
16.11.3 monthly	336
16.11.4 quarterly	336
16.11.5 yearly	336
16.12 ズーム (全スケールの例)	337
16.12.1 日単位スケールでのズーム	337
16.12.2 ズームなし	337
16.12.3 ズームあり	337
16.12.4 週単位スケールでのズーム	338
16.12.5 ズームなし	338
16.12.6 ズームあり	338
16.12.7 月単位スケールでのズーム	339
16.12.8 ズームなし	339
16.12.9 ズームあり	339
16.12.10 四半期単位スケールでのズーム	339
16.12.11 ズームなし	339
16.12.12 ズームあり	340
16.12.13 年単位スケールでのズーム	340
16.12.14 ズームなし	340
16.12.15 ズームあり	340
16.13 Weekscale with Weeknumbers or Calendar Date	341
16.13.1 With Weeknumbers (<i>by default</i>)	341
16.13.2 With Calendar Date	341
16.14 休業日	341
16.15 休業日に基づく 1 週間 (week) の定義	342
16.16 Working days	343
16.17 簡単なタスク継承	343
16.18 リソースを使用する	344
16.19 Hide resources	344
16.19.1 Without any hiding (<i>by default</i>)	344

16.19.2 Hide resources names	345
16.19.3 Hide resources footbox	345
16.19.4 Hide the both (resources names and resources footbox)	345
16.20 区切り線	346
16.21 Vertical Separator	346
16.22 複雑な例	346
16.23 コメント	347
16.24 スタイルの使用	347
16.24.1 スタイルなし (デフォルト)	347
16.24.2 スタイルあり	348
16.24.3 スタイルあり (完全な例)	349
16.24.4 簡潔なスタイル	351
16.25 ノートの追加	352
16.26 タスクの中断	354
16.27 リンクの色の変更	355
16.28 タスクやマイルストーンを同じ行に表示する	356
16.29 今日に色を付ける	356
16.30 2つのマイルストーンに挟まれたタスク	356
16.31 Grammar and verbal form	357
16.32 タイトル、ヘッダ、フッタ、キャプション、凡例を追加する	357
16.33 下部のボックスの除去 (すべてのスケールでの例)	357
16.34 カレンダーの言語	359
16.34.1 英語 (en、デフォルト)	359
16.34.2 ドイツ語 (de)	360
16.34.3 日本語 (ja)	360
16.34.4 中国語 (zh)	360
16.34.5 韓国語 (ko)	361
16.35 タスクやマイルストーンの削除	361
16.36 Start a project, a task or a milestone a number of days before or after today	362
16.37 Change Label position	362
16.37.1 The labels are near elements (<i>by default</i>)	362
16.37.2 Label on first column	363
16.37.3 Label on last column	364
17 マインドマップ	366
17.1 OrgMode の文法	366
17.2 Markdown 構文	366
17.3 算術記号による表記	367
17.4 複数行	367
17.5 Multiroot Mindmap	369
17.6 色	369
17.6.1 インラインの色指定	369
17.6.2 スタイルによる色指定	370
17.7 箱を消す	372
17.8 図の方向の変更	373
17.9 完全な例	373
17.10 スタイル変更	374
17.10.1 ノード (node)、深さ (depth)	374
17.10.2 枠無し (boxless)	375
17.11 単語の折り返し	376
17.12 Creole on Mindmap diagram	377
18 Work Breakdown Structure	380
18.1 OrgMode の文法	380
18.2 方向の変更	380
18.3 算術記号による表記	381
18.4 複数行	381
18.5 箱を消す	382
18.5.1 算術記号を使う場合	382



18.5.2	一部の箱を非表示にする	382
18.5.3	すべての箱を非表示にする	383
18.5.4	OrgMode の文法の場合	383
18.5.5	一部の箱を非表示にする	383
18.5.6	すべての箱を非表示にする	384
18.6	色 (インライン指定とスタイルの色)	384
18.7	スタイルを適用する	386
18.8	単語の折り返し	387
18.9	Add arrows between WBS elements	388
18.10	Creole on WBS diagram	389
19	数式	392
19.1	単体で使用する場合	393
19.2	どのように処理しているのか	393
20	ER 図	394
20.1	インフォメーションエンジニアリングの関係線	394
20.2	エンティティ	394
20.3	完全な例	395
21	共通コマンド	397
21.1	コメント	397
21.1.1	単一行コメント	397
21.1.2	ブロックコメント	397
21.2	拡大	397
21.3	タイトル	398
21.4	キャプション	399
21.5	フッタとヘッダ	399
21.6	図の凡例	400
21.7	付録: すべての図の例	400
21.7.1	アクティビティ図	400
21.7.2	アーキテクチャ図	401
21.7.3	クラス図	402
21.7.4	コンポーネント図、配置図、ユースケース図	402
21.7.5	ガントチャート	403
21.7.6	オブジェクト図	404
21.7.7	マインドマップ	404
21.7.8	ネットワーク図 (nwdiag)	405
21.7.9	シーケンス図	406
21.7.10	ステート図	406
21.7.11	タイミング図	407
21.7.12	Work Breakdown Structure (WBS)	408
21.7.13	Wireframe (SALT)	408
21.8	付録: すべての図でスタイルを指定した例	409
21.8.1	アクティビティ図	410
21.8.2	アーキテクチャ図	411
21.8.3	クラス図	413
21.8.4	コンポーネント図、配置図、ユースケース図	414
21.8.5	ガントチャート	416
21.8.6	オブジェクト図	417
21.8.7	マインドマップ	419
21.8.8	ネットワーク図 (nwdiag)	420
21.8.9	シーケンス図	422
21.8.10	ステート図	423
21.8.11	タイミング図	425
21.8.12	Work Breakdown Structure (WBS)	426
21.8.13	Wireframe (SALT)	427
21.9	メインフレーム	428
21.10	付録: すべての図に対するメインフレームの例	429

21.10.1	アクティビティ図	429
21.10.2	アーキテクチャ図	429
21.10.3	クラス図	430
21.10.4	コンポーネント図、配置図、ユースケース図	430
21.10.5	ガントチャート	430
21.10.6	オブジェクト図	431
21.10.7	マインドマップ	431
21.10.8	ネットワーク図 (nwdiag)	431
21.10.9	シーケンス図	432
21.10.10	ステート図	432
21.10.11	タイミング図	432
21.10.12	Work Breakdown Structure (WBS)	433
21.10.13	アイヤフレーム (SALT)	433
21.11	付録: すべての図に対するタイトル、ヘッダ、フッタ、キャプション、凡例、メインフレームの例	434
21.11.1	アクティビティ図	434
21.11.2	アーキテクチャ図	434
21.11.3	クラス図	435
21.11.4	コンポーネント図、配置図、ユースケース図	436
21.11.5	ガントチャート	436
21.11.6	オブジェクト図	437
21.11.7	マインドマップ	438
21.11.8	ネットワーク図 (nwdiag)	438
21.11.9	シーケンス図	439
21.11.10	ステート図	440
21.11.11	タイミング図	440
21.11.12	Work Breakdown Structure (WBS)	441
21.11.13	アイヤフレーム (SALT)	442
22	Creole	444
22.1	テキストの強調	444
22.2	リスト	444
22.3	エスケープ文字	445
22.4	見出し	445
22.5	絵文字	446
22.6	水平線	446
22.7	リンク	447
22.8	コード	447
22.9	テーブル	448
22.9.1	テーブルの作成	448
22.9.2	行とセルの色	449
22.9.3	枠線とテキストの色	449
22.9.4	枠線無し (背景と同色の枠線)	449
22.9.5	ヘッダを太字にするかどうか	450
22.10	ツリー	450
22.11	特殊文字	452
22.12	レガシー HTML	452
22.12.1	一般的な HTML 要素	453
22.12.2	上付き文字、下付き文字 [sub, sup]	454
22.13	OpenIconic	454
22.14	Appendix: Examples of "Creole List" on all diagrams	455
22.14.1	Activity	455
22.14.2	Class	456
22.14.3	Component, Deployment, Use-Case	457
22.14.4	Gantt project planning	458
22.14.5	Object	458
22.14.6	MindMap	459
22.14.7	Network (nwdiag)	459



22.14.8 Note	460
22.14.9 Sequence	460
22.14.10 State	461
22.14.11 WBS	462
22.15 Appendix: Examples of "Creole horizontal lines" on all diagrams	463
22.15.1 Activity	463
22.15.2 Class	464
22.15.3 Component, Deployment, Use-Case	464
22.15.4 Gantt project planning	466
22.15.5 Object	466
22.15.6 MindMap	467
22.15.7 Network (nwdiag)	467
22.15.8 Note	468
22.15.9 Sequence	468
22.15.10 State	469
22.15.11 WBS	470
22.16 スタイル対応表 (Creole と HTML)	471
23 スプライトの定義と使用	473
23.1 Inline SVG sprite	474
23.2 色の変更	474
23.3 スプライトへの変換	475
23.4 スプライトをインポートする	475
23.5 例	475
23.6 StdLib	476
23.7 スプライトを一覧表示する	476
24 skinparam コマンド	478
24.1 使用法	478
24.2 入れ子	478
24.3 白黒	478
24.4 影の有無	479
24.5 色の反転	479
24.6 色	480
24.7 文字色、フォント名、フォントサイズ	481
24.8 テキストの配置	481
24.9 例	482
24.10 すべての skinparam パラメータの一覧	486
24.10.1 コマンドライン: -language コマンド	486
24.10.2 コマンド: help skinparams	486
24.10.3 コマンド: skinparameters	486
24.10.4 All Skin Parameters on the Ashley's PlantUML Doc	488
25 前処理	489
25.1 変数定義 [=, ?=]	489
25.2 真偽値	490
25.2.1 真偽値の表現方法 [0 is false]	490
25.2.2 真偽値型の演算と演算子 [&&, , ()]	490
25.2.3 真偽値の組み込み関数 [%false(), %true(), %not(<exp>)]	490
25.3 条件分岐 [!if, !else, !elseif, !endif]	490
25.4 While ループ [!while, !endwhile]	491
25.4.1 While ループ (アクティビティ図での例)	491
25.4.2 While ループ (マインドマップでの例)	492
25.4.3 While ループ (コンポーネント図/配置図での例)	493
25.5 プロシージャ [!procedure, !endprocedure]	493
25.6 return 関数 [!function, !endfunction]	494
25.7 引数のデフォルト値	495
25.8 クォーテーション不要プロシージャ/関数 [!unquoted]	496
25.9 キーワード引数	497

25.10	ファイルや URL のインクルード [!include, !include_many, !include_once]	497
25.11	部分的なインポート [!startsub, !endsub, !includesub]	498
25.12	組み込み関数 [%]	499
25.13	ログ出力 [!log]	500
25.14	メモリーダンプ [!dump_memory]	500
25.15	アサーション [!assert]	500
25.16	カスタムライブラリの作成 [!import, !include]	501
25.17	検索パス	501
25.18	引数の文字列結合 [##]	501
25.19	動的呼び出し [%invoke_procedure(), %call_user_func()]	502
25.20	データ型に依存した加算演算子の評価 [+]	503
25.21	JSON 前処理	503
25.22	テーマのインクルード [!theme]	504
25.23	古いバージョンからの移行に関する注意事項	504
25.24	%Splitstr builtin function	504
26	Unicode	506
26.1	例	506
26.2	文字コード	508
26.3	Using Unicode Character on PlantUML	508
27	PlantUML 標準ライブラリ	509
27.0.1	標準ライブラリの概要	509
27.0.2	コミュニティからの貢献	509
27.1	標準ライブラリの一覧	509
27.2	ArchiMate (archimate)	511
27.2.1	List possible sprites	512
27.3	Amazon Labs AWS ライブラリ (awslib)	513
27.4	アジュールライブラリ [azure]	514
27.5	C4 ライブラリ (C4)	515
27.6	Cloud Insight (cloudinsight)	515
27.7	Cloudogu (cloudogu)	516
27.8	Elastic ライブラリ (elastic)	517
27.9	Google Material Icons (material)	519
27.10	Kubernetes (kubernetes)	520
27.11	Logos (logos)	521
27.12	Office (office)	523
27.13	Open Security Architecture (OSA) [osa]	525
27.14	Tupadr3 ライブラリ (tupadr3)	528
27.15	AWS ライブラリ (aws)	529

