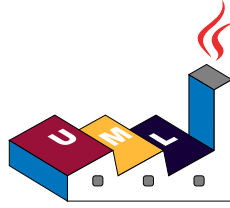


PlantUML を使った UML の描き方



PlantUML 言語リファレンスガイド

(Version 1.2019.4)

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
- アクティビティ図
- コンポーネント図
- 状態遷移図（ステートマシン図）
- オブジェクト図
- 配置図
- 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。

- 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
- MindMap diagram
- Work Breakdown Structure diagram
- AsciiMath や JLaTeXMath による、数学的記法

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

1.1 基本的な例

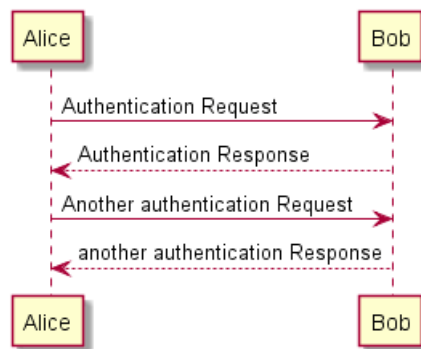
シーケンス図->を、2つの分類子間のメッセージを描画するために使います。分類子を、明示的に宣言する必要はありません。

点線の矢印を使う場合は、-->とします。

また、<-や<--を使うこともできます。これらによって図の見え方が変わることはありませんが、可読性を高めることができます。ただし、以上の方法はシーケンス図だけに当てはまります。ほかの種類の図には当てはまりません。

```
@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

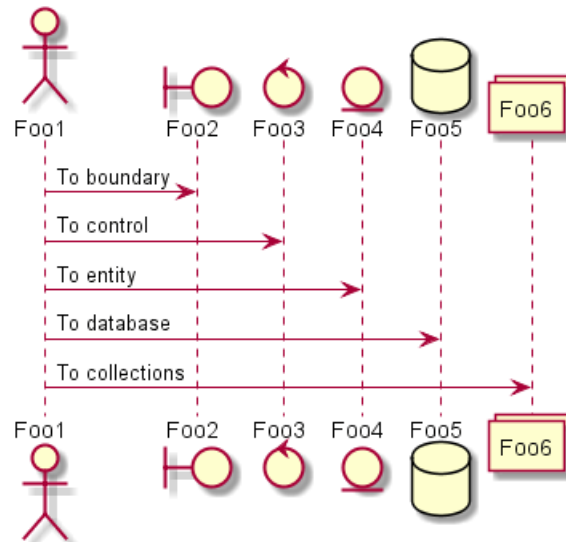
キーワード `participant` を使って、分類子の並び順を変えることができます。

分類子の宣言に別のキーワードを使用することも可能です:

- actor
- boundary
- control
- entity
- database

```
@startuml
actor Foo1
boundary Foo2
control Foo3
entity Foo4
database Foo5
collections Foo6
Foo1 -> Foo2 : To boundary
Foo1 -> Foo3 : To control
Foo1 -> Foo4 : To entity
Foo1 -> Foo5 : To database
Foo1 -> Foo6 : To collections
@enduml
```



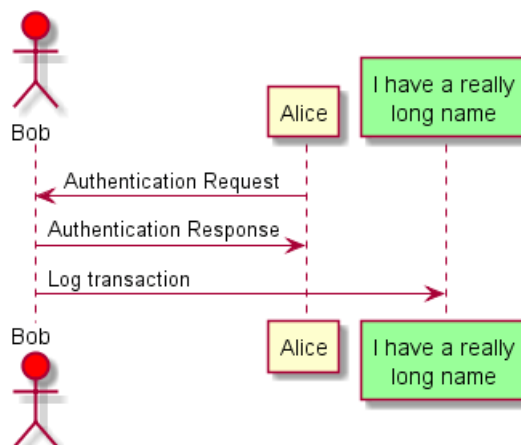


キーワード **as** を使って分類子の名前を変更することができます。

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。

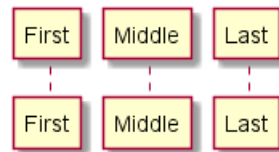
```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
    participant L as "I have a really\nlong name" #99FF99
/'
```

```
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
```



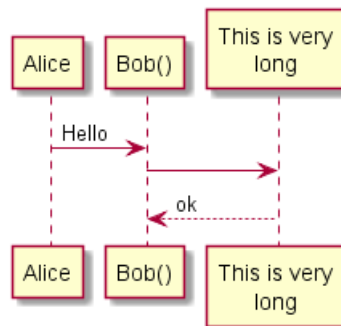
1.3 分類子名にアルファベット以外を使う

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

```



1.4 自分自身へのメッセージ

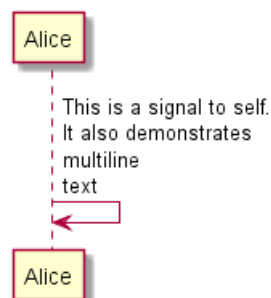
分類子は自分自身へメッセージを送信できます。

`\n` 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

```



1.5 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。

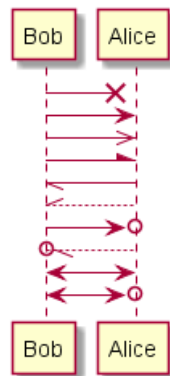


- 矢印の先端を繰り返す (たとえば >> や //) と、矢印の先端が細くなります。
- -- を - の代わりに使うと、矢印が点線になります。
- 矢じりに最後の "O" を追加
- 双方向の矢印を使用する

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

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

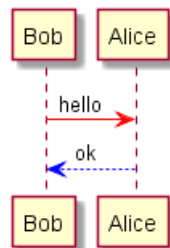
Bob <-> Alice
Bob <->o Alice
@enduml
```



1.6 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
```



1.7 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
```



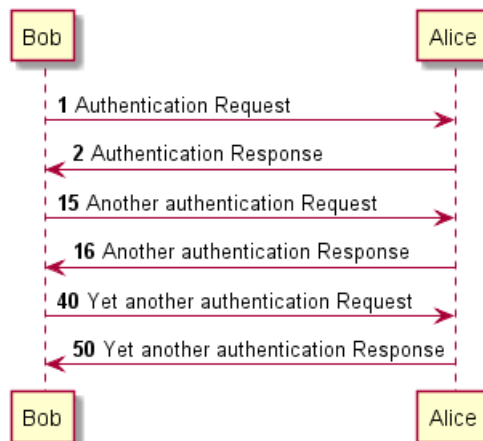
autonumber 開始で開始番号を、また、autonumber 開始 増分で増分も指定することができます。

```
@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 方式で行う（0 は桁を表し、# は存在しない場合は 0 で埋める桁を意味する）。

HTML タグを書式に使うこともできます。

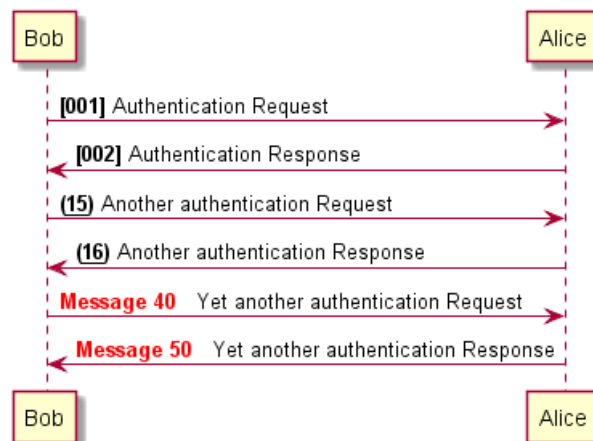
```
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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を使用することができます。

```

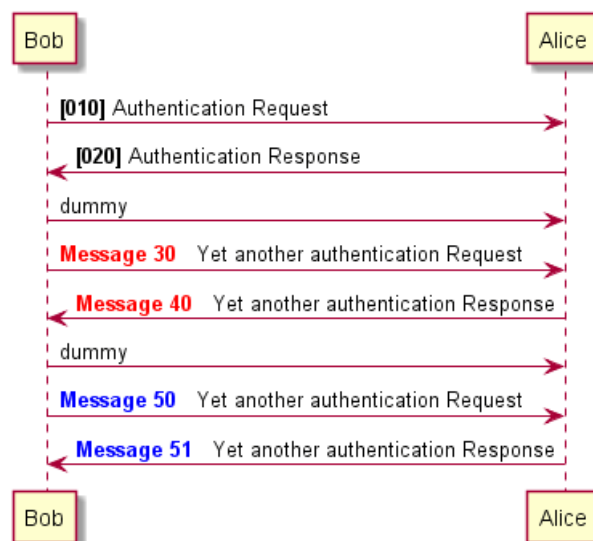
@startuml
autonumber 10 10 "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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
  
```



1.8 Page Title, Header and Footer

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。

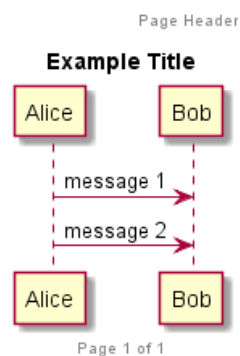
```
@startuml

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

title Example Title

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

@enduml
```



1.9 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

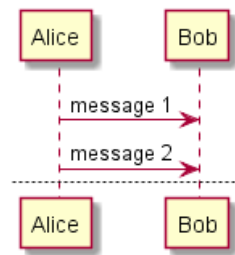
newpage

Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page

Alice -> Bob : message 5
Alice -> Bob : message 6

@enduml
```

1.10 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 を除く)

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

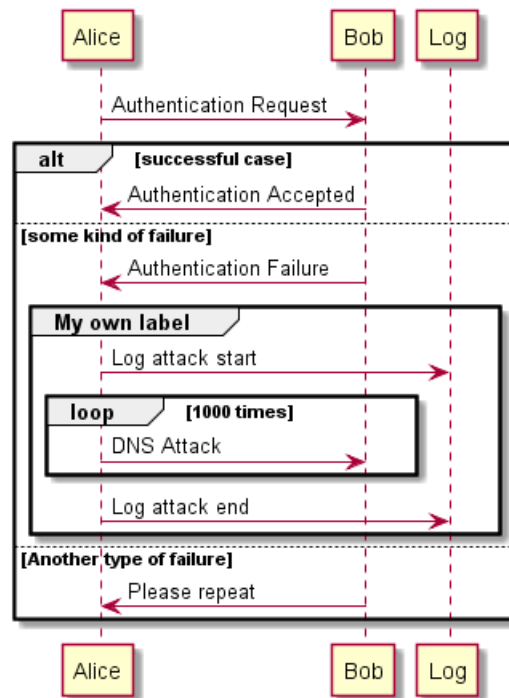
else some kind of failure

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
  
```



1.11 メッセージの注釈

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の注釈をつけることが可能です。

`end note` キーワードを使って、複数行の注釈を付けることができます。

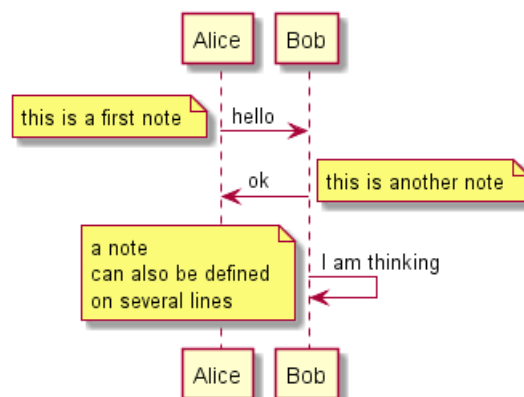
```

@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

```



1.12 その他の注釈

分類子との相対位置を指定して注釈を付けるには、次のものを使います:

注釈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` を使って複数行の注釈を付けることができます。

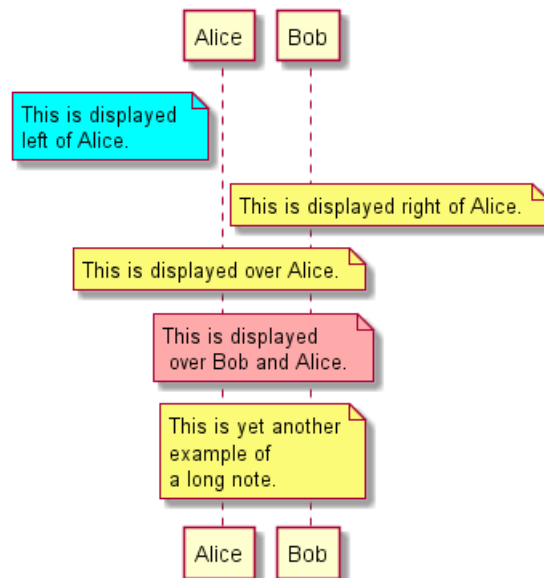
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

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
```



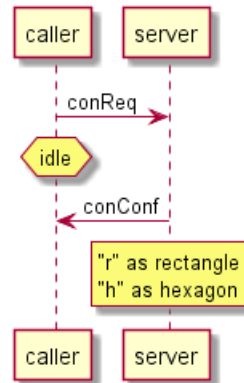
1.13 ノートの形を変える。

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

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
  "r" as rectangle
  "h" as hexagon
endrnote
```



@enduml



1.14 Creole と HTML

PlantUML では creole フォーマットを使うこともできます。

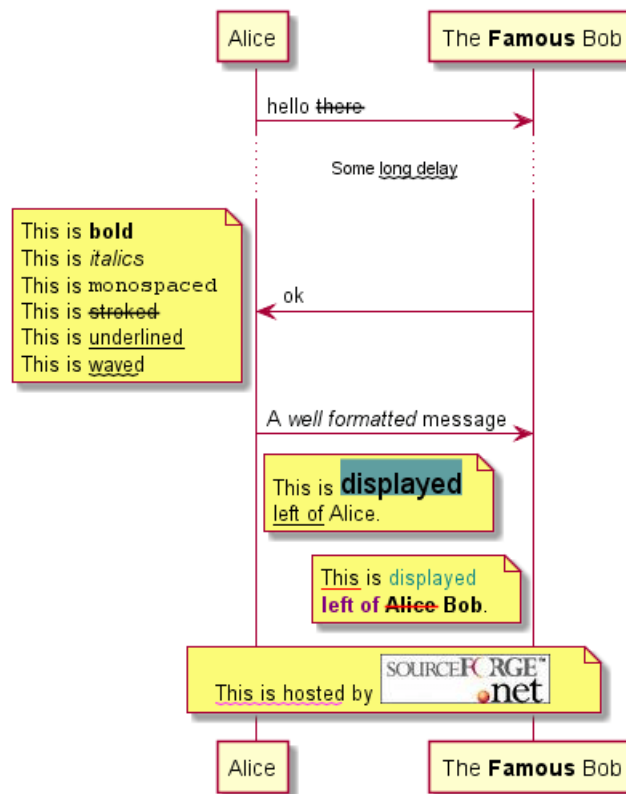
```

@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</b>.
end note
note over Alice, Bob
  <w:#FF33FF>This is hosted</w> by <img sourceforge.jpg>
end note
@enduml
  
```





1.15 境界線

== 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
```

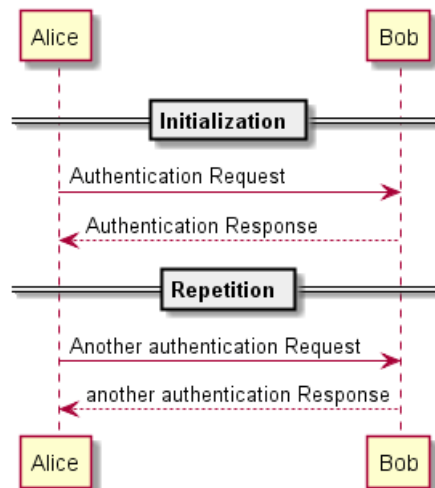
```
== Initialization ==
```

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

```
== Repetition ==
```

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

```
@enduml
```



1.16 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。

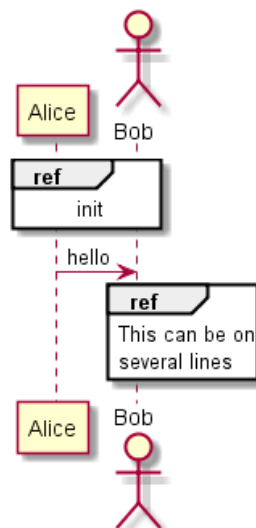
```

@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
  
```



1.17 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
  
```

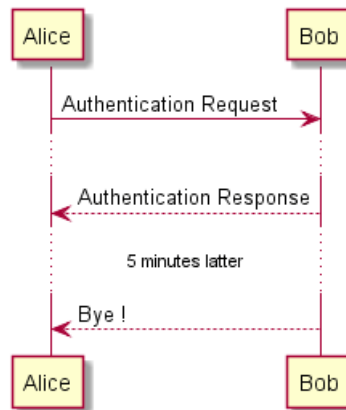


```

Alice -> Bob: Authentication Request
...
Bob --> Alice: Authentication Response
...5 minutes latter...
Bob --> Alice: Bye !

```

```
@enduml
```



1.18 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
```



1.19 ライフラインの活性化と破壊

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

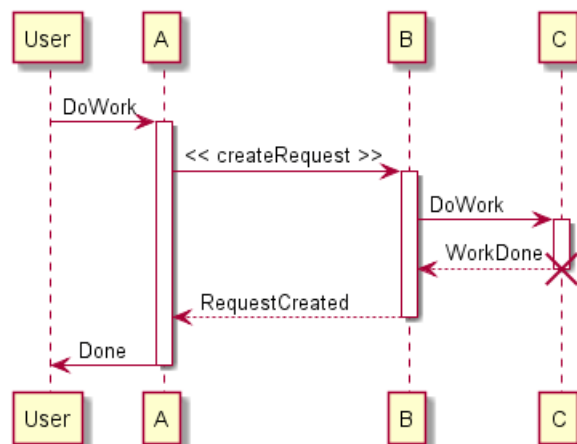
A -> B: << createRequest >>
activate B

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
```




```

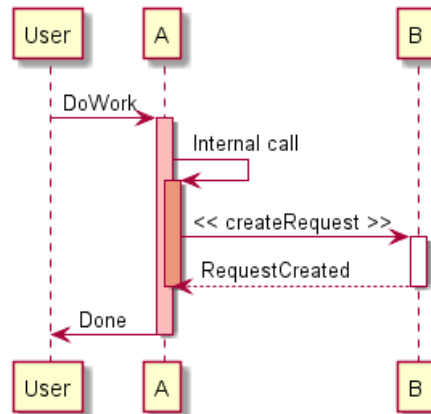
deactivate B
deactivate A
A -> User: Done
deactivate A

```

```

@enduml

```



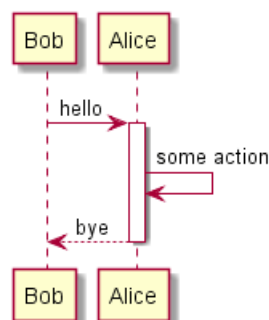
1.20 Return

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

```



1.21 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

```



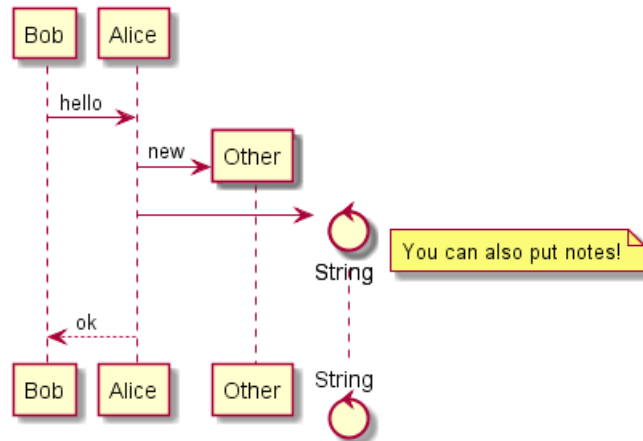
```

note right : You can also put notes!

Alice --> Bob : ok

@enduml

```



1.22 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、インまたはアウトの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

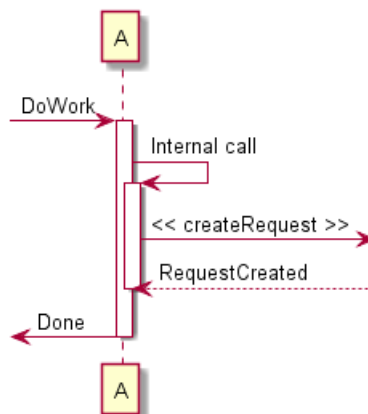
activate A

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

```

```

[-> Bob
[o-> Bob
[o->o Bob
[x-> Bob

[<- Bob
[x<- Bob

Bob ->]
Bob ->o]
Bob o->o]
Bob ->x]

Bob <-]
Bob x<-]
@enduml

```



1.23 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。

```

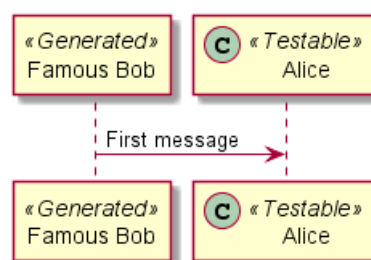
@startuml

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

Bob->>Alice: First message

@enduml

```



デフォルトでは *guillemet* キャラクターは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:

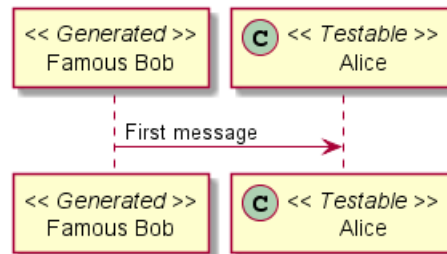
```
@startuml
```



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

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

```
@enduml
```

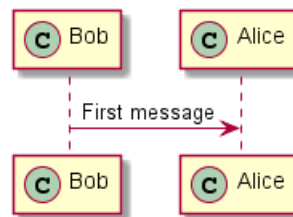


```
@startuml
```

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

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

```
@enduml
```



1.24 タイトルについての詳細

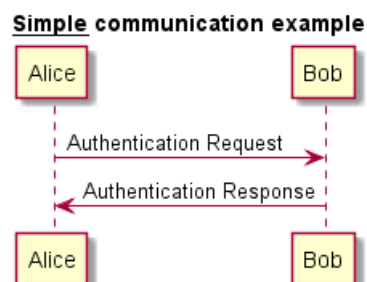
タイトルには creole フォーマットが使用できます。

```
@startuml
```

```
title __Simple__ **communication** example
```

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

```
@enduml
```



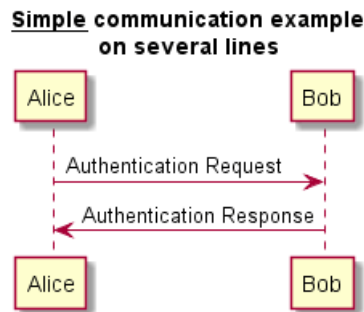
タイトルの記述では \n を使用して新しい行を追加することができます。

```
@startuml

title __Simple__ communication example\non several lines

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

@enduml
```



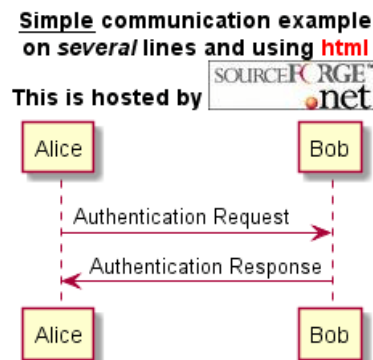
また、キーワード `title` と `end title` を使うことにより、タイトルを複数行にわたって記述できます。

```
@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
```



1.25 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
```

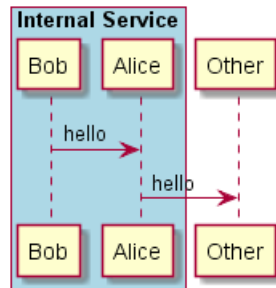


```
participant Other
```

```
Bob -> Alice : hello
```

```
Alice -> Other : hello
```

```
@enduml
```



1.26 フッターの除去

図の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 Footer removed
```

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

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

```
@enduml
```



1.27 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

```

```

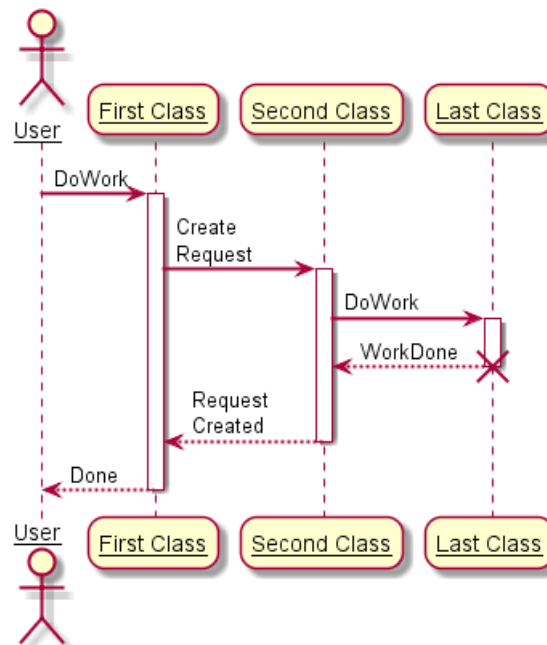
A --> User: Done
deactivate A

```

```

@enduml

```



```

@startuml
skinparam backgroundColor #EEEEBC
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

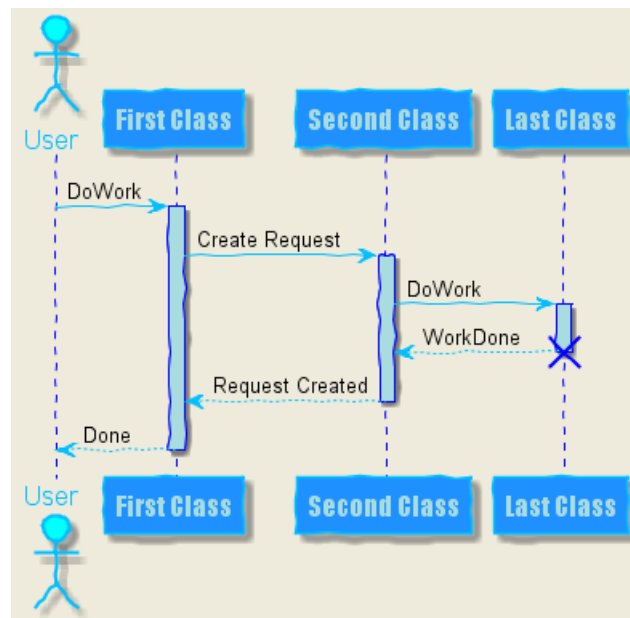
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



1.28 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。

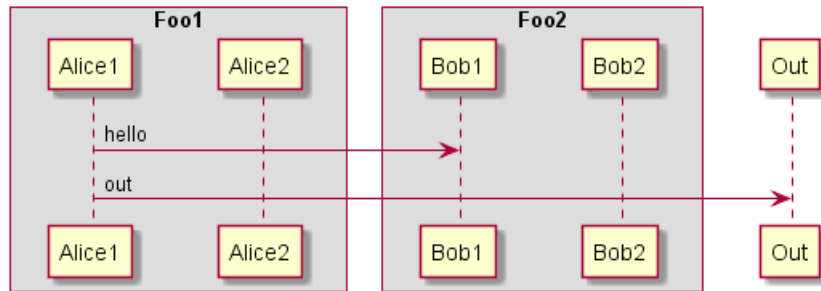
```

@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
```



2 ユースケース図

Let's have few examples :

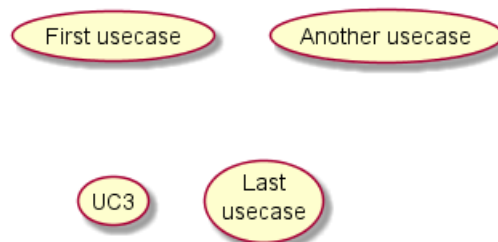
図の影は `skinparam shadowing false` コマンドにより非表示にすることができます。

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
```

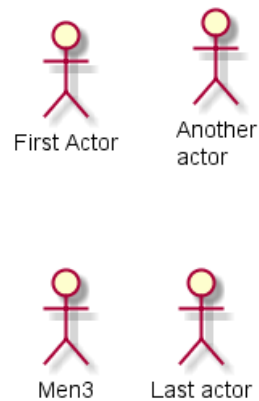


2.2 アクター

アクターは2つのコロンので囲まれます。

`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 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： -- .. == __。区切り記号の中には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.
```

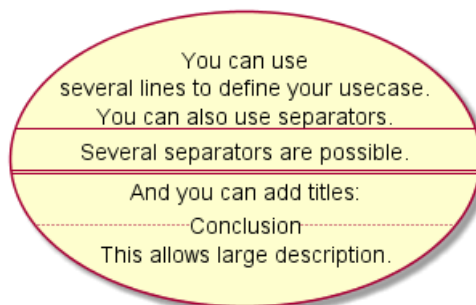
```
==
```

```
And you can add titles:
```

```
..Conclusion..
```

```
This allows large description."
```

```
@enduml
```



2.4 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
```

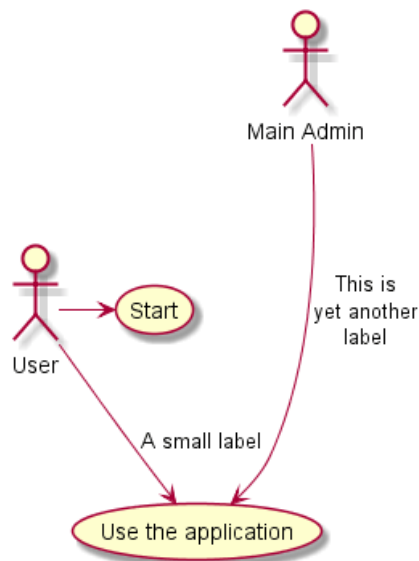
```
User -> (Start)
```

```
User --> (Use the application) : A small label
```

```
:Main Admin: ---> (Use the application) : This is\nyet another\nlabel
```



@enduml



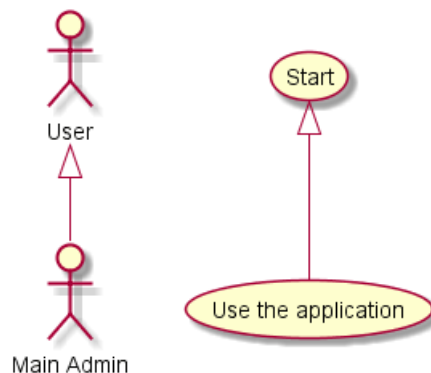
2.5 継承

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



2.6 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)

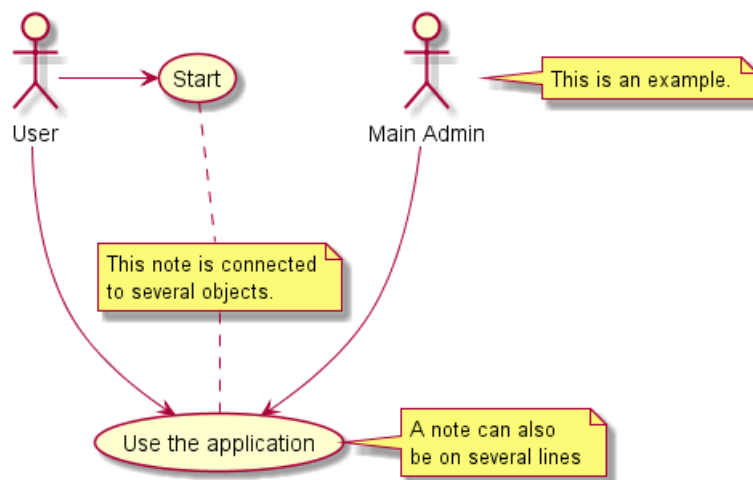
Admin ---> (Use)

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

```



2.7 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。

```

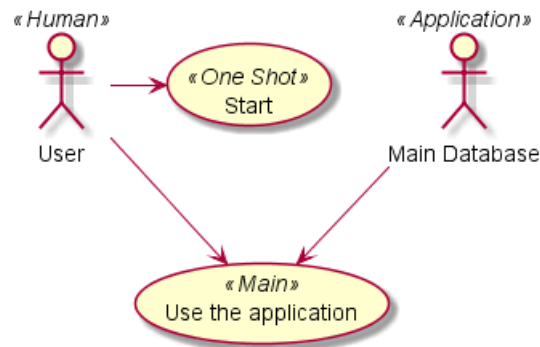
@startuml
User << Human >>
:Main Database: as MySql << Application >>
(Start) << One Shot >>
(Use the application) as (Use) << Main >>

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

MySql --> (Use)

@enduml

```

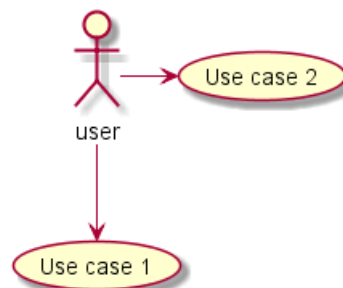


2.8 矢印の方向を変えるには

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。

```

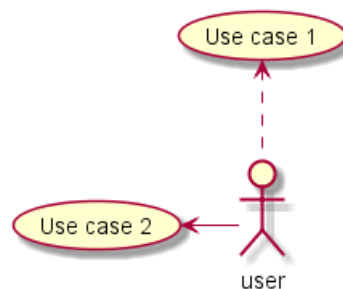
@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml
  
```



線を反対にすることでも方向を変えることができます。

```

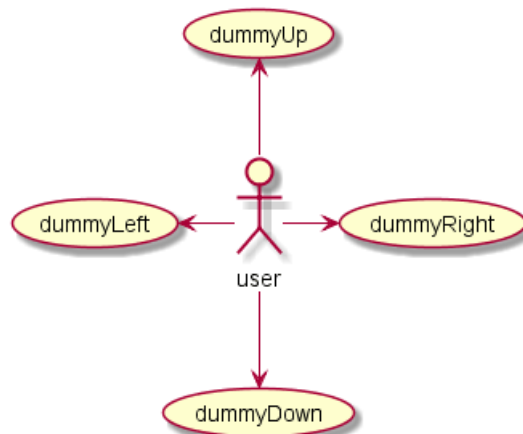
@startuml
(Use case 1) <.. :user:
(Use case 2) <- :user:
@enduml
  
```



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。

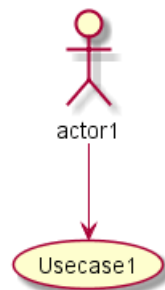
ただし、この機能の使いすぎには注意しましょう。ほとんどの場合、特別なことをしなくても *Graphviz* がその場にあった表示を選びます。

2.9 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
  
```



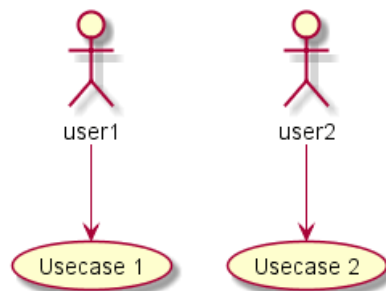
2.10 左から右に描画する

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。

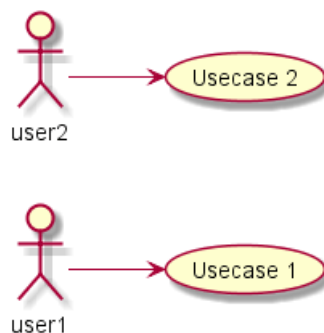
```
@startuml
```

```
left to right direction
```

```
user1 --> (Usecase 1)
```

```
user2 --> (Usecase 2)
```

```
@enduml
```



2.11 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)

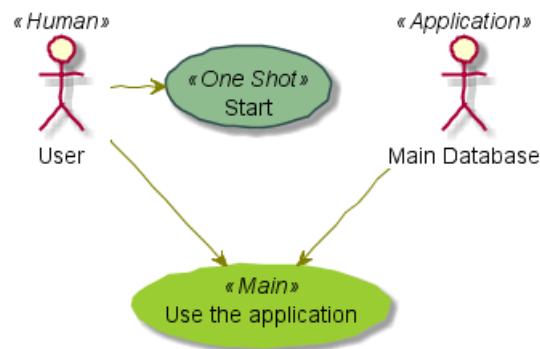
```

```
MySql --> (Use)

```

```
@enduml

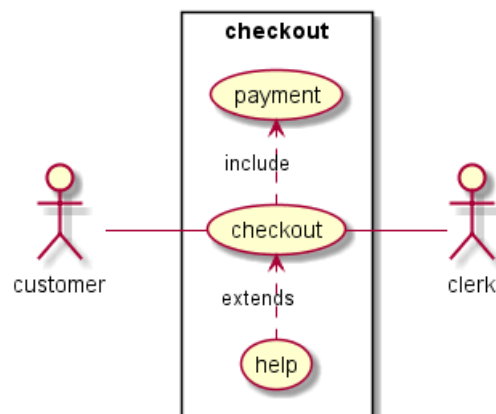
```



2.12 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

```



3 クラス図

3.1 クラス間の関係

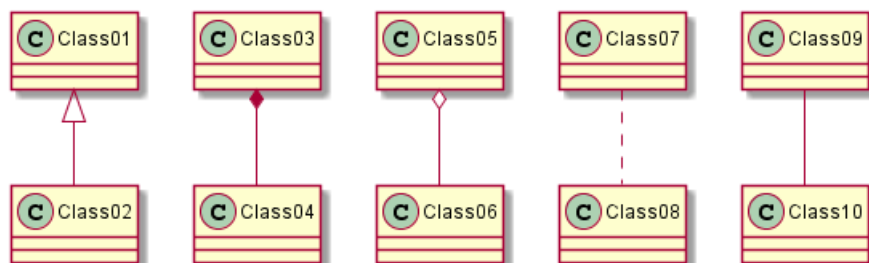
クラス間の関係は次の記号を使用して定義されています:

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

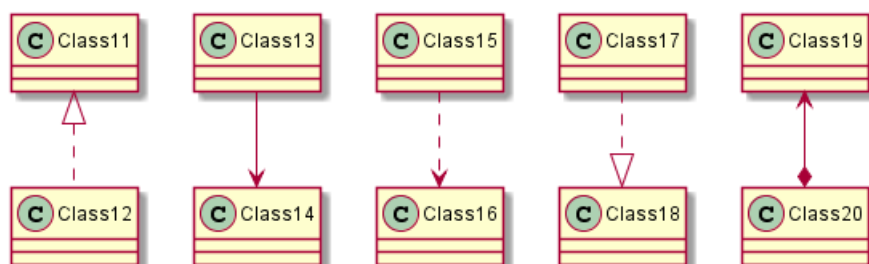
-- を .. に置き換えると点線にできます。

これらのルールを知ることによって、以下の図面を描くことができます:

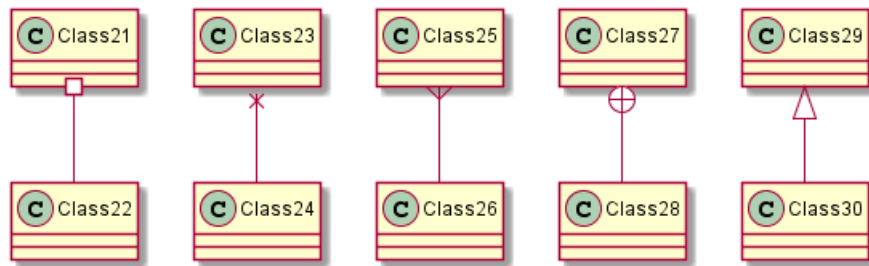
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



```
@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
```



3.2 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。

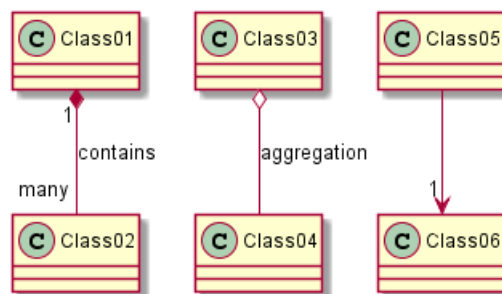
```
@startuml
```

```
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
```

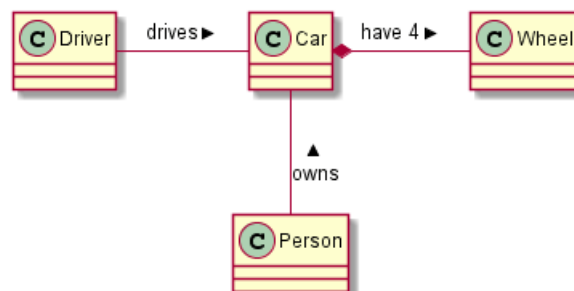
```
class Car
```

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

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

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

```
@enduml
```



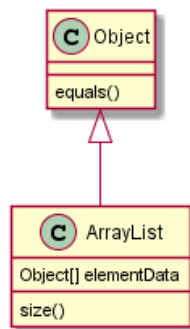
3.3 メソッドの追加

: 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
```



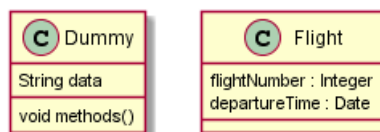
波括弧 {} を使って、フィールドやメソッドをくくることができます。

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
}

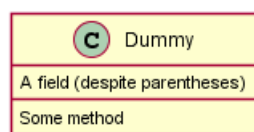
@enduml
```



You can use {field} and {method} modifiers to override default behaviour of the parser about fields and methods.









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

@enduml
```



3.4 可視性の定義

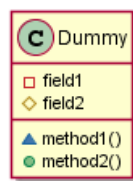
メソッドやフィールドを定義するときに対応する項目の可視性を定義する記号を使用することができます。

Character	Icon for field	Icon for method	Visibility
-			private
#			protected
~			package private
+			public

```
@startuml
```

```
class Dummy {
    -field1
    #field2
    ~method1()
    +method2()
}
```

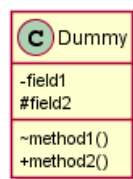
```
@enduml
```



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
```



3.5 Abstract と Static

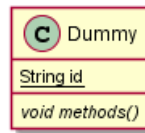
静的または抽象的なメソッドまたはフィールドは `{static}` または `{abstract}` 修飾子を使用することで定義することができます。

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



3.6 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：

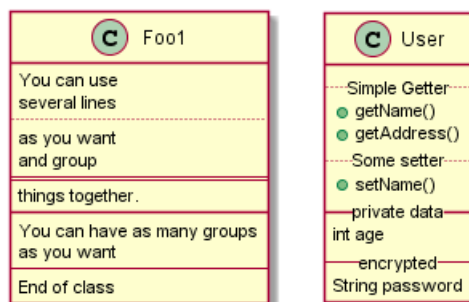
```
-- .. == __
```

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



3.7 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。

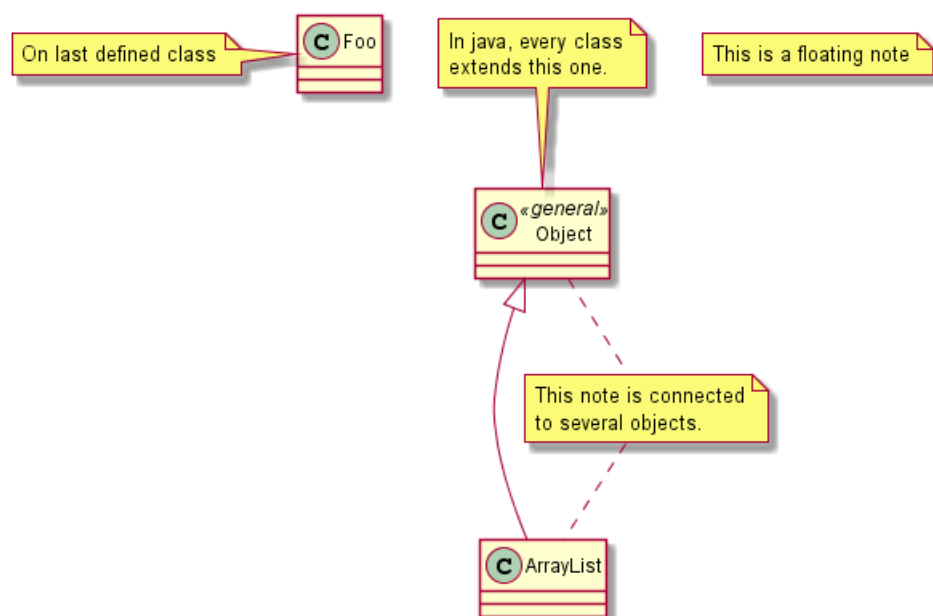
```
@startuml
class Object << general >>
Object <|--- ArrayList
```

```
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
```



3.8 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:

- ``
- `<u>`
- `<i>`
- `<s>`, ``, `<strike>`
- `` or ``
- `<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。

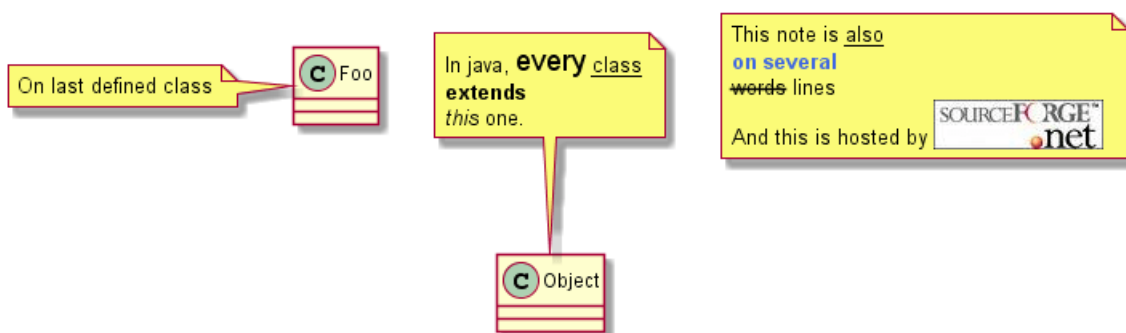
```
@startuml

class Foo
note left: On last defined class

note top of Object
  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
```



3.9 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。

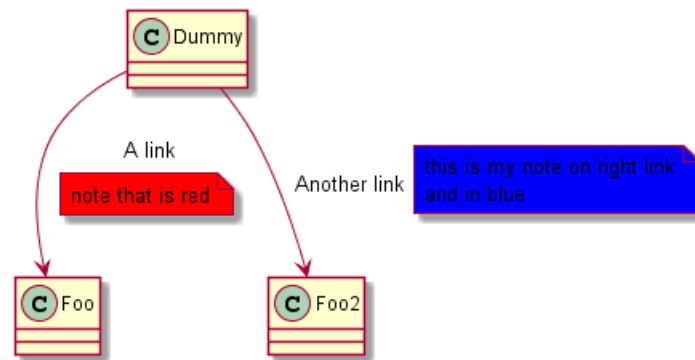
```
@startuml

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
```





3.10 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

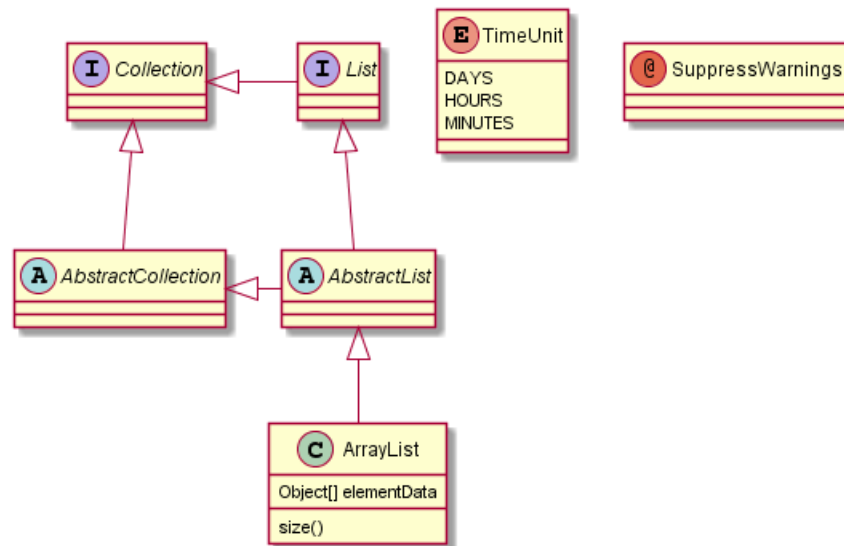
Collection <|-- List
AbstractCollection <|-- AbstractList
AbstractList <|-- ArrayList

class ArrayList {
    Object[] elementData
    size()
}

enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}

annotation SuppressWarnings

@enduml
  
```



3.11 非文字の使用

クラス（または列挙型...）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：

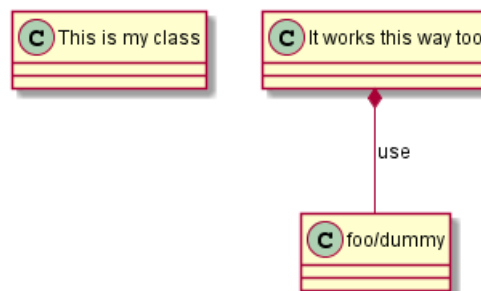
- クラス定義にはキーワード `as` を使用する
- クラス名の前後に引用符 `"` を入れる

```

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

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

```



3.12 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` は全てのインターフェースに、
- `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

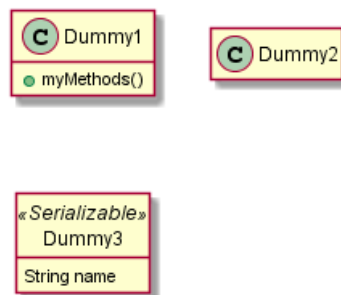
```
class Dummy1 {
    +myMethods()
}
```

```
class Dummy2 {
    +hiddenMethod()
}
```

```
class Dummy3 <<Serializable>> {
    String name
}
```

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

@enduml



3.13 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

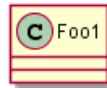
```
class Foo1
class Foo2
```

```
Foo2 *-- Foo1
```

```
hide Foo2
```



```
@enduml
```



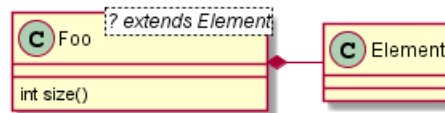
3.14 ジェネリクスの使用

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
```



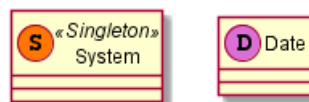
この描画は `skinparam genericDisplay old` コマンドにより非表示にすることができます。

3.15 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
```



3.16 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
```



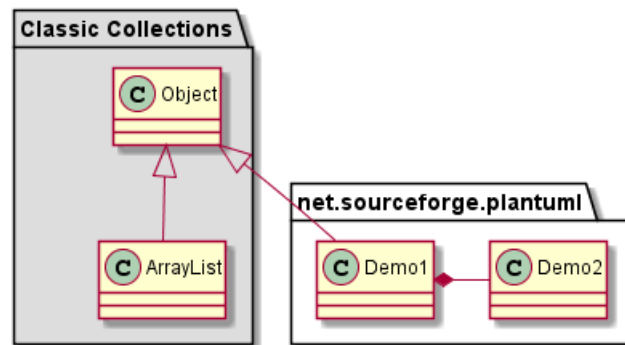
```

}

package net.sourceforge.plantuml {
    Object <|-- Demo1
    Demo1 *- Demo2
}

@enduml

```



3.17 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
}

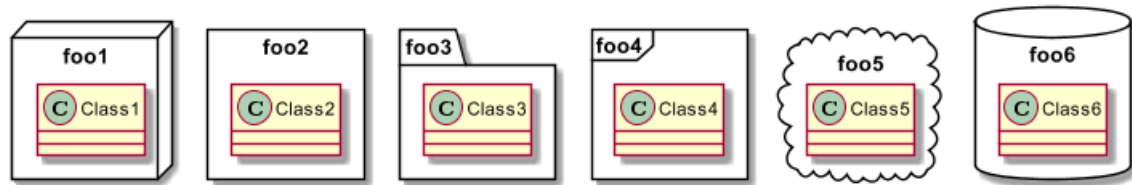
package foo5 <<Cloud>> {
    class Class5
}

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:

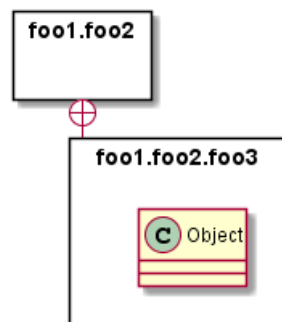
```
@startuml
skinparam packageStyle rectangle

package foo1.foo2 {
}

package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



3.18 名前空間

パッケージ内では、クラスの名前はこのクラスの一意な識別子です。それは、全く同じ名前の2つのクラスを異なるパッケージに持つことができないことを意味します。

そのような場合、パッケージの代わりに名前空間を使用したらいいでしょう。

名前空間からの完全修飾名によりクラスを参照することができます。デフォルトの名前空間からのクラスは、一つのドットで修飾します。

明示的に名前空間を作成する必要はないことに注意してください: 完全修飾されたクラスは自動的に適切な名前空間に置かれています。

```
@startuml
class BaseClass

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

    .BaseClass <|-- Meeting
}

namespace net.foo {
```



```

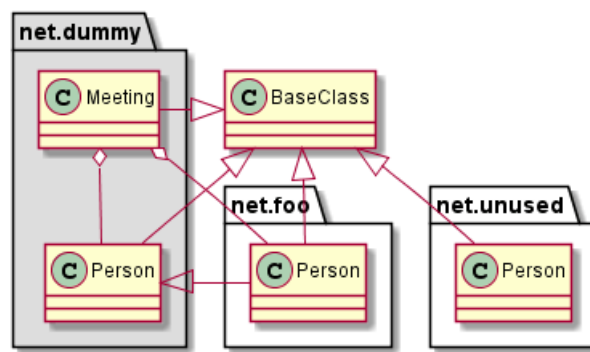
net.dummy.Person <|-- Person
.BaseClass <|-- Person

net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml

```



3.19 自動的に名前空間を作成する

コマンド `set namespaceSeparator ???` を使用して、(ドット以外の) 別の区切り文字を定義することができます。

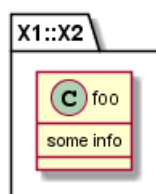
```

@startuml

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

@enduml

```



コマンド `set namespaceSeparator 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することができます。

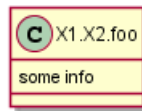
```

@startuml

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

@enduml

```

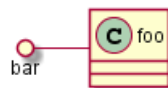


3.20 ロリポップ（棒付きキャンディー）インタフェース

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

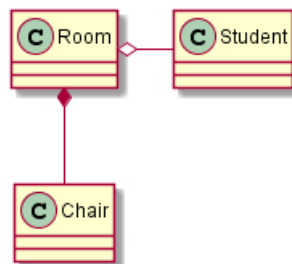
```
@startuml
class foo
bar ()- foo
@enduml
```



3.21 矢印の向きを変える

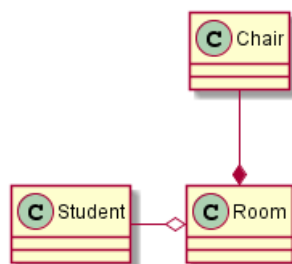
デフォルトではクラス間のリンクは2つのダッシュ--を持っており、垂直に配向されています。次のように単一のダッシュ（またはドット）を置くことによって水平方向にリンクを使用することが可能です。

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



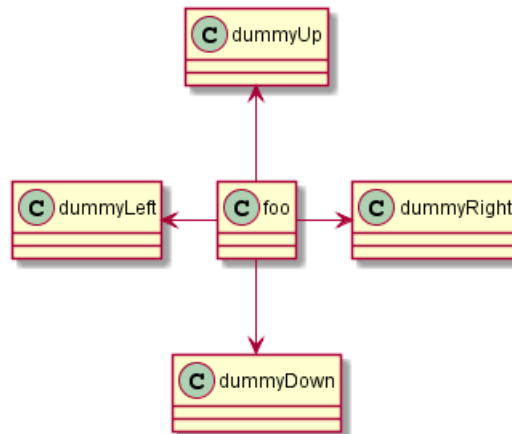
リンクをひっくり返すことにより向きを変えることができる：

```
@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-`)。

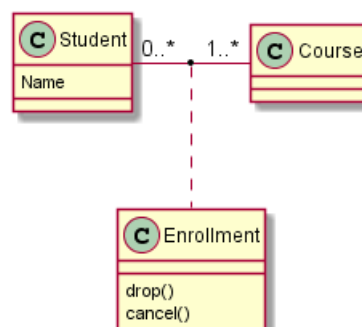
この機能を悪用してはならないことに注意してください。Graphviz は微調整のいらない良い結果を通常は与えてくれます。

3.22 関連クラス

この例のように、2 つのクラスの関係を定義した後で 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
```



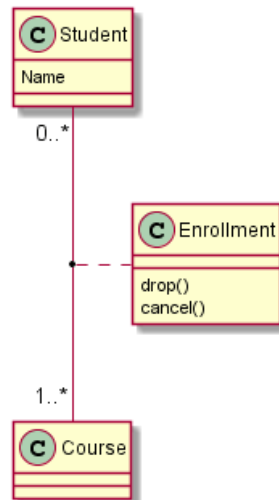
別の方向にそれを定義することができます:

```

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

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

```



3.23 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

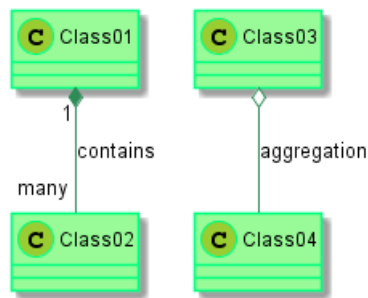
skinparam class {
    BackgroundColor PaleGreen
    ArrowColor SeaGreen
    BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

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

Class03 o-- Class04 : aggregation

@enduml

```



3.24 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。

```

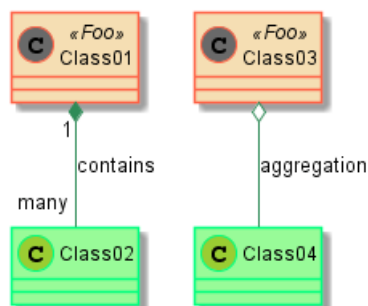
@startuml

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
  
```



3.25 色のグラデーション

表記を使用して、クラスや注釈の個々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使用することができます。

次の構文で背景に色のグラデーションをつけることもできます。2つの色の名前を次のいずれかで分割:

- |,
- /,
- \,
- or -



グラデーションの方向に依存します。

例えば、こんなふうに見えるかも：

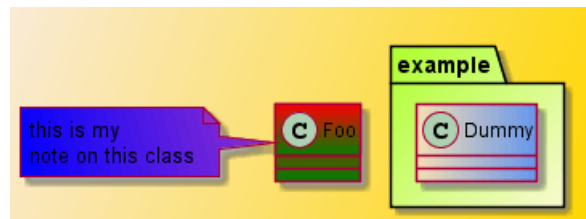
```
@startuml

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
```



3.26 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…

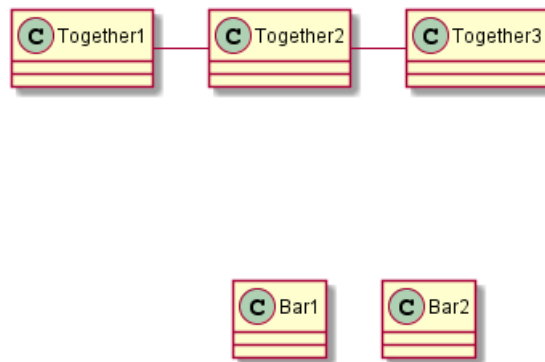
together キーワードを使って複数のクラスをグループにまとめることができます: レイアウトエンジンは、それらのクラスを（あたかも同じパッケージにあるかのように）グループにまとめようとします。

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
```



3.27 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

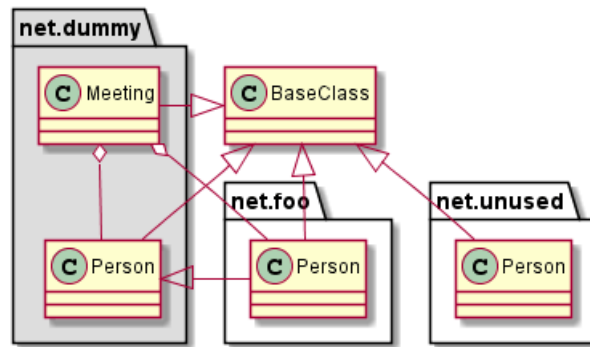
.BaseClass <|-- Meeting
}

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

  net.dummy.Meeting o-- Person
}

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

```



4 アクティビティ図

4.1 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
```



4.2 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。

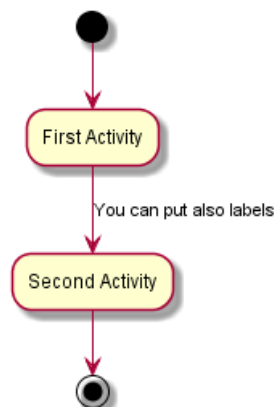
```
@startuml
```

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

```
-->[You can put also labels] "Second Activity"
```

```
--> (*)
```

```
@enduml
```



4.3 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)

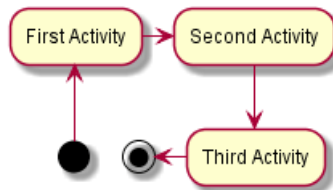


- -right-> or ->
- -left->
- -up->

```
@startuml
```

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

```
@enduml
```



4.4 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-> (*)
```

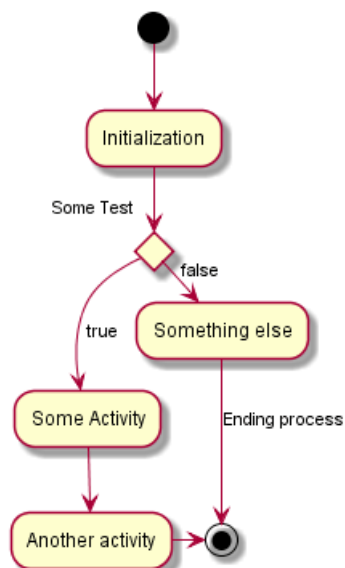
```
else
```

```
->[false] "Something else"
```

```
-->[Ending process] (*)
```

```
endif
```

```
@enduml
```



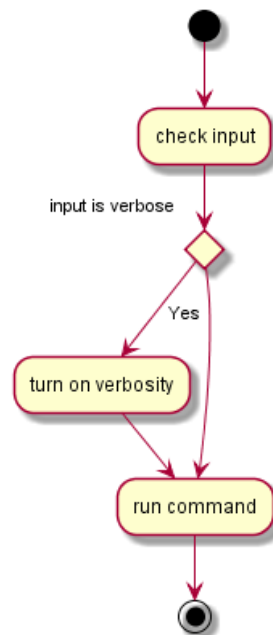
残念ながら、図のテキストで同じアクティビティを繰り返すことがあります：




```

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

```



4.5 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

else

    ->[false] "activity 2"

```



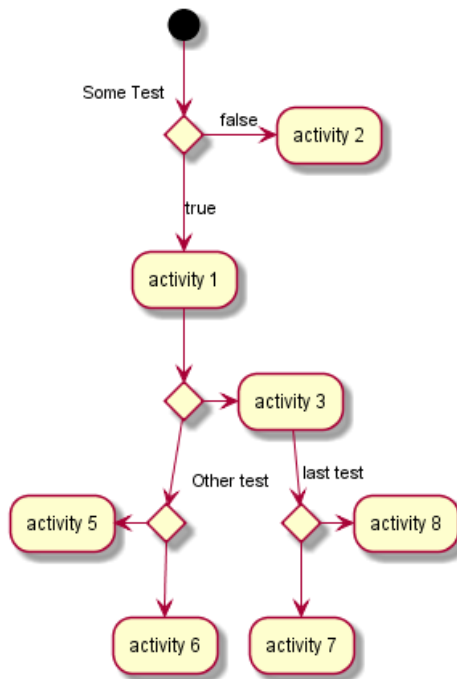
```

endif

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

@enduml

```



4.6 同期

=== code === を使用して同期バーを表示できます。

```

@startuml

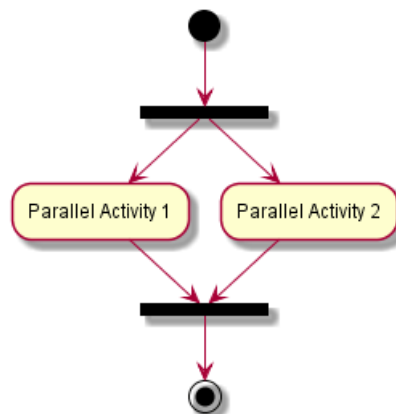
(*) --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

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

--> (*)

@enduml

```



4.7 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 `\n` を追加することもできます。

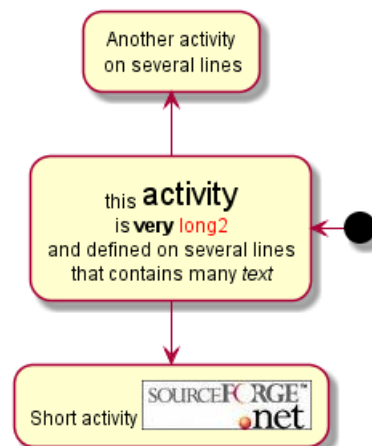
キーワード `as` を使ってアクティビティに短いコードを与えることもできます。このコードは、図の説明の後半で使用できます。

```

@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
  
```



4.8 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。

```

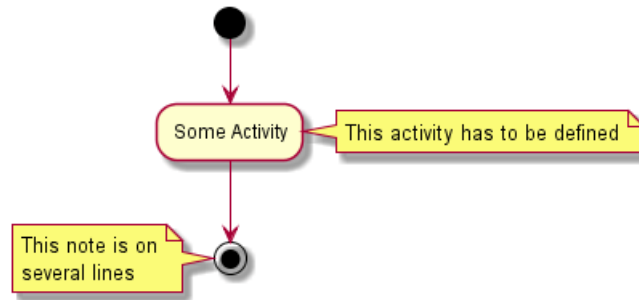
@startuml
  
```

```

(*) --> "Some Activity"
note right: This activity has to be defined
"Some Activity" --> (*)
note left
  This note is on
  several lines
end note

@enduml

```



4.9 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
}

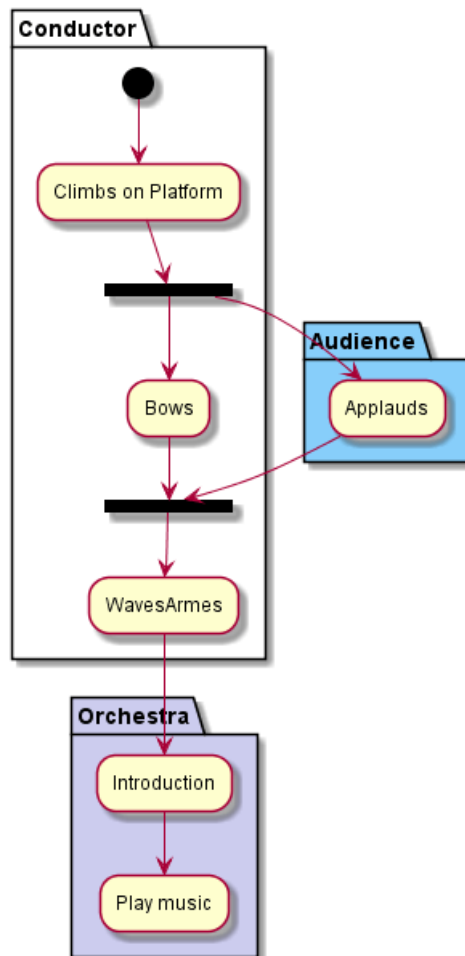
partition Audience #LightSkyBlue {
  === S1 === --> Applauds
}

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

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

@enduml

```



4.10 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
}

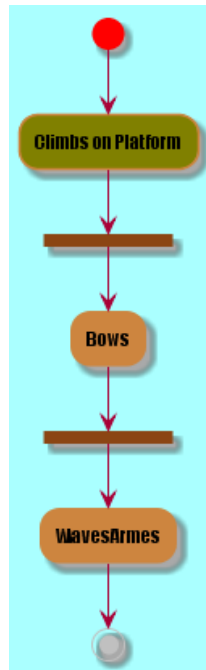
```

(*) --> "Climbs on Platform" << Begin >>



```
--> === S1 ===
--> Bows
--> === S2 ===
--> WavesArmes
--> (*)
```

```
@enduml
```



4.11 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
```



4.12 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

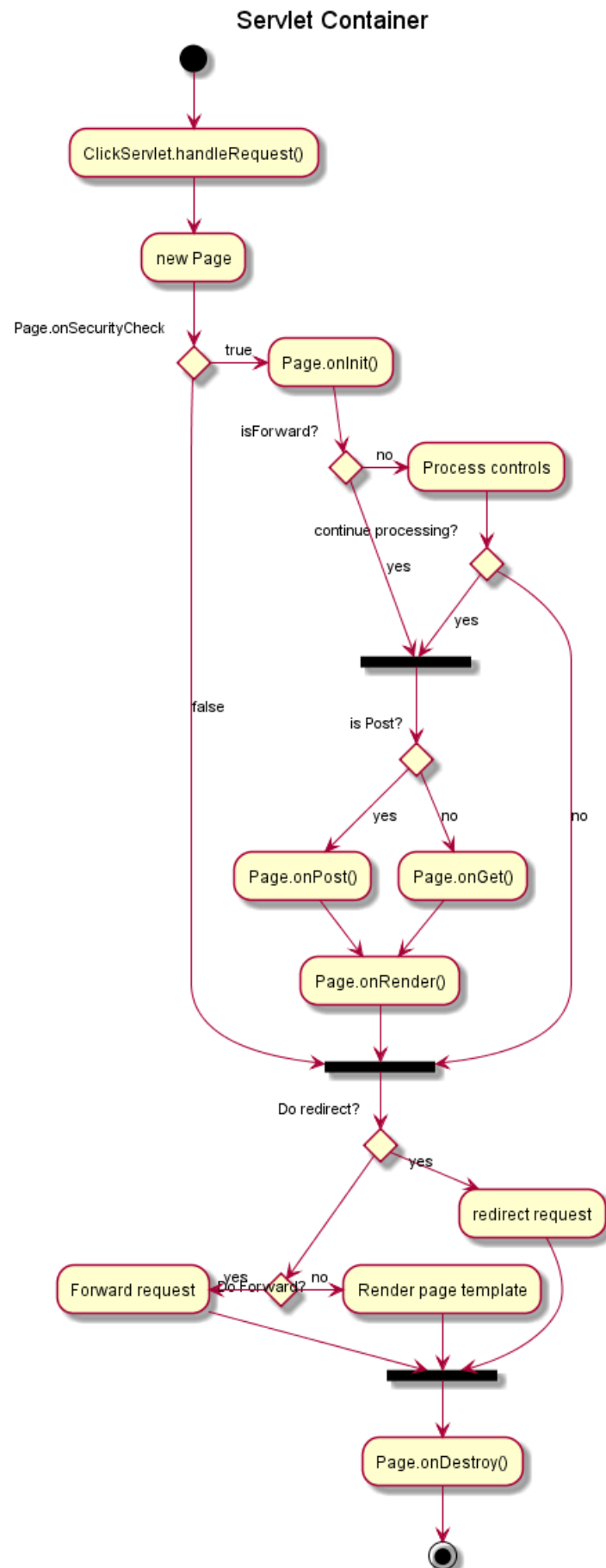
  if "is Post?" then
    -->[yes] "Page.onPost()"
    --> "Page.onRender()" as render
    --> ===REDIRECT_CHECK===
  else
    -->[no] "Page.onGet()"
    --> render
  endif

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
```



5 アクティビティ図 (ベータ版)

アクティビティ図の古い構文には、メンテナンスが難しいなど、いくつかの制限と欠点がありました。そのため、書式や構文をよりよく定義できるように、ベータ版として全く新しい構文と実装が提案されています (V7947 以降)。

この新しい実装には、(シーケンス図と同様に) Graphviz パッケージのインストールを必要としないという利点もあります。

将来的に古い構文は新しい構文に置換されるでしょう。しかし、上位互換性が確保され、古い構文もそのまま認識可能となる予定です。

新しい構文へ移行することが強く推奨されています。

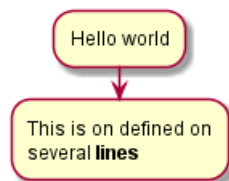
5.1 単純なアクティビティ

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。

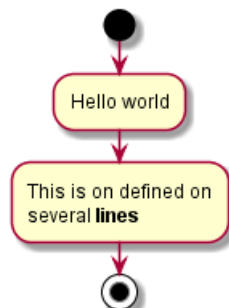
```
@startuml
:Hello world;
:This is on defined on
several **lines**;
@enduml
```



5.2 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 on defined on
several **lines**;
stop
@enduml
```

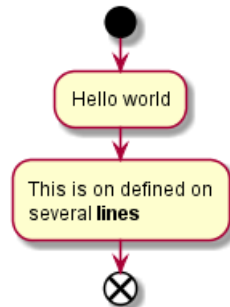


キーワード `end` もまた使用できます。

```

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

```



5.3 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。

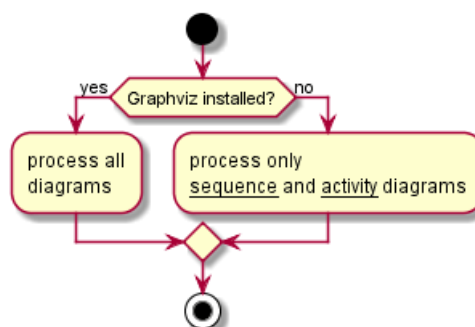
```

@startuml
start

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

stop
@enduml

```



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：

```

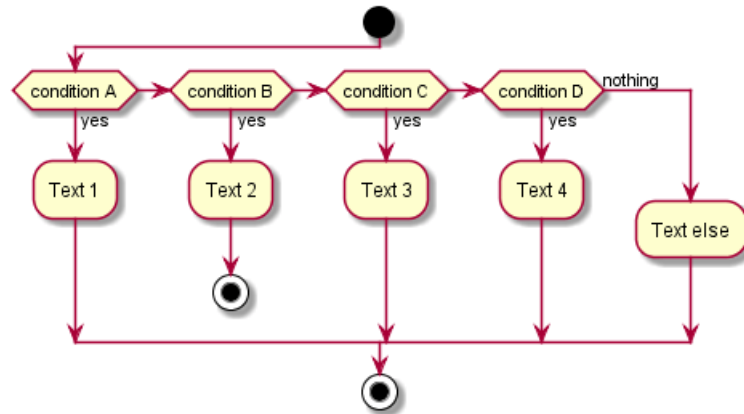
@startuml
start
if (condition A) then (yes)
    :Text 1;
elseif (condition B) then (yes)
    :Text 2;
stop

```

```

elseif (condition C) then (yes)
  :Text 3;
elseif (condition D) then (yes)
  :Text 4;
else (nothing)
  :Text else;
endif
stop
@enduml

```



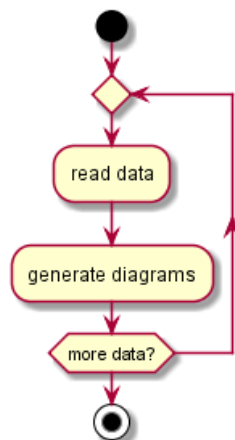
5.4 繰り返し（後判定）

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

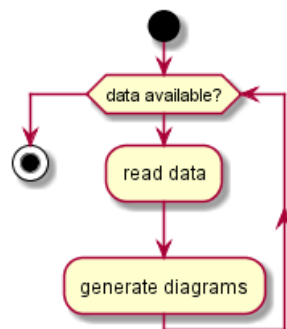
```



5.5 繰り返し（前判定）

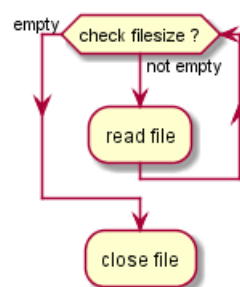
繰り返し処理（前判定）がある場合には、キーワード `while` と `end while` を使用できます。

```
@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
```



5.6 並列処理

並列処理を示すために、キーワード `fork`、`fork again` そして `end fork` が使用できます。

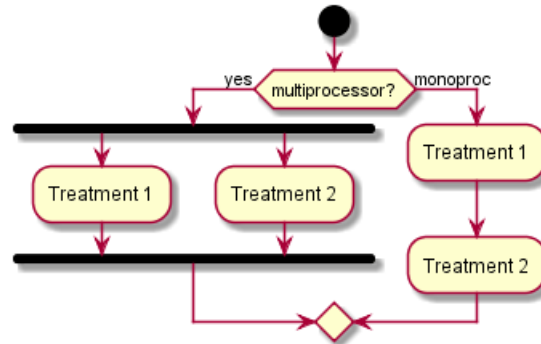
```
@startuml
start
if (multiprocessor?) then (yes)
    fork
    :Treatment 1;
end fork
```



```

fork again
:Treatment 2;
end fork
else (monoproc)
:Treatment 1;
:Treatment 2;
endif
@enduml

```



5.7 注釈

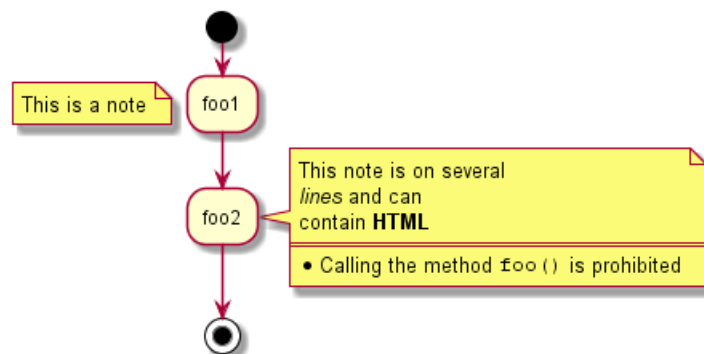
Creole 表記の Wiki 構文を使用することで、テキストの書式設定ができます。

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

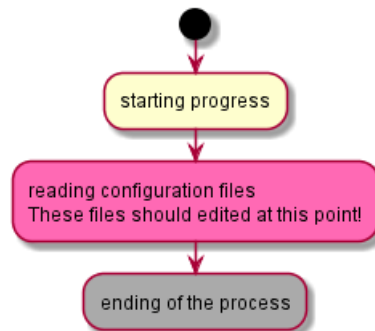
```



5.8 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
```

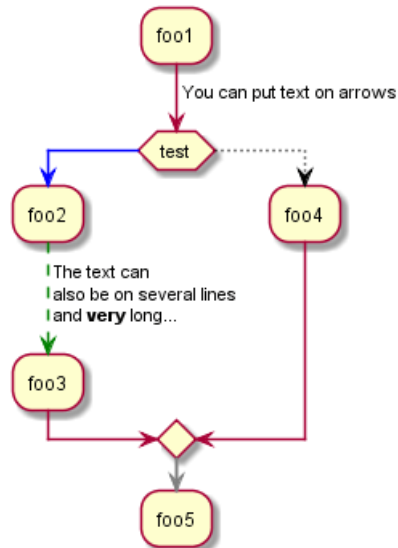


5.9 矢印

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
```

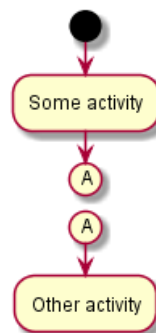


5.10 Connector

You can use parentheses to denote connector.

```

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



5.11 グループ化

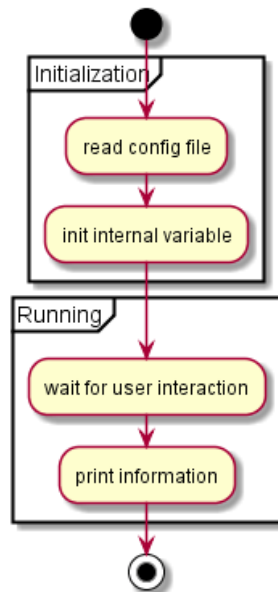
キーワード `partition` で複数のアクティビティを分割して、グループ化できます:

```

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



```
stop
@enduml
```

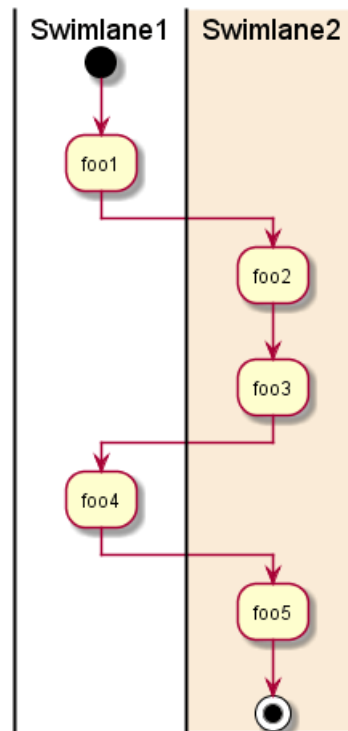


5.12 動線

パイプ記号 | を用いて、複数の動線を定義することができます。

さらに、動線毎に色を変えることができます。

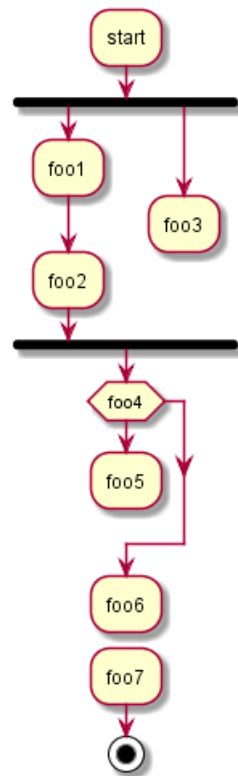
```
@startuml
|アイウエい|
start
:foo1;
|#AntiqueWhite|Swimlane2|
:foo2;
:foo3;
|Swimlane1|
:foo4;
|Swimlane2|
:foo5;
stop
@enduml
```

5.13 分離

キーワード `detach` 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
```



5.14 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:

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

```

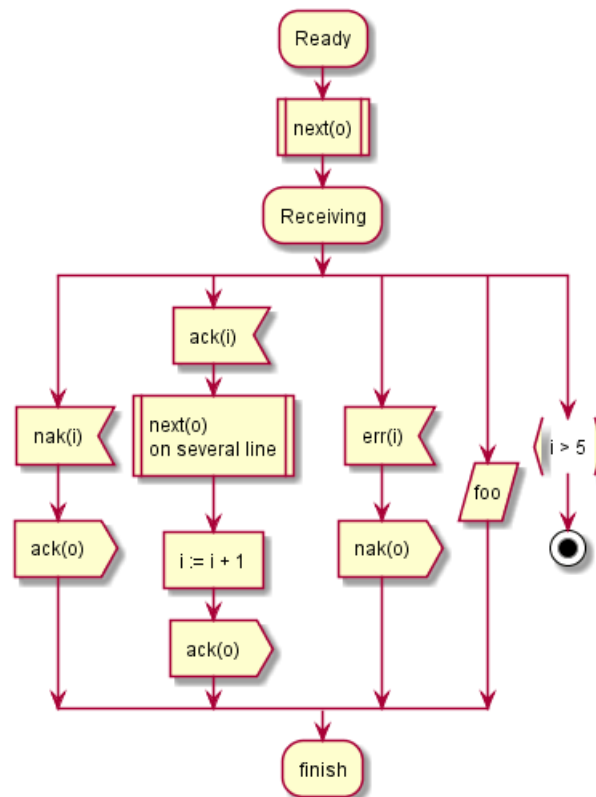
@startuml
:Ready;
:next(o)|
:Receiving;
split
:nak(i)<
:ack(o)>
split again
:ack(i)<
:next(o)
on several line|
:i := i + 1]
:ack(o)>
split again
:err(i)<
:nak(o)>
split again
:foo/
split again
:i > 5}
  
```



```

stop
end split
:finish;
@enduml

```



5.15 完全な例

```

@startuml

start
:「更新ボタン」を押す;
: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)

```



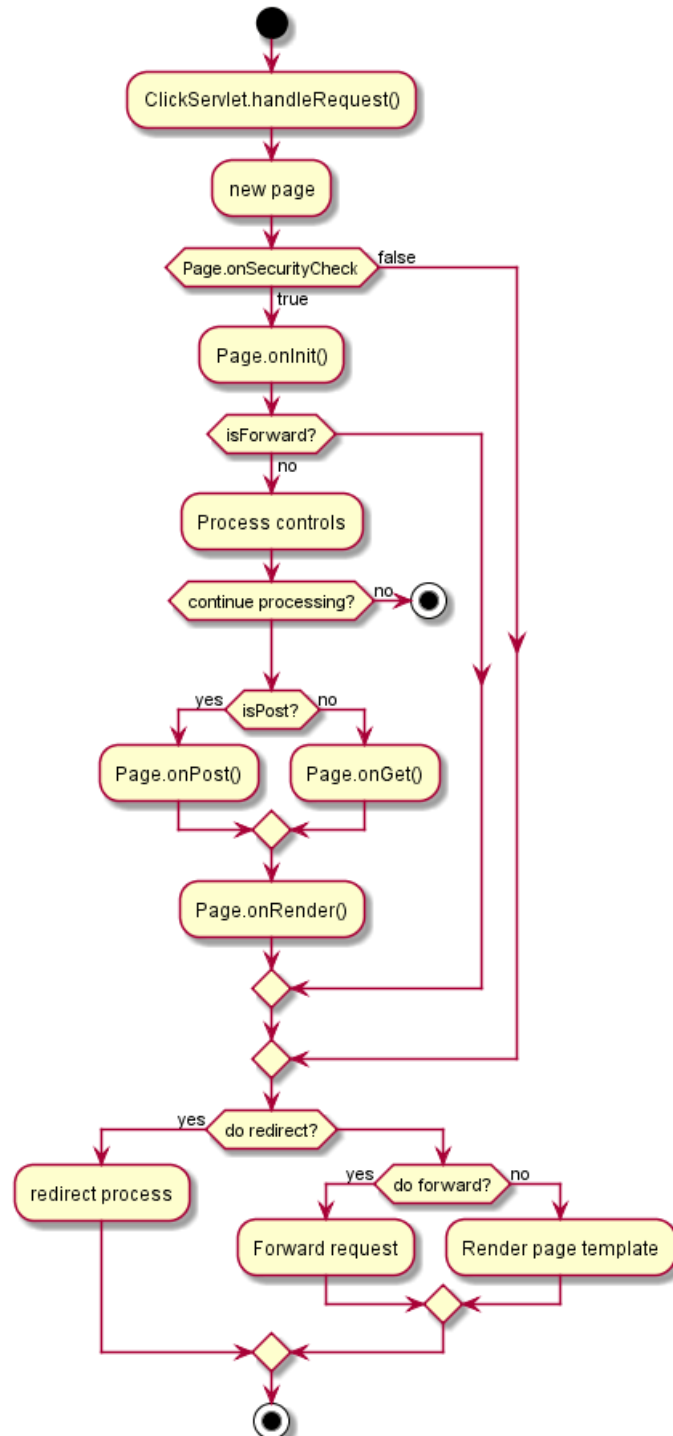
```

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

stop

@enduml

```



6 コンポーネント図

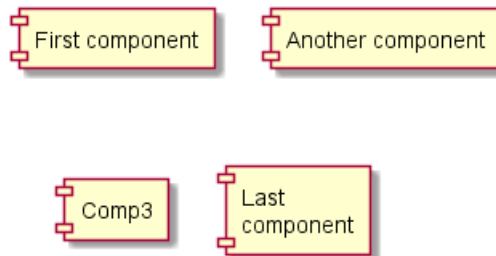
Let's have few examples :

6.1 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。そして、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
```



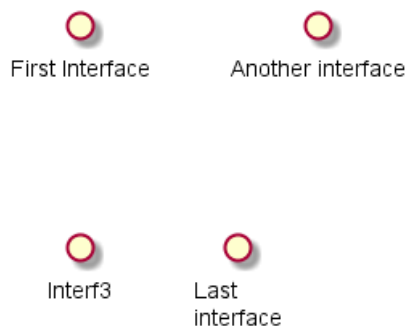
6.2 インタフェース

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
@enduml
```



6.3 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。

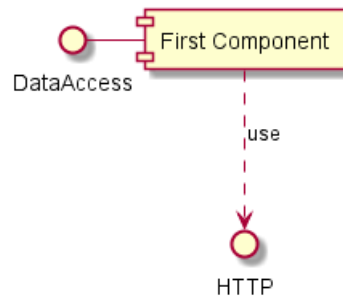
```
@startuml
```

```

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

```

```
@enduml
```



6.4 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。 `note left of`、`note right of`、`note top of`、`note bottom of`

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
```

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

```

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

```

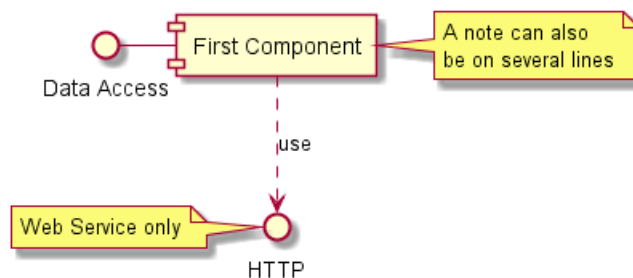
```
note left of HTTP : Web Service only
```

```

note right of [First Component]
    A note can also
    be on several lines
end note

```

```
@enduml
```



6.5 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]  
}
```

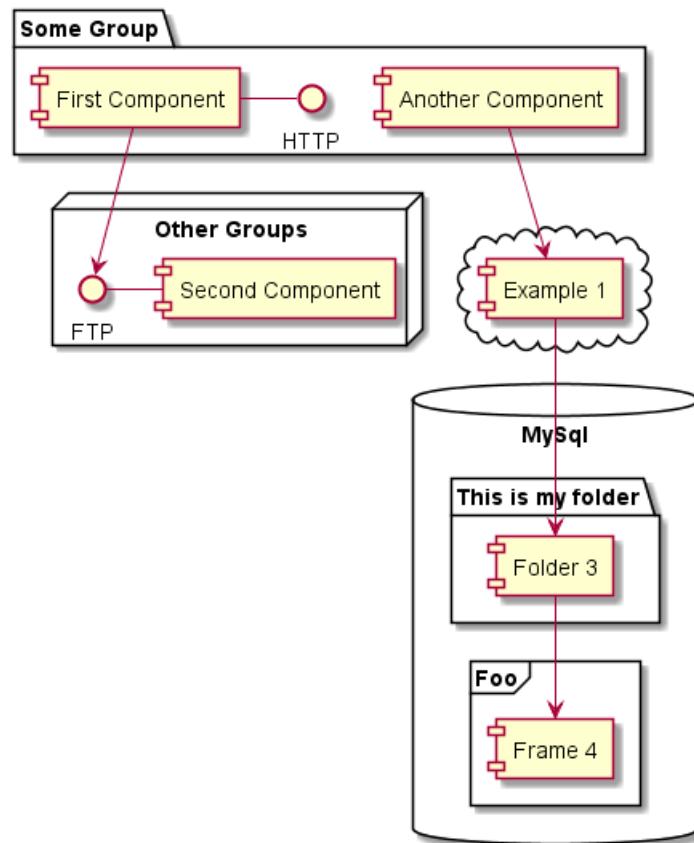
```
node "Other Groups" {  
    FTP - [Second Component]  
    [First Component] --> FTP  
}
```

```
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]
```

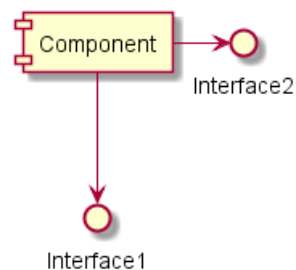
@enduml



6.6 矢印の方向を変える

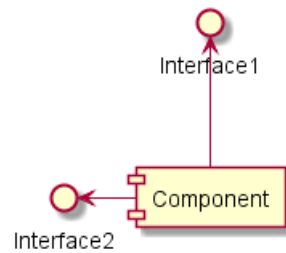
デフォルトではクラス間のリンクは2つのダッシュ--を持っており垂直方向に配向されています。次のように単一のダッシュ(またはドット)を置くことによって水平方向のリンクを使用することが可能です:

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



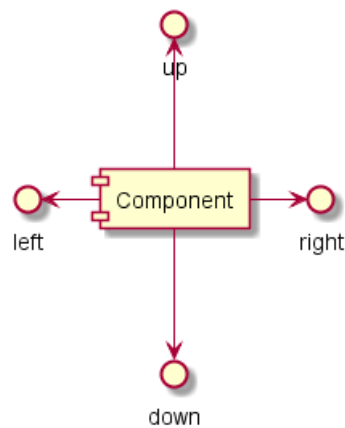
リンクを反対にすることで方向を変更することもできます。

```
@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。

6.7 UML2 表記の使用

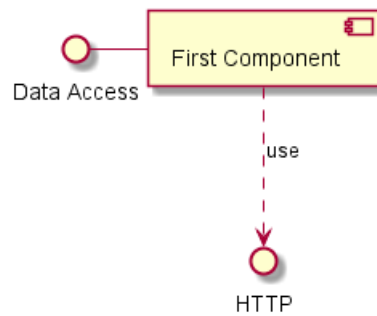
コマンド `skinparam componentStyle uml2` は、UML2 表記に切り替えるために使用されます。

```
@startuml
skinparam componentStyle uml2

interface "Data Access" as DA

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

@enduml
```

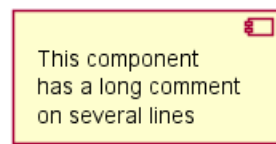


6.8 長い説明

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
  
```



6.9 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
  
```



6.10 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。

```

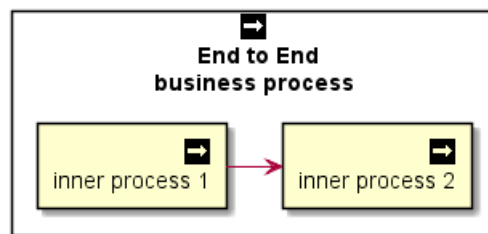
@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFF0FFFFF
FFFFFFFFF0FFFFF
FF000000000000FF
FF000000000000FF
FF000000000000FF
FFFFFFFFF0FFFFF
FFFFFFFFF0FFFFF
}
  
```

```

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

```



6.11 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>> Red
  BorderColor<<Apache>> #FF6655
  FontName Courier
  BorderColor black
  BackgroundColor gold
  ArrowFontName Impact
  ArrowColor #FF6655
  ArrowFontColor #777777
}

```

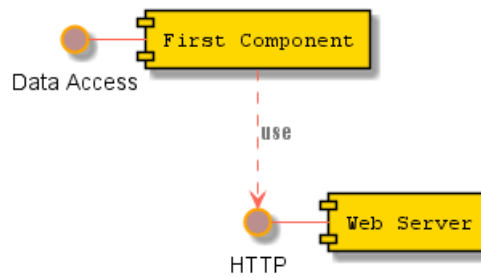
```
() "Data Access" as DA
```

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



```
[First Component] ..> () HTTP : use
HTTP - [Web Server] << Apache >>
```

```
@enduml
```



```
@startuml
```

```
[AA] <<static lib>>
```

```
[BB] <<shared lib>>
```

```
[CC] <<static lib>>
```

```
node node1
```

```
node node2 <<shared node>>
```

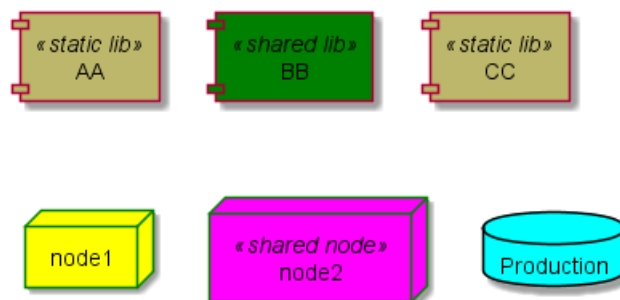
```
database Production
```

```
skinparam component {
  backgroundColor<<static lib>> DarkKhaki
  backgroundColor<<shared lib>> Green
}
```

```
skinparam node {
  borderColor Green
  backgroundColor Yellow
  backgroundColor<<shared node>> Magenta
}
```

```
skinparam databaseBackgroundColor Aqua
```

```
@enduml
```



7 ステート図

7.1 簡単なステート

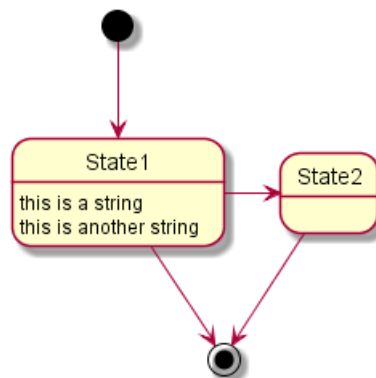
ステート図の始点と終点は、[*] で示します。

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
```



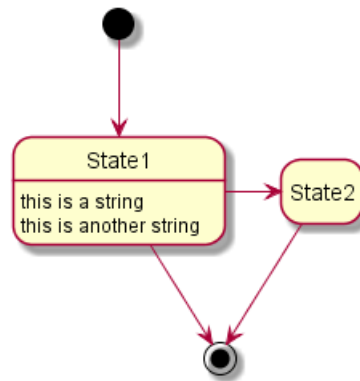
7.2 ステートの表現を変える

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
```



7.3 複合状態

状態は複合することができます。キーワード `state` と中括弧を使用して定義することができます。

```

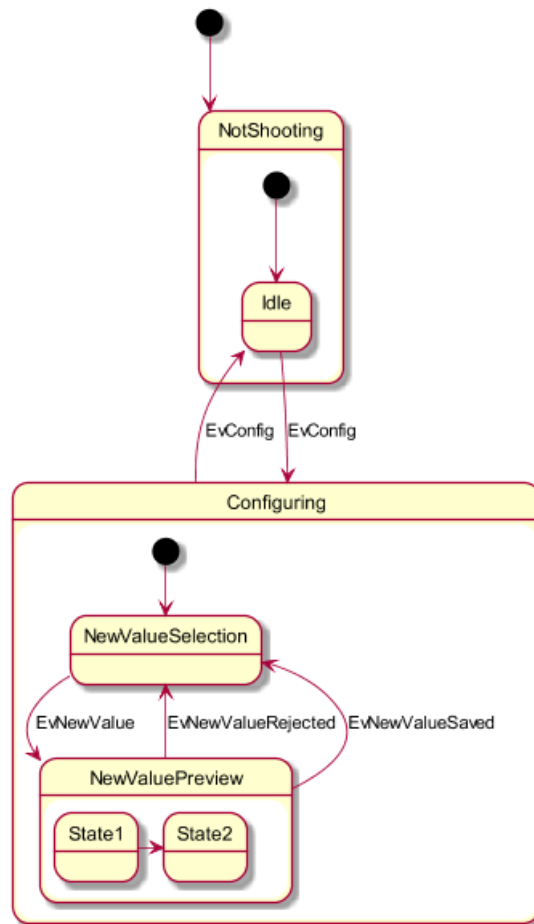
@startuml
scale 350 width
[*] --> NotShooting

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
  
```



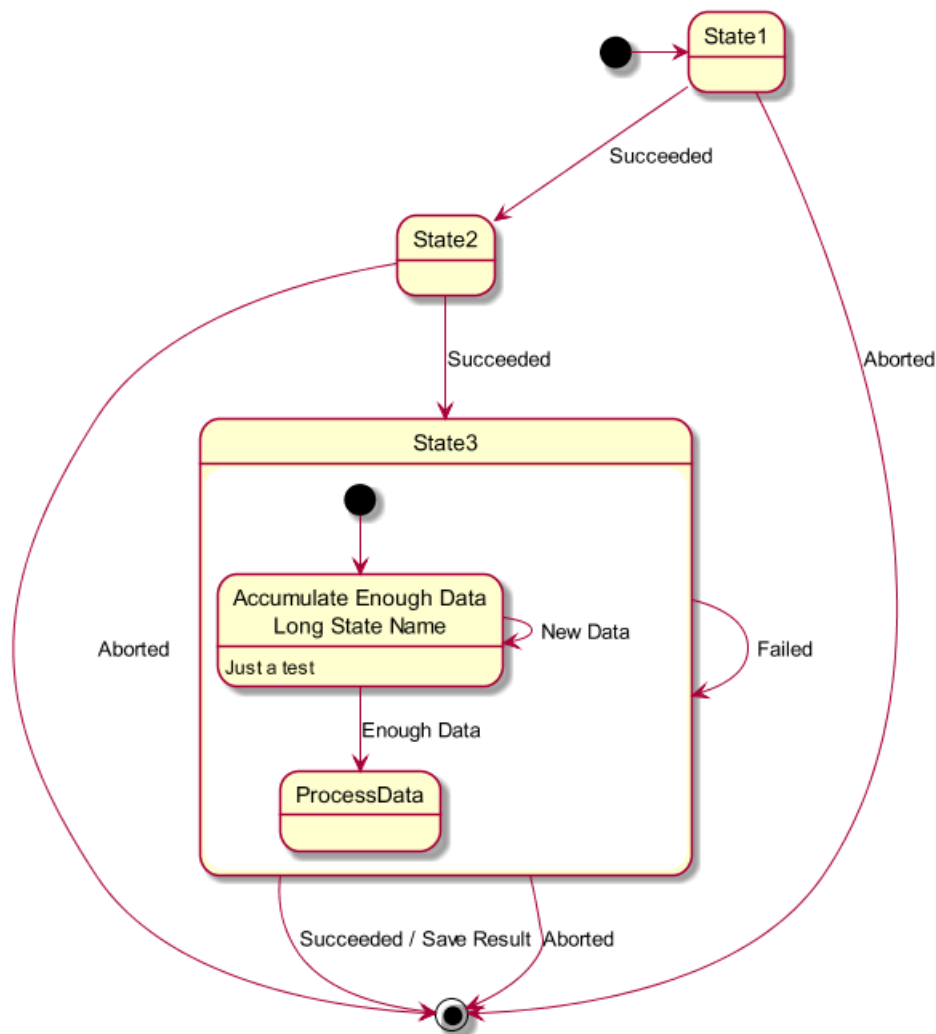
7.4 長い名前

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
```



7.5 同時状態

記号 `-- or ||` で分離することで、同時状態となる複合状態を定義することができます。

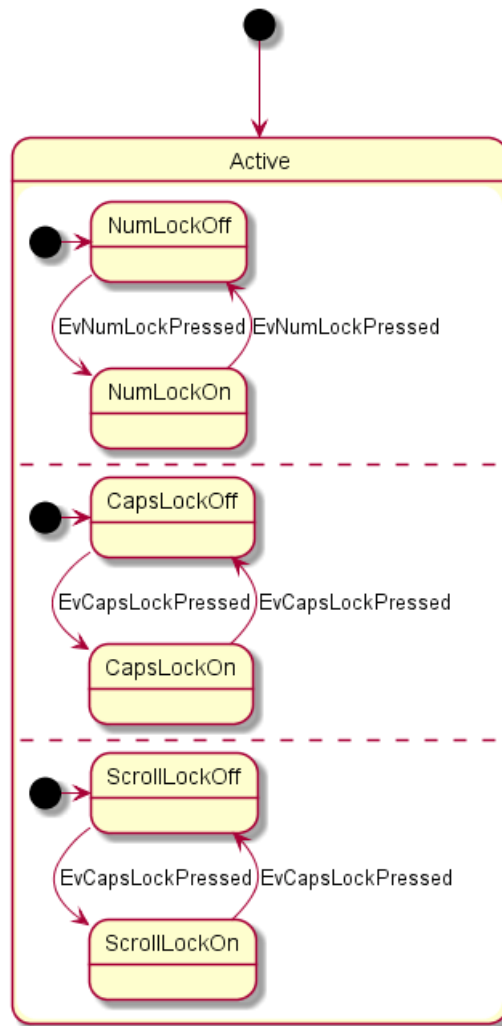
```

@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

```

7.6 矢印の方向

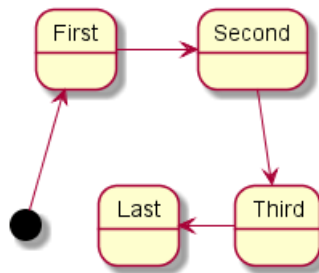
記号 `->` を水平矢印として使用でき、以下の構文を使用することで、矢印の方向を支配することができます。

- `-down->` (default arrow)
- `-right->` or `->`
- `-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。

7.7 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。

さらに、いくつもの行で注釈を定義できます。

```
@startuml
```

```
[*] --> Active
```

```
Active --> Inactive
```

```
note left of Active : this is a short\nnote
```

```
note right of Inactive
```

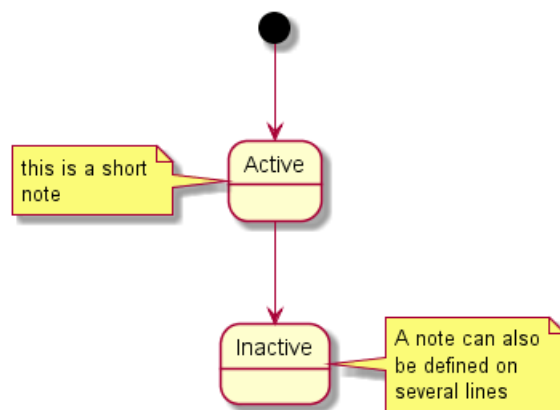
```
  A note can also
```

```
  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
```





7.8 もっと注釈

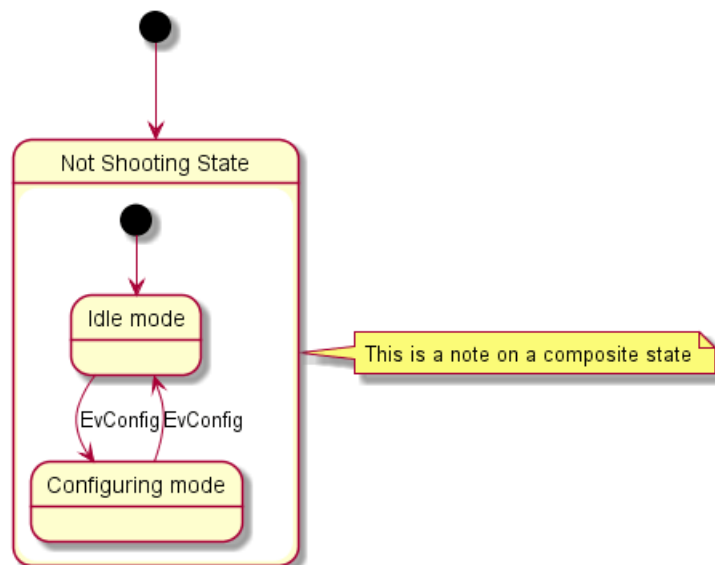
複合状態にも注釈をつけることができます。

```
@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
```



7.9 見栄え

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。

定型化した状態に、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

```

```

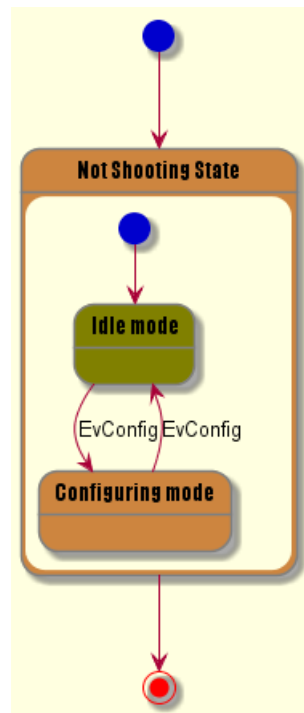
state "Not Shooting State" as NotShooting {
    state "Idle mode" as Idle <<Warning>>
    state "Configuring mode" as Configuring
    [*] --> Idle
    Idle --> Configuring : EvConfig
    Configuring --> Idle : EvConfig
}

```

```

NotShooting --> [*]
@enduml

```

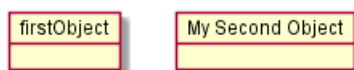


8 オブジェクト図

8.1 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
```



8.2 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。

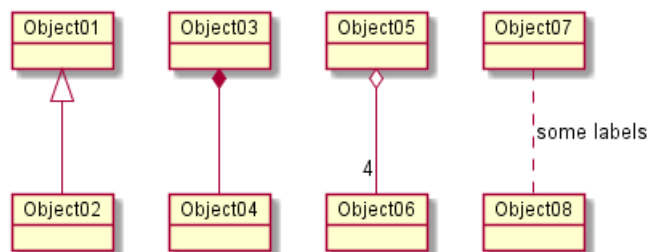
これらのルールを知ることで、以下の図を描くことができます。

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

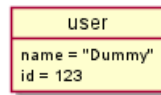
Object01 <|-- Object02
Object03 *-- Object04
Object05 o-- "4" Object06
Object07 .. Object08 : some labels
@enduml
```



8.3 フィールドの追加

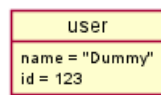
フィールドを宣言するには、シンボル : にフィールド名を続けます。

```
@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
```



8.4 クラス図と共通の機能

- 可視性
- 注釈の定義
- パッケージの使用
- 出力スキン

9 タイミング図

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！！

9.1 ライフライン

ライフラインは、**concise** か **robust** で定義できます。**concise** は状態ライフラインを、**robust** は汎用値ライフラインを、それぞれ作成します。

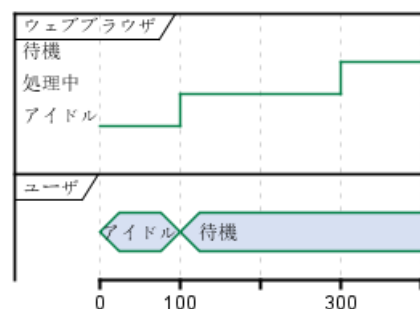
@と is を用いて、状態の変化を記述できます。

```
@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
```



9.2 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
```

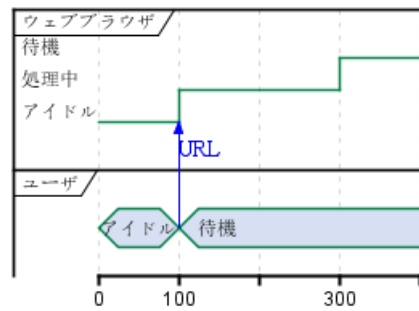
```
@0
WU is アイドル
WB is アイドル
```

```
@100
WU -> WB : URL
WU is 待機
WB is 処理中
```

```
@300
WB is 待機
```



@enduml



9.3 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
```

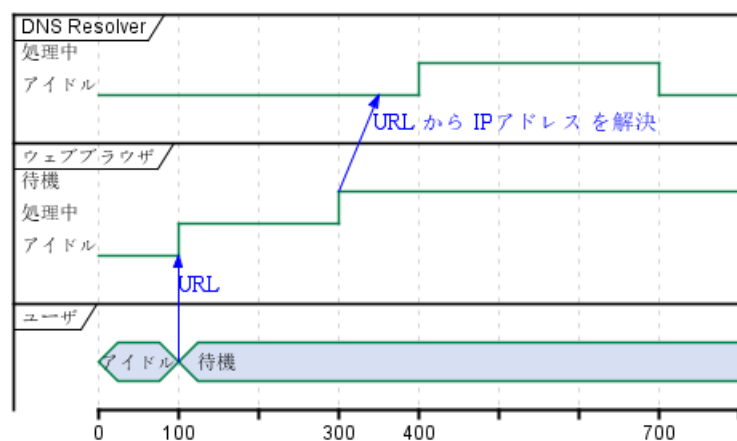
```
@0
WU is アイドル
WB is アイドル
DNS is アイドル
```

```
@+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
```



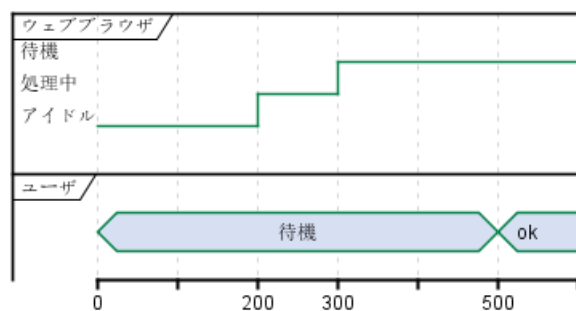
9.4 インスタンス指向

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
```

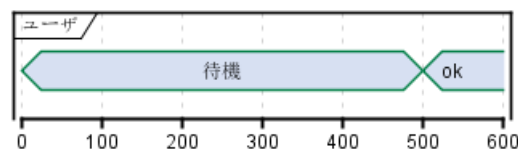


9.5 スケールの設定

スケール（メモリの数値の表示）を指定できます。以下の例では、「メモリを 100 ずつ表示、1 メモリの幅を 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
```



9.6 初期状態

「初期状態」を設定できます。

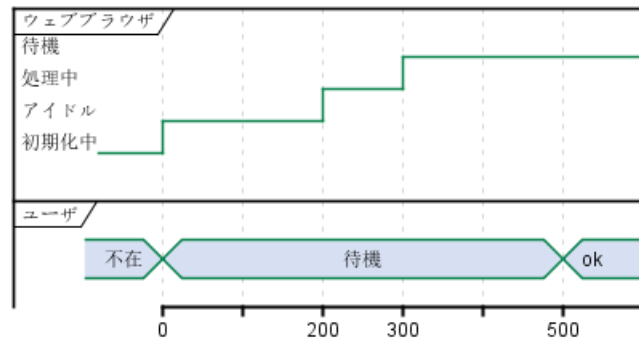
```
@startuml
robust "ウェブブラウザ" as WB
concise "ユーザ" as WU
```

```
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
```

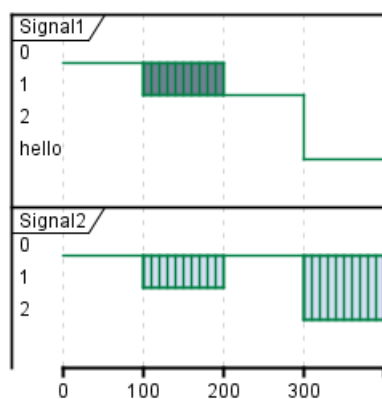
```
@WU
0 is 待機
+500 is ok
@enduml
```



9.7 Intricated state

A signal could be in some undefined state.

```
@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
```



9.8 Hidden state

It is also possible to hide some state.

```
@startuml
concise "Web User" as WU
```

```
@0
WU is {-}
```

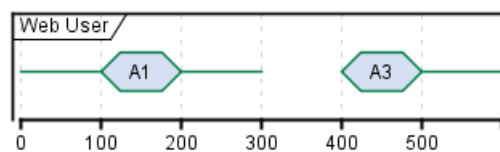
```
@100
WU is A1
```

```
@200
WU is {-}
```

```
@300
WU is {hidden}
```

```
@400
WU is A3
```

```
@500
WU is {-}
@enduml
```



9.9 時間定規（time constraint）の追加

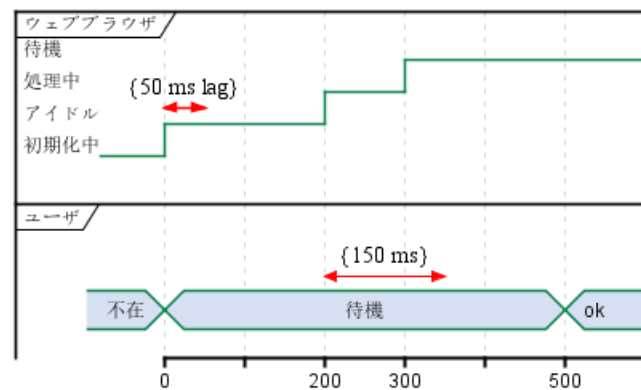
タイムラインのメモリとは別に、時間の尺度を示す矢印を表示することができます。

```
@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
```



9.10 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。

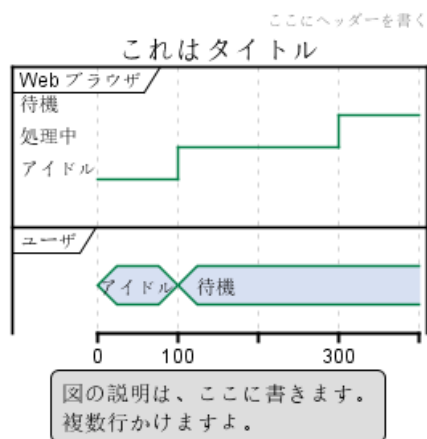
```
@startuml
Title これはタイトル
header: ここにヘッダーを書く
footer: ここにフッターを書く
legend
図の説明は、ここに書きます。
複数行かけますよ。
end legend
caption 一行の説明は、caption に書きましょう。
```

```
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 ガントチャート

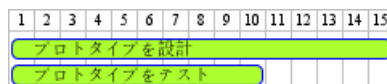
現在、このダイアグラムは提案段階です。将来的に変更されるかもしれません。

ガントチャートは、「主語と動詞の組み合わせ」という、単純で自然な英語を書くように記述できます。

10.1 タスクの定義

タスクは角カッコで定義します。期間は `lasts` で指定します。

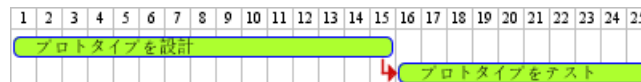
```
@startgantt
[プロトタイプを設計] lasts 15 days
[プロトタイプをテスト] lasts 10 days
@endgantt
```



10.2 依存関係

`start` と `end` で、タスク間の依存関係を定義します。

```
@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
[プロトタイプをテスト] starts at [プロトタイプを設計]'s end
@endgantt
```



複数のタスク同士をつなぐこともできます。

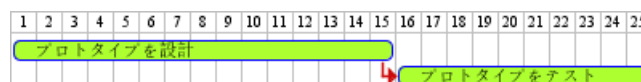
```
@startgantt
[プロトタイプを設計] lasts 10 days
[プロトタイプを実装] lasts 10 days
[テストを実装] lasts 5 days
[プロトタイプを実装] starts at [プロトタイプを設計]'s end
[テストを実装] starts at [プロトタイプを実装]'s start
@endgantt
```



10.3 エイリアス

`as` で、タスクに短い名前（エイリアス）を定義できます。

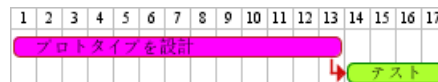
```
@startgantt
[プロトタイプを設計] as [設計] lasts 15 days
[プロトタイプをテスト] as [テスト] lasts 10 days
[テスト] starts at [設計]'s end
@endgantt
```



10.4 色の変更

colored in で、色の変更ができます。色の指定には、HTML のカラーネームを使用します。

```
@startgantt
[プロトタイプを設計] lasts 13 days
[テスト] lasts 4 days
[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
[テスト] is colored in GreenYellow/Green
@endgantt
```



10.5 マイルストーン

happens でマイルストーンを定義できます。

```
@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
```



10.6 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
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
```



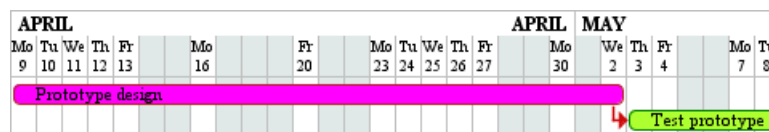
10.7 休業日

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
[Prototype design] lasts 14 days
[Test prototype] lasts 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
```



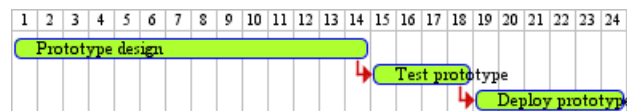
[Test prototype] is colored in GreenYellow/Green
 @endgantt



10.8 Simplified task succession

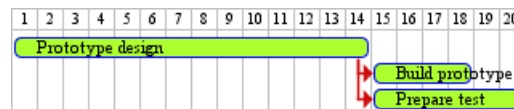
It's possible to use the then keyword to denote consecutive tasks.

```
@startgantt
[Prototype design] lasts 14 days
then [Test prototype] lasts 4 days
then [Deploy prototype] lasts 6 days
@endgantt
```



You can also use arrow ->

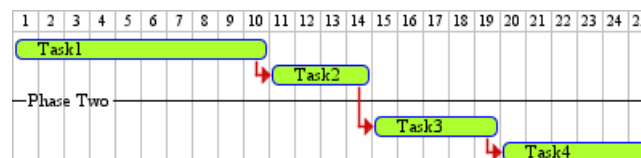
```
@startgantt
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@endgantt
```



10.9 Separator

You can use -- to group tasks together.

```
@startgantt
[Task1] lasts 10 days
then [Task2] lasts 4 days
-- Phase Two --
then [Task3] lasts 5 days
then [Task4] lasts 6 days
@endgantt
```



10.10 Working with resources

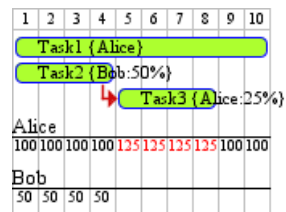
You can affect tasks on resources using the on keyword and brackets for resource name.




```

@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
then [Task3] on {Alice:25%} lasts 1 days
@endgantt

```



10.11 複雑な例

1つのタスクに対して、**and** で同時に複数の設定できます。

依存する前提タスクを、後から追加することができます。

```

@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]'s end
@endgantt

```



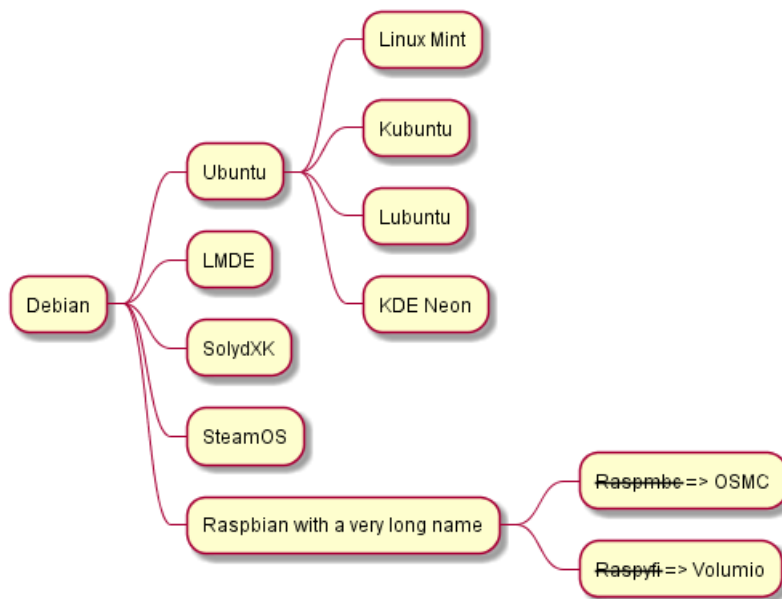
11 MindMap

MindMap diagram are still in beta: the syntax may change without notice.

11.1 OrgMode syntax

This syntax is compatible with OrgMode

```
@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
```

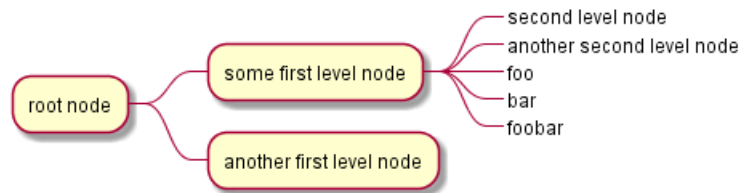


11.2 Removing box

You can remove the box drawing using an underscore.

```
@startmindmap
* root node
** some first level node
***_ second level node
***_ another second level node
***_ foo
***_ bar
***_ foobar
** another first level node
@endmindmap
```



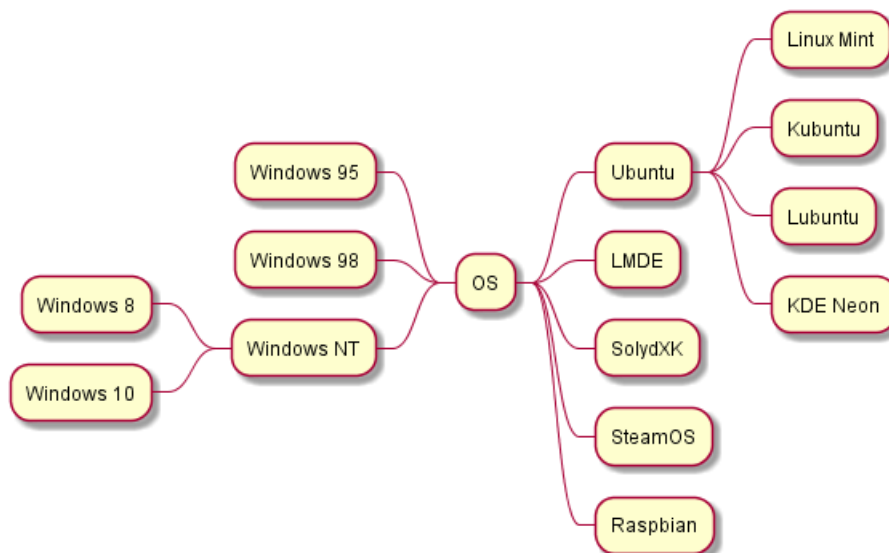


11.3 Arithmetic notation

You can use the following notation to choose diagram side.

```

@startmindmap
+ OS
++ Ubuntu
+++ Linux Mint
+++ Kubuntu
+++ Lubuntu
+++ KDE Neon
++ LMDE
++ SolydXK
++ SteamOS
++ Raspbian
-- Windows 95
-- Windows 98
-- Windows NT
--- Windows 8
--- Windows 10
@endmindmap
  
```



11.4 Markdown syntax

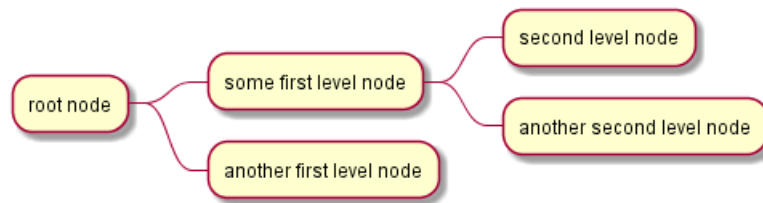
This syntax is compatible with Markdown

```

@startmindmap
* root node
* some first level node
* second level node
* another second level node
  
```



```
* another first level node
@endmindmap
```



11.5 Changing diagram direction

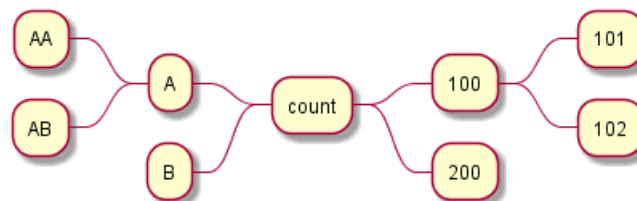
It is possible to use both sides of the diagram.

```
@startmindmap
```

```
* count
** 100
*** 101
*** 102
** 200
```

```
left side
```

```
** A
*** AA
*** AB
** B
@endmindmap
```



11.6 Complete example

```
@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

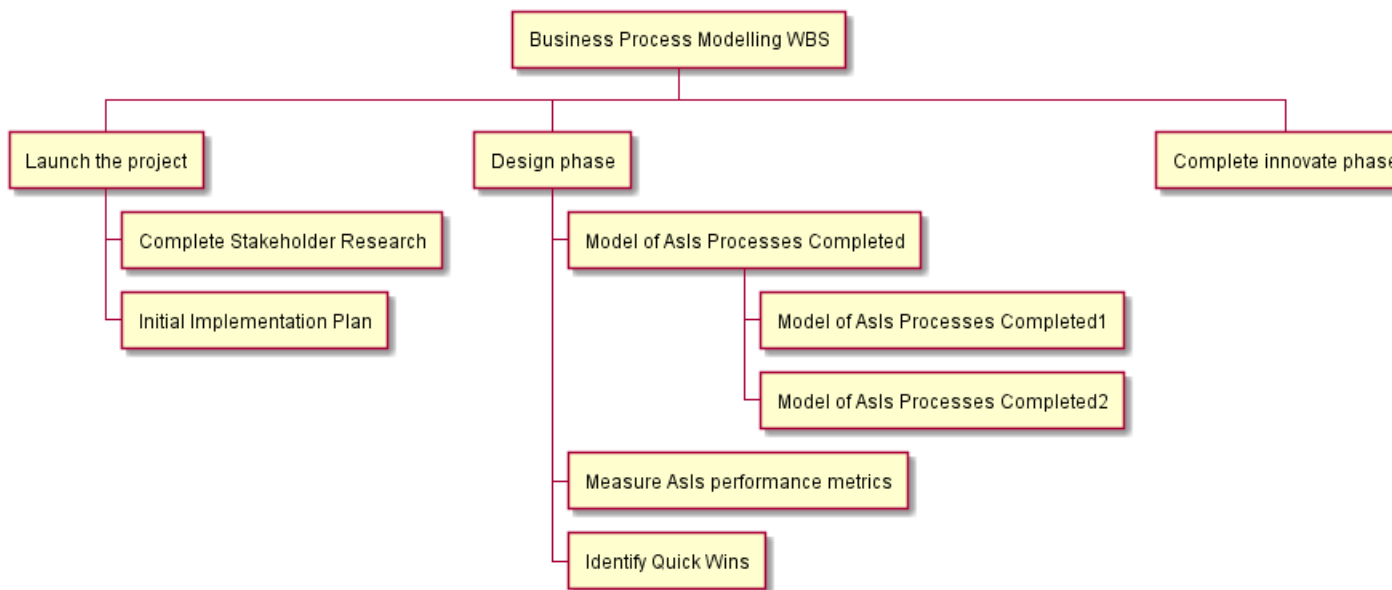
12 Work Breakdown Structure

WBS diagram are still in beta: the syntax may change without notice.

12.1 OrgMode syntax

This syntax is compatible with OrgMode

```
@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
```



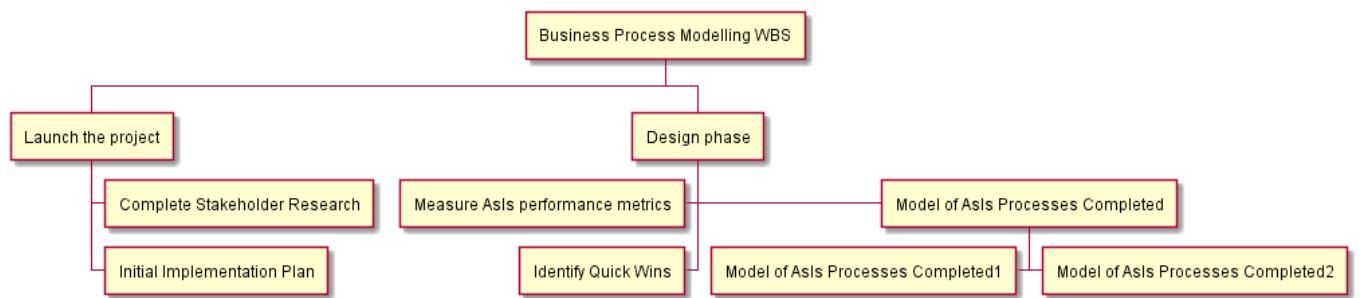
12.2 Change direction

You can change direction using < and >

```
@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

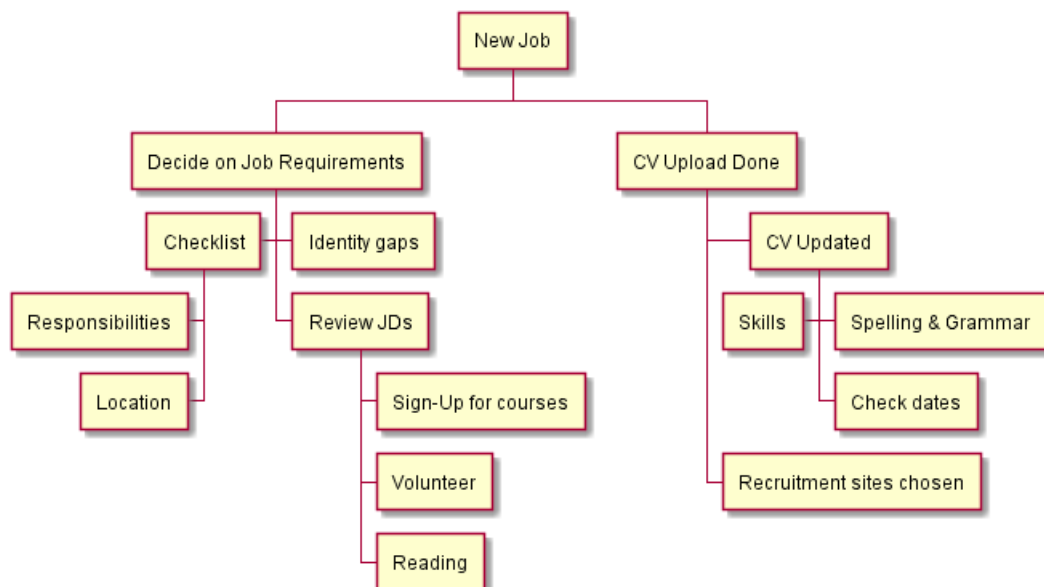


12.3 Arithmetic notation

You can use the following notation to choose diagram side.

```

@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
  
```



You can use underscore _ to remove box drawing.

```

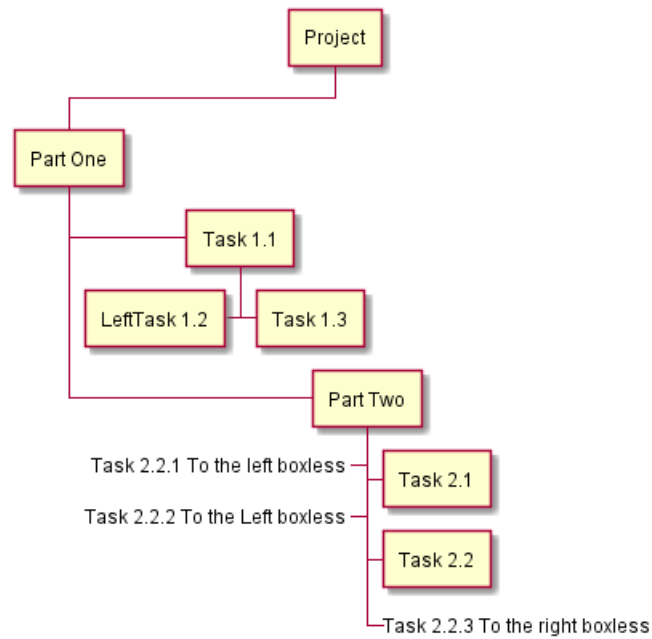
@startwbs
+ Project
  
```



```

+ 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

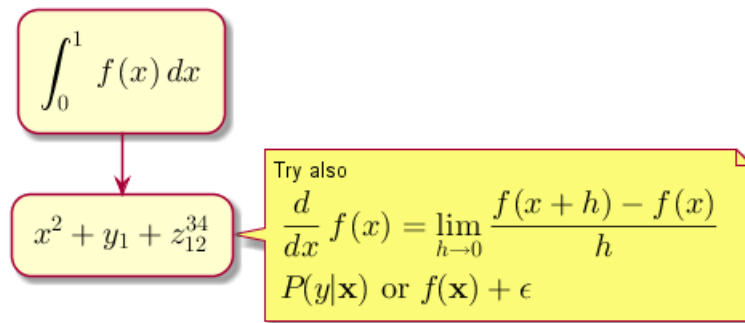
```



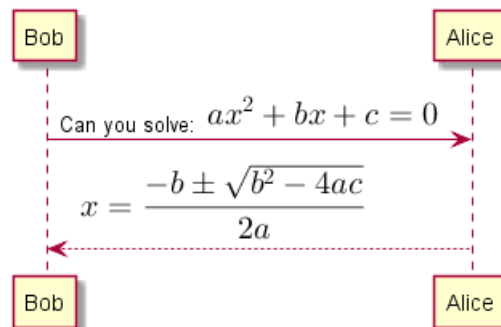
13 イントロダクション

PlantUML では、AsciiMath や JLaTeXMath 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} \frac{f(x+h) - f(x)}{h}</math>
<latex>P(y|\mathbf{x}) \ \mbox{ or } f(\mathbf{x}) + \epsilon</latex>
end note
@enduml
```



```
@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
```



13.1 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
```



$$\sum_{i=0}^{n-1} (a_i + b_i^2)$$

13.2 どのように処理しているのか

これらの式を表示するのに、PlantUML では 2 つのオープンソースプロジェクトを使用することができます。

- AsciiMath : AsciiMath を LaTeX へ変換。
- JLatexMath LaTeX でかかれた式の表示。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 PlantUML の標準ディストリビューションで使用する上で十分に小さいです。(訳者注: そのため、別途インストールする必要はありません。)

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。



14 Common commands

14.1 Comments

Everything that starts with `simple quote ' is a comment.`

You can also put comments on several lines using `/' to start and ' / to end.`

14.2 Footer and header

You can use the commands `header` or `footer` to add a footer or a header on any generated diagram.

You can optionally specify if you want a `center`, `left` or `right` footer/header, by adding a keyword.

As for title, it is possible to define a header or a footer on several lines.

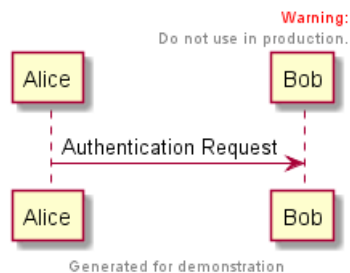
It is also possible to put some HTML into the header or footer.

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

```
header
<font color=red>Warning:</font>
Do not use in production.
endheader
```

```
center footer Generated for demonstration
```

```
@enduml
```



14.3 Zoom

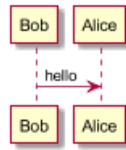
You can use the `scale` command to zoom the generated image.

You can use either a number or a fraction to define the scale factor. You can also specify either width or height (in pixel). And you can also give both width and height : the image is scaled to fit inside the specified dimension.

- `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
```



14.4 Title

The title keywords is used to put a title. You can add newline using \n in the title description.

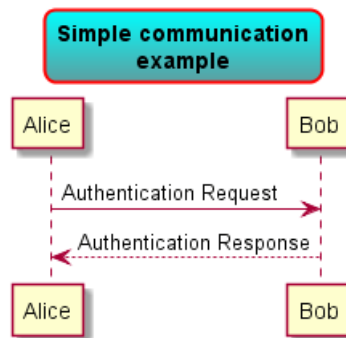
Some skinparam settings are available to put borders on the title.

```
@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
```



You can use creole formatting in the title.

You can also define title on several lines using title and end title keywords.

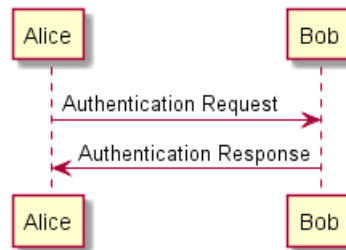
```
@startuml

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**



14.5 Caption

There is also a caption keyword to put a caption under the diagram.

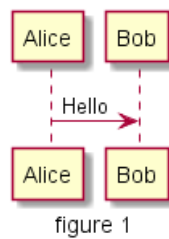
```

@startuml

caption figure 1
Alice -> Bob: Hello

@enduml

```



14.6 Legend the diagram

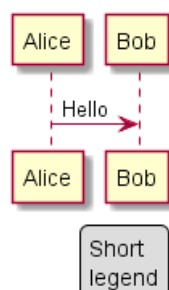
The legend and end legend are keywords is used to put a legend.

You can optionally specify to have left, right, top, bottom or center alignment for the legend.

```

@startuml
Alice -> Bob : Hello
legend right
  Short
  legend
endlegend
@enduml

```



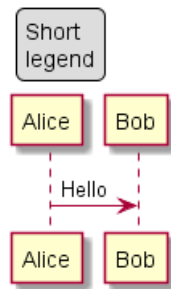
```

@startuml
Alice -> Bob : Hello
legend top left
  Short

```



```
legend
endlegend
@enduml
```



15 Salt (GUI 設計ツール)

Salt はグラフィカルインタフェースの設計を助ける PlantUML のサブプロジェクトです。

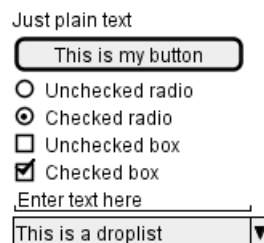
キーワード `@startsalt`、または、`@startuml` と次の行に続くキーワード `salt` の、いずれかを使用することができます。

15.1 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。

```
@startuml
salt
{
  Just plain text
  [This is my button]
  () Unchecked radio
  (X) Checked radio
  [] Unchecked box
  [X] Checked box
  "Enter text here  "
  ^This is a droplist^
}
@enduml
```



このツールの目標は簡単な見本でウィンドウについて議論することです。

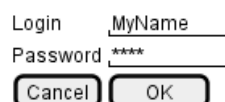
15.2 罫線の使用

表は括弧 `{` で開始すれば自動的に作成されます。

そして `|` で列を分割する必要があります。

例:

```
@startsalt
{
  Login      | "MyName  "
  Password   | "****   "
  [Cancel]   | [ OK   ]
}
@sendsalt
```



括弧で表を開始したら、行や列の罫線を表示したいがために定義された文字を使用することができます:

Symbol	Result
#	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
+	枠線を表示する

```
@startsalt
{+
    Login    | "MyName  "
    Password | "****   "
    [Cancel] | [ OK   ]
}
@endsalt
```

15.3 Group box

more info

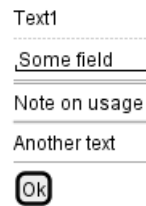
```
@startsalt
{~"My group box"
    Login    | "MyName  "
    Password | "****   "
    [Cancel] | [ OK   ]
}
@endsalt
```

15.4 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
```





15.5 木構造ウィジェット

木構造があるなら、{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。

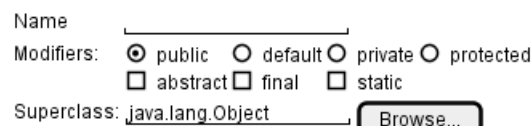
```
@startsalt
{
{T
+ World
++ America
+++ Canada
+++ USA
++++ New York
++++ Boston
+++ Mexico
++ Europe
+++ Italy
+++ Germany
++++ Berlin
++ Africa
}
}
@endsalt
```



15.6 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。

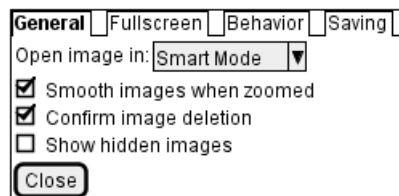
```
@startsalt
{
Name          | "          "
Modifiers:    | { (X) public | () default | () private | () protected
              | [] abstract | [] final   | [] static }
Superclass:   | { "java.lang.Object " | [Browse...] }
}
@endsalt
```



15.7 タブの追加

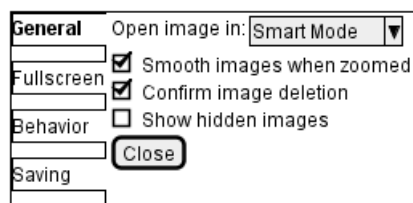
{/ 表記を使用してタブを追加することができます。太字のテキストを設けるように、HTML コードを使用できることに注意してください。

```
@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
```



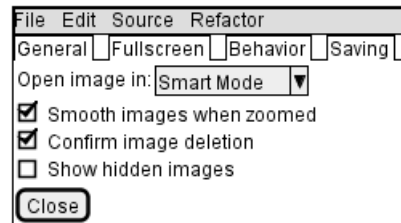
15.8 メニューの使用

{* 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^ }
```

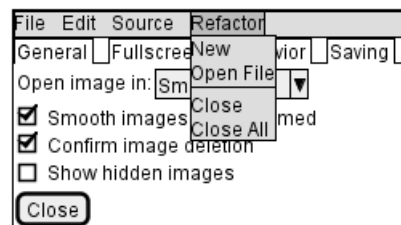


```
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



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
```



15.9 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。

- * は残りのセルとの範囲を示すために
- . は空のセルを示すために

```
@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	



15.10 OpenIconic

OpenIconic is an very nice open source icon set. Those icons have been integrated into the creole parser, so you can use them out-of-the-box.

You can use the following syntax: `<&ICON_NAME>`.

```
@startsalt
{
  Login<&person> | "MyName"
  Password<&key> | "****"
  [Cancel <&circle-x>] | [OK <&account-login>]
}
@endsalt
```



The complete list is available on OpenIconic Website, or you can use the following special diagram:

```
@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	random	vertical-align-center
arrow-thick-right	caret-left	data-transfer-upload	graph	map	reload	vertical-align-top
arrow-thick-top	caret-right	delete	grid-four-up	media-pause	resize-both	video
arrow-top	caret-top	dial	grid-three-up	media-play	resize-height	volume-high
audio-spectrum	cart	document	grid-two-up	media-record	resize-width	volume-low
audio	chat	dollar	hard-drive	media-skip-backward	rss-alt	volume-off
badge	check	double-quote-sans-left	header	media-skip-forward	rss	warning
ban	chevron-bottom	double-quote-sans-right	headphones	media-step-backward	script	wifi
bar-chart	chevron-left	double-quote-serif-left	heart	media-step-forward	share-boxed	wrench
basket	chevron-right	double-quote-serif-right	home	media-stop	share	x
battery-empty	chevron-top	droplet	image	medical-cross	shield	yen
battery-full	circle-check	eject	inbox	menu	signal	zoom-in
beaker	circle-x	elevator	infinity	microphone	signpost	zoom-out
	clipboard	ellipses	info	minus	sort-ascending	
	clock	envelope-closed	italic	monitor	sort-descending	
	cloud-download	envelope-open	justify-center	moon	spreadsheet	
	cloud-upload	euro	justify-left	move		

15.11 Include Salt

see: <http://forum.plantuml.net/2427/salt-with-minimum-flowchat-capabilities?show=2427#q2427>

```
@startuml
(*) --> "
{{
salt
{+
<b>an example
choose one option
```



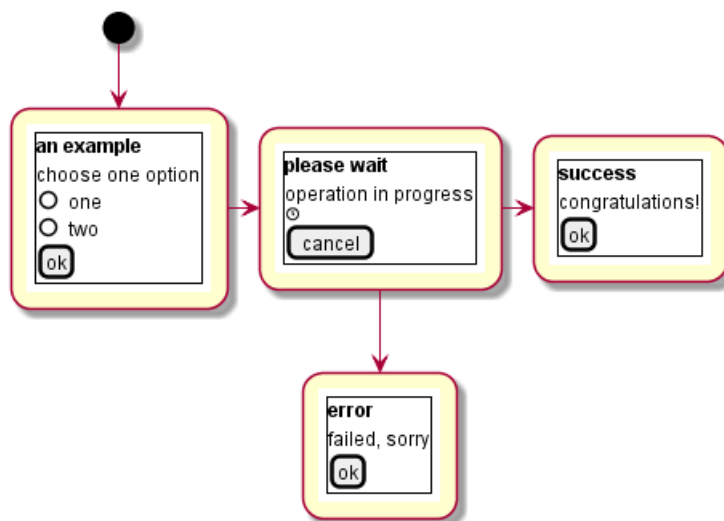
```

()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

```



It can also be combined with define macro.

```
@startuml
!definelong SALT(x)
"{{
salt
_##x
}}
" as x
!enddefinelong

!definelong _choose
{+
<b>an example
choose one option
()one
()two
[ok]
}
!enddefinelong

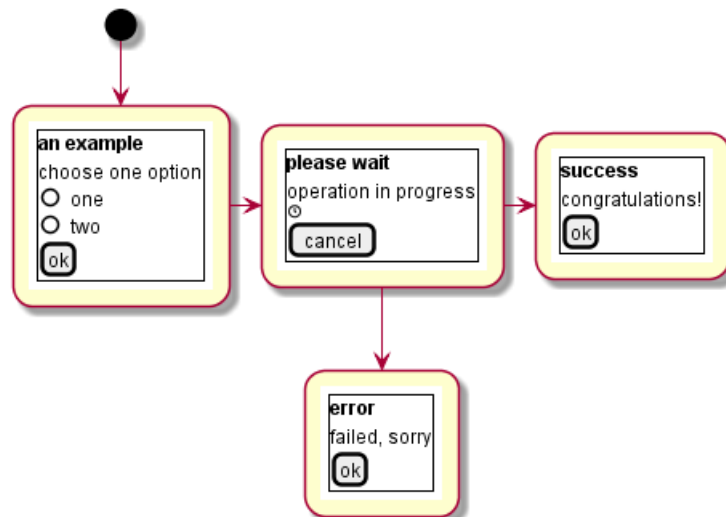
!definelong _wait
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!enddefinelong

!definelong _success
{+
<b>success
congratulations!
[ok]
}
!enddefinelong

!definelong _error
{+
<b>error
failed, sorry
[ok]
}
!enddefinelong

(*) --> SALT(choose)
-right-> SALT(wait)
wait -right-> SALT(success)
wait -down-> SALT(error)

@enduml
```



15.12 Scroll Bars

You can use "S" as scroll bar like in following examples:

```
@startsalt
{S
Message
.
.
.
.
}
@endsalt
```



```
@startsalt
{SI
Message
.
.
.
.
}
@endsalt
```

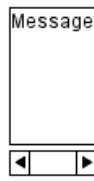


```
@startsalt
{S-
Message
.
.
.

```



```
.  
}  
@endsalt
```



16 Creole

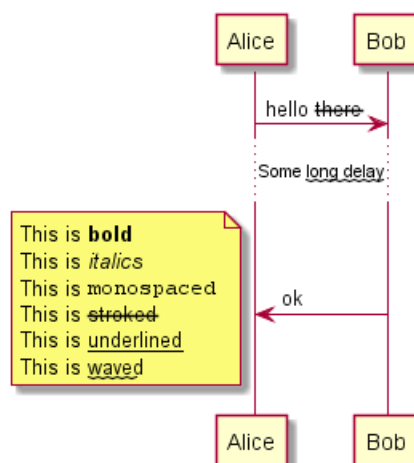
A light Creole engine has been integrated into PlantUML to have a standardized way of defining text style.

All diagrams are now supporting this syntax.

Note that ascending compatibility with HTML syntax is preserved.

16.1 Emphasized text

```
@startuml
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
@enduml
```



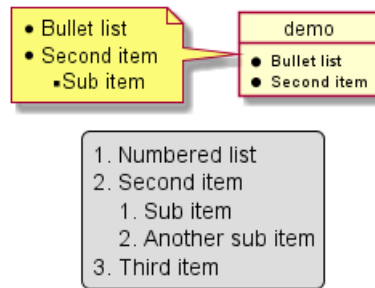
16.2 List

```
@startuml
object demo {
  * Bullet list
  * Second item
}
note left
  * Bullet list
  * Second item
  ** Sub item
end note

legend
  # Numbered list
  # Second item
  ## Sub item
  ## Another sub item
end
```



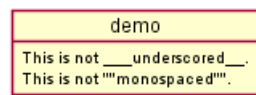
```
# Third item
end legend
@enduml
```



16.3 Escape character

You can use the tilde ~ to escape special creole characters.

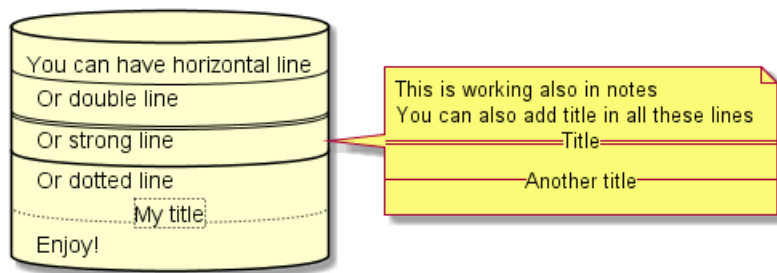
```
@startuml
object demo {
  This is not ~___underscored___.
  This is not ~""monospaced"".
}
@enduml
```



16.4 Horizontal lines

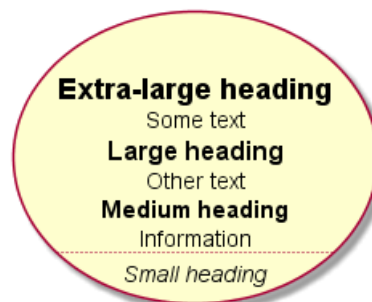
```
@startuml
database DB1 as "
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
Enjoy!
"
note right
  This is working also in notes
  You can also add title in all these lines
  ==Title==
  --Another title--
end note
@enduml
```





16.5 Headings

```
@startuml
usecase UC1 as "
= Extra-large heading
Some text
== Large heading
Other text
=== Medium heading
Information
....
==== Small heading"
@enduml
```



16.6 Legacy HTML

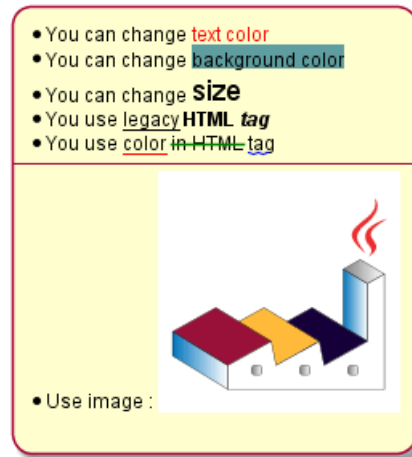
Some HTML tags are also working:

- `` for bold text
- `<u>` or `<u:#AAAAAA>` or `<u:colorName>` for underline
- `<i>` for italic
- `<s>` or `<s:#AAAAAA>` or `<s:colorName>` for strike text
- `<w>` or `<w:#AAAAAA>` or `<w:colorName>` for wave underline text
- `<color:#AAAAAA>` or `<color:colorName>`
- `<back:#AAAAAA>` or `<back:colorName>` for background color
- `<size:nn>` to change font size
- `<img:file>`: the file must be accessible by the filesystem
- `<img:http://plantuml.com/logo3.png>`: the URL must be available from the Internet

```
@startuml
:* You can change <color:red>text color</color>
* You can change <back:cadetblue>background color</back>
* You can change <size:18>size</size>
```



```
* You use <u>legacy</u> <b>HTML <i>tag</i></b>
* You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
----
* Use image : <img:http://plantuml.com/logo3.png>
;
@enduml
```



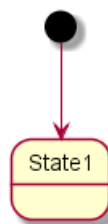
16.7 Table

It is possible to build table.

```
@startuml
skinparam titleFontSize 14
title
  Example of simple table
  | = | = table | = header |
  | a | table | row |
  | b | table | row |
end title
[*] --> State1
@enduml
```

Example of simple table

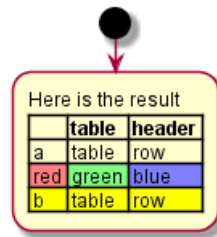
	table	header
a	table	row
b	table	row



You can specify background colors for cells and lines.

```
@startuml
start
:Here is the result
| = | = table | = header |
| a | table | row |
|<#FF8080> red |<#80FF80> green |<#8080FF> blue |
<#yellow>| b | table | row |;
@enduml
```



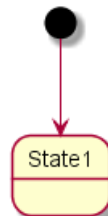


16.8 Tree

You can use |_ characters to build a tree.

```
@startuml
skinparam titleFontSize 14
title
    Example of Tree
    |_ First line
    |_ **Bom(Model)**
|_ prop1
|_ prop2
|_ prop3
    |_ Last line
end title
[*] --> State1
@enduml
```

```
Example of Tree
|_ First line
|_ Bom(Model)
    |_ prop1
    |_ prop2
    |_ prop3
    |_ Last line
```



16.9 Special characters

It's possible to use any unicode characters with &# syntax or <U+XXXX>

```
@startuml
usecase foo as "this is &#8734; long"
usecase bar as "this is also <U+221E> long"
@enduml
```



16.10 OpenIconic

OpenIconic is an very nice open source icon set. Those icons have been integrated into the creole parser, so you can use them out-of-the-box.



You can use the following syntax: `<&ICON_NAME>`.

```
@startuml
title: <size:20><&heart>Use of OpenIconic<&heart></size>
class Wifi
note left
    Click on <&wifi>
end note
@enduml
```

♥Use of OpenIconic♥



The complete list is available on OpenIconic Website, or you can use the following special diagram:

```
@startuml
listopeniconic
@enduml
```

List Open Iconic

Credit to
<https://useiconic.com/open>

→ account-login	🔔 bell	☁ cloud	📄 excerpt	⌵ expand-down	🔑 key	🎵 musical-note	★ star
→ account-logout	📶 bluetooth	☁️ cloudy	⌵ expand-left	⌵ expand-right	💻 laptop	📄 paperclip	☀ sun
↶ action-redo	⚙ bolt	🔗 code	⌵ expand-up	👁 eye	📱 layers	✎ pencil	📱 tablet
↷ action-undo	📖 book	⌵ collapse-down	👁 eyedropper	🔗 link-broken	👤 person	📊 pie-chart	🏷 tag
≡ align-center	📁 bookmark	⌵ collapse-left	📁 file	🔗 link-intact	📌 pin	📊 task	🏷 tags
≡ align-left	📦 box	⌵ collapse-right	🔥 fire	📋 list	🎮 play-circle	📌 terminal	🎯 target
≡ align-right	📧 briefcase	⌵ collapse-up	🚩 flag	📍 location	➕ plus	📄 text	👍 thumb-down
⚙ aperture	£ british-pound	📄 command	⚡ flash	🔒 lock-locked	🔌 power-standby	🖨 print	👍 thumb-up
↓ arrow-bottom	🐛 bug	📄 comment-square	📁 folder	🔓 lock-unlocked	📄 project	📄 timer	➡ transfer
⦿ arrow-circle-bottom	📊 calculator	📄 compass	🍴 fork	🔄 loop-circular	📌 pulse	🗑 trash	🗑 underline
⦿ arrow-circle-left	📅 calendar	📄 contrast	🖥 fullscreen-enter	⦿ loop-square	🧩 puzzle-piece	🗑 vertical-align-bottom	🗑 vertical-align-center
⦿ arrow-circle-right	📷 camera-slr	📄 copywriting	🖥 fullscreen-exit	📄 loop	❓ question-mark	🗑 vertical-align-top	🗑 video
⦿ arrow-circle-top	↶ caret-bottom	📄 credit-card	🌐 globe	🔍 magnifying-glass	🌧 rain	🔊 volume-high	🔊 volume-low
← arrow-left	↶ caret-left	📄 crop	📊 graph	📍 map-marker	⚡ random	🔊 volume-off	⚠ warning
→ arrow-right	↷ caret-right	🗑 delete	📊 grid-four-up	📄 map	🔄 reload	📶 wifi	🔧 wrench
↓ arrow-thick-bottom	⬆ caret-top	📄 dial	📊 grid-three-up	📄 media-pause	↕ resize-both	📶 x	🔧 x
← arrow-thick-left	🚗 cart	📄 document	📊 grid-two-up	▶ media-play	↕ resize-height	🔧 yen	🔧 yen
→ arrow-thick-right	💬 chat	💵 dollar	💾 hard-drive	● media-record	↕ resize-width	📶 zoom-in	📶 zoom-out
↑ arrow-thick-top	✓ check	💵 double-quote-sans-left	📄 header	⏮ media-skip-backward	📶 rss-alt		
⚙ audio-spectrum	↶ chevron-bottom	💵 double-quote-sans-right	🎧 headphones	⏪ media-skip-backward	📶 rss		
🔊 audio	↶ chevron-left	💵 double-quote-serif-left	♥ heart	⏩ media-skip-forward	📶 script		
🏷 badge	↷ chevron-right	💵 double-quote-serif-right	🏠 home	⏮ media-step-backward	📶 share-boxed		
🚫 ban	⬆ chevron-top	💧 droplet	🖼 image	⏪ media-step-backward	📶 share		
📊 bar-chart	🔍 circle-check	🚪 eject	📁 inbox	⏩ media-step-forward	🛡 shield		
🛒 basket	⦿ circle-x	🚶 elevator	∞ infinity	⏮ media-stop	📶 signal		
🔋 battery-empty	📄 clipboard	📄 ellipses	📄 info	⏪ media-stop	📶 signpost		
🔋 battery-full	🕒 clock	✉ envelope-closed	📄 italic	⏩ media-stop	📶 sort-ascending		
🧴 beaker	☁ cloud-download	✉ envelope-open	≡ justify-center	🌙 moon	📶 sort-descending		
	☁ cloud-upload	€ euro	≡ justify-left	➡ move	📶 spreadsheet		



17 Defining and using sprites

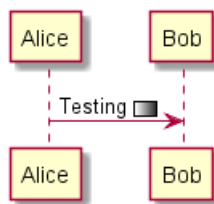
A *Sprite* is a small graphic element that can be used in diagrams.

In PlantUML, sprites are monochrome and can have either 4, 8 or 16 gray level.

To define a sprite, you have to use a hexadecimal digit between 0 and F per pixel.

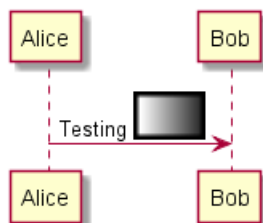
Then you can use the sprite using <\$XXX> where XXX is the name of the sprite.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1>
@enduml
```



You can scale the sprite.

```
@startuml
sprite $foo1 {
  FFFFFFFFFFFFFFFF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  F0123456789ABCF
  FFFFFFFFFFFFFFFF
}
Alice -> Bob : Testing <$foo1{scale=3}>
@enduml
```



17.1 Encoding Sprite

To encode sprite, you can use the command line like:

```
java -jar plantuml.jar -encodesprite 16z foo.png
```

where `foo.png` is the image file you want to use (it will be converted to gray automatically).

After `-encodesprite`, you have to specify a format: 4, 8, 16, 4z, 8z or 16z.

The number indicates the gray level and the optional `z` is used to enable compression in sprite definition.

17.2 Importing Sprite

You can also launch the GUI to generate a sprite from an existing image.

Click in the menubar then on File/Open Sprite Window.

After copying an image into you clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

17.3 Examples

```
@startuml
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvF
start
:click on <$printer> to print the page;
@enduml
```



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p__FEjQEgB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrrlw
sprite $printer [15x15/8z] N0tH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvF
sprite $disk {
    444445566677881
    436000000009991
    43600000000ACA1
    53700000001A7A1
    53700000012B8A1
    53800000123B8A1
    63800001233C9A1
    634999AABBC99B1
    744566778899AB1
    7456AAAAA99AAB1
    8566AFC228AABB1
    8567AC8118BBBB1
    867BD4433BBBBB1
    39AAAAABBBBBBC1
}

```

title Use of sprites (<\$printer>, <\$bug>...)

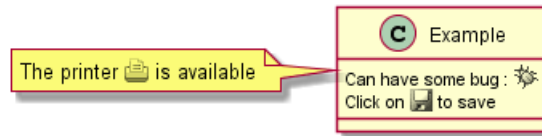
```
class Example {
    Can have some bug : <$bug>
    Click on <$disk> to save
}
```




```
note left : The printer <$printer> is available
```

```
@enduml
```

Use of sprites (🖨️, 🐛...)



18 Skinparam command

You can change colors and font of the drawing using the `skinparam` command.

Example:

```
skinparam backgroundColor transparent
```

18.1 Usage

You can use this command :

- In the diagram definition, like any other commands,
- In an included file,
- In a configuration file, provided in the command line or the ANT task.

18.2 Nested

To avoid repetition, it is possible to nest definition. So the following definition :

```
skinparam xxxxParam1 value1
skinparam xxxxParam2 value2
skinparam xxxxParam3 value3
skinparam xxxxParam4 value4
```

is strictly equivalent to:

```
skinparam xxxx {
    Param1 value1
    Param2 value2
    Param3 value3
    Param4 value4
}
```

18.3 List

Since the documentation is not always up to date, you can have the complete list of parameters using this command:

```
java -jar plantuml.jar -language
```

18.4 Black and White

You can force the use of a black&white output using `skinparam monochrome true` command.

```
@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
```



```

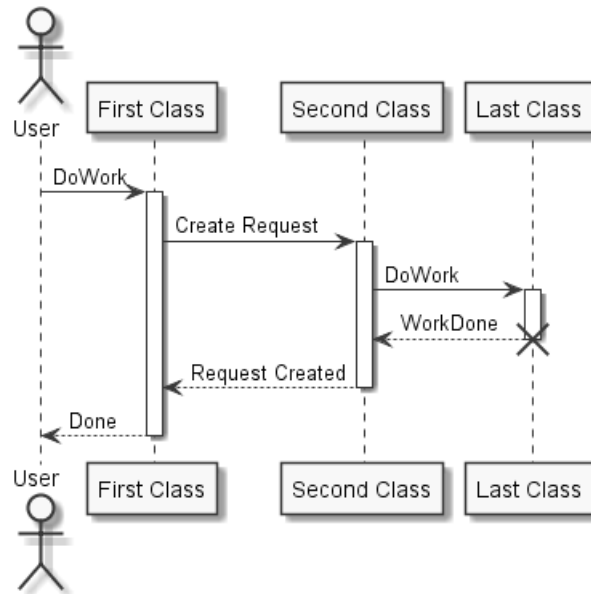
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



18.5 Reverse colors

You can force the use of a black&white output using `skinparam monochrome reverse` command. This can be useful for black background environment.

```

@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

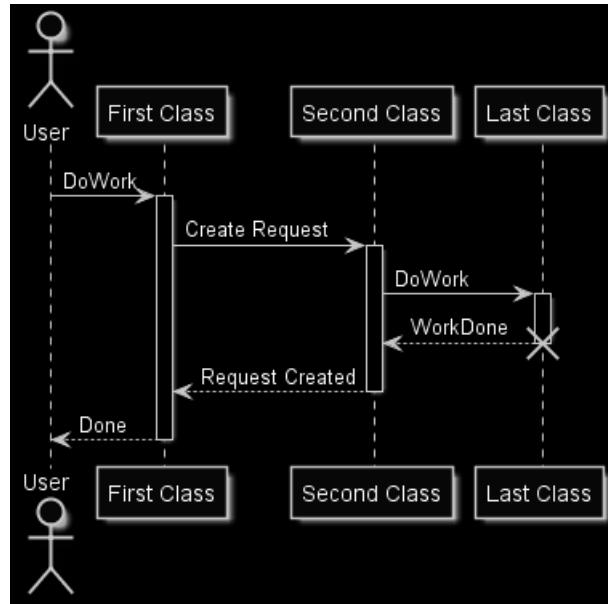
```



```
deactivate B
```

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

```
@enduml
```



18.6 Colors

You can use either standard color name or RGB code.

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 can only be used for background of the image.

18.7 Font color, name and size

You can change the font for the drawing using xxxFontColor, xxxFontSize and xxxFontName parameters.

Example:



```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Aapex
```

You can also change the default font for all fonts using `skinparam defaultFontName`.

Example:

```
skinparam defaultFontName Aapex
```

Please note the fontname is highly system dependent, so do not over use it, if you look for portability. Helvetica and Courier should be available on all system.

A lot of parameters are available. You can list them using the following command:

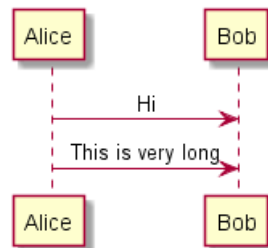
```
java -jar plantuml.jar -language
```

18.8 Text Alignment

Text alignment can be set up to left, right or center. You can also use `direction` or `reverseDirection` values for `sequenceMessageAlign` which align text depending on arrow direction.

Param name	Default value	Comment
<code>sequenceMessageAlign</code>	left	Used for messages in sequence diagrams
<code>sequenceReferenceAlign</code>	center	Used for ref over in sequence diagrams

```
@startuml
skinparam sequenceMessageAlign center
Alice -> Bob : Hi
Alice -> Bob : This is very long
@enduml
```



18.9 Examples

```
@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

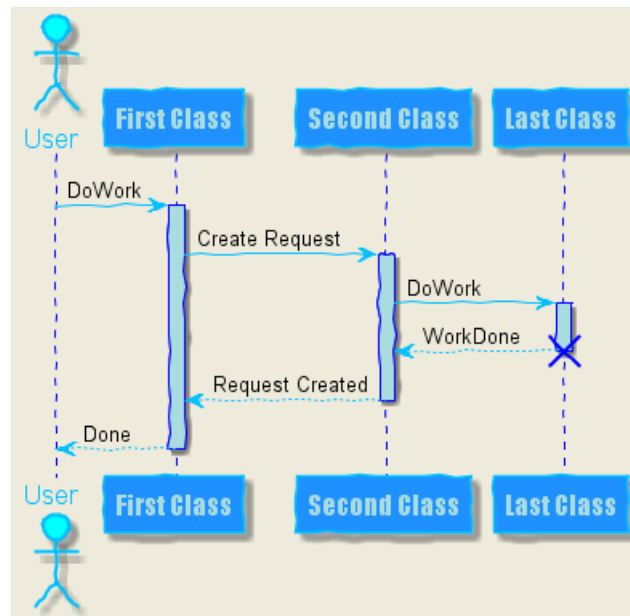
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

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

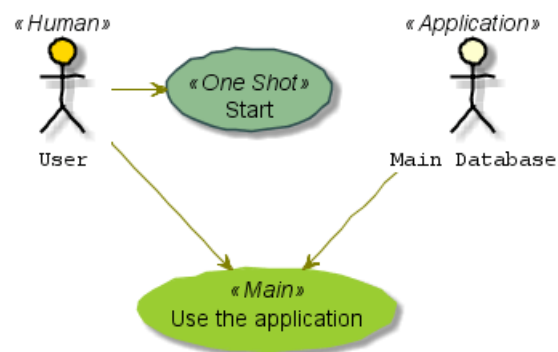
ArrowColor Olive
}

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

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

MySql --> (Use)
@enduml

```



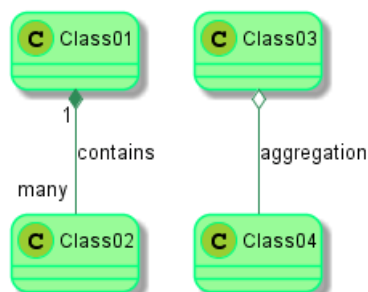
```

@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
}

```

```

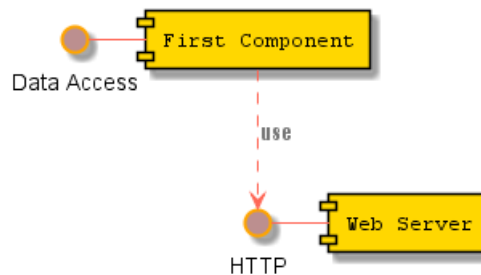
}

skinparam component {
  FontSize 13
  BackgroundColor<<Apache>> Red
  BorderColor<<Apache>> #FF6655
  FontName Courier
  BorderColor black
  BackgroundColor gold
  ArrowFontName Impact
  ArrowColor #FF6655
  ArrowFontColor #777777
}

() "Data Access" as DA

DA - [First Component]
[First Component] ..> () HTTP : use
HTTP - [Web Server] << Apache >>
@enduml

```



```

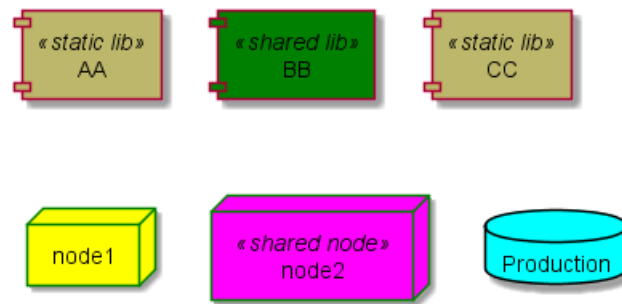
@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

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

```

19 Preprocessing

Some minor preprocessing capabilities are included in **PlantUML**, and available for *all* diagrams.

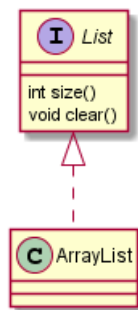
Those functionalities are very similar to the C language preprocessor, except that the special character # has been changed to the exclamation mark !.

19.1 Including files

Use the `!include` directive to include file in your diagram.

Imagine you have the very same class that appears in many diagrams. Instead of duplicating the description of this class, you can define a file that contains the description.

```
@startuml
!include List.iuml
List <|.. ArrayList
@enduml
```



File List.iuml

```
interface List
List : int size()
List : void clear()
```

The file `List.iuml` can be included in many diagrams, and any modification in this file will change all diagrams that include it.

A file can be only be included once. If you want to include several times the very same file, you have to use the directive `!include_many` instead of `!include`.

You can also put several `@startuml/@enduml` text block in an included file and then specify which block you want to include adding `!0` where 0 is the block number.

For example, if you use `!include foo.txt!1`, the second `@startuml/@enduml` block within `foo.txt` will be included.

You can also put an id to some `@startuml/@enduml` text block in an included file using `@startuml(id=MY_OWN_ID)` syntax and then include the block adding `!MY_OWN_ID` when including the file, so using something like `!include foo.txt!MY_OWN_ID`.

19.2 Including URL

Use the `!includeurl` directive to include file from Internet/Intranet in your diagram.

You can also use `!includeurl http://someurl.com/mypath!0` to specify which `@startuml/@enduml` block from `http://someurl.com/mypath` you want to include. The `!0` notation denotes the first diagram.

19.3 Constant definition

You can define constant using the `!define` directive. As in C language, a constant name can only use alphanumeric and underscore characters, and cannot start with a digit.



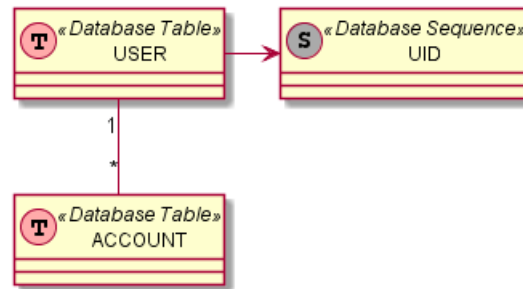
```

@startuml

!define SEQUENCE (S,#AAAAAA) Database Sequence
!define TABLE (T,#FFAAAA) Database Table

class USER << TABLE >>
class ACCOUNT << TABLE >>
class UID << SEQUENCE >>
USER "1" -- "*" ACCOUNT
USER -> UID
@enduml

```



Of course, you can use the `!include` directive to define all your constants in a single file that you include in your diagram.

Constant can be undefined with the `!undef XXX` directive.

You can also specify constants within the command line, with the `-D` flags.

```
java -jar plantuml.jar -DTITLE="My title" atest1.txt
```

Note that the `-D` flag must be put after the `"-jar plantuml.jar"` section.

19.4 Macro definition

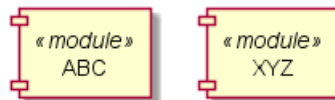
You can also define macro with arguments.

```

@startuml

!define module(x) component x <<module>>
module(ABC)
module(XYZ)
@enduml

```



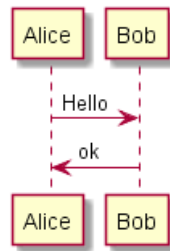
Macro can have several arguments.

```

@startuml
!define send(a,b,c) a->b : c
send(Alice, Bob, Hello)
send(Bob, Alice, ok)
@enduml

```





19.5 Adding date and time

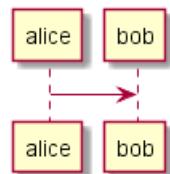
You can also expand current date and time using the special variable `%date%`.

Date format can be specified using format specified in SimpleDataFormat documentation.

```

@startuml
!define ANOTHER_DATE %date[yyyy.MM.dd 'at' HH:mm]%
Title Generated %date% or ANOTHER_DATE
alice -> bob
@enduml
  
```

Generated Wed Apr 03 22:49:22 CEST 2019 or 2019.04.03 at 22:49



19.6 Other special variables

You can also use the following special variables:

Variable	Content
<code>%dirpath%</code>	Path of the current file
<code>%filename%</code>	Name of the current file

19.7 Macro on several lines

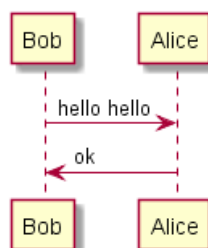
You can also define macro on several lines using `!definelong` and `!enddefinelong`.

```

@startuml
!define DOUBLE(x) x x
!definelong AUTHEN(x,y)
x -> y : DOUBLE(hello)
y -> x : ok
!enddefinelong
  
```

```

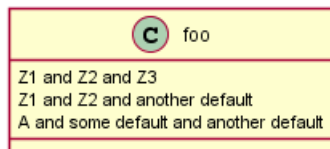
AUTHEN(Bob,Alice)
@enduml
  
```



19.8 Default values for macro parameters

It is possible to assign default values to macro parameters.

```
@startuml
!define some_macro(x, y = "some default" , z = 'another default' ) x and y and z
class foo {
    some_macro(Z1, Z2, Z3)
    some_macro(Z1, Z2)
    some_macro(A)
}
@enduml
```



19.9 Conditions

You can use `!ifdef XXX` and `!endif` directives to have conditionnal drawings.

The lines between those two directives will be included only if the constant after the `!ifdef` directive has been defined before.

You can also provide a `!else` part which will be included if the constant has **not** been defined.

```
@startuml
!include ArrayList.iuml
@enduml
```

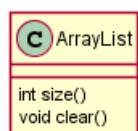


File ArrayList.iuml:

```
class ArrayList
!ifdef SHOW_METHODS
class ArrayList {
    int size()
    void clear()
}
!endif
```

You can then use the `!define` directive to activate the conditionnal part of the diagram.

```
@startuml
!define SHOW_METHODS
!include ArrayList.iuml
@enduml
```



You can also use the `!ifndef` directive that includes lines if the provided constant has NOT been defined.

You can use boolean expression with parenthesis, operators `&&` and `||` in the test.

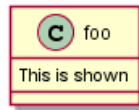
```
@startuml
!define SHOW_FIELDS
```



```

!undef SHOW_METHODS
class foo {
!ifdef SHOW_FIELDS || SHOW_METHODS
This is shown
!endif
!ifdef SHOW_FIELDS && SHOW_METHODS
This is NOT shown
!endif
}
@enduml

```



19.10 Building custom library

It's possible to package a set of included files into a single .zip or .jar archive. This single zip/jar can then be imported into your diagram using !import directive.

Once the library has been imported, you can !include file from this single zip/jar.

Example:

```

@startuml
!import /path/to/customLibrary.zip
' This just adds "customLibrary.zip" in the search path

!include myFolder/myFile.iuml
' Assuming that myFolder/myFile.iuml is located somewhere
' either inside "customLibrary.zip" or on the local filesystem

...

```

19.11 Search path

You can specify the java property plantuml.include.path in the command line.

For example:

```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

Note the this -D option has to put before the -jar option. -D options after the -jar option will be used to define constants within plantuml preprocessor.

19.12 Advanced features

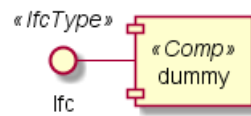
It is possible to append text to a macro argument using the ## syntax.

```

@startuml
!definelong COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!enddefinelong
COMP_TEXTGENCOMP(dummy)
@enduml

```

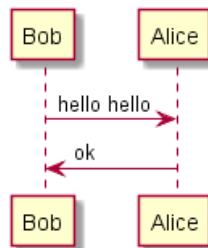




A macro can be defined by another macro.

```

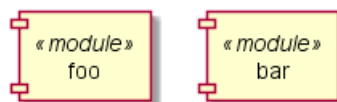
@startuml
!define DOUBLE(x) x x
!definelong AUTHEN(x,y)
x -> y : DOUBLE(hello)
y -> x : ok
!enddefinelong
AUTHEN(Bob,Alice)
@enduml
  
```



A macro can be polymorphic with argument count.

```

@startuml
!define module(x) component x <<module>>
!define module(x,y) component x as y <<module>>
module(foo)
module(bar, barcode)
@enduml
  
```



You can use system environment variable or constant definition when using include:

```

!include %windir%/test1.txt
!define PLANTUML_HOME /home/foo
!include PLANTUML_HOME/test1.txt
  
```

20 Unicode

The PlantUML language use *letters* to define actor, usecase and soon.

But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language*.

20.1 Examples

```
@startuml
skinparam handwritten true
skinparam backgroundColor #EEEEBD

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
```

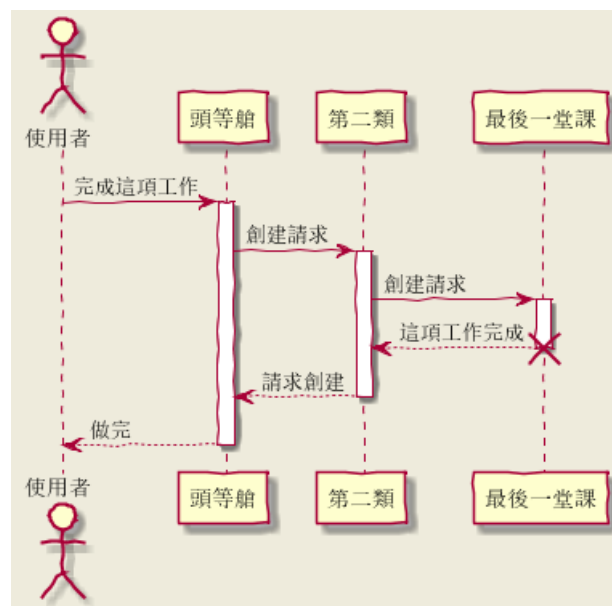
```
使用者 -> A: 完成這項工作
activate A
```

```
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
```



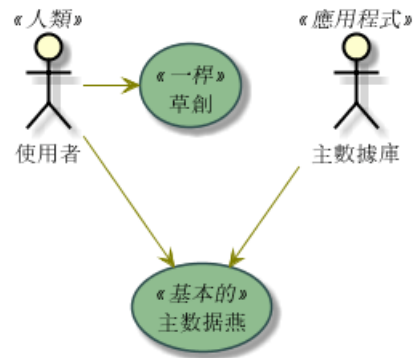
```
@startuml
skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray
```

```
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
```





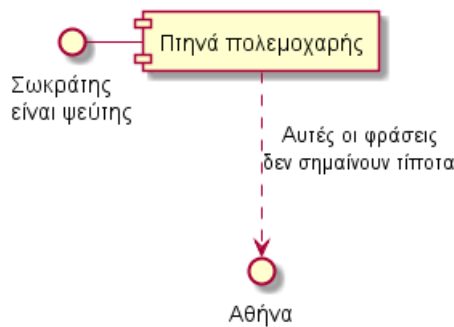
@startuml

() "Σωκράτηςψεύτης" as Σωκράτης

Σωκράτης - [Πτηνά πολεμοχαρής]

[Πτηνά πολεμοχαρής] ..> () Αθήνα : Αυτές οι φράσειςσημαίνουν τίποτα

@enduml



20.2 Charset

The default charset used when *reading* the text files containing the UML text description is system dependent.

Normally, it should just be fine, but in some case, you may want to the use another charset. For example, with the command line:

```
java -jar plantuml.jar -charset UTF-8 files.txt
```

Or, with the ant task:

```
<!-- Put images in c:/images directory -->
<target name="main">
<plantuml dir="./src" charset="UTF-8" />
```

Depending of your Java installation, the following charset should be available: ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16.



21 Standard Library

This page explains the official Standard Library for PlantUML. This Standard Library is now included in official releases of PlantUML. Including files follows the C convention for "C standard library" (see https://en.wikipedia.org/wiki/C_standard_library)

Contents of the library come from third party contributors. We thank them for their useful contribution!

21.1 AWS library

<https://github.com/milo-minderbinder/AWS-PlantUML>

The AWS library consists of Amazon AWS icons, it provides icons of two different sizes.

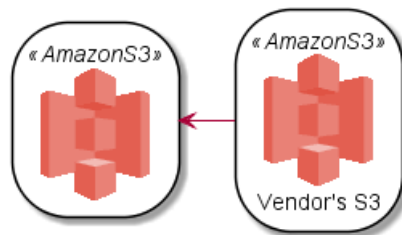
Use it by including the file that contains the sprite, eg: `!include <aws/Storage/AmazonS3/AmazonS3>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <aws/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

```
@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
```



21.2 Azure library

<https://github.com/RicardoNiepel/Azure-PlantUML/>

The Azure library consists of Microsoft Azure icons.

Use it by including the file that contains the sprite, eg: `!include <azure/Analytics/AzureEventHub.puml>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `AzureCommon.puml` file, eg: `!include <azure/AzureCommon.puml>`, which contains helper macros defined. With the `AzureCommon.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

```
@startuml
!include <azure/AzureCommon.puml>
!include <azure/Analytics/AzureEventHub.puml>
!include <azure/Analytics/AzureStreamAnalytics.puml>
!include <azure/Databases/AzureCosmosDb.puml>
```

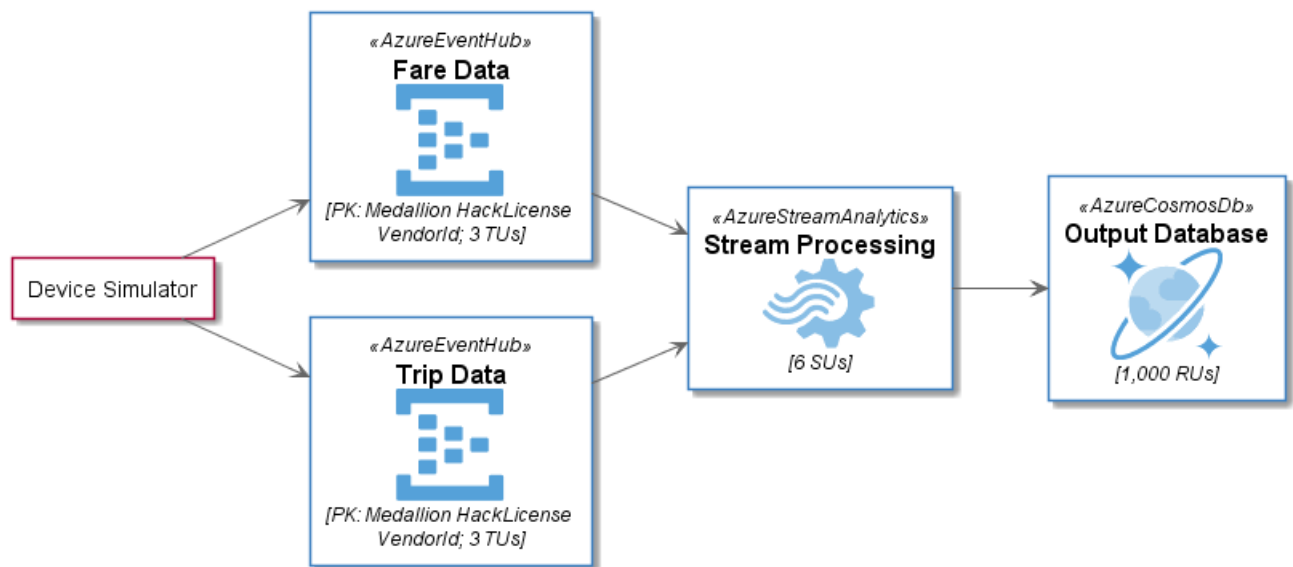
left to right direction



```
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")
AzureStreamAnalytics(streamAnalytics, "Stream Processing", "6 SUs")
AzureCosmosDb(outputCosmosDb, "Output Database", "1,000 RUs")
```

```
devices --> fareDataEventHub
devices --> tripDataEventHub
fareDataEventHub --> streamAnalytics
tripDataEventHub --> streamAnalytics
streamAnalytics --> outputCosmosDb
@enduml
```



21.3 Cloud Insight

<https://github.com/rabelenda/cicon-plantuml-sprites>

This repository contains PlantUML sprites generated from Cloudinsight icons, which can easily be used in PlantUML diagrams for nice visual representation of popular technologies.

```
@startuml
!include <cloudinsight/tomcat>
!include <cloudinsight/kafka>
!include <cloudinsight/java>
!include <cloudinsight/cassandra>

title Cloudinsight sprites example

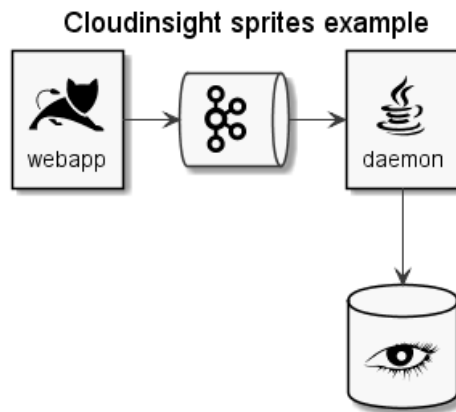
skinparam monochrome true

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
```



21.4 Devicons and Font Awesome library

<https://github.com/tupadr3/plantuml-icon-font-sprites>

These two library consists respectively of Devicons and Font Awesome libraries of icons.

Use it by including the file that contains the sprite, eg: `!include <font-awesome/align_center>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

You may also include the `common.puml` file, eg: `!include <font-awesome/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `NAME_OF_SPRITE(parameters...)` macro.

Example of usage:

```

@startuml
!include <tupadr3/common>
!include <tupadr3/font-awesome/server>
!include <tupadr3/font-awesome/database>

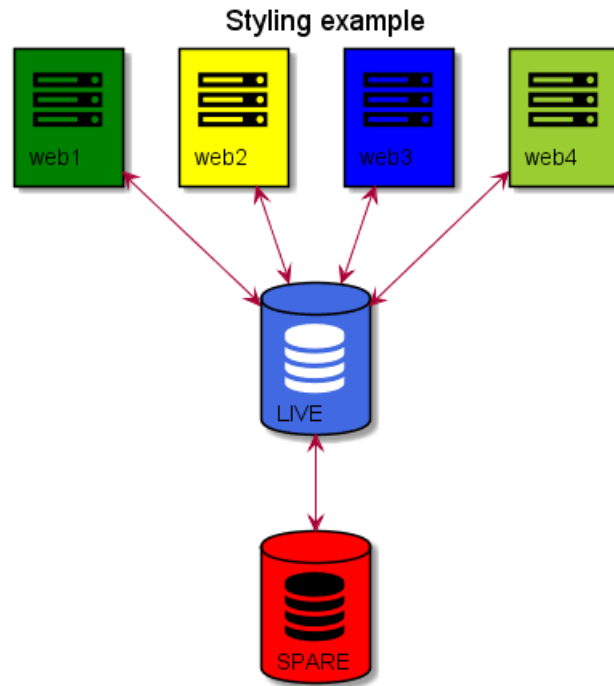
title Styling example

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
  
```

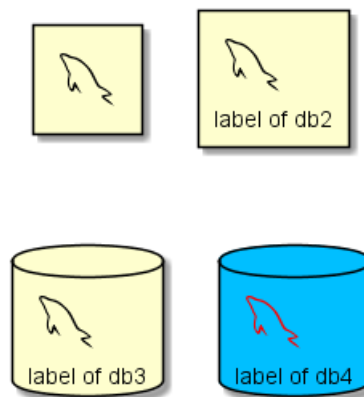


```

@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

```



21.5 Google Material Icons

<https://github.com/Templarian/MaterialDesign>

This library consists of a free Material style icons from Google and other artists.

Use it by including the file that contains the sprite, eg: `!include <material/ma_folder_move>`. When imported, you can use the sprite as normally you would, using `<$ma_sprite_name>`. Notice that this library requires an `ma_` prefix on sprites names, this is to avoid clash of names if multiple sprites have the same name on different libraries.

You may also include the `common.puml` file, eg: `!include <material/common>`, which contains helper macros defined. With the `common.puml` imported, you can use the `MA_NAME_OF_SPRITE(parameters...)` macro, note

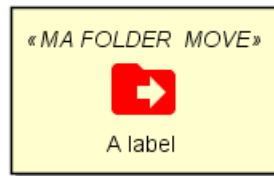


again the use of the prefix MA_.

Example of usage:

```
@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
```



Notes

When mixing sprites macros with other elements you may get a syntax error if, for example, trying to add a rectangle along with classes. In those cases, add { and } after the macro to create the empty rectangle.

Example of usage:

```
@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
```



21.6 Office

<https://github.com/Roemer/plantuml-office>

There are sprites (*.puml) and colored png icons available. Be aware that the sprites are all only monochrome even if they have a color in their name (due to automatically generating the files). You can either color the sprites with the macro (see examples below) or directly use the fully colored pngs. See the following examples on how to use the sprites, the pngs and the macros.

Example of usage:

```
@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>
```

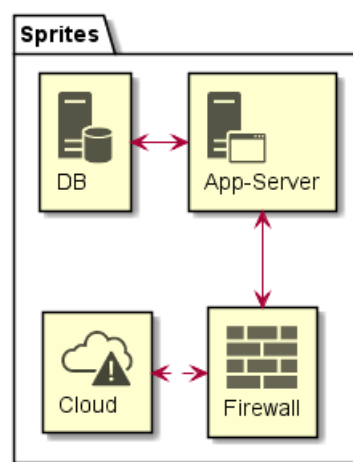


```
title Office Icons Example
```

```
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
```

Office Icons Example



```
@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
```

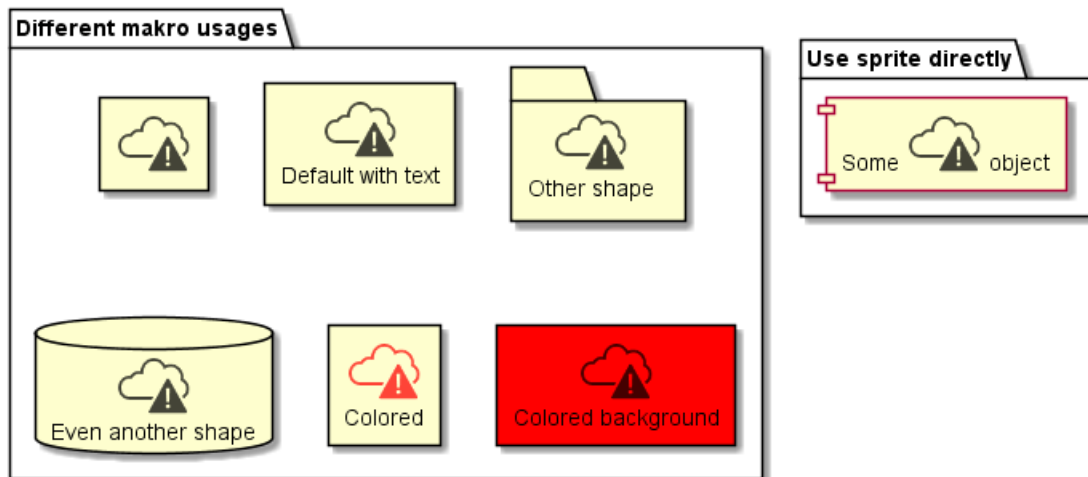
```
package "Use sprite directly" {
  [Some <$cloud_disaster_red> object]
}
```

```
package "Different makro 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
```



Extended Office Icons Example



21.7 ArchiMate

<https://github.com/ebbypeter/Archimate-PlantUML>

This repository contains ArchiMate PlantUML macros and other includes for creating Archimate Diagrams easily and consistantly.

@startuml Internet Browser Example

!includeurl <https://raw.githubusercontent.com/ebbypeter/Archimate-PlantUML/master/Archimate.puml>

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_Specilization_Up(webpageBehaviour, itSupportService, "")

Rel_Flow_Right(dataObject, webpageBehaviour, "")

Rel_Specilization_Up(dataObject, businessObject, "")

Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")

Rel_Specilization_Up(inMemoryItem, dataObject, "")

Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")

Rel_Specilization_Right(inMemoryItem, internetBrowser, "")

Rel_Serving_Up(internetBrowser, webpageBehaviour, "")

Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")

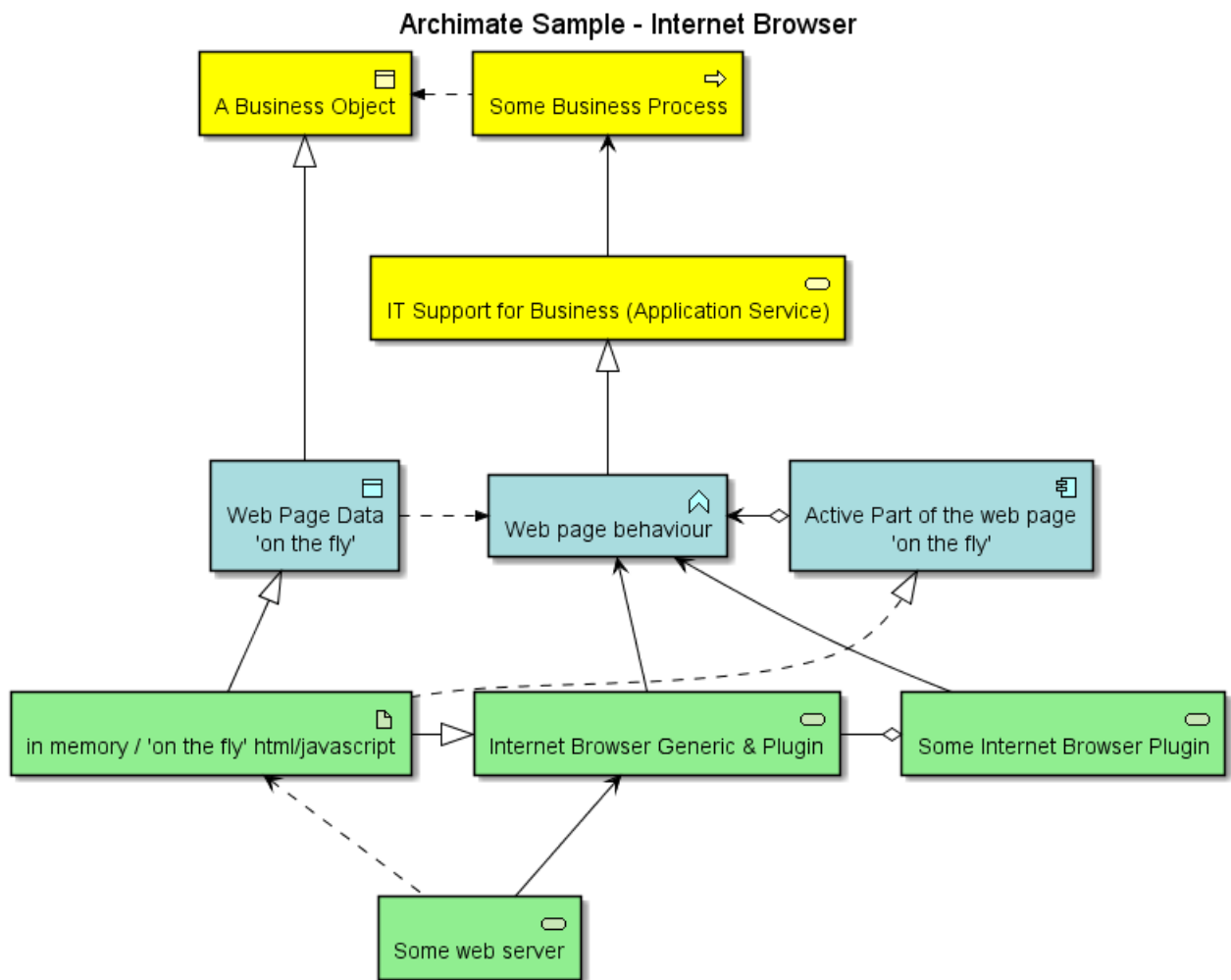
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")

Rel_Access_Up(webServer, inMemoryItem, "")



```
Rel_Serving_Up(webServer, internetBrowser, "")
```

```
@enduml
```



21.8 Miscellaneous

You can list standard library folders using the special diagram:

```
@startuml
stdlib
@enduml
```

aws

Version 18.02.22

Delivered by <https://github.com/milo-minderbinder/AWS-PlantUML>**azure**

Version 2.1.0

Delivered by <https://github.com/RicardoNiepel/Azure-PlantUML>**c4**

Version 1.0.0

Delivered by <https://github.com/RicardoNiepel/C4-PlantUML>**cloudinsight**

Version 0.0.1

Delivered by <https://github.com/rabelenda/cicon-plantuml-sprites/>**cloudogu**

Version 0.0.1

Delivered by <https://github.com/cloudogu/plantuml-cloudogu-sprites>**material**

Version 0.0.1

Delivered by <https://github.com/Templarian/MaterialDesign>**office**

Version 0.0.1

Delivered by <https://github.com/Roemer/plantuml-office>**tupadr3**

Version 2.0.0

Delivered by <https://github.com/tupadr3/plantuml-icon-font-sprites>

It is also possible to use the command line `java -jar plantuml.jar -stdlib` to display the same list.

Finally, you can extract the full standard library sources using `java -jar plantuml.jar -extractstdlib`. All files will be extracted in the folder `stdlib`.

Sources used to build official PlantUML releases are hosted here <https://github.com/plantuml/plantuml-stdlib>. You can create Pull Request to update or add some library if you find it relevant.



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	1
1.3	分類子名にアルファベット以外を使う	3
1.4	自分自身へのメッセージ	3
1.5	矢印の見た目を変える	3
1.6	矢印の色を替える	4
1.7	メッセージシーケンスの番号付け	4
1.8	Page Title, Header and Footer	7
1.9	図の分割	7
1.10	メッセージのグループ化	8
1.11	メッセージの注釈	9
1.12	その他の注釈	10
1.13	ノートの形を変える。	10
1.14	Creole と HTML	11
1.15	境界線	12
1.16	リファレンス	13
1.17	遅延	13
1.18	間隔	14
1.19	ライフラインの活性化と破壊	15
1.20	Return	16
1.21	分類子の生成	16
1.22	インとアウトのメッセージ	17
1.23	ステレオタイプとスポット	18
1.24	タイトルについての詳細	19
1.25	分類子の囲み	20
1.26	フッターの除去	21
1.27	スキンパラメータ	21
1.28	パディングの変更	23
2	ユースケース図	25
2.1	ユースケース	25
2.2	アクター	25
2.3	ユースケースの説明	26
2.4	簡単な例	26
2.5	継承	27
2.6	ノートの使用方法	27
2.7	ステレオタイプ	28
2.8	矢印の方向を変えるには	29
2.9	図を分割する	30
2.10	左から右に描画する	30
2.11	スキン設定 (Skinparam)	31
2.12	完全な例	32
3	クラス図	33
3.1	クラス間の関係	33
3.2	関係のラベル	34
3.3	メソッドの追加	35
3.4	可視性の定義	36
3.5	Abstract と Static	36
3.6	高等なクラス本体	37
3.7	注釈とステレオタイプ	38
3.8	注釈の詳細	38
3.9	リンクへの注釈	39
3.10	抽象クラスとインタフェース	40
3.11	非文字の使用	41
3.12	属性、メソッド等の非表示	41



3.13	非表示クラス	42
3.14	ジェネリクスの使用	43
3.15	特殊な目印	43
3.16	パッケージ	43
3.17	パッケージスタイル	44
3.18	名前空間	45
3.19	自動的に名前空間を作成する	46
3.20	ロリポップ（棒付きキャンディー）インタフェース	47
3.21	矢印の向きを変える	47
3.22	関連クラス	48
3.23	化粧をする	49
3.24	ステレオタイプの化粧	50
3.25	色のグラデーション	50
3.26	レイアウトの手助け	51
3.27	大きなファイルの分割	52
4	アクティビティ図	54
4.1	単純なアクティビティ	54
4.2	矢印のラベル	54
4.3	矢印の方向を変える	54
4.4	分岐	55
4.5	もっと分岐	56
4.6	同期	57
4.7	長いアクティビティの記述	58
4.8	注釈	58
4.9	パーティション	59
4.10	スキンパラメータ	60
4.11	八角形	61
4.12	完全な例	61
5	アクティビティ図（ベータ版）	64
5.1	単純なアクティビティ	64
5.2	開始 / 終了	64
5.3	条件文	65
5.4	繰り返し（後判定）	66
5.5	繰り返し（前判定）	67
5.6	並列処理	67
5.7	注釈	68
5.8	色指定	69
5.9	矢印	69
5.10	Connector	70
5.11	グループ化	70
5.12	動線	71
5.13	分離	72
5.14	SDL 図	73
5.15	完全な例	74
6	コンポーネント図	76
6.1	コンポーネント	76
6.2	インタフェース	76
6.3	基本的な例	77
6.4	ノートの使用方法	77
6.5	コンポーネントのグループ化	78
6.6	矢印の方向を変える	79
6.7	UML2 表記の使用	80
6.8	長い説明	81
6.9	個々の色	81
6.10	ステレオタイプでスプライトを使用	81
6.11	見かけを変える	82



7	ステート図	84
7.1	簡単なステート	84
7.2	ステートの表現を変える	84
7.3	複合状態	85
7.4	長い名前	86
7.5	同時状態	87
7.6	矢印の方向	88
7.7	注釈	89
7.8	もっと注釈	90
7.9	見栄え	90
8	オブジェクト図	92
8.1	オブジェクトの定義	92
8.2	オブジェクト間の関係	92
8.3	フィールドの追加	92
8.4	クラス図と共通の機能	93
9	タイミング図	94
9.1	ライフライン	94
9.2	メッセージ（相互作用）	94
9.3	相対時間での指定	95
9.4	インスタンス指向	96
9.5	スケールの設定	96
9.6	初期状態	96
9.7	Intricated state	97
9.8	Hidden state	98
9.9	時間定規（time constraint）の追加	98
9.10	タイトルなどを追加する	99
10	ガントチャート	101
10.1	タスクの定義	101
10.2	依存関係	101
10.3	エイリアス	101
10.4	色の変更	102
10.5	マイルストーン	102
10.6	日付の表示	102
10.7	休業日	102
10.8	Simplified task succession	103
10.9	Separator	103
10.10	Working with resources	103
10.11	複雑な例	104
11	MindMap	105
11.1	OrgMode syntax	105
11.2	Removing box	105
11.3	Arithmetic notation	106
11.4	Markdown syntax	106
11.5	Changing diagram direction	107
11.6	Complete example	107
12	Work Breakdown Structure	109
12.1	OrgMode syntax	109
12.2	Change direction	109
12.3	Arithmetic notation	110
13	イントロダクション	112
13.1	単体で使用する場合	112
13.2	どのように処理しているのか	113



14 Common commands	114
14.1 Comments	114
14.2 Footer and header	114
14.3 Zoom	114
14.4 Title	115
14.5 Caption	116
14.6 Legend the diagram	116
15 Salt (GUI 設計ツール)	118
15.1 基本のウィジェット	118
15.2 罫線の使用	118
15.3 Group box	119
15.4 セパレータの使用	119
15.5 木構造ウィジェット	120
15.6 括弧で括る	120
15.7 タブの追加	121
15.8 メニューの使用	121
15.9 テーブル (上級)	122
15.10OpenIconic	123
15.11Include Salt	123
15.12Scroll Bars	126
16 Creole	128
16.1 Emphasized text	128
16.2 List	128
16.3 Escape character	129
16.4 Horizontal lines	129
16.5 Headings	130
16.6 Legacy HTML	130
16.7 Table	131
16.8 Tree	132
16.9 Special characters	132
16.10OpenIconic	132
17 Defining and using sprites	134
17.1 Encoding Sprite	135
17.2 Importing Sprite	135
17.3 Examples	135
18 Skinparam command	137
18.1 Usage	137
18.2 Nested	137
18.3 List	137
18.4 Black and White	137
18.5 Reverse colors	138
18.6 Colors	139
18.7 Font color, name and size	139
18.8 Text Alignment	140
18.9 Examples	140
19 Preprocessing	145
19.1 Including files	145
19.2 Including URL	145
19.3 Constant definition	145
19.4 Macro definition	146
19.5 Adding date and time	147
19.6 Other special variables	147
19.7 Macro on several lines	147
19.8 Default values for macro parameters	148



19.9 Conditions	148
19.10 Building custom library	149
19.11 Search path	149
19.12 Advanced features	149
20 Unicode	151
20.1 Examples	151
20.2 Charset	153
21 Standard Library	154
21.1 AWS library	154
21.2 Azure library	154
21.3 Cloud Insight	155
21.4 Devicons and Font Awesome library	156
21.5 Google Material Icons	157
21.6 Office	158
21.7 ArchiMate	160
21.8 Miscellaneous	161