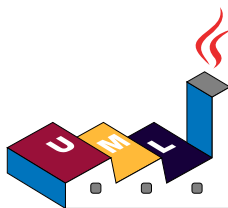


Построение диаграмм UML с использованием PlantUML



Справочное руководство по языку PlantUML
(Version 1.2019.9)

PlantUML — проект с открытым кодом, позволяющий быстро создавать:

- Диаграммы последовательности
- Диаграммы прецедентов
- Диаграммы классов
- Диаграммы активности
- Диаграммы компонентов
- Диаграммы состояний
- Диаграммы объектов
- Диаграмма развёртывания
- Диаграмма синхронизации

Также поддерживаются следующие не-UML-диаграммы:

- Wireframe graphical interface
- Archimate diagram
- Specification and Description Language (SDL)
- Dita diagram
- Gantt diagram
- MindMap diagram
- Work Breakdown Structure diagram
- Mathematic with AsciiMath or JLaTeXMath notation

Для создания диаграмм применяется простой и интуитивно понятный язык.

1 Диаграмма последовательности

1.1 Основные примеры

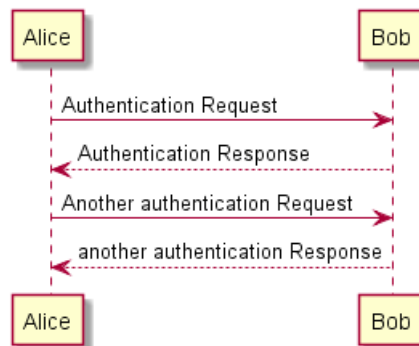
Последовательность -> используется, чтобы отобразить сообщение между двумя участниками (participants). Не обязательно явно объявлять участников.

Для получения пунктирной стрелки (dotted arrow), используйте -->.

Также возможно использовать <- и <--. Это не изменит отображение, но может улучшить читабельность. Заметьте, что это верно только для диаграмм последовательности, для других диаграмм правила другие.

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

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



1.2 Объявление участников

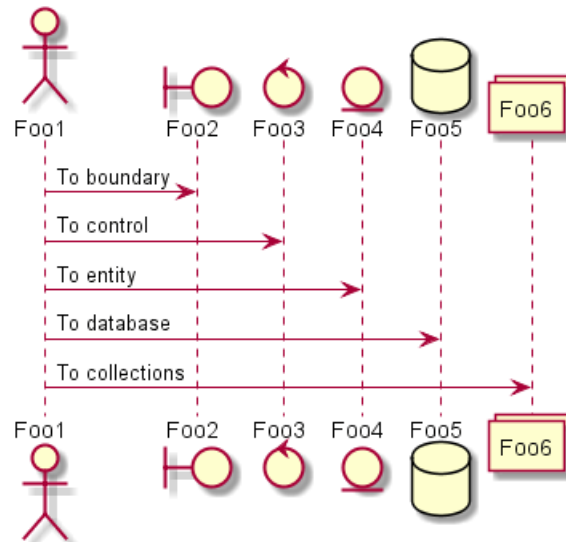
Возможно изменить порядок участников, используя ключевое слово participant.

Так же возможно использование других ключевых слов, для объявления participant'a:

- 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
```





Можно переименовать участника используя ключевое слово `as`

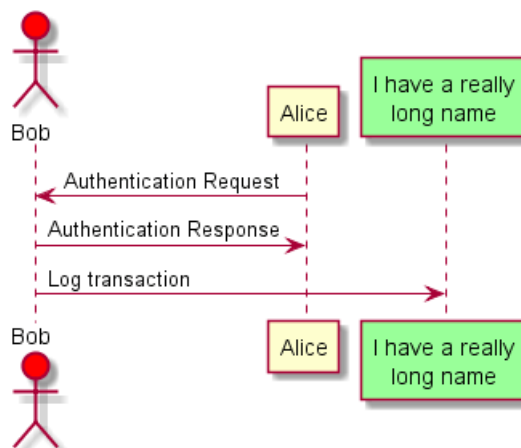
Также возможно изменить цвет фона actor-а или участника, используя имя цвета или его html-код

```

@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
  
```

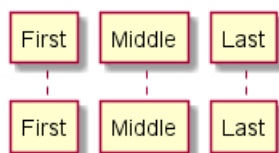


Используя ключевое поле `order` (порядок) можно настроить порядок печати participant (участника).

```

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



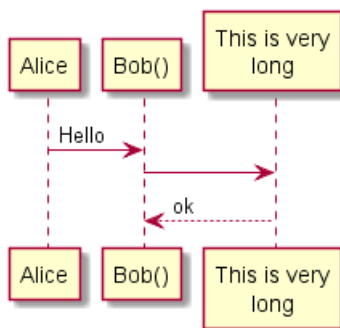


1.3 Использование небуквенных символов в названиях участников

Вы можете использовать кавычки для задания участников. Также Вы можете использовать ключевое слово `as` для присвоения псевдонимов к этим участникам.

```

@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
  
```



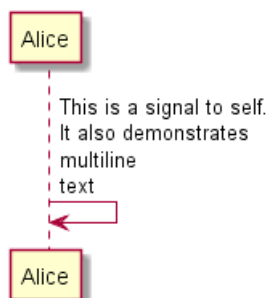
1.4 Сообщения к самому себе

Участник может посылать сообщения сам себе.

Также возможно создание многострочных используя `\n`.

```

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



1.5 Изменить стиль стрелок

Вы можете изменить стиль стрелок следующими способами:

- закончить стрелку с помощью `x` для обозначения потерянного сообщения
- используя `\` или `/` вместо `<` или `>` для создания только верхней или нижней части стрелки.
- повторите окончание стрелки (например, `>>` or `//`) для тонкой отрисовки.

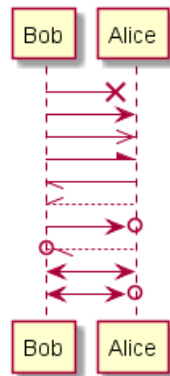


- используйте -- вместо - для создания пунктирной стрелки
- заканчивать символом "o" в острие стрелки
- использовать двунаправленные стрелки <->

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

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

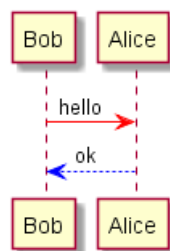
Bob <-> Alice
Bob <->o Alice
@enduml
```



1.6 Изменить цвет стрелок

Вы можете изменить цвет отдельных стрелок, используя следующие правила:

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



1.7 Нумерация сообщений в последовательностях

Ключевое слово autonumber используется для автоматической нумерации сообщений.

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





Вы можете обозначить число с которого начнется отсчет `autonumber start`, и число которое будет использоваться в качестве инкремента `autonumber start increment`.

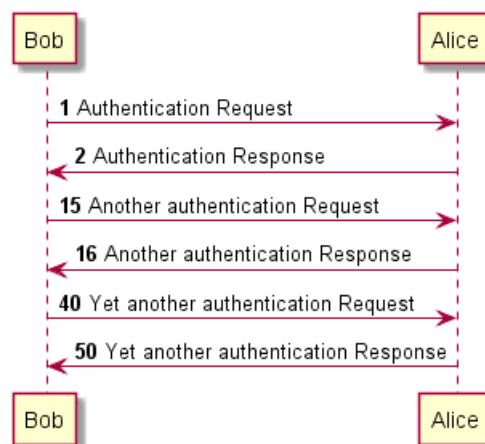
```

@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
  
```



Можно задавать формат чисел, указав его в двойных кавычках.

Форматирование выполнено с использованием класса Java `DecimalFormat` (0 означает цифру, # означает цифру или ноль если отсутствует).

При форматировании также можно использовать теги html.

```

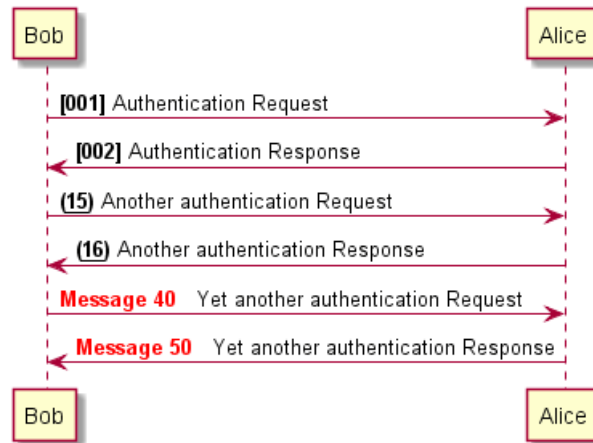
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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
  
```



@endum1



Вы так же можете использовать `autonumber stop` и `autonumber resume increment format` чтобы соответственно остановить и продолжить автоматическое нумерование.

```

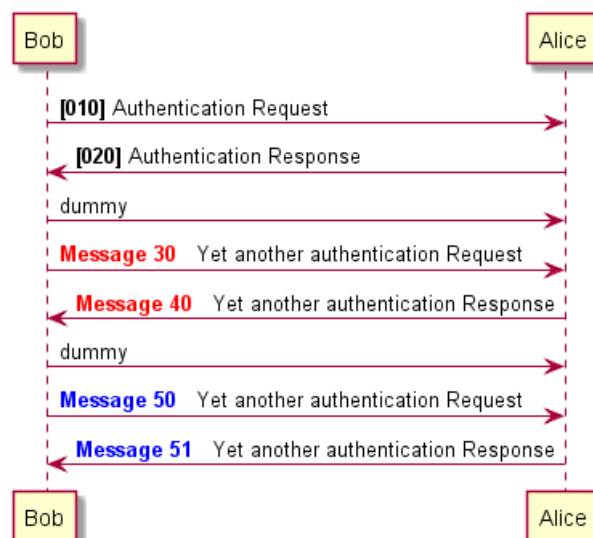
@startuml
autonumber 10 10 "<b>[000] "
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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

The title keyword is used to add a title to the page.

Pages can display headers and footers using header and footer.

```
@startuml
```

```
header Page Header
```

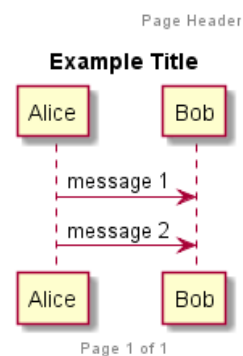
```
footer Page %page% of %lastpage%
```

```
title Example Title
```

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

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

```
@enduml
```



1.9 Разбиение диаграмм

Ключевое слово `newpage` используется для разбиения диаграмм на несколько изображений.

Вы можете указать название страницы сразу после ключевого слова `newpage`.

Это очень полезно для печати длинных диаграмм на нескольких страницах.

```
@startuml
```

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

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

```
newpage
```

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

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

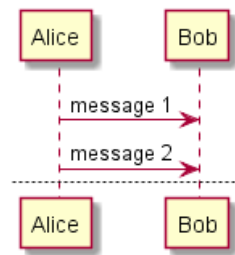
```
newpage A title for the\nlast page
```

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

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

```
@enduml
```





1.10 Группировка сообщений

Группировать сообщения возможно используя следующие ключевые слова:

- alt/else
- opt
- loop
- par
- break
- critical
- group, соответствует тексту который должен быть отображен

Имеется возможность добавить текст который должен быть отображен в заголовке. Ключевое слово end используется для завершения группы. Имейте ввиду что допускаются вложенные группы.

Ключевое слово end закрывает группу.

Допустимо вложение группы в группу.

```
@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
```

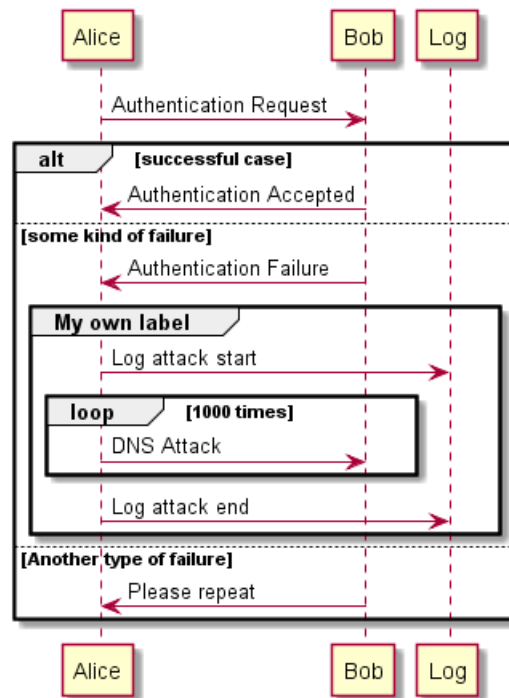
```
else Another type of failure
```

```
    Bob -> Alice: Please repeat
```

```
end
```

```
@enduml
```





1.11 Примечания в сообщениях

Можно помещать заметки к сообщениям, используя ключевые слова `note left` или `note right` сразу после сообщения.

Можно делать многострочные заметки используя ключевое слово `end note` для завершения.

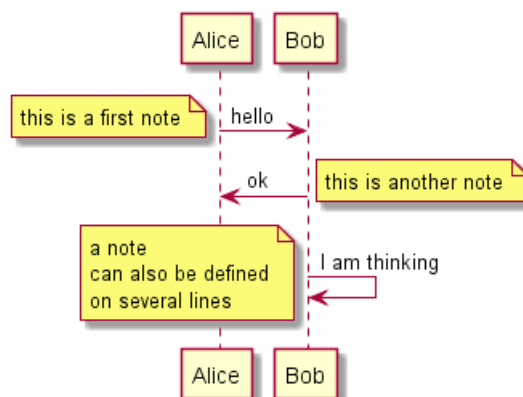
```

@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 Другие примечания

Так же возможно размещение примечаний относительно участников с использованием ключевых слов `<code>note left of</code>`

, `note right of` или `note over`.

Возможно выделить примечание изменив цвет фона.

Так же возможно многострочное примечание, для этого существует ключевое слово `end note`.

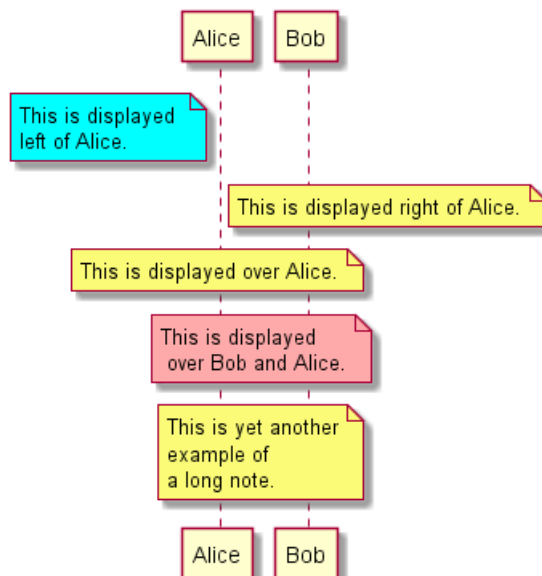
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

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
```



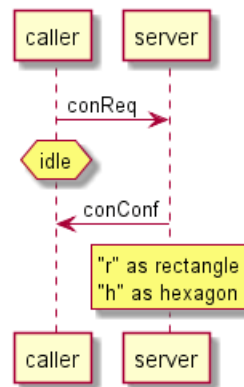
1.13 Изменение формы примечаний

Вы можете использовать `hnote` и `rnote` для изменения формы примечаний.

```
@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 и HTML

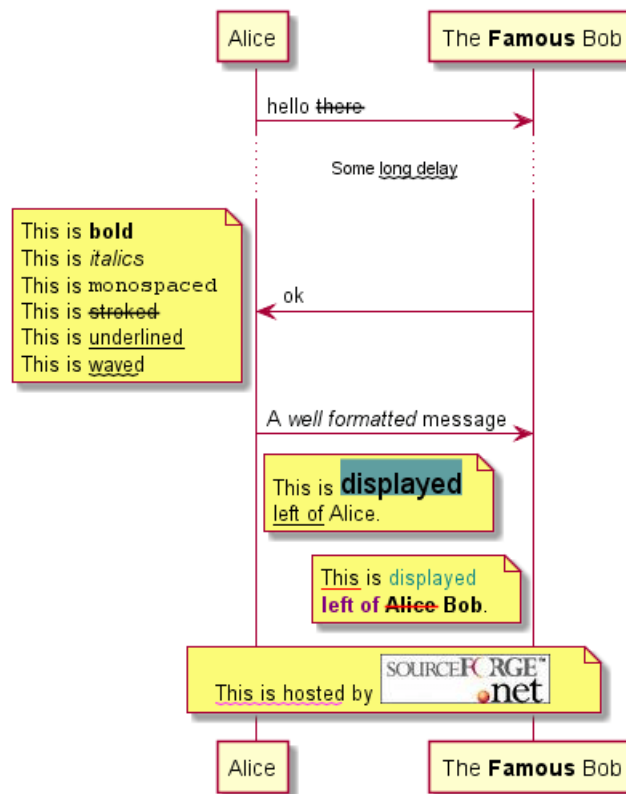
Так же можно использовать форматирование на Creole:

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

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

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





1.15 Разделитель

Вы можете использовать разделитель "==", чтобы разбить диаграмму на несколько этапов.

```
@startuml
```

```
== Initialization ==
```

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

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

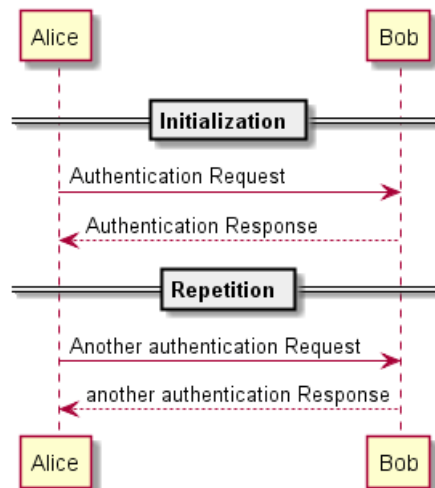
```
== Repetition ==
```

```
Alice -> Bob: Another authentication Request
```

```
Alice <-- Bob: another authentication Response
```

```
@enduml
```





1.16 Ссылки

Вы можете использовать ссылки в диаграммах с помощью ключевого слова `ref over`.

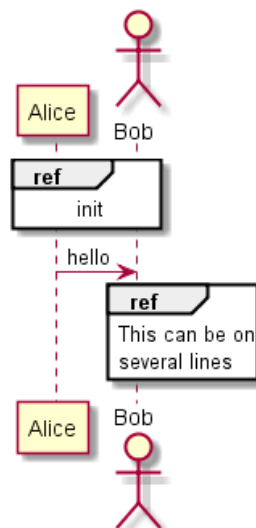
```

@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 Задержка на диаграммах

Вы можете использовать конструкцию `...` для представления временной задержки в процессе на диаграмме. При необходимости можно снабдить задержку комментарием.

```

@startuml
  
```

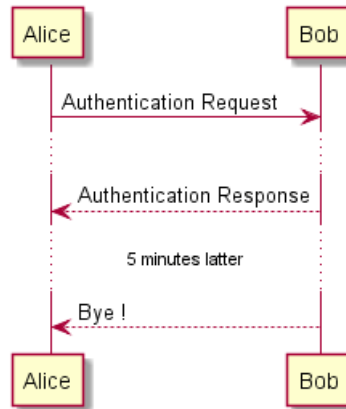


```

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

```

```
@enduml
```



1.18 Промежутки

Вы можете использовать ||| чтобы показать промежутки в диаграммах..

Так же возможно указать промежуток в пикселях.

```
@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 Активация и деактивация линии существования

activate и deactivate используются чтобы обозначить активацию участника.

Линия существования появляется в момент активации участника.

activate и deactivate применяются к предыдущему сообщению.

destroy обозначает конец линии существования участника.

```
@startuml
participant User

User -> A: DoWork
activate A

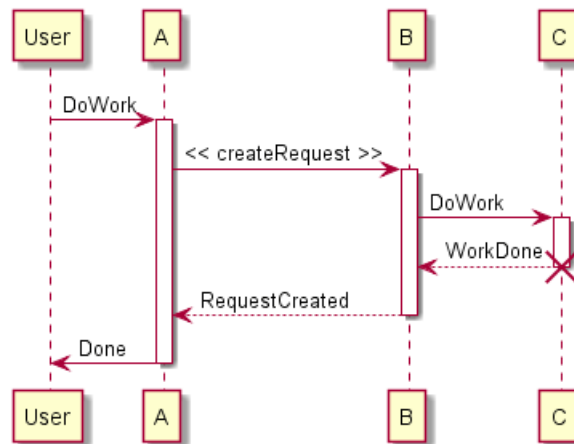
A -> B: << createRequest >>
activate B

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

B --> A: RequestCreated
deactivate B

A -> User: Done
deactivate A

@enduml
```



Можно использовать вложенные линии существования, и возможно добавлять цвет линии существования

```
@startuml
participant User

User -> A: DoWork
activate A #FFBBBB

A -> A: Internal call
activate A #DarkSalmon

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

B --> A: RequestCreated
```




```

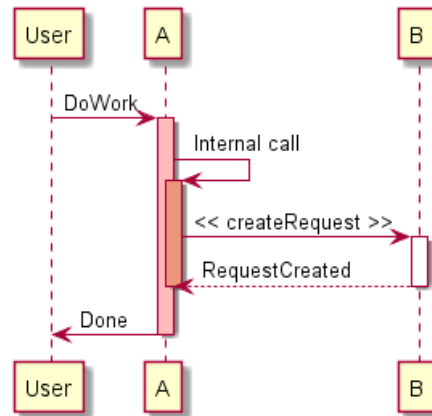
deactivate B
deactivate A
A -> User: Done
deactivate A

```

```

@enduml

```



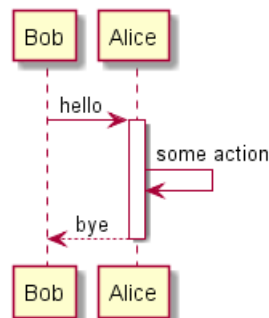
1.20 Return

A new command `return` for generating a return message with optional text label. The point returned to is the point that caused the most recently activated life-line. The syntax is simply `return label` where label, if provided, can be any string acceptable on conventional messages.

```

@startuml
Bob -> Alice : hello
activate Alice
Alice -> Alice : some action
return bye
@enduml

```



1.21 Отображение создания участника процессом

Вы можете использовать ключевое слово `create` перед декларацией сообщения для акцентирования факта, что принимающий участник *создается* данным сообщением.

```

@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

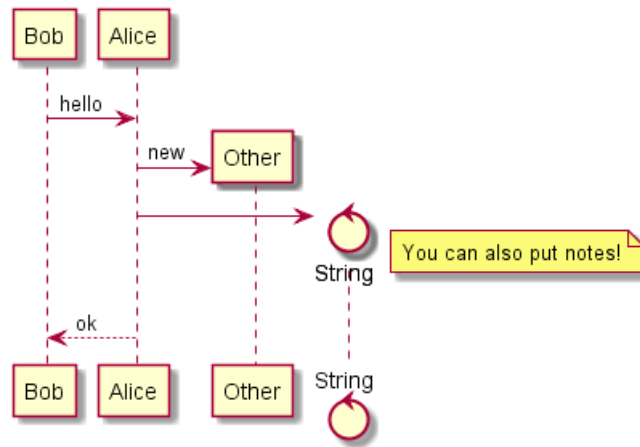
create control String
Alice -> String
note right : You can also put notes!

```



```
Alice --> Bob : ok
```

```
@enduml
```



1.22 Входящие и исходящие сообщения

Вы можете использовать входящие или исходящие стрелки если вы хотите сфокусироваться на части диаграммы.

Используйте квадратные скобки для указания левой "[" или правой "]" стороны диаграммы

```
@startuml
```

```
[-> A: DoWork
```

```
activate A
```

```
A -> A: Internal call
```

```
activate A
```

```
A ->] : << createRequest >>
```

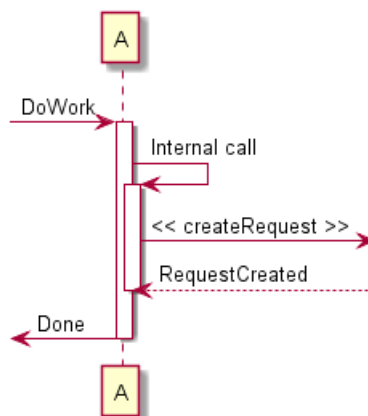
```
A<--] : RequestCreated
```

```
deactivate A
```

```
[<- A: Done
```

```
deactivate A
```

```
@enduml
```



Вы также можете использовать следующий синтаксис:

```
@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 Шаблоны и отметки

Можно добавить шаблоны к участникам используя << и >>.

В шаблоне вы можете добавить отмеченного участника в цветном круге используя синтаксис (X,color).

```

@startuml

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

```

```

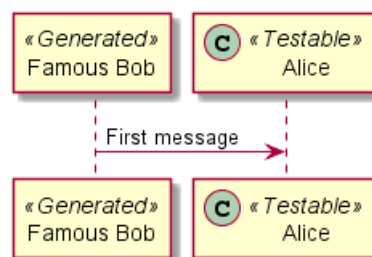
Bob->>Alice: First message

```

```

@enduml

```



По умолчанию, символ *guillemet* используется для отображения шаблона. Вы можете изменить это поведение, используя `skinparam guillemet`:

```

@startuml

skinparam guillemet false

```



```

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

```

```

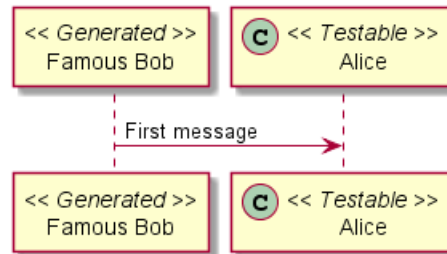
Bob->Alice: First message

```

```

@enduml

```



```

@startuml

```

```

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

```

```

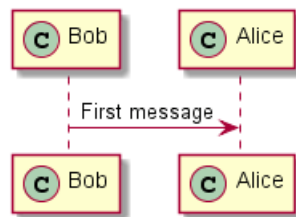
Bob->Alice: First message

```

```

@enduml

```



1.24 Больше информации в заголовках

Вы можете использовать форматирование на Creole для заголовков.

```

@startuml

```

```

title __Simple__ **communication** example

```

```

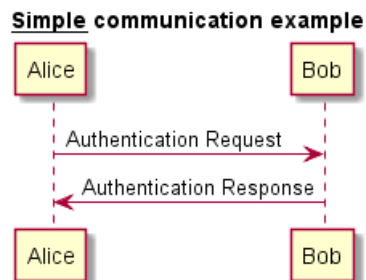
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Response

```

```

@enduml

```



С помощью последовательности символов \n вы можете добавить перевод строки в заголовок.

```

@startuml

```

```

title __Simple__ communication example\nnon several lines

```



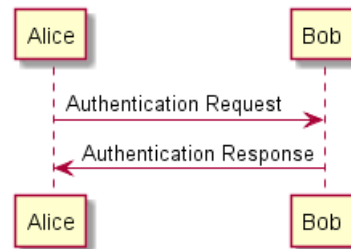
```

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

```

```
@enduml
```

**Simple communication example
on several lines**



Вы также можете задать заголовок на нескольких строках, используя ключевые слова `title` и `end title`.

```
@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

```

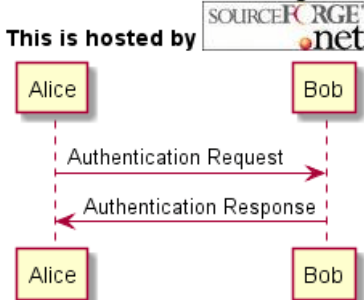
```

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

```

```
@enduml
```

**Simple communication example
on several lines and using html**



1.25 Группировка участников

Можно создать прямоугольник вокруг участников, используя команды `box` и `end box`.

Вы можете задать опциональный заголовок и цвет фона, после команды `the box`.

```
@startuml
```

```

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

```

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

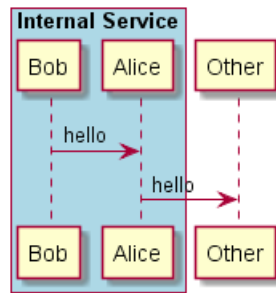


```

Alice -> Other : hello

@enduml

```



1.26 Удаление футера

Вы можете использовать ключевое слово `hide footbox` для удаления футера из диаграммы.

```

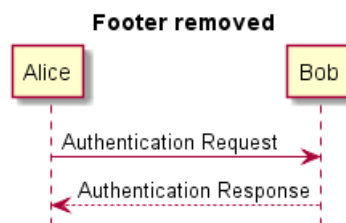
@startuml

hide footbox
title Footer removed

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

@enduml

```



1.27 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете изменить другие параметры отображения, как видно из следующих примеров:

```

@startuml
skinparam sequenceArrowThickness 2
skinparam roundcorner 20
skinparam maxmessageSize 60
skinparam sequenceParticipant underline

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

```



```

User -> A: DoWork
activate A

A -> B: Create Request
activate B

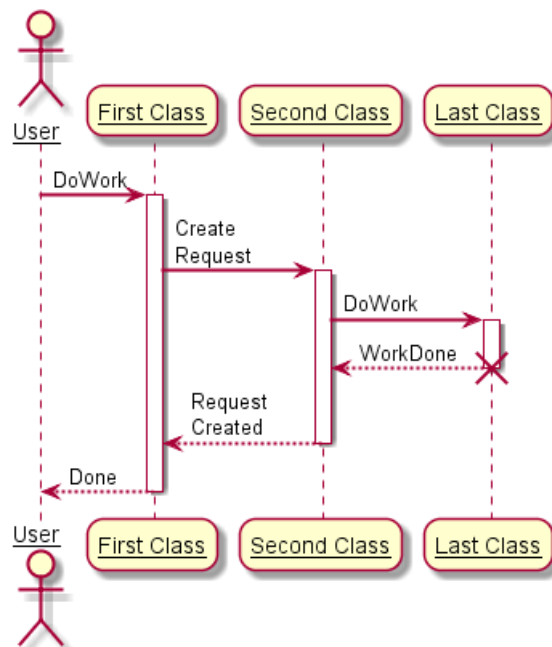
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



```

@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

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

ParticipantBorderColor DeepSkyBlue
ParticipantBackgroundColor DodgerBlue
ParticipantFontName Impact
ParticipantFontSize 17
ParticipantFontColor #A9DCDF

ActorBackgroundColor aqua
ActorFontColor DeepSkyBlue

```



```

ActorFontSize 17
ActorFontName 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

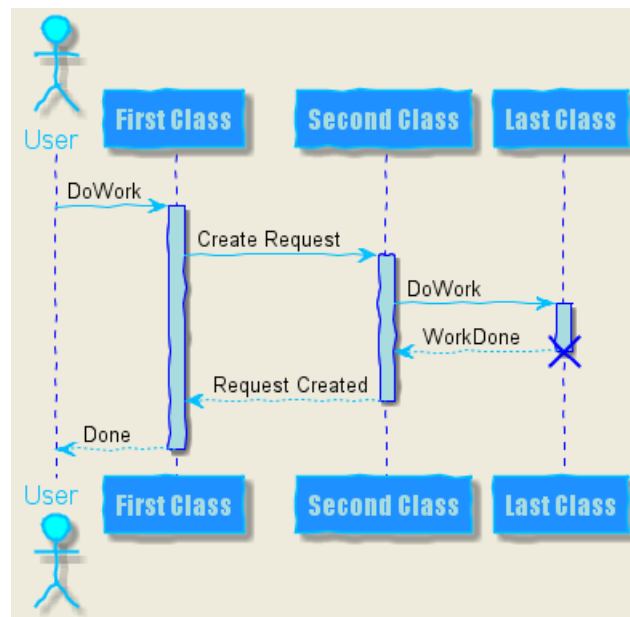
B -> C: DoWork
activate C
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



1.28 Изменение отступов

Вы можете изменить некоторые настройки отступов

```

@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1

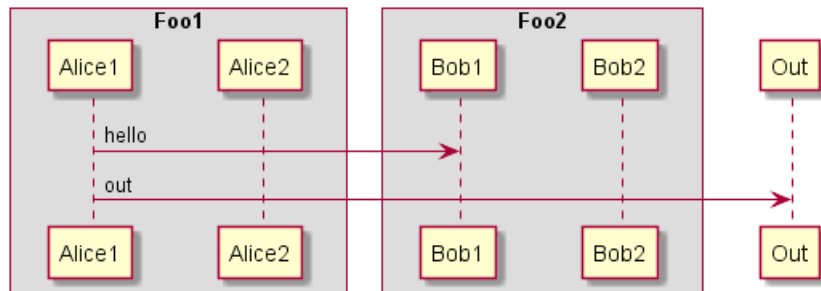
```




```

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

```



2 Диаграмма прецедентов

Рассмотрим несколько примеров:

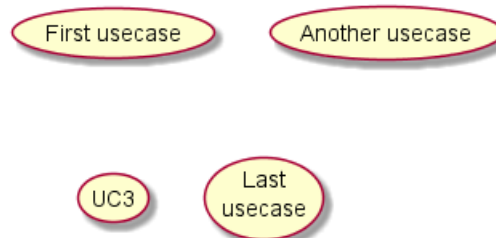
Заметьте, что Вы можете отключить тени, используя команду `skinparam shadowing false`.

2.1 Прецеденты

Прецеденты заключаются в две скобки (потому что две скобки выглядят как овал).

Вы можете использовать `usecase` для создания прецедента. также вы можете создать псевдоним, используя `as keyword`. Этот псевдоним будет использоваться позже во время определения связей

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



2.2 Актёры

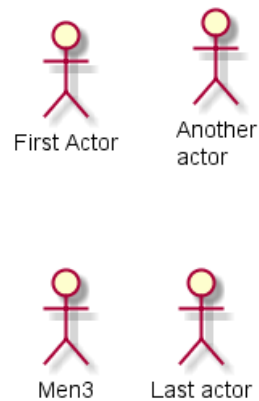
Актеры обозначаются заключёнными между двумя точками.

Также Вы можете использовать ключевое слово `actor` для определения актёра. И вы можете создать псевдоним, используя ключевое слово `as`. Этот псевдоним будет использован позднее, при определении отношений.

Мы увидим, что определения актеров не обязательны.

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





2.3 Описание прецедентов

Если вы хотите описание на несколько строк, можете использовать кавычки.

Вы также можете использовать следующие разделители: -- .. == __. И вы можете вставлять заголовки внутри разделителей.

```
@startuml
```

```
usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.
```

```
--
```

```
Several separators are possible.
```

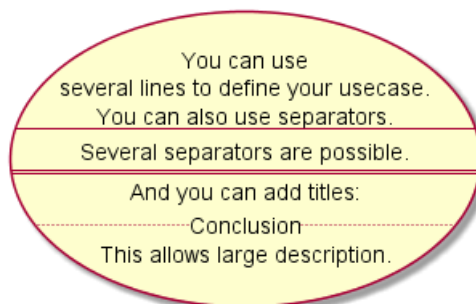
```
==
```

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

```
..Conclusion..
```

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

```
@enduml
```



2.4 Простой пример

Для соединения актеров и прецедентов, используется стрелка -->.

Чем больше тире - в стрелке, тем она длиннее. Вы можете добавить метку на стрелку, добавив символ : при определении стрелки.

В этом примере, вы можете видеть, что *User* не определён ранее и используется как актёр.

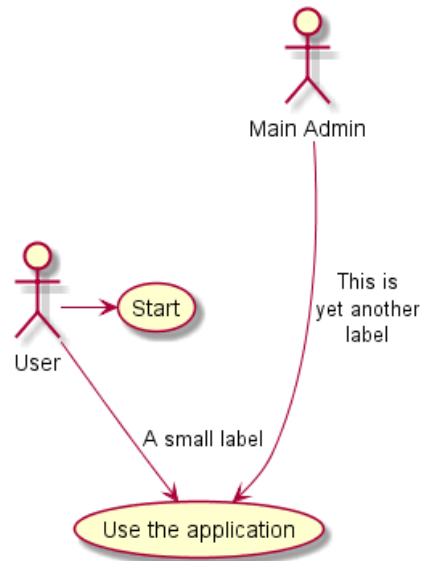
```
@startuml
```

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

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



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



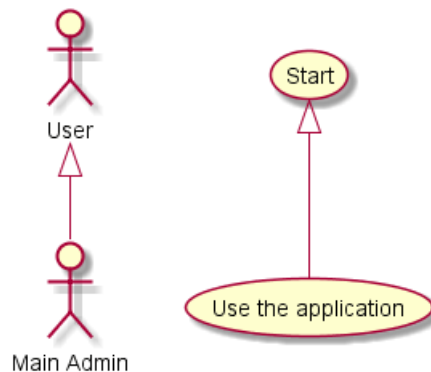
2.5 Расширение

Если один актёр/прецедент расширяют другой, вы можете использовать символ <|--.

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

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

@enduml
```



2.6 Использование заметок

Вы можете использовать ключевые слова `note left of`, `note right of`, `note top of`, `note bottom of` чтобы создать заметку относящуюся к одному объекту.

Заметка так же может быть создана с помощью ключевого слова `note`, а затем прикреплена к другому объекту используя символ `...`

```
@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)
```

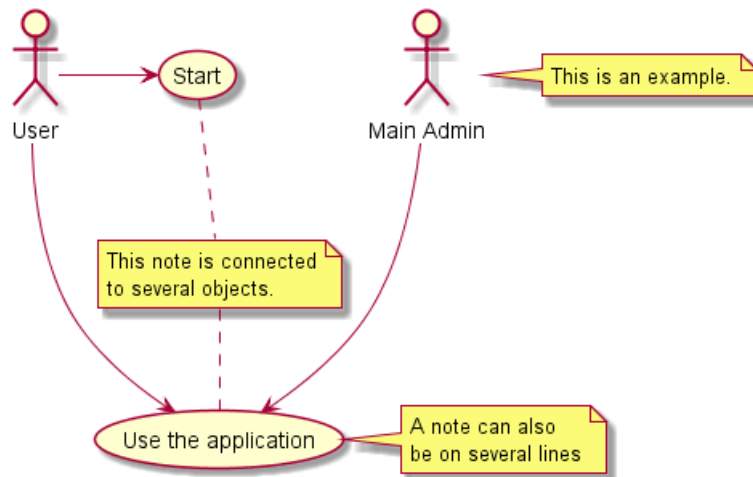
```
  A note can also  
  be on several lines  
end note
```

```
note "This note is connected\nto several objects." as N2
```

```
(Start) .. N2
```

```
N2 .. (Use)
```

```
@enduml
```



2.7 Шаблоны

Вы можете добавить шаблоны когда определяете актёров и прецеденты, используя << и >>.

```
@startuml
```

```
User << Human >>
```

```
:Main Database: as MySql << Application >>
```

```
(Start) << One Shot >>
```

```
(Use the application) as (Use) << Main >>
```

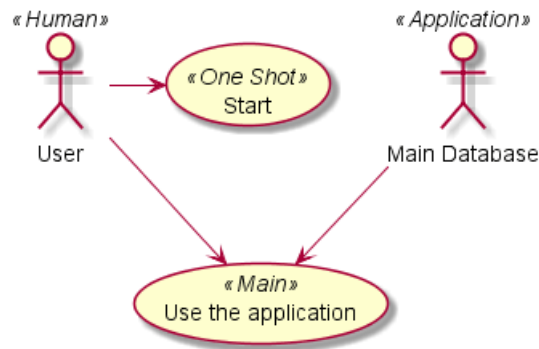
```
User -> (Start)
```

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

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

```
@enduml
```



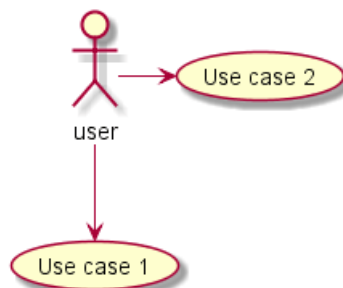


2.8 Смена направления стрелок

По умолчанию, связи между классами имеют два типа -- и вертикально ориентированны. можно использовать горизонтальные связи, с помощью написание одного типа (или точки), вот так:

```

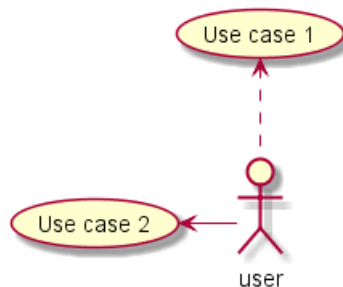
@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml
  
```



Вы так же можете изменить направление с помощью переворачивания связи:

```

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

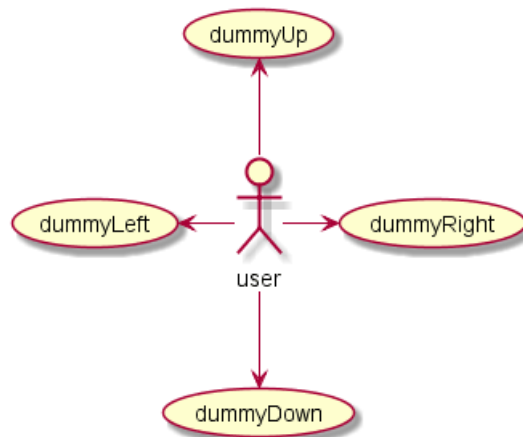


Так же возможно сменить направление добавляя ключевые слова left, right, up или down внутри стрелки:

```

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





Вы можете записать короче, используя только первый символ названия направления (например, `-d-` вместо `-down-`) или первые два символа (`-do-`).

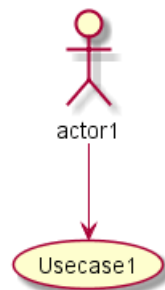
Пожалуйста, помните, что Вы не должны использовать эту функциональность без реальной необходимости: *GraphViz* обычно даёт хороший результат без дополнительных настроек.

2.9 Разделение диаграмм

Ключевое слово `newpage` используется для разделения диаграмм на несколько страниц или изображений.

```

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



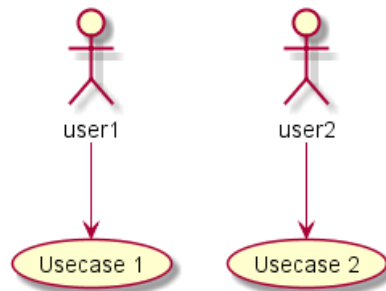
2.10 Направление слева направо

Общее поведение по умолчанию - построение диаграмм сверху вниз.

```

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

@enduml
  
```



Вы можете изменить направление на слева направо используя команду `left to right direction`. Часто результат с таким направлением выглядит лучше.

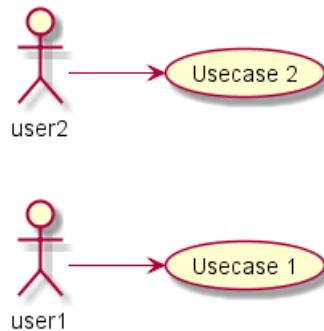
```
@startuml
```

```
left to right direction
```

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

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

```
@enduml
```



2.11 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете задать цвет или шрифт для актёров или прецедентов с шаблонами.

```
@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)

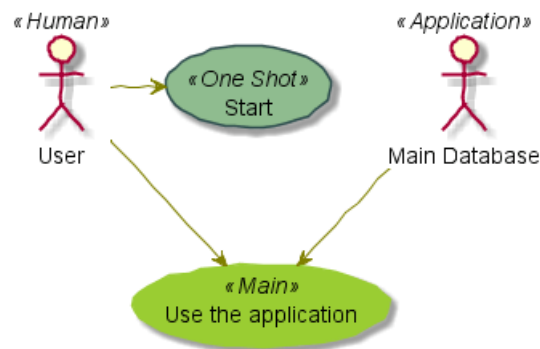
```

```

MySql --> (Use)

```

@enduml

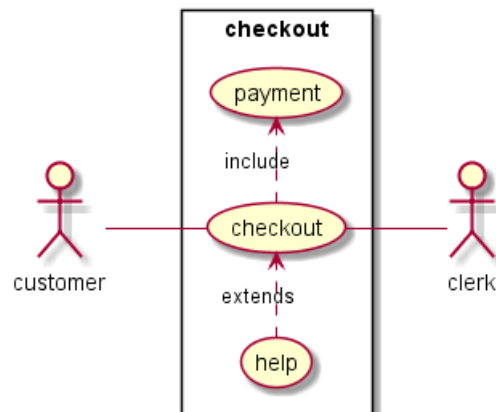


2.12 Полноценный пример

```

@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 Диаграмма классов

3.1 Отношения между классами

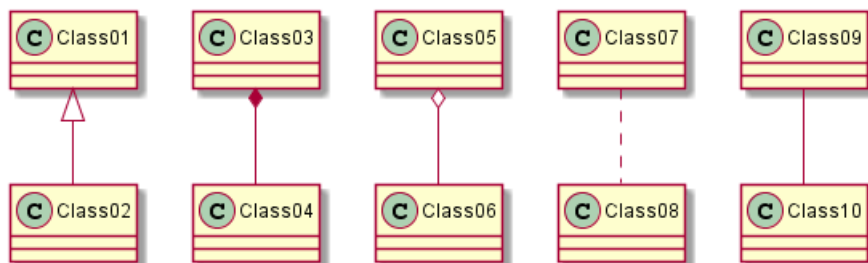
Отношения между классами определяются с помощью следующих символов:

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

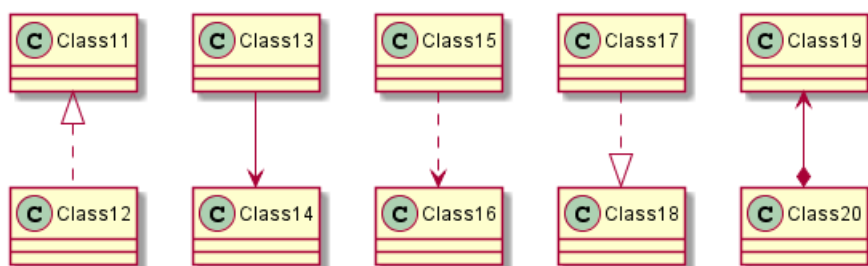
Можно заменить - на .., чтобы создать пунктирную линию.

Зная эти правила можно нарисовать следующие изображения:

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

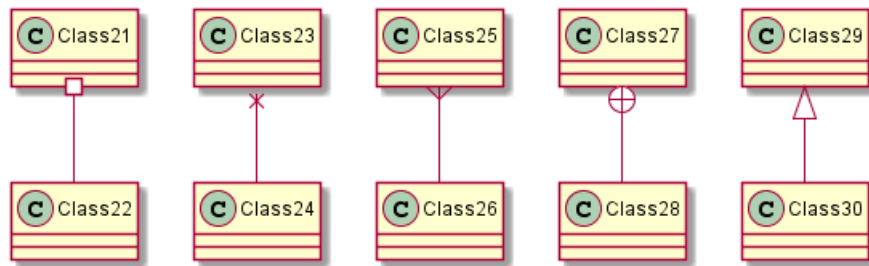


```
@startuml
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
Class19 <--* Class20
@enduml
```



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



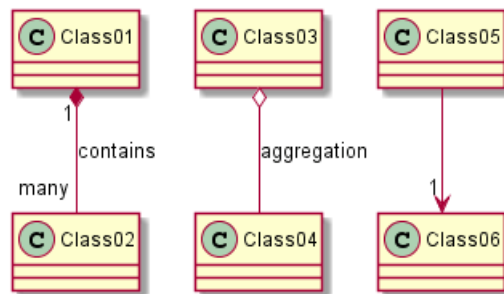


3.2 Метки на отношениях

Для отношения можно добавить метку. Делается это с помощью указания символа :, после которого указывается текст метки.

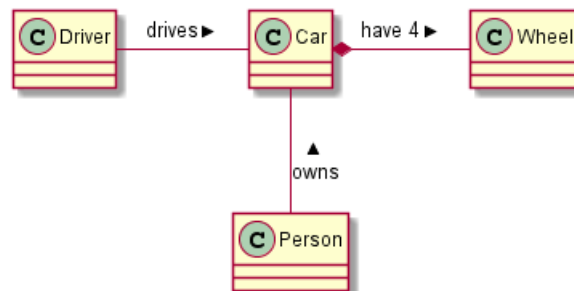
Для указания количества элементов на каждой стороне отношения можно использовать двойные кавычки "".

```
@startuml
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
@enduml
```



Вы можете добавить дополнительные стрелки < или > в начале или в конце метки, указывающие на использование одного из объектов другим объектом.

```
@startuml
class Car
Driver - Car : drives >
Car *- Wheel : have 4 >
Car -- Person : < owns
@enduml
```



3.3 Добавление методов

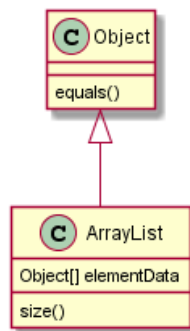
Для объявления полей и методов вы можете использовать символ `:`, после которого указывается имя поля или метода.

Для определения того, что вы указали метод или поле, система ищет скобки.

```
@startuml
Object <|-- ArrayList

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

@enduml
```



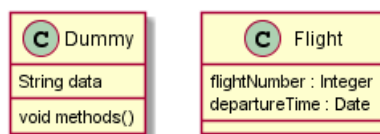
Также можно группировать все поля и методы между фигурными скобками `{}`.

Синтаксис порядка описания типа/имени довольно гибок.

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

class Flight {
    flightNumber : Integer
    departureTime : Date
}

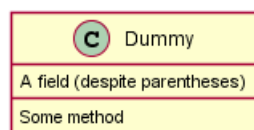
@enduml
```



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 Указание видимости

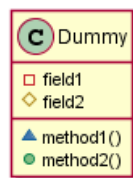
Определяя методы и поля, вы можете использовать символы указания видимости, приведённые в таблице ниже:

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

```
@startuml
```

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

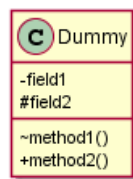
```
@enduml
```



Убрать значки можно командой `skinparam classAttributeIconSize 0`:

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

```
@enduml
```



3.5 Абстрактные и статические

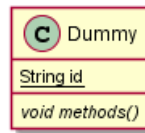
Вы можете определить статические или абстрактные методы и поля используя модификаторы `{static}` и `{abstract}` соответственно.

Эти модификаторы могут располагаться как в начале так и в конце строки. Вы так же можете использовать `{classifier}` как замену для `{static}`.

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



@enduml



3.6 Расширенное тело класса

По умолчанию, методы и поля автоматически группируются PlantUML. Вы можете использовать разделители, чтобы определить собственный порядок полей и методов. Можно использовать следующие разделители:

-- .. == __.

Вы также можете использовать заголовки внутри разделителей:

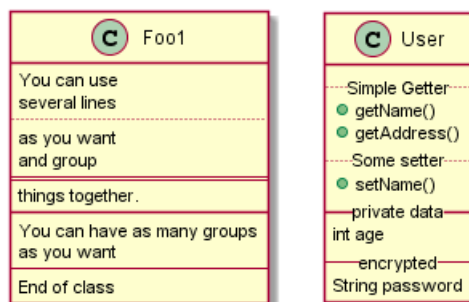
```

@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 Заметки и шаблоны

Шаблоны задаются ключевым словом `class`, `<< и >>`.

Также вы можете создать заметку, используя ключевые слова `note left of`, `"note right of"`, `note top of`, `note bottom of`.

Вы также можете добавить заметку к последнему определённому классу, используя `note left`, `note right`, `note top`, `note bottom`.

Ключевым словом `note` легко создать заметку без привязки, а после, используя символ `..`, привязать её к другим объектам.

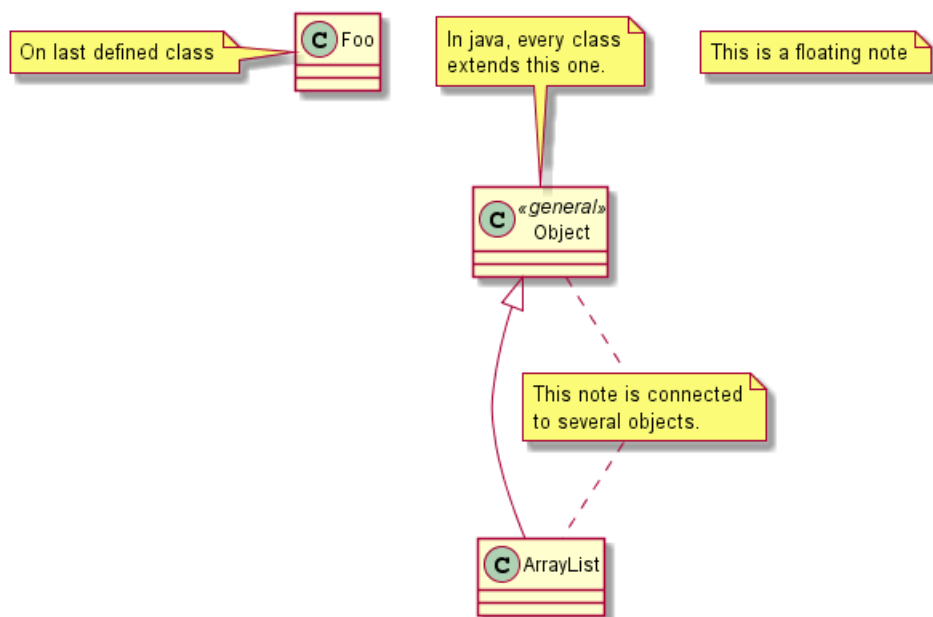
```
@startuml
class Object << general >>
Object <|--- ArrayList
```

```
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 Больше о заметках

Также допускается использование некоторых HTML-тегов, таких как:

- ``
- `<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

Заметка может быть из нескольких строк.

Можно определить заметку для класса, заданного последним, с помощью `note left`, `note right`, `note top`, `note bottom`.

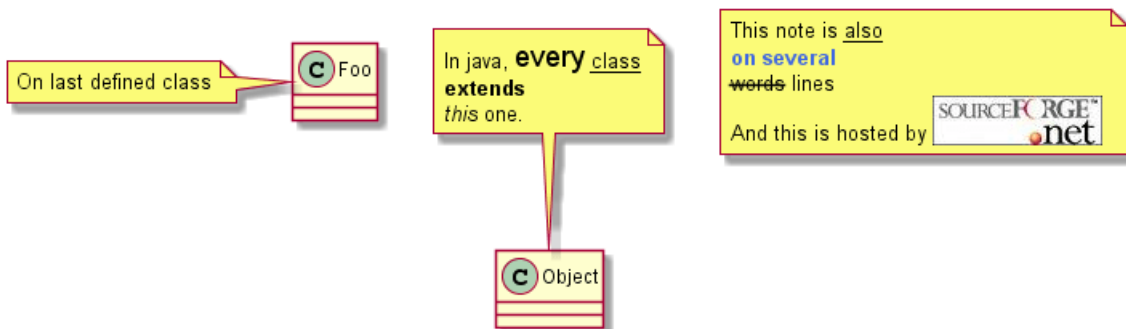
```
@startuml

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 Заметки на связях

Возможно добавить заметку на связь, сразу после определения связи, используя `note on link`.

Вы также можете использовать `note left on link`, `note right on link`, `note top on link`, `note bottom on link` если вы хотите изменить относительную позицию заметки с надписью.

```
@startuml

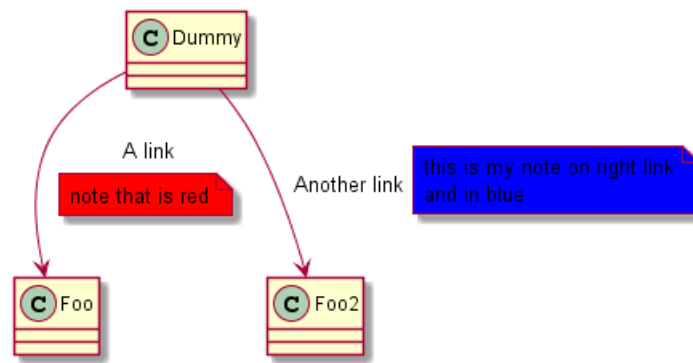
class Dummy
Dummy --> Foo : A link
note on link #red: note that is red

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

@enduml
```



@enduml



3.10 Абстрактные классы и интерфейсы

Вы можете определить класс как абстрактный, используя ключевые слова `abstract` или `abstract class`.

Классы будут нарисованы *курсивом*.

Вы также можете использовать ключевые слова `interface`, `annotation` и `enum`.

@startuml

```

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

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

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

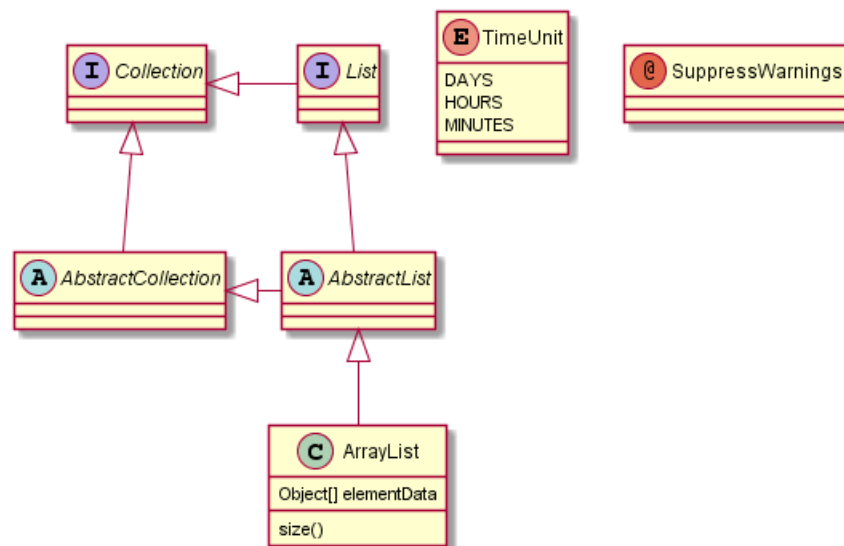
class ArrayList {
    Object[] elementData
    size()
}

enum TimeUnit {
    DAYS
    HOURS
    MINUTES
}

annotation SuppressWarnings

@enduml

```



3.11 Использование не буквенных символов

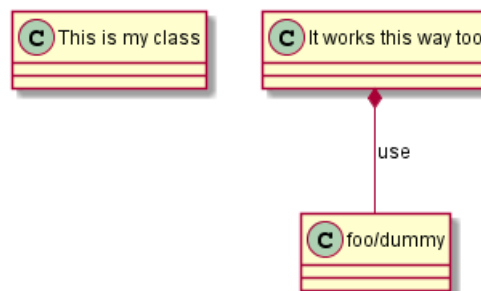
Если вы хотите использовать не буквенные символы в названии класса (или другого объекта), вы можете использовать 2 способа :

- Использовать ключевое слово `as` в определении класса
- Поставить кавычки `" "` вокруг имени класса

```

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

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



3.12 Скрытие атрибутов, методов...

Вы можете управлять видимостью классов с помощью команды `hide/show`.

Базовая команда это - `hide empty members`. Команда скроет атрибуты или методы, если они пусты.

Вместо `empty members`, вы можете использовать:

- `empty fields` или `empty attributes` для пустых полей,
- `empty methods` для пустых методов,
- `fields` или `attributes`, которые скроют поля, даже если они были описаны,
- `methods`, которые скроют методы, даже если они были описаны,
- `members`, которые скроют поля и методы, даже если они были описаны,



- circle для круглых символов перед именем класса,
- stereotype для шаблона.

Вы также можете указать ключевое слово, сразу за hide или show:

- class для всех классов,
- interface для всех интерфейсов,
- enum для всех перечислений,
- <<foo1>> для классов, к которым применен шаблон с помощью *foo1*,
- имя существующего названия класса.

Для определения большого набора, состоящего из правил и исключений, можно использовать несколько команд show/hide.

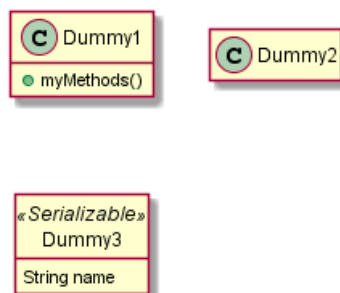
```
@startuml
class Dummy1 {
+myMethods()
}

class Dummy2 {
+hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

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

@enduml
```



3.13 Скрытие классов

Вы также можете использовать команду show/hide, чтобы скрывать классы.

Это может быть полезно, если вы определяете большой !подключенный файл, и если вы хотите скрыть некоторые классы после включения.

```
@startuml
class Foo1
class Foo2

Foo2 *-- Foo1
```



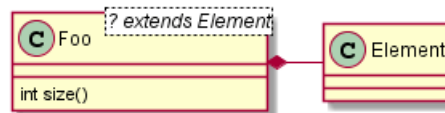
```
hide Foo2
@enduml
```



3.14 Использование дженериков

Вы также можете использовать скобки < и > чтобы указать на использование дженериков в классе.

```
@startuml
class Foo<? extends Element> {
    int size()
}
Foo *- Element
@enduml
```



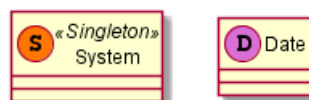
Вы можете отключить отрисовку этих элементов, используя команду `skinparam genericDisplay old`.

3.15 Определение метки

Обычно, метка с буквой (C, I, E or A) используется для классов, интерфейсов, перечисления и абстрактных классов.

Но также вы можете использовать свою собственную метку для класса, когда создаёте шаблон, добавляя одну букву и цвет, как в этом примере:

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



3.16 Пакеты

Вы можете определить пакет, используя ключевое слово `package`, с возможностью объявить ещё и цвет его фона, (используя html-код цвета или его имя).

Обратите внимание, что определения пакета могут быть вложенными.



```

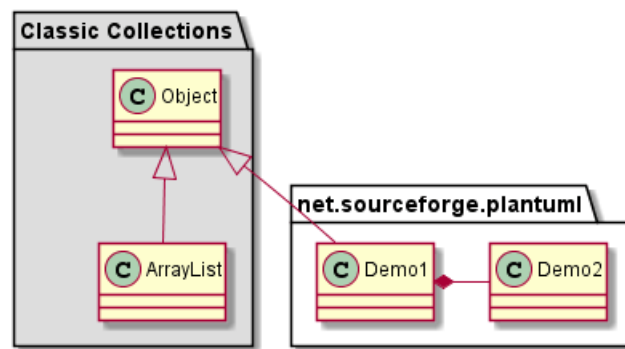
@startuml

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

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

@enduml

```



3.17 Стили пакетов

Доступны различные стили для пакетов.

Можно задать стили по умолчанию с помощью команды: `skinparam packageStyle`, или применить шаблоны на пакет:

```

@startuml
scale 750 width
package foo1 <<Node>> {
    class Class1
}

package foo2 <<Rectangle>> {
    class Class2
}

package foo3 <<Folder>> {
    class Class3
}

package foo4 <<Frame>> {
    class Class4
}

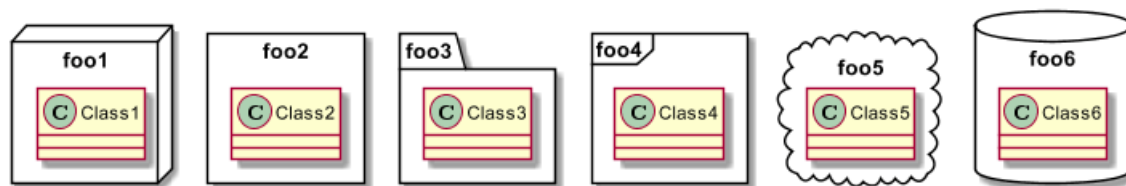
package foo5 <<Cloud>> {
    class Class5
}

package foo6 <<Database>> {
    class Class6
}

```



@enduml



Вы также можете определить связи между пакетами, как в данном примере:

@startuml

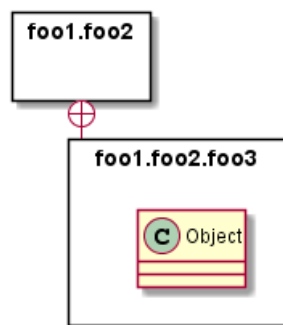
```
skinparam packageStyle rectangle
```

```
package foo1.foo2 {
}
```

```
package foo1.foo2.foo3 {
    class Object
}
```

```
foo1.foo2 +-- foo1.foo2.foo3
```

@enduml



3.18 Пространства имён

В пакетах, имя класса является уникальным идентификатором этого класса. Это значит, что у вас не может быть двух одноименных классов в разных блоках.

В этом случае, вам следует использовать пространства имен вместо пакетов.

Вы можете ссылаться на классы из других пространств имён по их полному определению. Классы из пространства имён по умолчанию определяются ведущей точкой.

Обратите внимание, что вы не обязаны явно создавать пространство имен: полностью определенный класс автоматически попадает в правильное пространство имен.

@startuml

```
class BaseClass
```

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

```
    .BaseClass <|-- Meeting
}
```



```

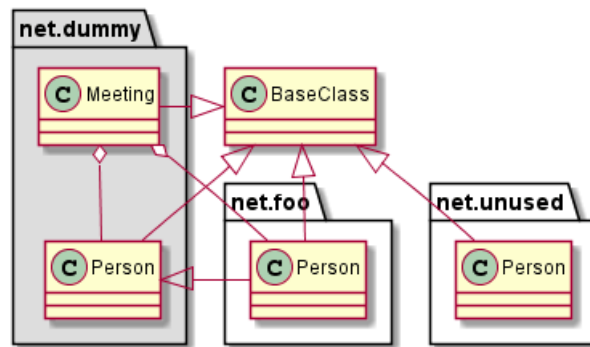
namespace net.foo {
  net.dummy.Person <|-- Person
  .BaseClass <|-- Person

  net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml

```



3.19 Автоматическое создание пространств имён

Вы также можете задать другой разделитель (не точку) используя команду : `set namespaceSeparator ???`.

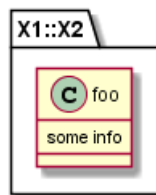
```

@startuml

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

@enduml

```



Вы можете отключить автоматическое создание пакетов используя команду `set namespaceSeparator none`.

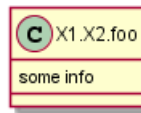
```

@startuml

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

@enduml

```

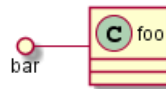


3.20 Lollipop интерфейс

Вы также можете задать lollipop интерфейсы на классах, используя следующий синтаксис:

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

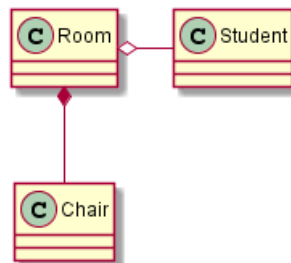
```
@startuml
class foo
bar ()- foo
@enduml
```



3.21 Изменение направления стрелок

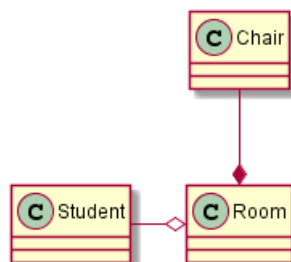
По умолчанию, связи между классами имеют два типа -- и вертикально ориентированны. Возможно создать горизонтальную связь, используя одно типе (or dot) вот так:

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



Вы можете изменить направление перевернув связь:

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



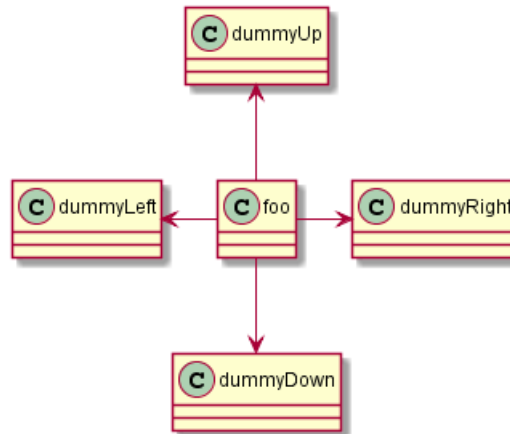
Также возможно изменять направление стрелок, добавляя ключевые слова left, right, up или down внутри стрелки:




```

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

```



Вы можете укоротить запись, используя только первую букву направления (например, `-d-` вместо `-down-`) или две первые буквы (`-do-`).

Заметьте, что вам не стоит пользоваться этой функциональностью без особой надобности: *Graphviz* обычно предоставляет хорошие результаты без дополнительной настройки.

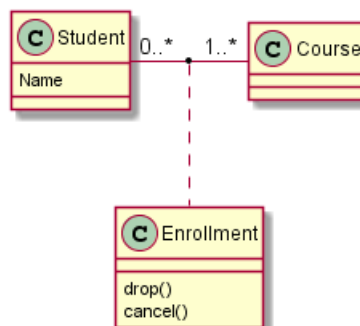
3.22 Ассоциация классов

Вы можете задать *ассоциацию класса* после того, как была задана связь между двумя классами, как в примере:

```

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

```



Вы можете задать это в другом направлении:

```

@startuml
class Student {

```

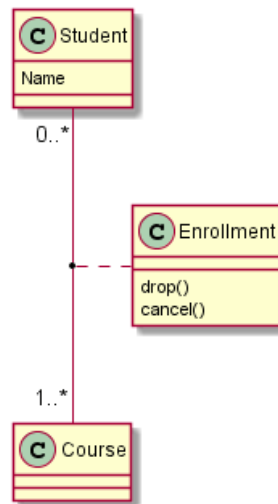


```

    Name
}
Student "0..*" -- "1..*" Course
(Student, Course) . Enrollment

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

```



3.23 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

```

@startuml

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

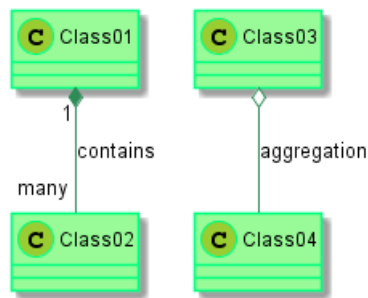
Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation

@enduml

```





3.24 Шаблоны со Skinparam

Вы можете задать цвет или шрифт для шаблонов классов.

```

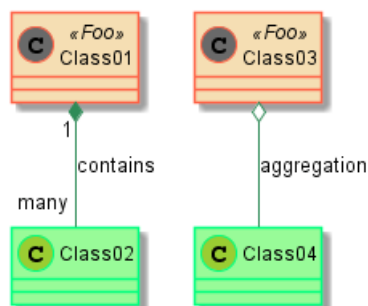
@startuml

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 Цветовой градиент

Можно объявить индивидуальный цвет для классов или примечаний, используя # обозначения.

Можно использовать как стандартные названия цветов, так и RGB-код.

Так же возможно использование градиента для фона, используя следующие символы для разделения пары цветов:

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



в зависимости от направления градиента

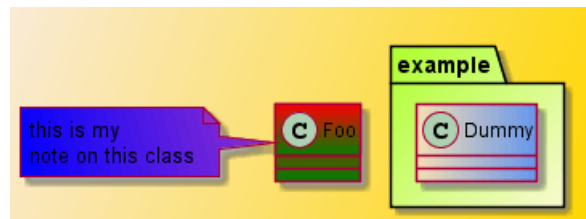
Например так :

```
@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 Помощь в расположении классов

Sometimes, the default layout is not perfect...

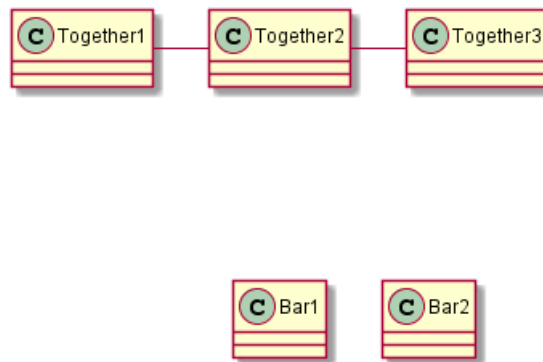
You can use `together` keyword to group some classes together : the layout engine will try to group them (as if they were in the same package).

You can also use hidden links to force the layout.

```
@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 Разделение больших файлов

Иногда могут получиться очень большие файлы изображений.

Вы можете использовать команду `page (hpages)x(vpages)` чтобы разделить создаваемое изображение на несколько файлов (страниц) :

`hpages` - это задание числа горизонтальных страниц, и `vpages` - это задание числа вертикальных страниц..

Здесь также можно использовать специфику `skinparam` настроек как цвета разделённых страниц, так и их границы (смотри пример).

```

@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

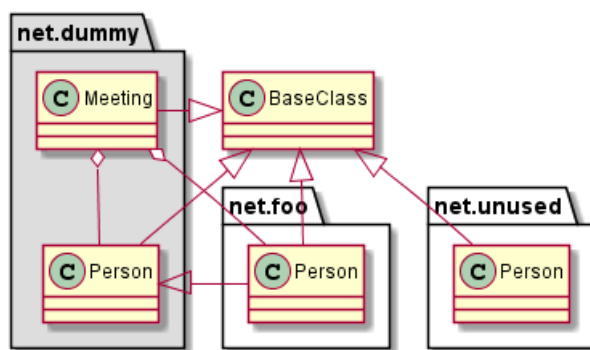
.BaseClass <|-- Meeting
}

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

  net.dummy.Meeting o-- Person
}

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

```



4 Диаграмма деятельности

4.1 Простая деятельность

Вы можете использовать (*) для начальных и конечных точек диаграммы деятельности.

В некоторых случаях, вы можете использовать (*top) чтобы указать что начальная точка должна быть в верху диаграммы.

Используйте --> для стрелок.

```
@startuml
```

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

```
@enduml
```



4.2 Метка на стрелках

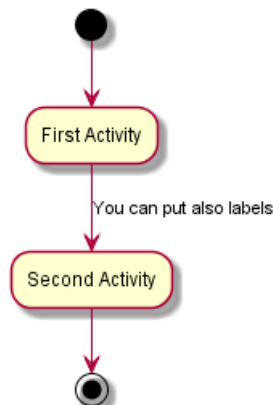
По умолчанию, стрелка начинается с последней использованной активности.

Вы можете пометить стрелку при помощи скобок [и] сразу после определения стрелки.

```
@startuml
```

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

```
@enduml
```



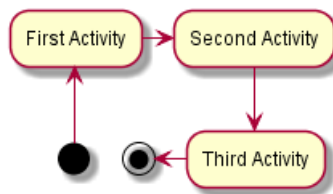
4.3 Изменение направления стрелки

Вы можете использовать -> для горизонтальных стрелок. Возможно задать направление стрелки используя следующий синтаксис:



- -down-> (default arrow)
- -right-> or ->
- -left->
- -up->

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

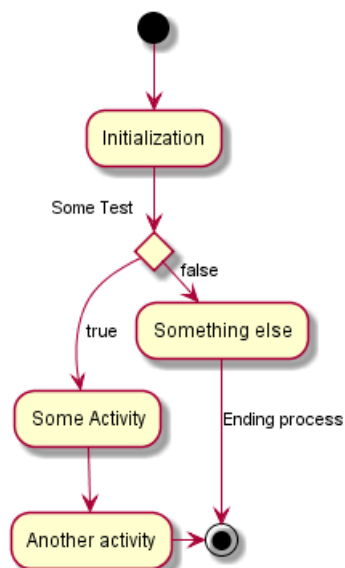


4.4 Ветвления

Вы можете использовать ключевые слова if/then/else чтобы определять ветки.

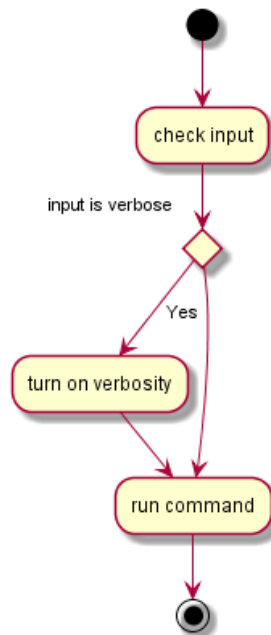
```
@startuml
(*) --> "Initialization"

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



К сожалению, вам иногда придётся повторять ту же активность в тексте диаграммы:

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



4.5 Больше о ветках

По умолчанию, ветка соединена к последней заданной активности, но возможно переопределить это и задать связь с помощью ключевого слова `if`.

Также возможно создавать вложенные ветки.

```
@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
```



```

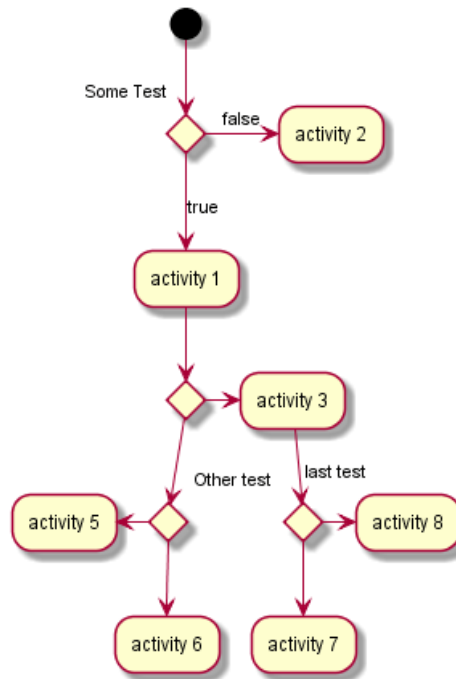
->[false] "activity 2"

endif

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

@enduml

```



4.6 Синхронизация

Вы можете использовать `=== code ===`, чтобы отобразить барьеры синхронизации.

```

@startuml

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

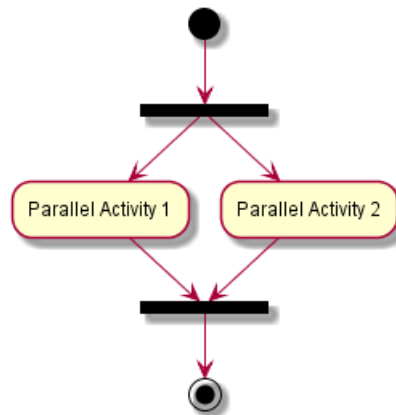
===B1=== --> "Parallel Activity 2"
--> ===B2===

--> (*)

@enduml

```





4.7 Длинное описание активности

Когда вы задаёте активность, вы можете разделить её описание на несколько линий. Вы также можете добавить \n в описание.

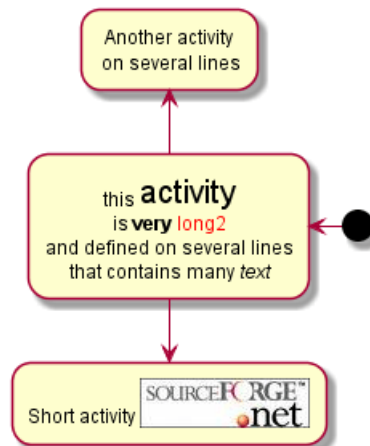
Вы также можете задать короткий код активности в помощью ключевого слова as. Этот код может быть использован позже в описании диаграммы.

```

@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 Заметки

Вы можете добавить заметки к активности используя команды `note left`, `note right`, `note top` or `note bottom`, Сразу после описания активности. к которой вы хотите прикрепить заметку.

Если вы хотите прикрепить заметку к точку начала, задайте метку в самом начале описания диаграммы.

Вы также можете создать заметку на нескольких линиях, используя ключевое слово `endnote`.

```

@startuml
  
```

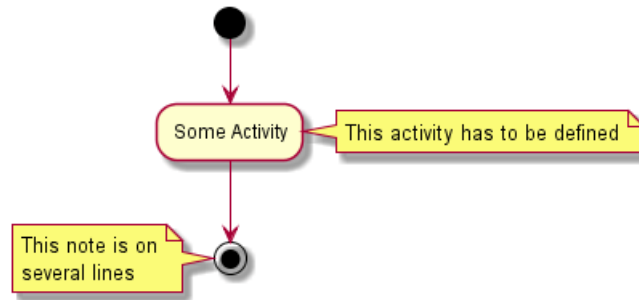


```

(*) --> "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 Разделы

Вы можете задать раздел используя ключевое слово `partition`, и опционально задать цвет фона для своего раздела (Используя код цвета `html` или название цвета)

Когда вы задаёте активность, они автоматически попадают в последнюю заданную активность.

Вы можете закрыть раздел используя закрывающую скобку `}`.

```

@startuml

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

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

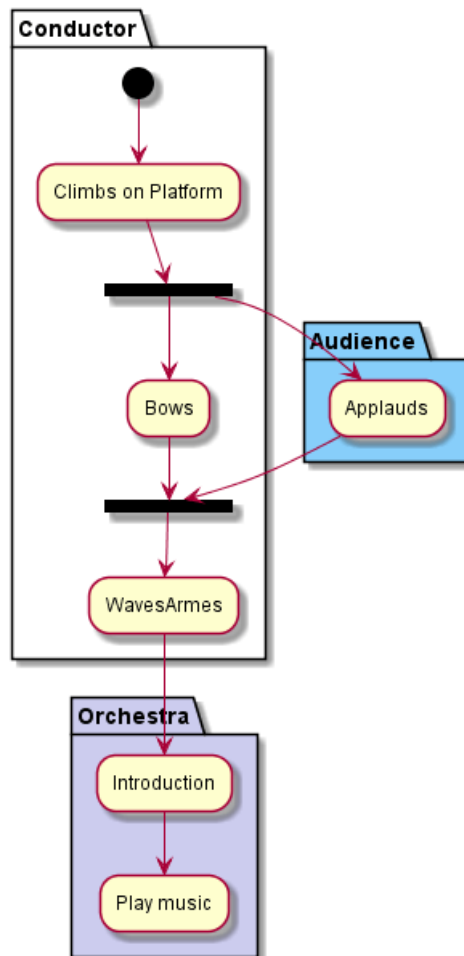
partition Conductor {
  Bows --> === S2 ===
  --> WavesArmes
  Applauds --> === S2 ===
}

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

@enduml

```





4.10 Skinparam

Вы можете использовать команду `skinparam` чтобы изменить цвет и шрифт рисования.

Вы можете использовать команду :

- В определении диаграммы, как любую другую команду,
- В подключаемом файле,
- В конфигурационном файле, подставленный в командной строке ANT задания.

Вы можете задать определённый цвет и шрифт для активностей с шаблоном.

```
@startuml
```

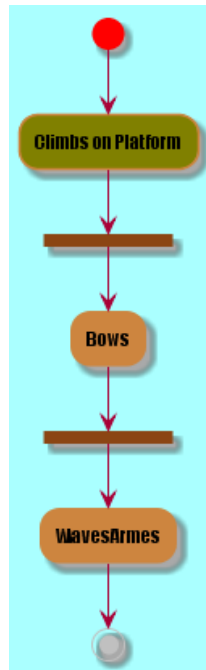
```
skinparam backgroundColor #AAFFFF
skinparam activity {
  StartColor red
  BarColor SaddleBrown
  EndColor Silver
  BackgroundColor Peru
  BackgroundColor<< Begin >> Olive
  BorderColor Peru
  FontName Impact
}
```

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



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

```
@enduml
```



4.11 Восьмиугольник

Вы можете изменить форму активностей на восьмиугольник, используя команду `skinparam activityShape octagon`.

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

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

```
@enduml
```



4.12 Полноценный пример

```
@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

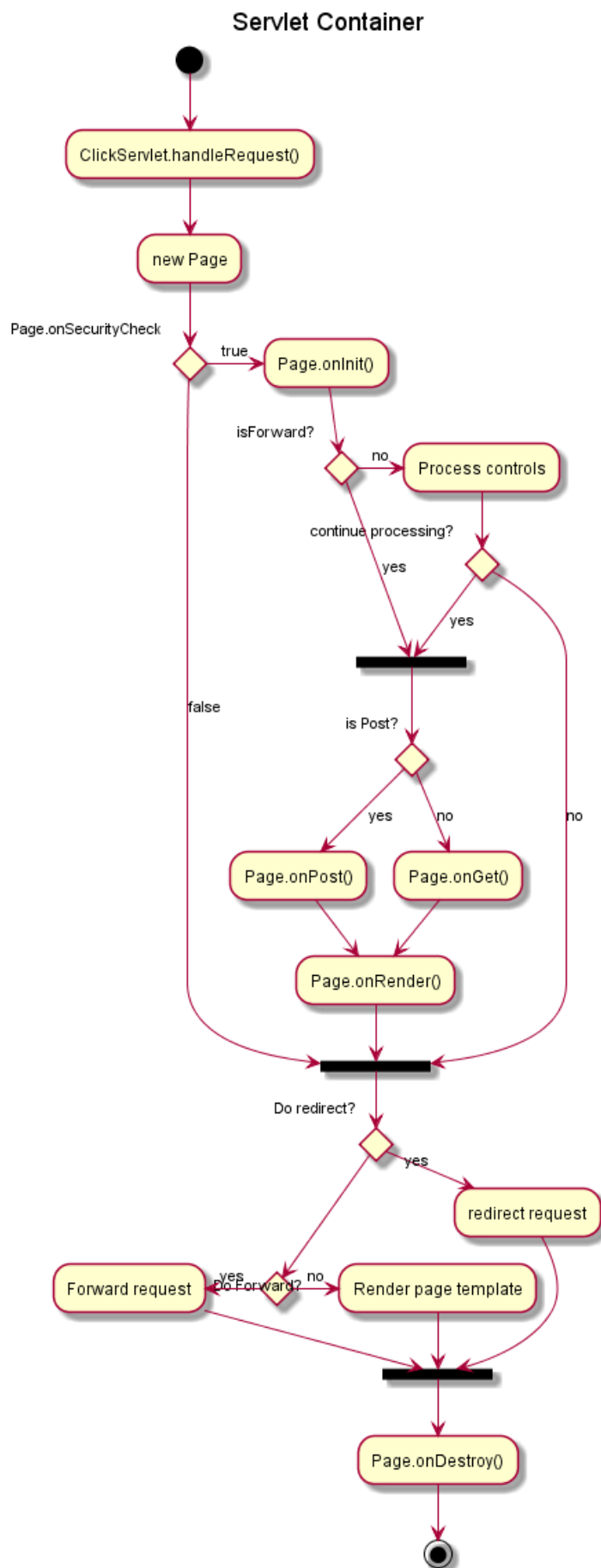
else
  -->[false] ===REDIRECT_CHECK===
endif

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 Диаграмма активности (бета)

Текущий синтаксис диаграммы активности имеет несколько ограничений и недостатков (например, её сложно поддерживать).

Таким образом, новый синтаксис и реализация предложены как **бета версия** пользователям (начиная с V7947), так что мы сможем определить новый формат и синтаксис.

Другое преимущество этой новой реализации, это то, что для неё не будет требоваться установленный Graphviz (как для диаграмм последовательностей).

Новый синтаксис заменит старый. Однако, по причине совместимости, старый синтаксис всё ещё будет распознаваться, чтобы обеспечивать *восходящую совместимость*.

Пользователи будут просто поощряться при миграции на новый синтаксис.

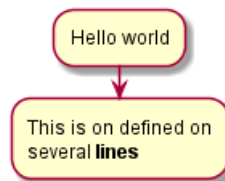
5.1 Простая активность

Описания активностей начинаются с : и заканчиваются ;.

Форматировать текст возможно, используя синтаксис creole.

Активности косвенно связаны в порядке их определения.

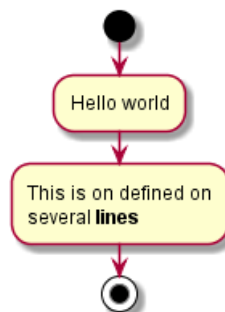
```
@startuml
:Hello world;
:This is on defined on
several **lines**;
@enduml
```



5.2 Старт/Стоп

Вы можете использовать ключевые слова `start` и `stop`, чтобы обозначать начало и конец диаграммы.

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



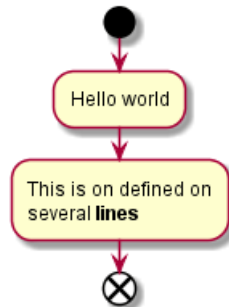
Вы также можете использовать ключевое слово `end`.



```

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

```



5.3 Условия

Вы можете использовать ключевые слова `if`, `then` и `else`, чтобы добавить проверяющие условия на вашу диаграмму. Описания можно добавить, используя круглые скобки.

```

@startuml

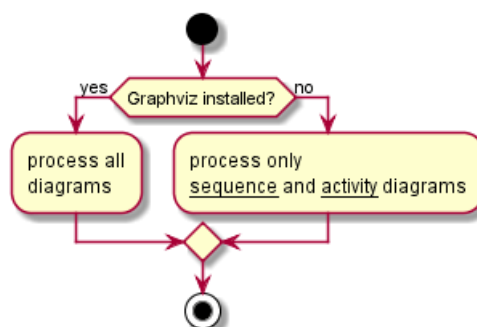
start

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

stop

@enduml

```



Вы можете использовать ключевые слова `elseif`, чтобы создать несколько проверок:

```

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

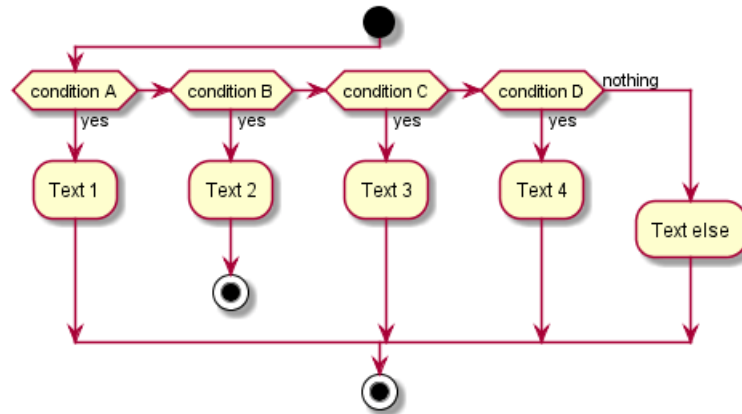
```



```

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

```



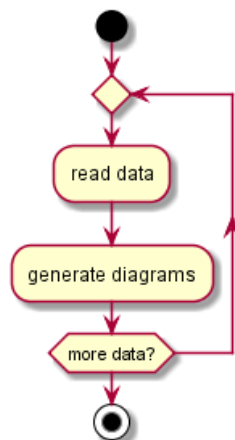
5.4 Повторяющийся цикл

Вы можете использовать ключевые слова `repeat` и `repeatwhile` чтобы создать повторяющиеся циклы.

```

@startuml
start
repeat
  :read data;
  :generate diagrams;
repeat while (more data?)
stop
@enduml

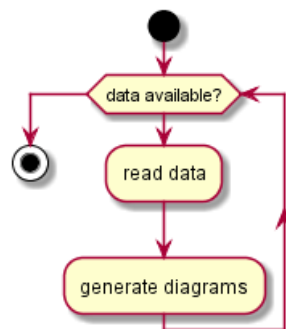
```



5.5 Цикл while

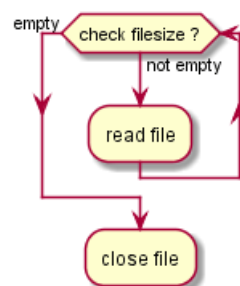
Вы можете использовать ключевые слова `while` и `end while` чтобы создавать повторяющиеся циклы.

```
@startuml
start
while (data available?)
    :read data;
    :generate diagrams;
endwhile
stop
@enduml
```



Возможно добавить описание используя ключевое слово `endwhile`, или используя ключевое слово `is..`

```
@startuml
while (check filesize ?) is (not empty)
    :read file;
endwhile (empty)
:close file;
@enduml
```



5.6 Паралельные процессы

Вы можете использовать ключевые слова `fork`, `fork again` и `end fork` чтобы определить параллельные процессы.

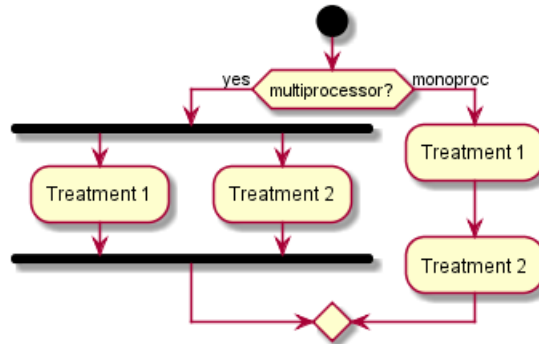
```
@startuml
start
if (multiprocessor?) then (yes)
    fork
    :Treatment 1;
  
```



```

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

```



5.7 Заметки

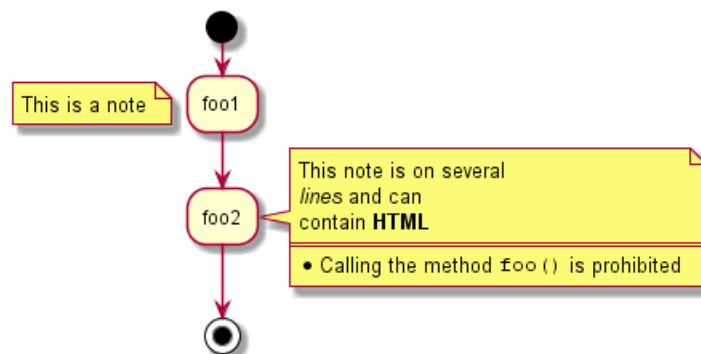
Форматирование текста может быть сделано синтаксисом creole.

Заметку можно сделать плавающей, с помощью ключевого слова `floating`.

```

@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

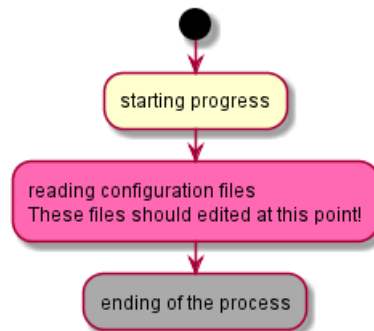
```



5.8 Цвета

Вы можете задать цвет некоторым активностям.

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



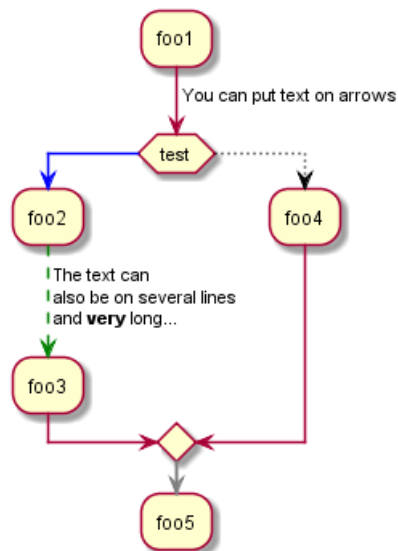
5.9 Стрелки

Используя нотацию `->`, вы можете добавить текст к стрелке, аи поменять их цвет.

Так же можно сделать стрелки: из точек, из дефисов, жирные и скрытые.

```
@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
```



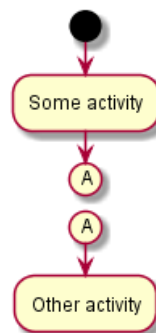


5.10 Connector

You can use parentheses to denote connector.

```

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



5.11 Группирование

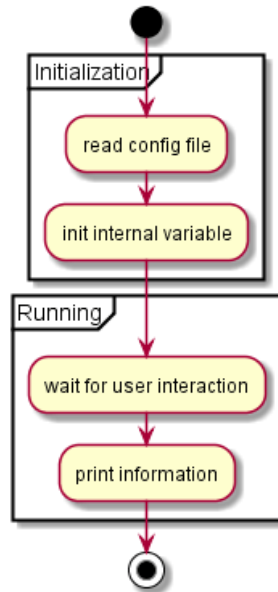
Вы можете группировать активности вместе, определяя раздел:

```

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



```
stop
@enduml
```



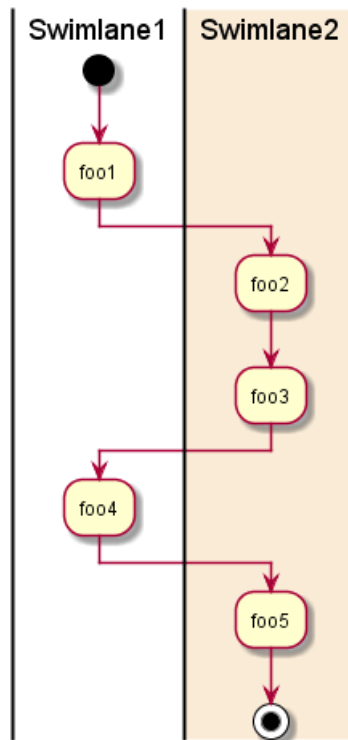
5.12 Дорожки

Используя символ |, вы можете определять плавающие линии.

Также возможно изменять цвет плавающих линий.

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





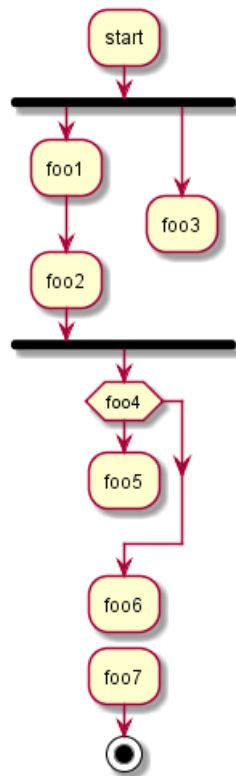
5.13 Отсоединение

Возможно убрать стрелку используя ключевое слово `detach`.

```

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

```



5.14 SDL

Изменяя последний разделитель ;, вы можете установить различный рендеринг для активности:

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

```

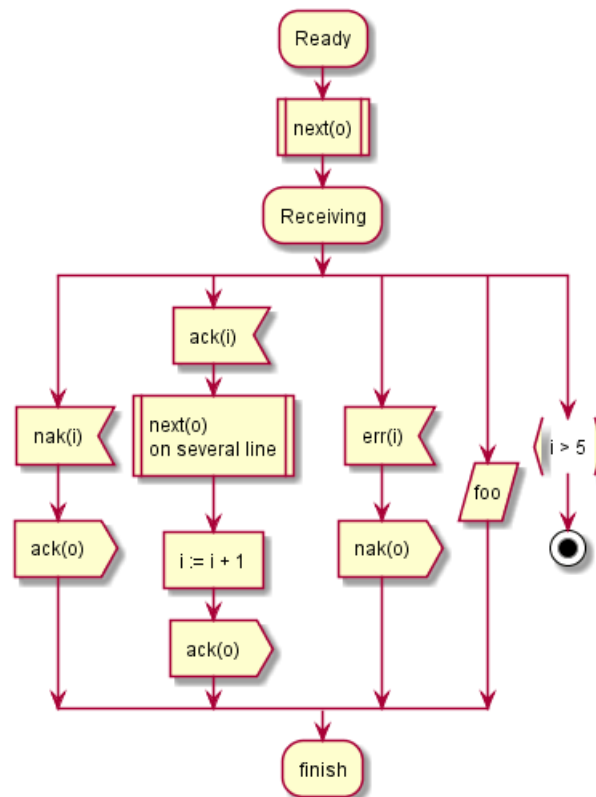
@startuml
:Ready;
:next(o)|
:Receiving;
split
:na<(i)<
:ack(o)>
split again
:ack(i)<
:next(o)
on several line|
:i := i + 1]
:ack(o)>
split again
:err(i)<
:na<(o)>
split again
:foo/
split again
:i > 5}
  
```



```

stop
end split
:finish;
@enduml

```



5.15 Полноценный пример

```

@startuml

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

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

if (do redirect?) then (yes)

```

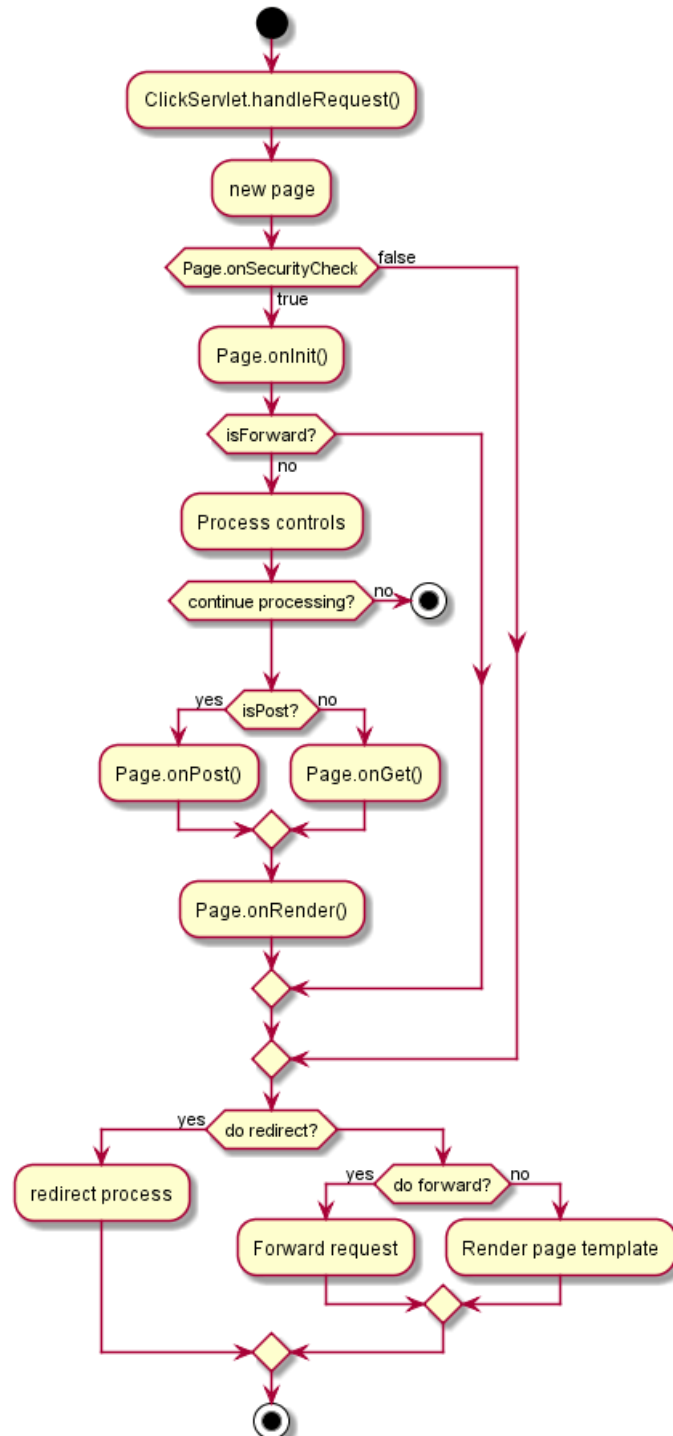


```

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

@enduml

```



6 Диаграмма компонентов

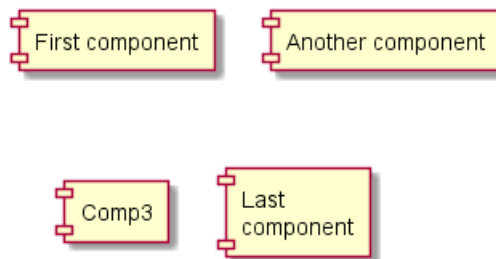
Let's have few examples :

6.1 Компоненты

Обозначения компонентов должны быть заключены в квадратные скобки.

Также можно использовать ключевое слово `component` для объявления компонента. Вы можете объявить алиас с помощью ключевого слова `as`. Этот алиас может быть использован позже, при объявлении связей.

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



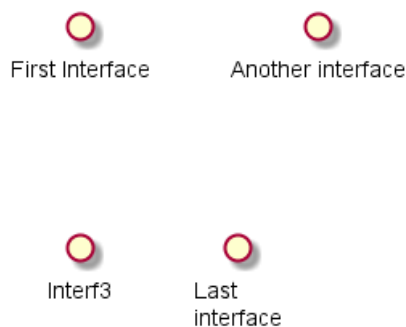
6.2 Интерфейсы

Для обозначения интерфейса используется символ `()` (потому что он выглядит как круг).

Также возможно использование ключевого слова `interface` для объявления интерфейса. Вы можете объявить алиас с помощью ключевого слова `as`. Этот алиас может быть использован позднее, когда будут задаваться связи.

Далее мы увидим, что задание интерфейсов опционально.

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



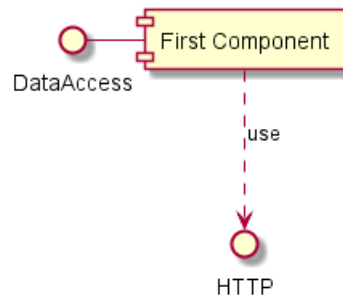
6.3 Простой пример

Отношения между элементами создаются с помощью комбинации точечных линий (. .), прямых линий (--) и стрелок (-->).

```
@startuml
```

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

```
@enduml
```



6.4 Использование заметок

Вы можете использовать ключевые слова `note left of`, `note right of`, `note top of`, `note bottom of` чтобы задать метки, относящиеся к одному объекту.

Заметка также может быть задана не прикрепленной, используя ключевое слово `note`, а затем прикреплена к другим объектам, используя символ `...`

```
@startuml
```

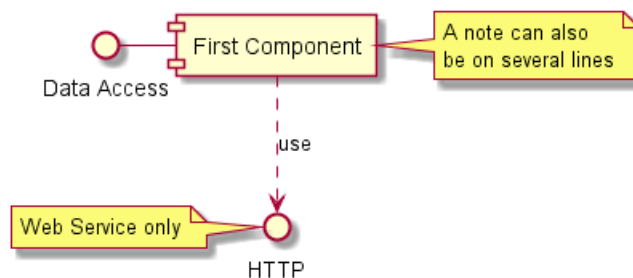
```
interface "Data Access" as DA
```

```
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 Группирование компонентов

Вы можете использовать несколько ключевых слов `package`, чтобы группировать компоненты и интерфейсы вместе.

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

```
@startuml
```

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

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

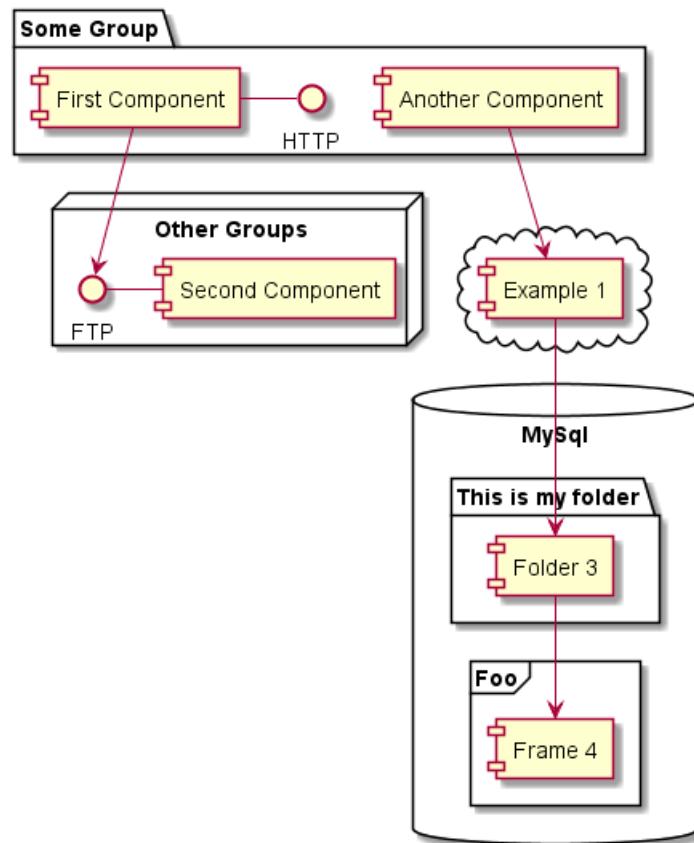
```
cloud {  
    [Example 1]  
}
```

```
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
```

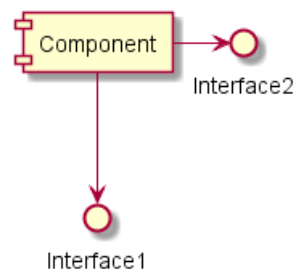




6.6 Изменение направления стрелок

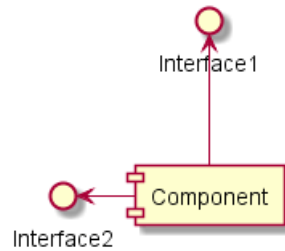
По умолчанию, связи между классами имеют два типа -- и ориентированы вертикально. Можно создавать горизонтальные связи с помощью одного типа (или точки), вот так:

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



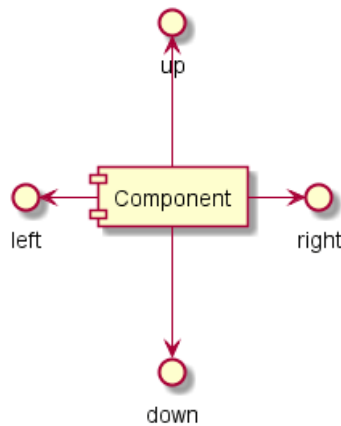
Вы также можете изменять направления, перевернув связь:

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

Также, можно изменить направление стрелки добавлением ключевых слов `left`, `right`, `up` или `down` внутри стрелки:

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



Вы можете сократить запись, используя только первую букву направления (например, `-d-` вместо `-down-`) или две первые буквы (`-do-`).

Пожалуйста, заметьте, что не стоит использовать эту функциональность без особой надобности: *Graphviz* обычно даёт хорошие результаты без дополнительной настройки.

6.7 Использование нотации UML2

Команда `skinparam componentStyle uml2` используется, чтобы переключиться на нотацию UML2.

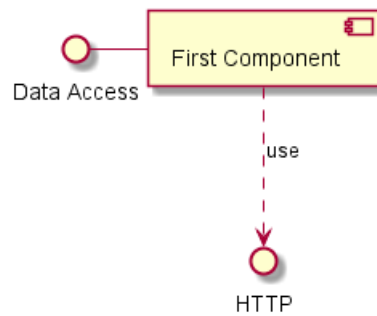
```
@startuml
skinparam componentStyle uml2

interface "Data Access" as DA

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

@enduml
```



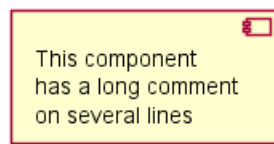


6.8 Длинное описание

Для помещения многострочного текста в тело компонента используются квадратные скобки

```

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



6.9 Индивидуальные цвета

Вы можете задать цвет после определения компонента.

```

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



6.10 Использование Sprite в стереотипах

Можно использовать спрайты внутри компонентов стереотипа.

```

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



```

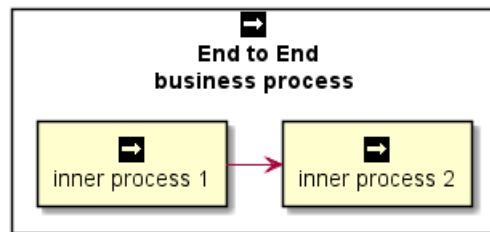
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}

```

```

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 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете задать цвет и шрифт для компонентов и интерфейсов с заданными шаблонами.

```
@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
```

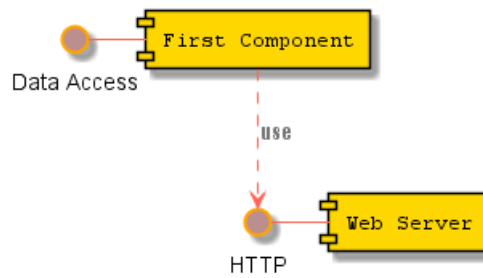
```
DA - [First Component]
```

```
[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

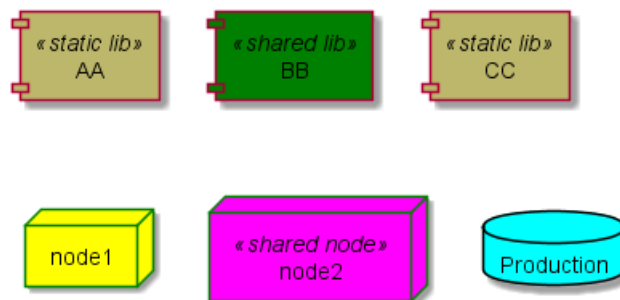
```

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 Диаграмма состояний

7.1 Простое состояние

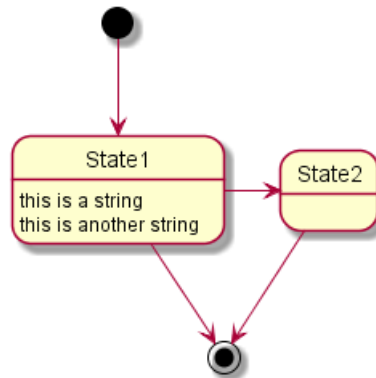
Для изображения начального и конечного псевдосостояний используется [*].

Используйте --> для изображения переходов.

```
@startuml
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```



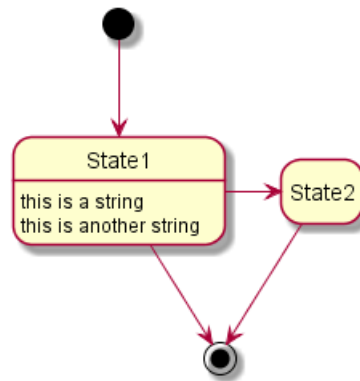
7.2 Change state rendering

You can use `hide empty description` to render state as simple box.

```
@startuml
hide empty description
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string

State1 -> State2
State2 --> [*]

@enduml
```



7.3 Составное состояние

Также можно изображать составные состояния. Для этого его следует объявить, используя конструкцию `state { ... }`.

```

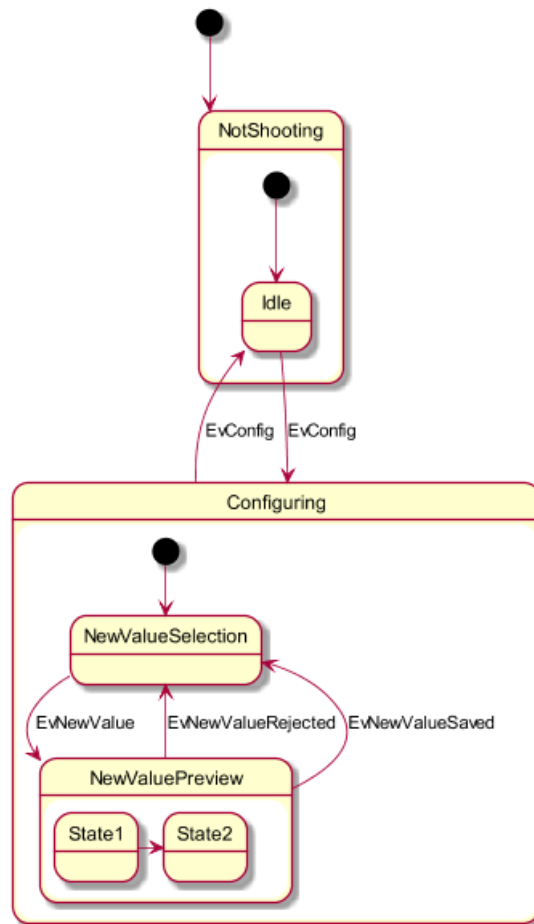
@startuml
scale 350 width
[*] --> NotShooting

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 Длинные имена

Вы также можете использовать ключевое слово `state` для сокращения длинного имени состояния.

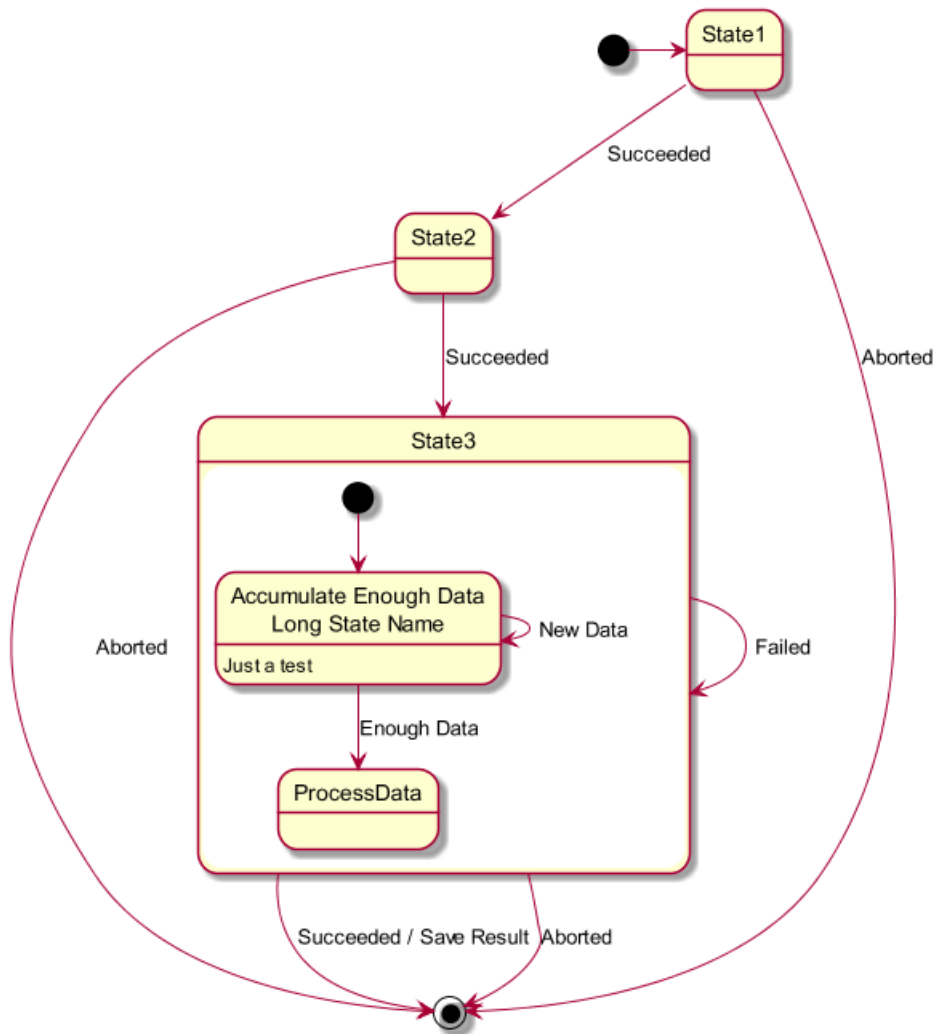
```

@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 Параллельные состояния

Используя оператор `--` или `||`, вы можете объявлять параллельные подсостояния внутри составного состояния.

```

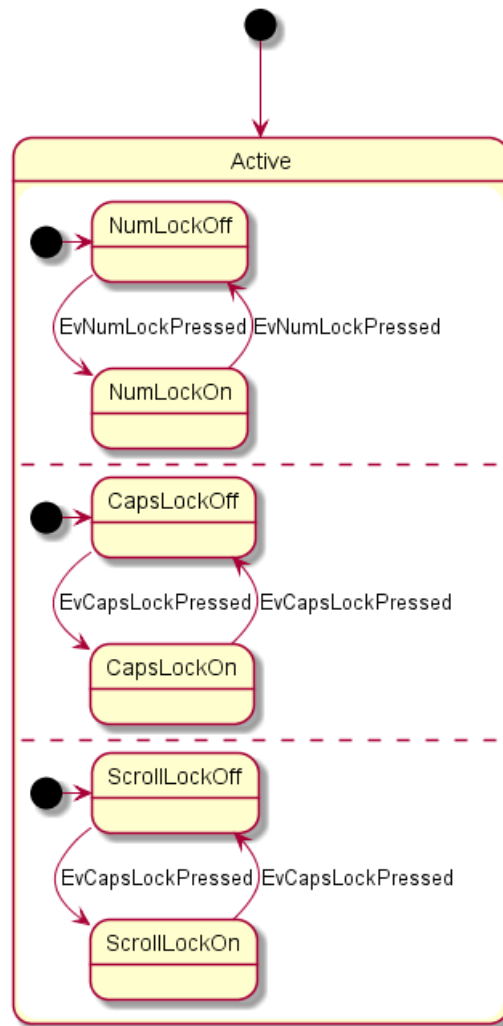
@startuml
[*] --> Active

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 Направления стрелок

Для изображения стрелок перехода горизонтально используется оператор `->`. Следующий синтаксис позволяет задать другое направление.

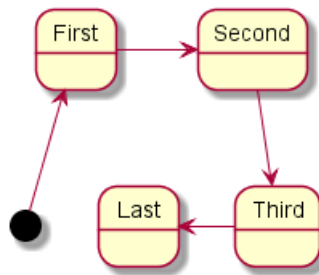
- `-down->` (default arrow)
- `-right->` or `->`
- `-left->`
- `-up->`

```
@startuml
```

```
[*] -up-> First
First -right-> Second
Second --> Third
Third -left-> Last
```

```
@enduml
```





Вы также можете сокращать слова в описании стрелок (например, -d-> или -do-> вместо -down->).

Не следует злоупотреблять этой функциональностью: *GraphViz* в большинстве случаев дает хороший результат без лишних манипуляций.

7.7 Заметки

К состоянию можно добавлять заметки, используя специальные ключевые слова: `note left of`, `note right of`, `note top of`, `note bottom of`.

Заметки можно определять в несколько строк.

```
@startuml
```

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

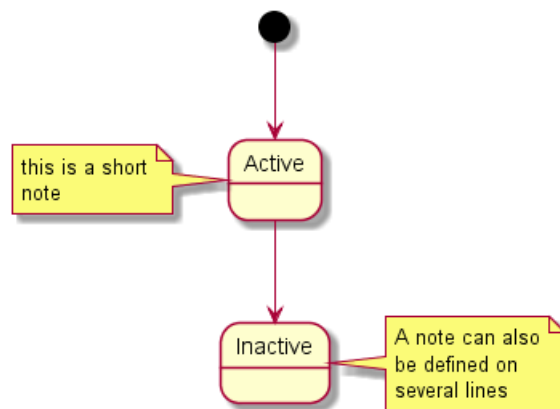
```
Active --> Inactive
```

```
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
```



Можно создавать заметки, не привязанные ни к какому объекту.

```
@startuml
```

```
state foo
```

```
note "This is a floating note" as N1
```

```
@enduml
```





7.8 Еще о заметках

Также заметки можно прикреплять к составным состояниям.

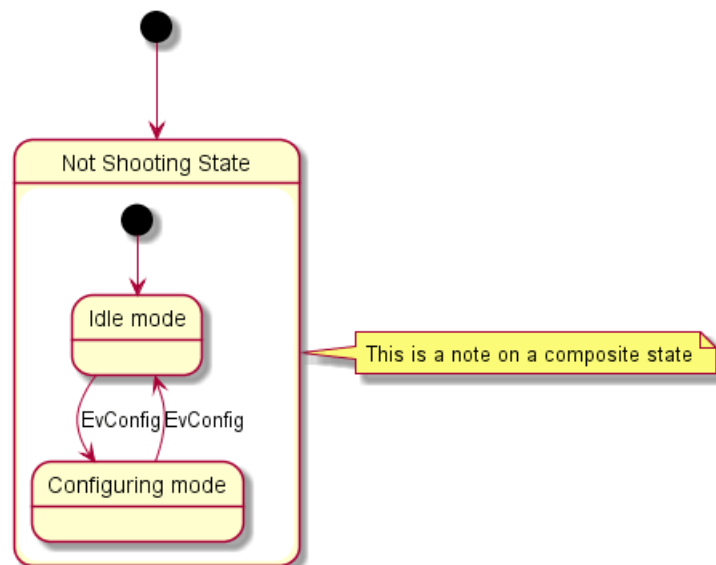
```
@startuml
```

```
[*] --> 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 Skinparam

Вы можете использовать команду `skinparam` для изменения шрифтов и цветов диаграммы

Вы можете использовать данную команду :

- В определении диаграммы, как любую другую команду,
- В подключенном файле,
- В конфигурационном файле, указанном в командной строке в задании ANT.

Вы можете задавать цвета и шрифты для именованных шаблонов состояний.

```
@startuml
skinparam backgroundColor LightYellow
skinparam state {
    StartColor MediumBlue
}
```



```

EndColor Red
BackgroundColor Peru
BackgroundColor<<Warning>> Olive
BorderColor Gray
FontName Impact
}

```

```

[*] --> NotShooting

```

```

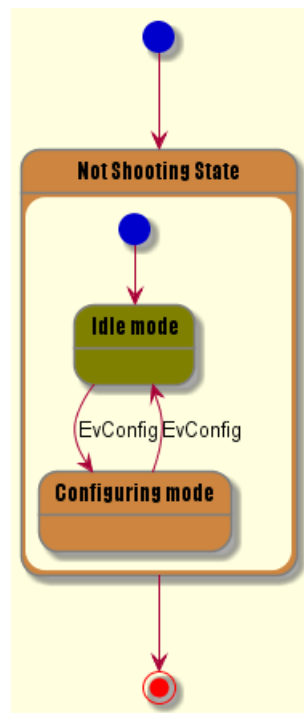
state "Not Shooting State" as NotShooting {
  state "Idle mode" as Idle <<Warning>>
  state "Configuring mode" as Configuring
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

```

```

NotShooting --> [*]
@enduml

```

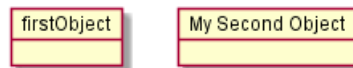


8 Диаграмма объектов

8.1 Определение объектов

Вы можете определить экземпляр объекта используя ключевое слово `object`.

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



8.2 Отношения между объектами

Отношения между объектами определяются с использованием следующих символов :

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

Возможно заменить `--` на `..` чтобы получить линию из точек.

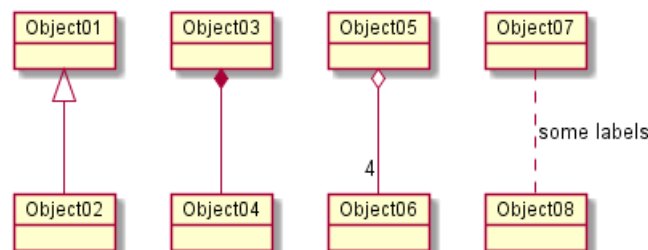
Зная данные правила, можно создать следующие картинки.

Возможно добавить описание к связи, используя `:`, с последующим текстом описания.

Для определения количества элементов, вы можете использовать двойные кавычки `"` на каждой стороне связи.

```
@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
```

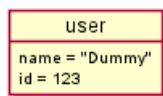


8.3 Добавление полей

Для определения свойств (полей) объекта, задайте префикс `:`, указав вслед за ним имя свойства.

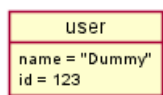


```
@startuml  
  
object user  
  
user : name = "Dummy"  
user : id = 123  
  
@enduml
```



Также возможно разместить все поля между скобками {}.

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



8.4 Общие с диаграммами классов функции

- Видимость
- Задание меток
- Использование пакетов
- Стилизованные выводы

9 Timing Diagram

This is only a proposal and subject to change.

You are very welcome to create a new discussion on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

9.1 Declaring participant

You declare participant using concise or robust keyword, depending on how you want them to be drawn.

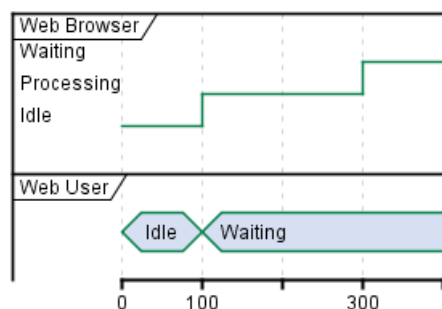
You define state change using the @ notation, and the is verb.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@0
WU is Idle
WB is Idle
```

```
@100
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
@enduml
```



9.2 Adding message

You can add message using the following syntax.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

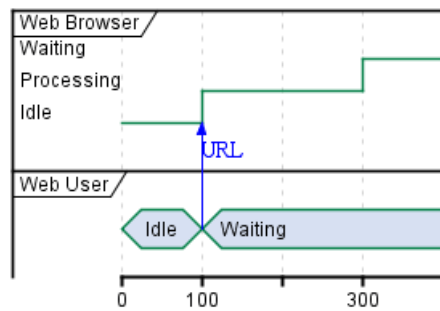
```
@0
WU is Idle
WB is Idle
```

```
@100
WU -> WB : URL
WU is Waiting
WB is Processing
```

```
@300
WB is Waiting
```



@enduml



9.3 Relative time

It is possible to use relative time with @.

```

@startuml
robust "DNS Resolver" as DNS
robust "Web Browser" as WB
concise "Web User" as WU
  
```

```

@0
WU is Idle
WB is Idle
DNS is Idle
  
```

```

@+100
WU -> WB : URL
WU is Waiting
WB is Processing
  
```

```

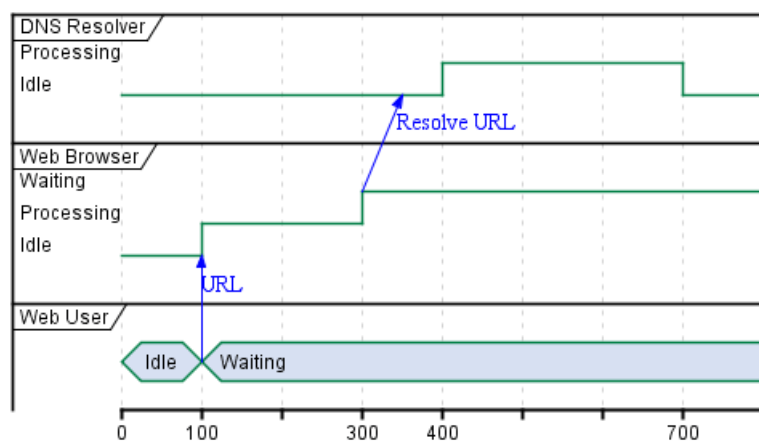
@+200
WB is Waiting
WB -> DNS@+50 : Resolve URL
  
```

```

@+100
DNS is Processing
  
```

```

@+300
DNS is Idle
@enduml
  
```



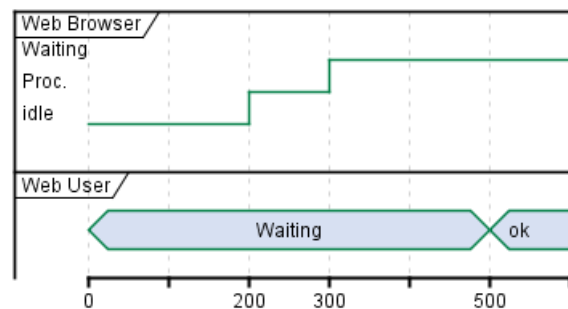
9.4 Participant oriented

Rather than declare the diagram in chronological order, you can define it by participant.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

```
@WB
0 is idle
+200 is Proc.
+100 is Waiting
```

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

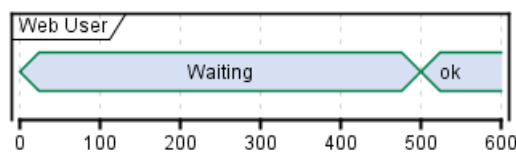


9.5 Setting scale

You can also set a specific scale.

```
@startuml
concise "Web User" as WU
scale 100 as 50 pixels
```

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



9.6 Initial state

You can also define an initial state.

```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU
```

```
WB is Initializing
WU is Absent
```

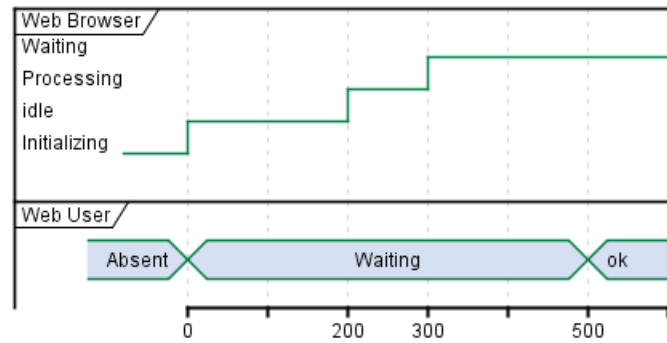


```

@WB
0 is idle
+200 is Processing
+100 is Waiting

@WU
0 is Waiting
+500 is ok
@enduml

```



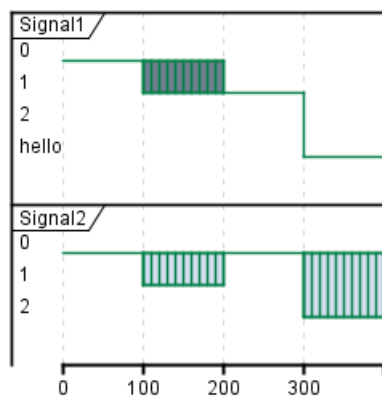
9.7 Intricated state

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 {-}

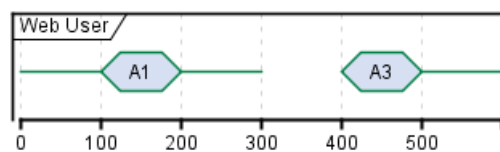
@100
WU is A1

@200
WU is {-}

@300
WU is {hidden}

@400
WU is A3

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



9.9 Adding constraint

It is possible to display time constraints on the diagrams.

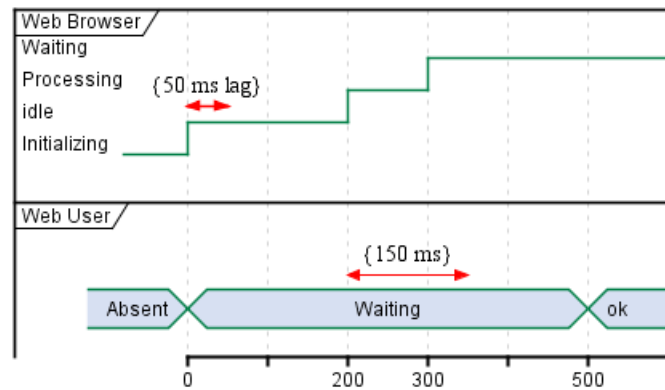
```
@startuml
robust "Web Browser" as WB
concise "Web User" as WU

WB is Initializing
WU is Absent

@WB
0 is idle
+200 is Processing
+100 is Waiting
WB@0 <-> @50 : {50 ms lag}

@WU
0 is Waiting
+500 is ok
@200 <-> @+150 : {150 ms}
@enduml
```





9.10 Adding texts

You can optionally add a title, a header, a footer, a legend and a caption:

```

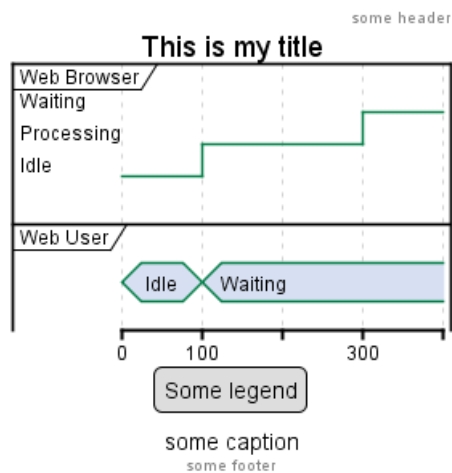
@startuml
Title This is my title
header: some header
footer: some footer
legend
Some legend
end legend
caption some caption

robust "Web Browser" as WB
concise "Web User" as WU

@0
WU is Idle
WB is Idle

@100
WU is Waiting
WB is Processing

@300
WB is Waiting
@enduml
  
```



10 Gantt Diagram

This is only a proposal and subject to change.

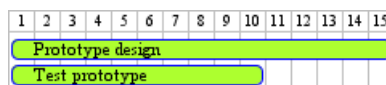
You are very welcome to create a new discussion on this future syntax. Your feedbacks, ideas and suggestions help us to find the right solution.

The Gantt is described in *natural* language, using very simple sentences (subject-verb-complement).

10.1 Declaring tasks

Tasks defined using square bracket. Their durations are defined using the last verb:

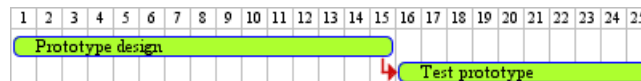
```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
@endgantt
```



10.2 Adding constraints

It is possible to add constraints between task.

```
@startgantt
[Prototype design] lasts 15 days
[Test prototype] lasts 10 days
[Test prototype] starts at [Prototype design]'s end
@endgantt
```



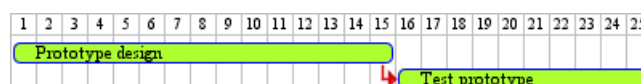
```
@startgantt
[Prototype design] lasts 10 days
[Code prototype] lasts 10 days
[Write tests] lasts 5 days
[Code prototype] starts at [Prototype design]'s end
[Write tests] starts at [Code prototype]'s start
@endgantt
```



10.3 Short names

It is possible to define short name for tasks with the as keyword.

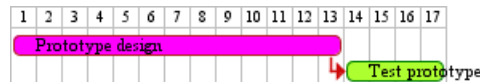
```
@startgantt
[Prototype design] as [D] lasts 15 days
[Test prototype] as [T] lasts 10 days
[T] starts at [D]'s end
@endgantt
```



10.4 Customize colors

It also possible to customize colors.

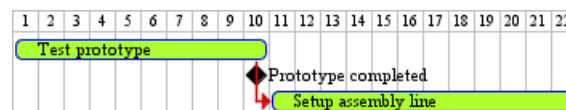
```
@startgantt
[Prototype design] lasts 13 days
[Test prototype] lasts 4 days
[Test prototype] starts at [Prototype design]'s end
[Prototype design] is colored in Fuchsia/FireBrick
[Test prototype] is colored in GreenYellow/Green
@endgantt
```



10.5 Milestone

You can define Milestones using the happens verb.

```
@startgantt
[Test prototype] lasts 10 days
[Prototype completed] happens at [Test prototype]'s end
[Setup assembly line] lasts 12 days
[Setup assembly line] starts at [Test prototype]'s end
@endgantt
```



10.6 Calendar

You can specify a starting date for the whole project. By default, the first task starts at this date.

```
@startgantt
Project starts the 20th of september 2017
[Prototype design] as [TASK1] lasts 13 days
[TASK1] is colored in Lavender/LightBlue
@endgantt
```



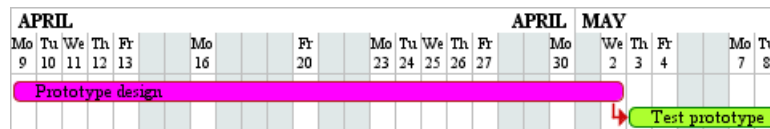
10.7 Close day

It is possible to close some day.

```
@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
```



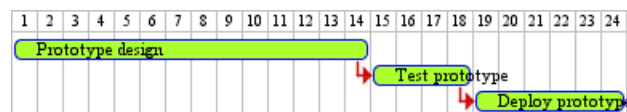
```
@endgantt
```



10.8 Simplified task succession

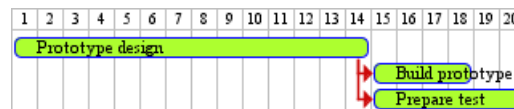
It's possible to use the then keyword to denote consecutive tasks.

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



You can also use arrow ->

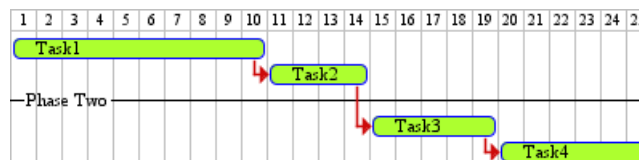
```
@startgantt
[Prototype design] lasts 14 days
[Build prototype] lasts 4 days
[Prepare test] lasts 6 days
[Prototype design] -> [Build prototype]
[Prototype design] -> [Prepare test]
@endgantt
```



10.9 Separator

You can use -- to separate sets of tasks.

```
@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.

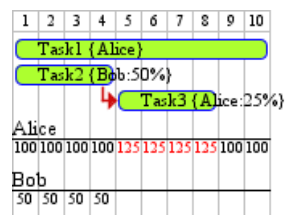
```
@startgantt
[Task1] on {Alice} lasts 10 days
[Task2] on {Bob:50%} lasts 2 days
```



```

then [Task3] on {Alice:25%} lasts 1 days
@endgant

```



10.11 Complex example

It also possible to use the and conjunction.

You can also add delays in constraints.

```
@startgant
```

```
[Prototype design] lasts 13 days and is colored in Lavender/LightBlue
```

```
[Test prototype] lasts 9 days and is colored in Coral/Green and starts 3 days after [Prototype design]'s end
```

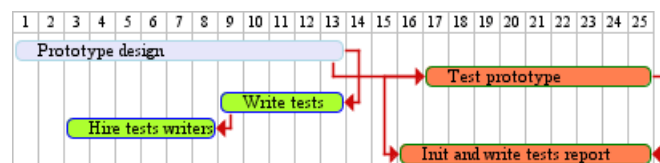
```
[Write tests] lasts 5 days and ends at [Prototype design]'s end
```

```
[Hire tests writers] lasts 6 days and ends at [Write tests]'s start
```

```
[Init and write tests report] is colored in Coral/Green
```

```
[Init and write tests report] starts 1 day before [Test prototype]'s start and ends at [Test prototype]'s end
```

```
@endgant
```



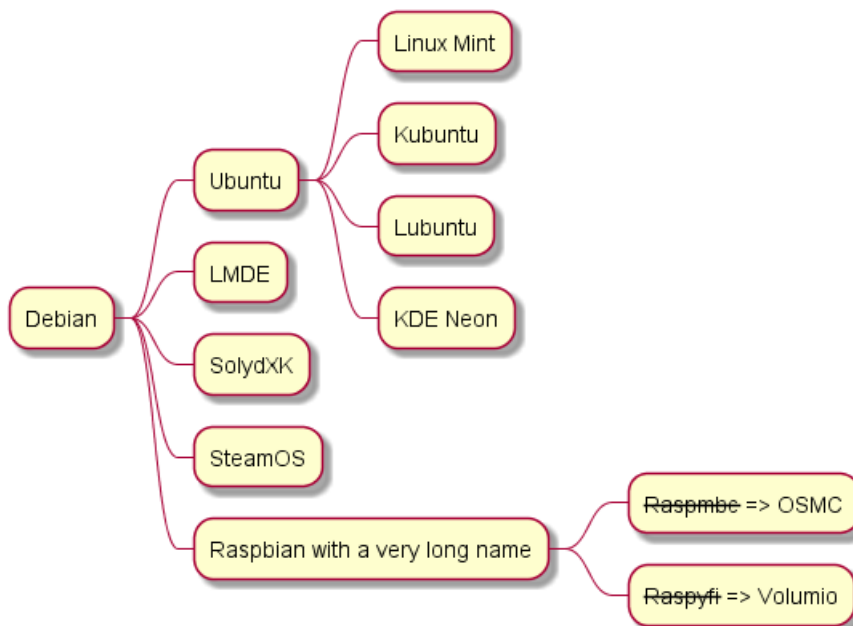
11 MindMap

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
```



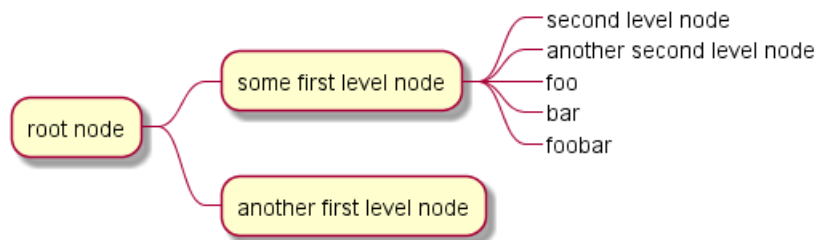
11.2 Removing box

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
```

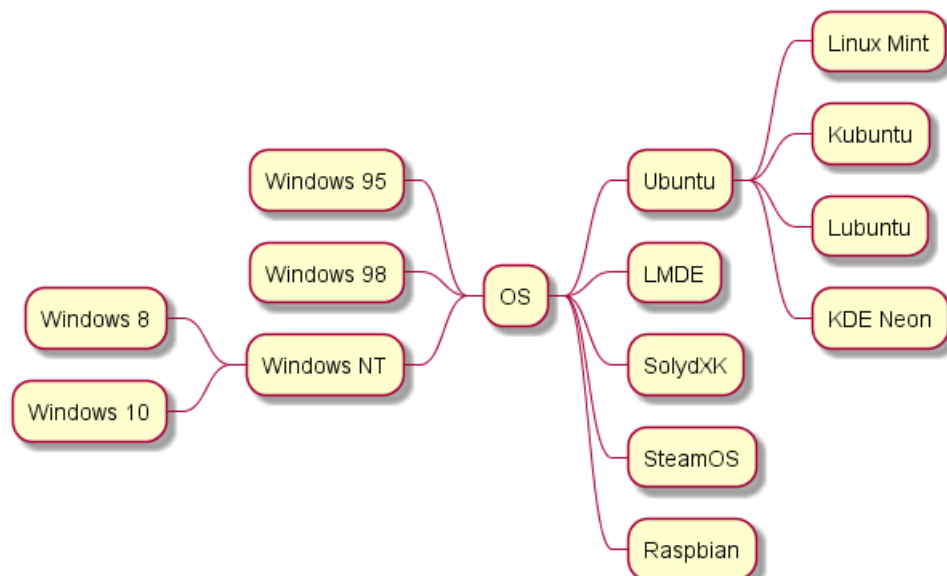


11.3 Arithmetic notation

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

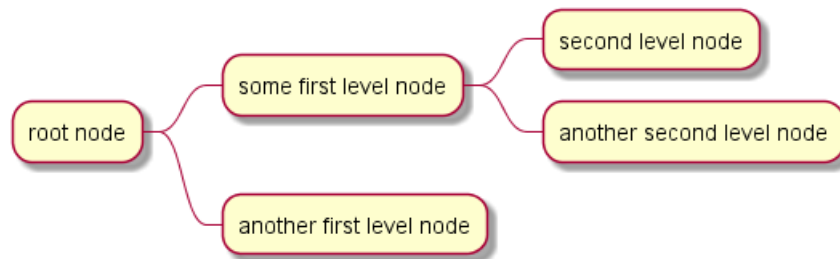
```
@startmindmap
```



```

* 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

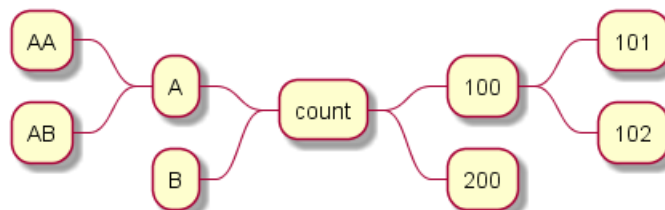
```

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
```

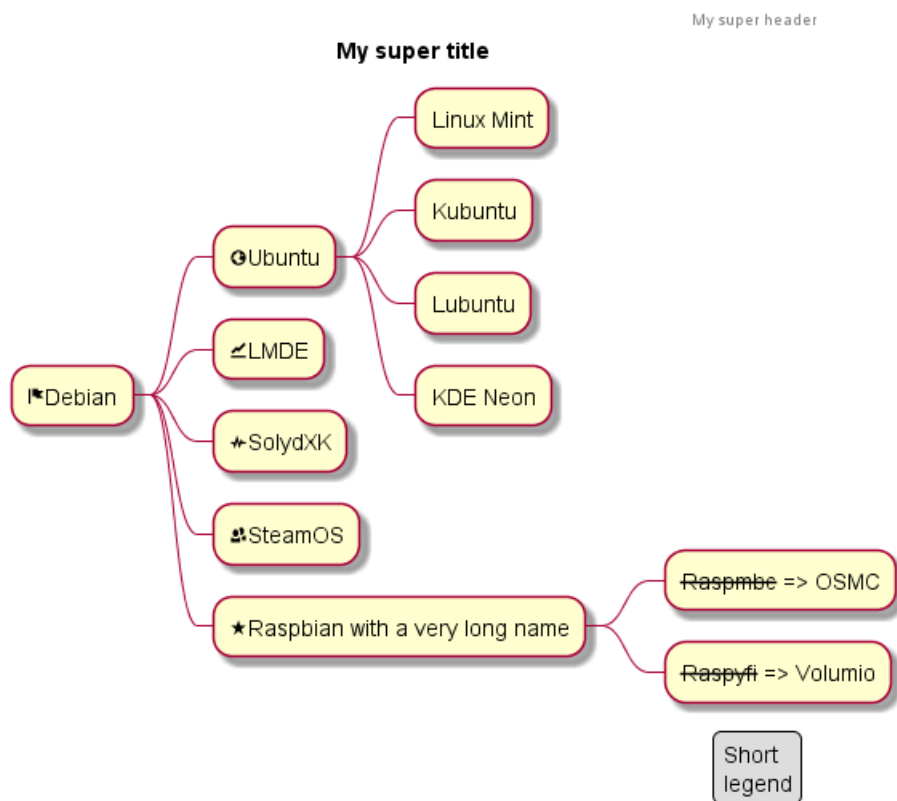


figure 1

My super footer

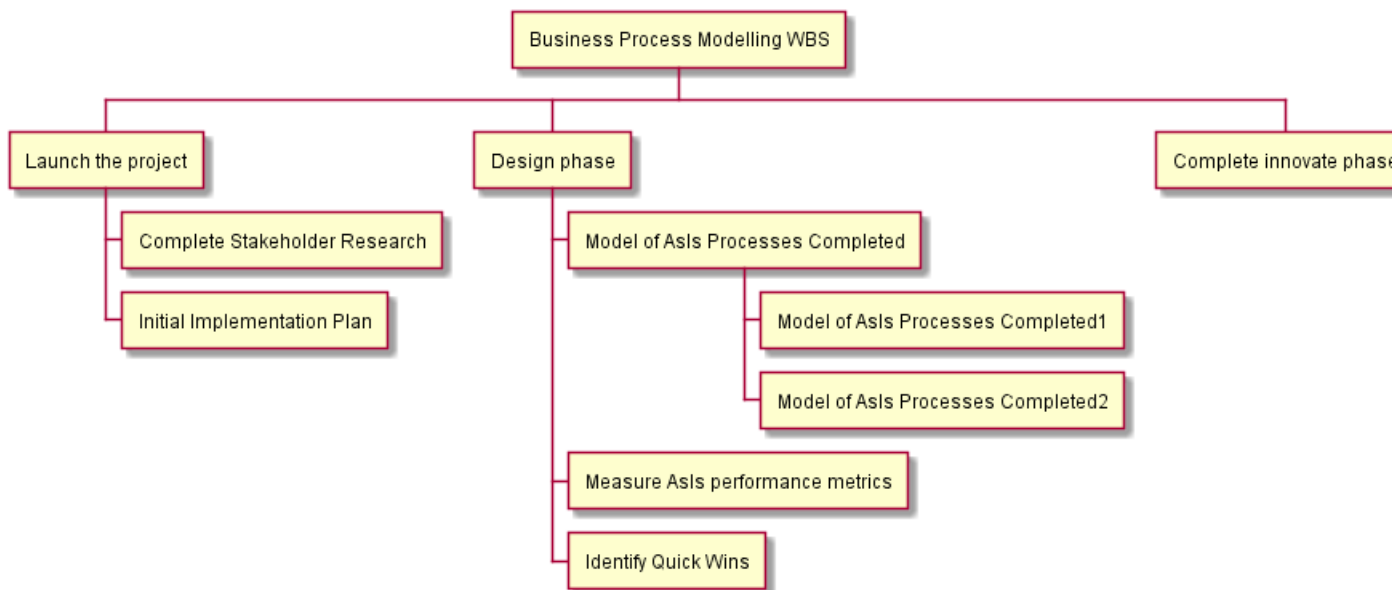
12 Work Breakdown Structure

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
```



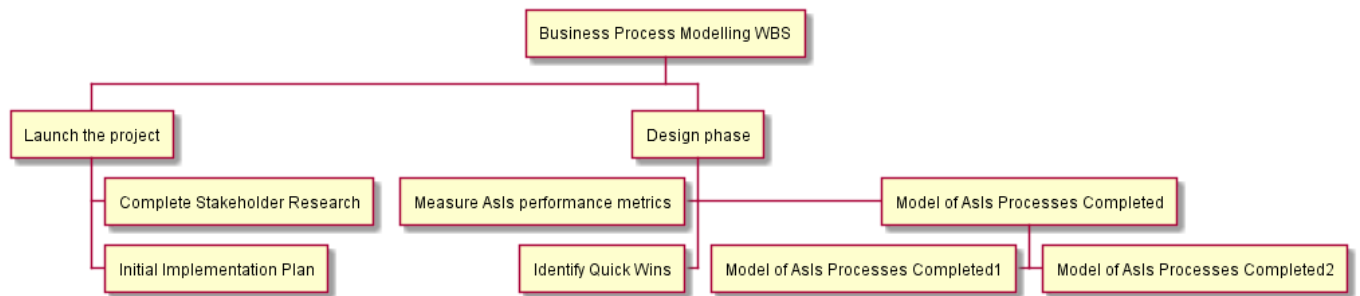
12.2 Change direction

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

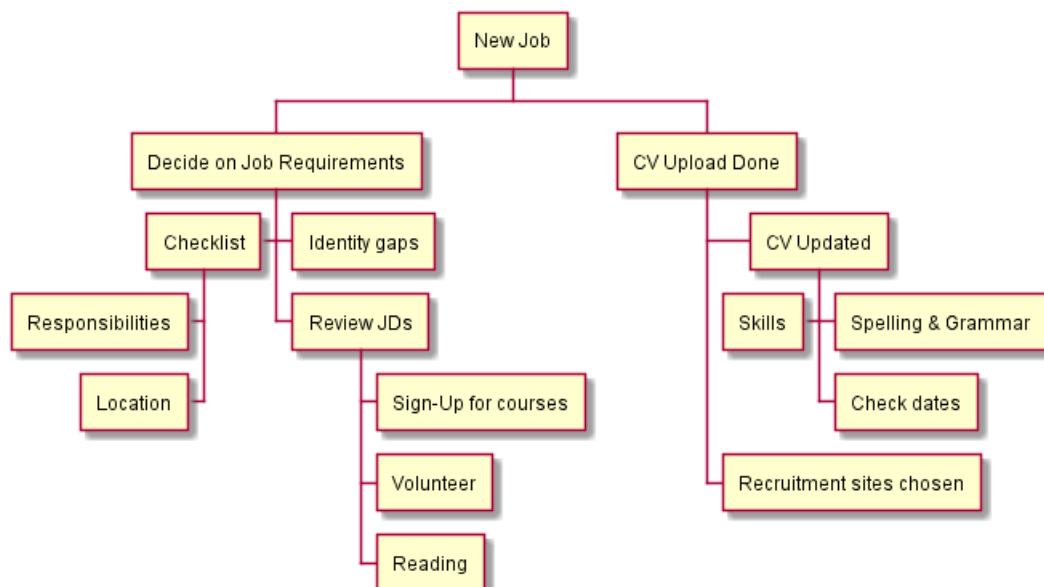


12.3 Arithmetic notation

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.

```

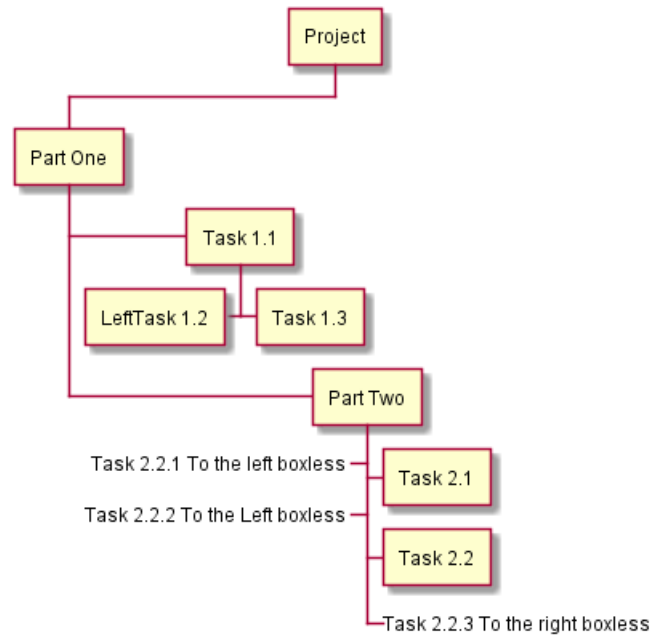
@startwbs
+ Project
  
```



```

+ 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

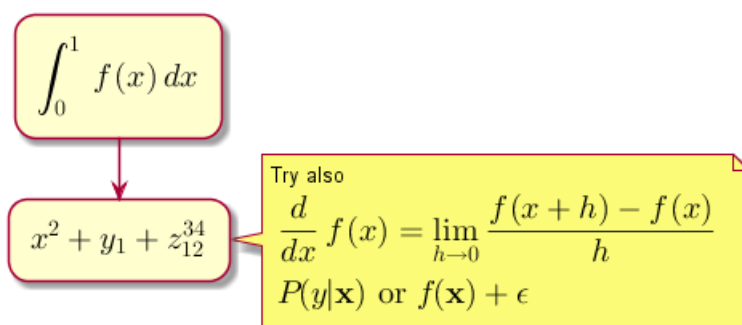
```



13 Maths

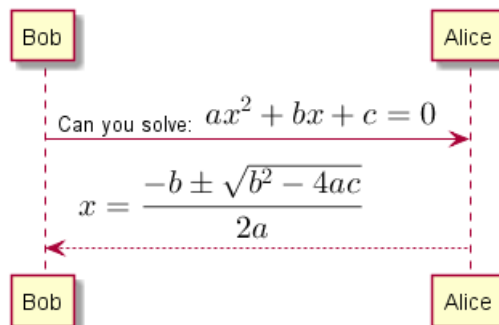
You can use AsciiMath or JLaTeXMath notation within PlantUML:

```
@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
```



or:

```
@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 Standalone diagram

You can also use @startmath/@endmath to create standalone AsciiMath formula.

```
@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
```

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi t}{L}\right) + \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi t}{L}\right)$$

Or use @startlatex/@endlatex to create standalone JLaTeXMath formula.

```
@startlatex
\sum_{i=0}^{n-1} (a_i + b_i^2)
@endlatex
```



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

13.2 How is this working ?

To draw those formulas, PlantUML uses two OpenSource projects:

- AsciiMath that converts AsciiMath notation to LaTeX expression.
- JLatexMath that displays mathematical formulas written in LaTeX. JLaTeXMath is the best Java library to display LaTeX code.

ASCIIMathTeXImg.js is small enough to be integrated into PlantUML standard distribution.

Since JLatexMath is bigger, you have to download it separately, then unzip the 4 jar files (*batik-all-1.7.jar*, *jlatexmath-minimal-1.0.3.jar*, *jlm_cyrillic.jar* and *jlm_greek.jar*) in the same folder as *PlantUML.jar*.



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.

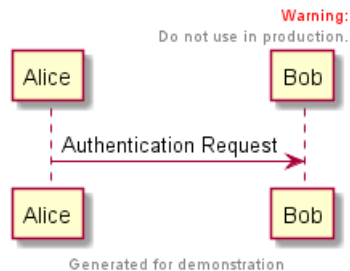
It is also possible to put some HTML into the header or footer.

```
@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

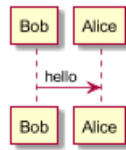
You can use the `scale` command to zoom the generated image.

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.

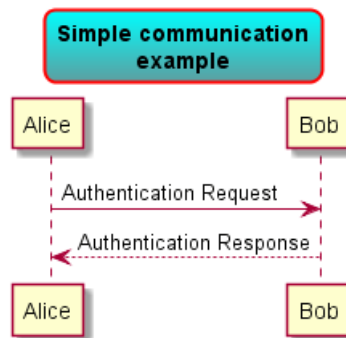
Some skinparam settings are available to put borders on the title.

```
@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

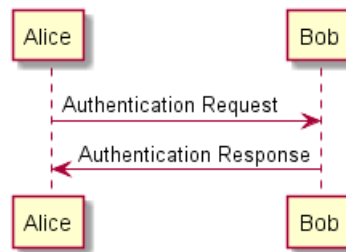
title
  <u>Simple</u> communication example
  on <i>several</i> lines and using <back:cadetblue>creole tags</back>
end title
```

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

```
@enduml
```



**Simple communication example
on several lines and using creole tags**



14.5 Caption

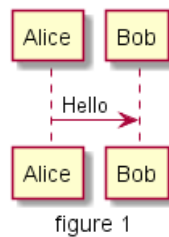
There is also a caption keyword to put a caption under the diagram.

```

@startuml

caption figure 1
Alice -> Bob: Hello

@enduml
  
```



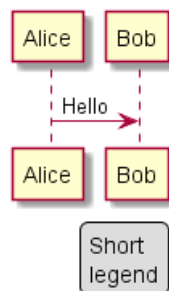
14.6 Legend the diagram

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
  
```

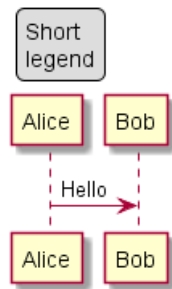


```

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



```
legend
endlegend
@enduml
```



15 Salt

Salt - это подпроект, включенный в PlantUML и призванный помочь вам проектировать графический интерфейс.

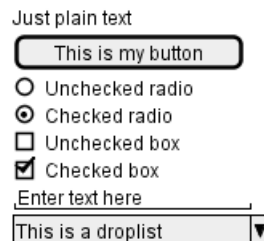
Вы можете использовать ключевое слово `@startsalt`, или `@startuml`, после которого стоит ключевое слово `salt`.

15.1 Простые виджеты

Окно должно начинаться и заканчиваться скобками. Вы можете определить:

- Кнопку, используя `[и]`.
- Radio button (переключатель), используя `(и)`.
- Checkbox (флажок), используя `[и]`.
- Поле ввода текста, используя `"`.

```
@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
```



Цель этого инструмента - обсуждать простые и типовые окна.

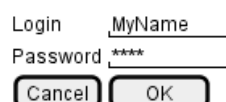
15.2 Использование сетки

Таблица автоматически создаётся, когда вы используете открывающую скобку `{`.

Ещё вам нужно использовать `|` для деления на колонки.

Например:

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



Сразу после открывающей скобки вы можете использовать символ, чтобы определить, где закончится строка и начнётся колонка сетки:

| Symbol | Result |
|--------|--|
| # | Чтобы показать вертикальные и горизонтальные линии |
| ! | Чтобы показать вертикальные линии |
| - | Чтобы показать горизонтальные линии |
| + | Чтобы показать внешние линии |

```
@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 Использование разделителя

Вы можете использовать несколько горизонтальных линий как разделитель.

```
@startsalt
```

```
{
```

```
    Text1
```

```
    ..
```

```
    "Some field"
```

```
    ==
```

```
    Note on usage
```

```
    ~~
```

```
    Another text
```

```
    --
```

```
    [Ok]
```

```
}
```

```
@endsalt
```



Text1

Some field

Note on usage

Another text

Ok

15.5 Древовидный виджет

Чтобы создать дерево, вам нужно начать с {T и использовать + чтобы определять иерархию.

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



15.6 Окружающие скобки

Вы можете задать подэлементы, открывая новую скобку.

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

Name

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

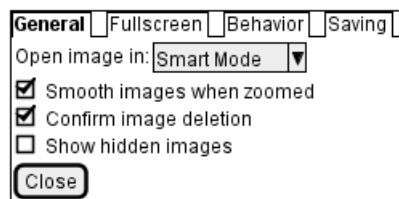
Superclass: java.lang.Object



15.7 Добавление вкладок

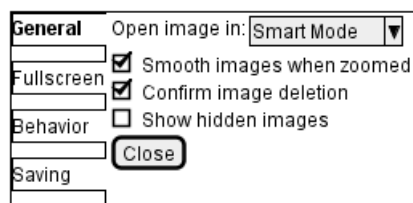
Вы можете добавить вкладки, используя нотацию {/. Заметьте, что вы можете использовать HTML код для выделения текста жирным.

```
@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
```



Вкладки также могут быть вертикально ориентированы:

```
@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
```



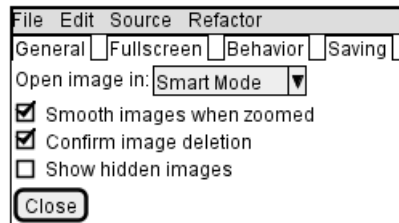
15.8 Использование меню

Вы можете добавить меню используя нотацию {*.

```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
```

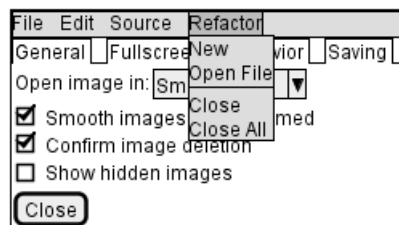


```
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



Также можно открыть меню:

```
@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 Продвинутая таблица

Вы можете использовать две специальные нотации для таблиц :

- * чтобы показать что ячейка должна быть объединена с левой
- . чтобы обозначить пустую ячейку

```
@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 | 🔩 bolt | 🔗 code | ⌵ expand-left | 💻 laptop | 📝 pencil | 📱 tablet |
| ↷ action-undo | 📖 book | ⚙ cog | ⌵ expand-right | 📂 layers | 👤 people | 🏷 tag |
| ≡ align-center | 🔖 bookmark | ⌵ collapse-down | ⌵ expand-up | 💡 lightbulb | 👤 person | 🏷 tags |
| ≡ align-left | 📦 box | ⌵ collapse-left | 🔗 external-link | 🔗 link-broken | 📞 phone | 🎯 target |
| ≡ align-right | 📁 briefcase | ⌵ collapse-right | 👁 eye | 🔗 link-intact | 📺 pie-chart | 📋 task |
| 🔍 aperture | 💷 british-pound | ⌵ collapse-up | 🔍 eyedropper | 📋 list-rich | 📌 pin | 💻 terminal |
| ↓ arrow-bottom | 🌐 browser | 📄 comment-square | 📁 file | 📋 list | 🎮 play-circle | 📄 text |
| 🕒 arrow-circle-bottom | 🖌 brush | 📄 compass | 🔥 fire | 📍 location | ➕ plus | 👍 thumb-down |
| 🕒 arrow-circle-left | 🐛 bug | 🕒 contrast | 🚩 flag | 🔒 lock-locked | 🔌 power-standby | 👍 thumb-up |
| 🕒 arrow-circle-right | 📣 bullhorn | 📄 copywriting | ⚡ flash | 🔓 lock-unlocked | 🖨 print | ⌚ timer |
| ↶ arrow-left | 📊 calculator | 📄 credit-card | 📁 folder | 🔄 loop-circular | 📁 project | ⇄ transfer |
| → arrow-right | 📅 calendar | 📄 crop | 🍴 fork | 📐 loop-square | ⚡ pulse | 🗑 trash |
| ↓ arrow-thick-bottom | 📷 camera-slr | 📄 dashboard | 🖥 fullscreen-enter | 🔄 loop | 🧩 puzzle-piece | 📏 underline |
| ↶ arrow-thick-left | 📉 caret-bottom | 📄 data-transfer-download | 🖥 fullscreen-exit | 🔍 magnifying-glass | ❓ question-mark | 📏 vertical-align-bottom |
| → arrow-thick-right | ⏪ caret-left | 📄 data-transfer-upload | 🌐 globe | 📍 map-marker | 🎲 random | 📏 vertical-align-center |
| ↑ arrow-top | ⏩ caret-right | 🗑 delete | 📊 graph | 📍 map | 🔄 reload | 📏 vertical-align-top |
| 🔊 audio-spectrum | ⬆ caret-top | 📞 dial | 📊 grid-four-up | ⏸ media-pause | 🔄 resize-both | 📹 video |
| 🔊 audio | 📄 cart | 📄 document | 📊 grid-three-up | ▶ media-play | 🔄 resize-height | 🔊 volume-high |
| 🏷 badge | 💬 chat | 💵 dollar | 📊 grid-two-up | ⏮ media-skip-backward | ↔ resize-width | 🔊 volume-low |
| 📊 bar-chart | ✓ check | 🗣 double-quote-sans-left | 💾 hard-drive | ▶ media-skip-forward | 📶 rss | 🔊 volume-off |
| 📊 basket | 📉 chevron-bottom | 🗣 double-quote-sans-right | 📄 header | ⏮ media-step-backward | 📄 rss-alt | ⚠ warning |
| 🔋 battery-empty | ↶ chevron-left | 🗣 double-quote-serif-left | 🎧 headphones | ⏮ media-step-forward | 📄 script | 🔧 wrench |
| 🔋 battery-full | ↷ chevron-right | 🗣 double-quote-serif-right | 🏠 home | ⏮ media-stop | 📦 share-boxed | ✂ x |
| 🥼 beaker | ⬆ chevron-top | 💧 droplet | 🖼 image | ⚕ medical-cross | ➦ share | ¥ yen |
| | 🕒 circle-check | 🚀 eject | 📧 inbox | ☰ menu | 🛡 shield | 🔍 zoom-in |
| | 🕒 circle-x | 🚶 elevator | ♾ infinity | 🎤 microphone | 📶 signal | 🔍 zoom-out |
| | 📋 clipboard | ⋯ ellipses | 🔍 info | ➖ minus | 📍 signpost | |
| | 🕒 clock | 📄 envelope-closed | 📄 italic | 📺 monitor | 📊 sort-ascending | |
| | ☁ cloud-download | 📄 envelope-open | ⌵ justify-center | 🌙 moon | 📊 sort-descending | |
| | ☁ cloud-upload | € euro | ⌵ justify-left | ➡ move | 📊 spreadsheet | |

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
```



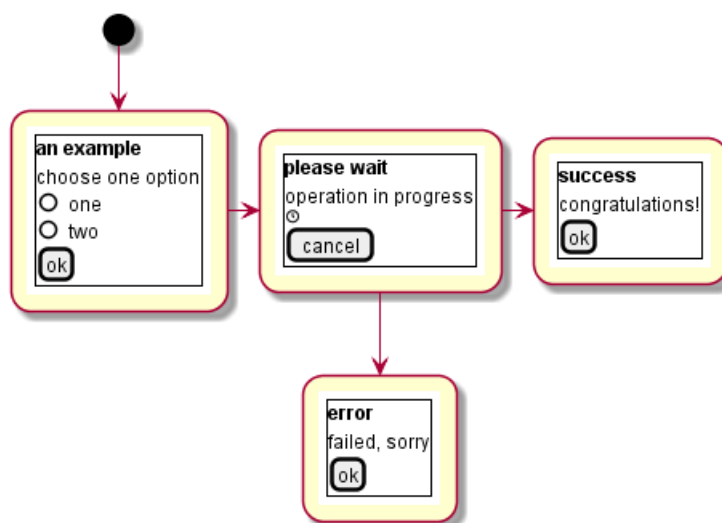
```

()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
!unquoted function SALT($x)
"{{
salt
%invoke_void_func("_"+"$x)
}}" as $x
!endfunction

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

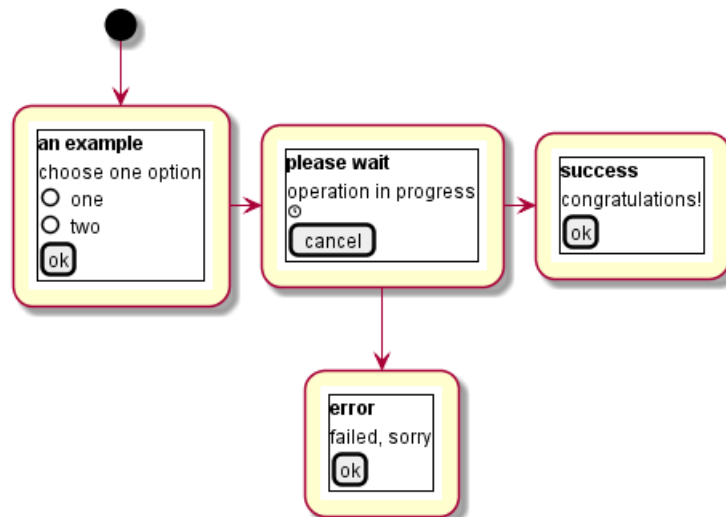
!function _wait()
{+
<b>please wait
operation in progress
<&clock>
[cancel]
}
!endfunction

!function _success()
{+
<b>success
congratulations!
[ok]
}
!endfunction

!function _error()
{+
<b>error
failed, sorry
[ok]
}
!endfunction

(*) --> 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
```



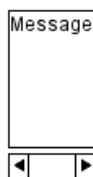
```
@startsalt
{SI
Message
.
.
.
.
}
@endsalt
```



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



```
.  
}  
@endsalt
```



16 Creole

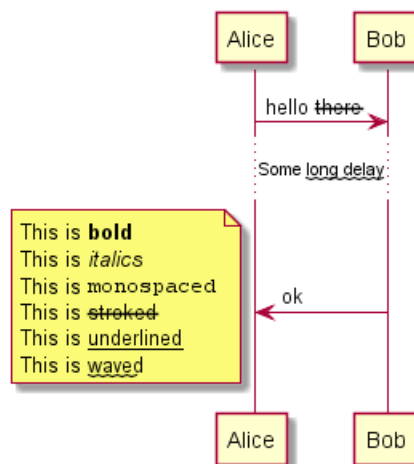
A light Creole engine has been integrated into PlantUML to have a standardized way of defining text style.

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
```



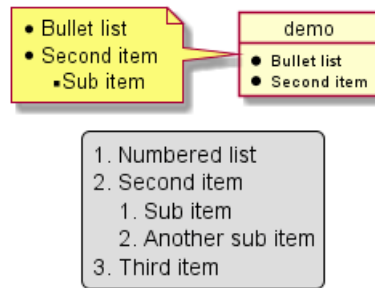
16.2 List

```
@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
```



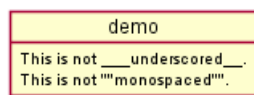

```
# Third item
end legend
@enduml
```



16.3 Escape character

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

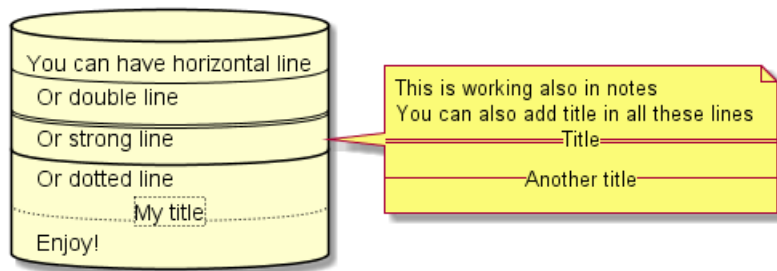
```
@startuml
object demo {
  This is not ~__underscored__.
  This is not ~""monospaced"".
}
@enduml
```



16.4 Horizontal lines

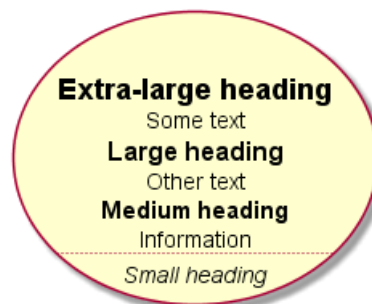
```
@startuml
database DB1 as "
You can have horizontal line
----
Or double line
====
Or strong line
----
Or dotted line
..My title..
Enjoy!
"
note right
  This is working also in notes
  You can also add title in all these lines
  ==Title==
  --Another title--
end note
@enduml
```





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

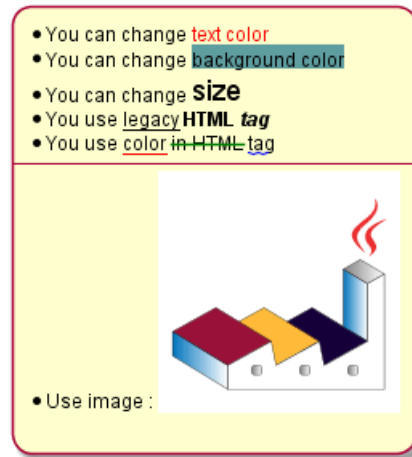
Some HTML tags are also working:

- `` 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
```



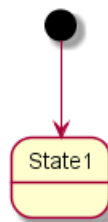
16.7 Table

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

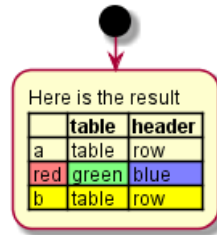
| | table | header |
|---|-------|--------|
| a | table | row |
| b | table | row |



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
```



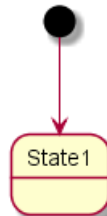


16.8 Tree

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 | ☑ justify-right | 🎵 musical-note | ★ star |
| → account-logout | 📶 bluetooth | ☁️ cloudy | ⏏ expand-down | 🔑 key | 📎 paperclip | ☀ sun |
| ↶ action-redo | ⚙ bolt | 🔗 code | ⏪ expand-left | 💻 laptop | ✎ pencil | 📱 tablet |
| ↷ action-undo | 📖 book | ⚙️ cog | ⏩ expand-right | 📷 layers | 👤 people | 🏷 tag |
| ☐ align-center | 📌 bookmark | ⏴ collapse-down | ⏴ expand-up | 💡 lightbulb | 👤 person | 🏷 tags |
| ☐ align-left | 📦 box | ⏵ collapse-left | 🔗 external-link | 🔗 link-broken | 📱 phone | 🎯 target |
| ☐ align-right | 📁 briefcase | ⏵ collapse-right | 👁 eye | 🔗 link-intact | 📊 pie-chart | 📋 task |
| 🔍 aperture | £ british-pound | ⏴ collapse-up | 👉 eyedropper | 📋 list-rich | 📌 pin | 💻 terminal |
| ↓ arrow-bottom | 📅 browser | 🔑 command | 📁 file | 📋 list | 🎮 play-circle | 📄 text |
| 🕒 arrow-circle-bottom | 🖌 brush | 📄 comment-square | 🔥 fire | 📍 location | ➕ plus | 👇 thumb-down |
| 🕒 arrow-circle-left | 🐛 bug | 📄 compass | 🚩 flag | 🔒 lock-locked | 🔌 power-standby | 👍 thumb-up |
| 🕒 arrow-circle-right | 📢 bullhorn | ⚙ contrast | ⚡ flash | 🔓 lock-unlocked | 🖨 print | ⌚ timer |
| 🕒 arrow-circle-top | 📊 calculator | 📄 copywriting | 📁 folder | 🔄 loop-circular | 📁 project | ➡ transfer |
| ↶ arrow-left | 📅 calendar | 📄 credit-card | 🍴 fork | 📐 loop-square | ★ pulse | 🗑 trash |
| → arrow-right | 📷 camera-slr | 📄 crop | 🖥 fullscreen-enter | 📐 loop | 🧩 puzzle-piece | 📄 underline |
| ↓ arrow-thick-bottom | ↶ caret-bottom | 📄 dashboard | 🔍 fullscreen-exit | 🔍 magnifying-glass | ? question-mark | 📄 vertical-align-bottom |
| ↶ arrow-thick-left | ↶ caret-left | ⬇ data-transfer-download | 🌐 globe | 📍 map-marker | 🌧 rain | 📄 vertical-align-center |
| → arrow-thick-right | ↶ caret-right | ⬆ data-transfer-upload | 📊 graph | 📍 map | 🎲 random | 📄 vertical-align-top |
| ↑ arrow-thick-top | ↶ caret-top | 🗑 delete | 📊 grid-four-up | ⏸ media-pause | 🔄 reload | 📄 video |
| ↑ arrow-top | 📄 cart | 📞 dial | 📊 grid-three-up | ▶ media-play | ↔ resize-both | 🔊 volume-high |
| 🔊 audio-spectrum | 🗨 chat | 📄 document | 📊 grid-two-up | 🎵 media-record | ↕ resize-height | 🔊 volume-low |
| 🏷 badge | ✓ check | 💵 dollar | 💾 hard-drive | ⏮ media-skip-backward | ↔ resize-width | 🔊 volume-off |
| 🚫 ban | ↶ chevron-bottom | 💵 double-quote-sans-left | 📄 header | ▶ media-skip-forward | 📡 rss-alt | ⚠ warning |
| 📊 bar-chart | ↶ chevron-left | 💵 double-quote-sans-right | 🎧 headphones | ⏮ media-step-backward | 📡 rss | 📶 wifi |
| 📊 battery-empty | ↶ chevron-right | 💵 double-quote-serif-left | ♥ heart | ▶ media-step-forward | 📄 script | 🔧 wrench |
| 📊 battery-full | ↶ chevron-top | 💵 double-quote-serif-right | 🏠 home | ⏮ media-stop | 📄 share-boxed | ✕ x |
| 📄 beaker | 🔍 circle-check | 💧 droplet | 🖼 image | 🏥 medical-cross | ➦ share | ¥ yen |
| | 🔍 circle-x | 🚮 eject | 📁 inbox | ☰ menu | 🛡 shield | 📶 zoom-in |
| | 📄 clipboard | 🚶 elevator | ∞ infinity | 🎤 microphone | 📶 signal | 🔍 zoom-out |
| | 🕒 clock | 📄 ellipses | ⌚ info | ➖ minus | 📶 signpost | |
| | ☁ cloud-download | ✉ envelope-closed | 🇮🇹 italic | 📺 monitor | ↗ sort-ascending | |
| | ☁ cloud-upload | ✉ envelope-open | ☰ justify-center | 🌙 moon | ↘ sort-descending | |
| | | € euro | ☰ justify-left | ➡ move | 📄 spreadsheet | |



17 Defining and using sprites

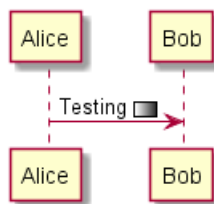
A *Sprite* is a small graphic element that can be used in diagrams.

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.

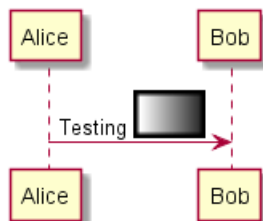
Then you can use the sprite using <\$XXX> where XXX is the name of the sprite.

```
@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

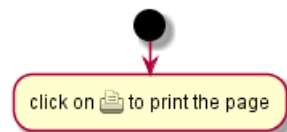
You can also launch the GUI to generate a sprite from an existing image.

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__FEjQEqB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_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>...)

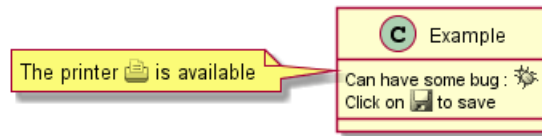
```
class Example {
  Can have some bug : <$bug>
  Click on <$disk> to save
}
```



```
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 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
```



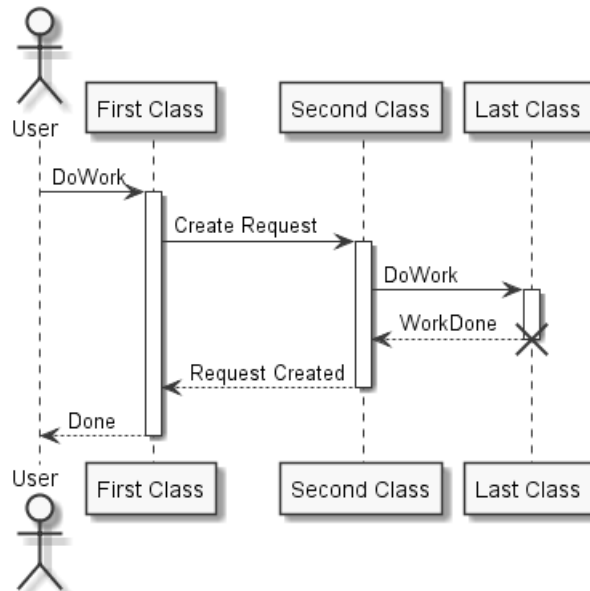
```

deactivate B

A --> User: Done
deactivate A

@enduml

```



18.4 Shadowing

You can disable the shadowing using the `skinparam shadowing false` command.

```

@startuml

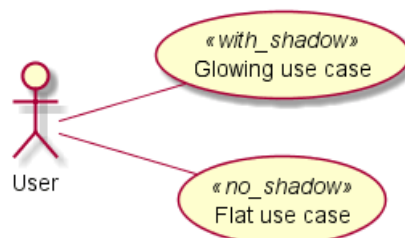
left to right direction

skinparam shadowing<<no_shadow>> false
skinparam shadowing<<with_shadow>> true

actor User
(Glowing use case) <<with_shadow>> as guc
(Flat use case) <<no_shadow>> as fuc
User -- guc
User -- fuc

@enduml

```



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

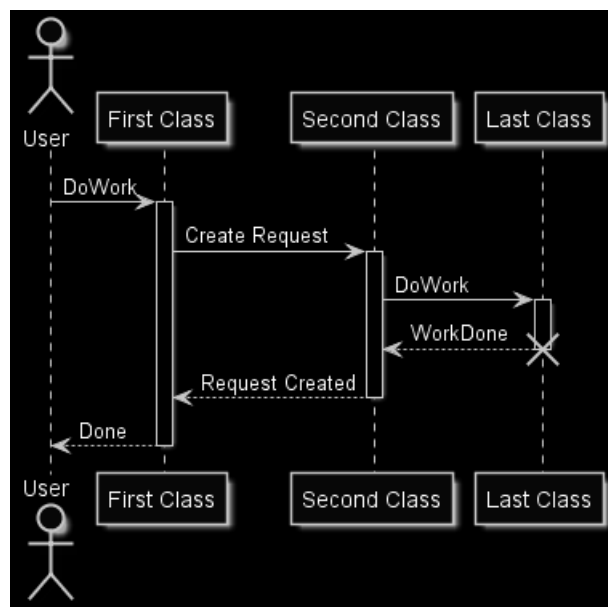
A -> B: Create Request
activate B

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
```

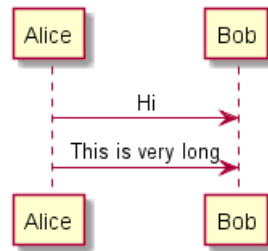
18.8 Text Alignment

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

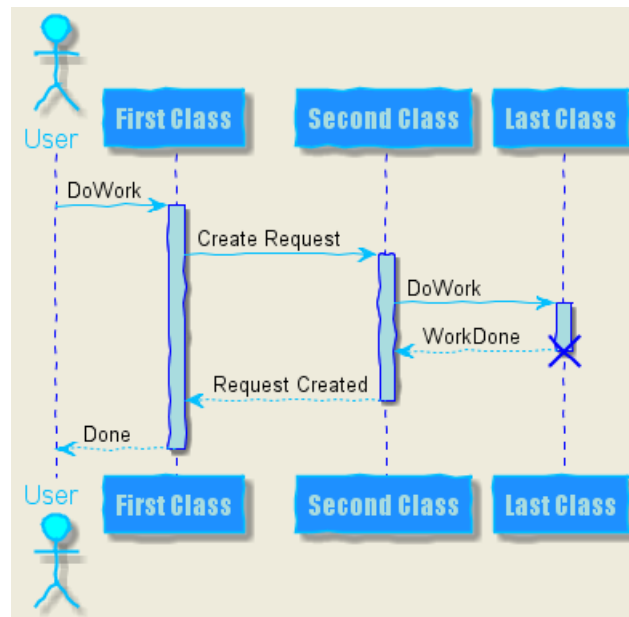
B --> A: Request Created
deactivate B

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

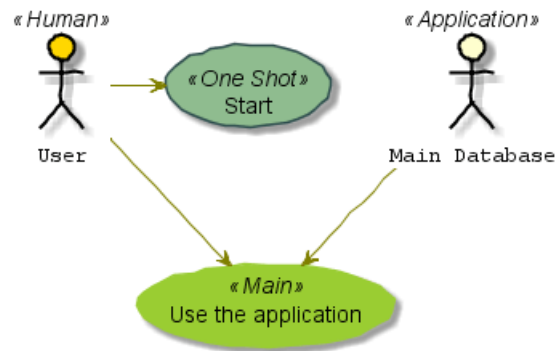
ArrowColor Olive
}

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

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

MySql --> (Use)
@enduml

```



```

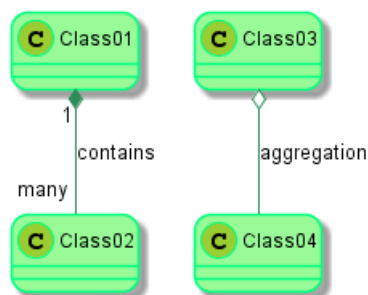
@startuml
skinparam roundcorner 20
skinparam class {
  BackgroundColor PaleGreen
  ArrowColor SeaGreen
  BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen
  
```

```

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
}
  
```

```

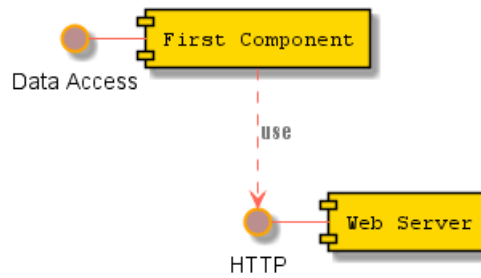
() "Data Access" as DA
  
```

```

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



```
HTTP - [Web Server] << Apache >>
@enduml
```

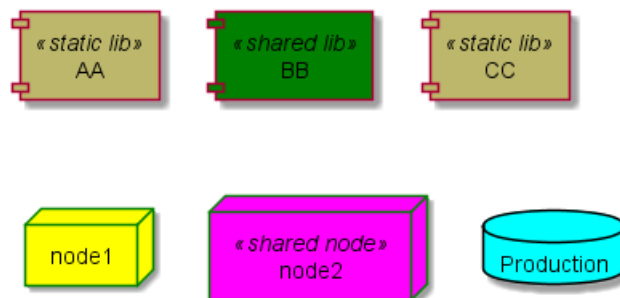


```
@startuml
[AA] <<static lib>>
[BB] <<shared lib>>
[CC] <<static lib>>

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
```



18.10 List of all skinparam parameters

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
```

Or you can generate a "diagram" with a list of all the skinparam parameters using:

```
@startuml
help skinparams
@enduml
```

That will give you the following result:



Help on skinparam

The code of this command is located in *net.sourceforge.plantuml.help* package.

You may improve it on <https://github.com/plantuml/plantuml/tree/master/src/net/sourceforge/plantuml/help>

The possible skinparam are :

- ActivityBackgroundColor
- ActivityBarColor
- ActivityBorderColor
- ActivityBorderThickness
- ActivityDiamondBackgroundColor
- ActivityDiamondBorderColor
- ActivityDiamondFontColor
- ActivityDiamondFontName
- ActivityDiamondFontSize
- ActivityDiamondFontStyle
- ActivityEndColor
- ActivityFontColor
- ActivityFontName
- ActivityFontSize
- ActivityFontStyle
- ActivityStartColor
- ActorBackgroundColor
- ActorBorderColor
- ActorFontColor
- ActorFontName
- ActorFontSize
- ActorFontStyle
- ActorStereotypeFontColor
- ActorStereotypeFontName
- ActorStereotypeFontSize
- ActorStereotypeFontStyle
- AgentBackgroundColor
- AgentBorderColor
- AgentBorderThickness
- AgentFontColor
- AgentFontName
- AgentFontSize
- AgentFontStyle
- AgentStereotypeFontColor
- AgentStereotypeFontName
- AgentStereotypeFontSize
- AgentStereotypeFontStyle
- ArchimateBackgroundColor
- ArchimateBorderColor
- ArchimateBorderThickness
- ArchimateFontColor
- ArchimateFontName
- ArchimateFontSize
- ArchimateFontStyle
- ArchimateStereotypeFontColor
- ArchimateStereotypeFontName
- ArchimateStereotypeFontSize
- ArchimateStereotypeFontStyle
- ArrowColor
- ArrowFontColor
- ArrowFontName
- ArrowFontSize
- ArrowFontStyle
- ArrowLollipopColor
- ArrowMessageAlignment
- ArrowThickness
- ArtifactBackgroundColor

You can also view each skinparam parameters with its results displayed at <https://plantuml-documentation.readthedocs.io/en/latest/formatting/all-skin-params.html>.

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 Migration notes

The actual preprocessor is an update from some legacy preprocessor.

Even if some legacy features are still supported with the actual preprocessor, you should not use them any more (they might be removed in some long term future).

- You should not use `!define` and `!definelong` anymore. Use `!function` and variable definition instead. `!define` should be replaced by `return function` and `!definelong` should be replaced by `void function`.
- `!include` now allows multiple inclusions : you don't have to use `!include_many` anymore
- `!include` now accepts a URL, so you don't need `!includeurl`
- Some features (like `%date%`) have been replaced by builtin functions (for example `%date()`)
- When calling a legacy `!definelong` macro with no arguments, you do have to use parenthesis. You have to use `my_own_definelong()` because `my_own_definelong` without parenthesis is not recognized by the new preprocessor.

Please contact us if you have any issues.

19.2 Variable definition

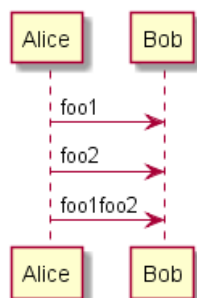
Although this is not mandatory, we highly suggest that variable names start with a `$`. There are two types of data:

- Integer number
- String - these must be surrounded by single quote or double quote.

Variables created outside function are **global**, that is you can access them from everywhere (including from functions). You can emphasize this by using the optional `global` keyword when defining a variable.

```
@startuml
!$ab = "foo1"
!$cd = "foo2"
!global $ef = $ab + $cd
```

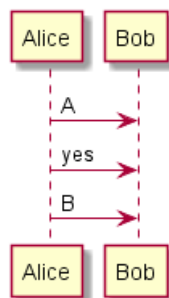
```
Alice -> Bob : $ab
Alice -> Bob : $cd
Alice -> Bob : $ef
@enduml
```



19.3 Conditions

- You can use expression in condition.
- *else* is also implemented

```
@startuml
!$a = 10
!$ijk = "foo"
Alice -> Bob : A
!if ($ijk == "foo") && ($a+10>=4)
Alice -> Bob : yes
!else
Alice -> Bob : This should not appear
!endif
Alice -> Bob : B
@enduml
```



19.4 Void function

- Function names *must* start with a \$
- Argument names *must* start with a \$
- Void functions can call other void functions

Example:

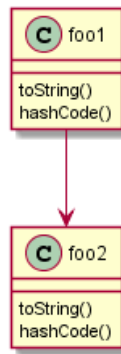
```
@startuml
!function msg($source, $destination)
$source --> $destination
!endfunction
```

```
!function init_class($name)
class $name {
$addCommonMethod()
}
!endfunction
```

```
!function $addCommonMethod()
    toString()
    hashCode()
!endfunction
```

```
init_class("foo1")
init_class("foo2")
msg("foo1", "foo2")
@enduml
```





Variables defined in functions are **local**. It means that the variable is destroyed when the function ends.

19.5 Return function

A return function does not output any text. It just define a function that you can call:

- directly in variable definition or in diagram text
- from other return function
- from other void function
- Function name *should* start by a \$
- Argument names *should* start by a \$

```

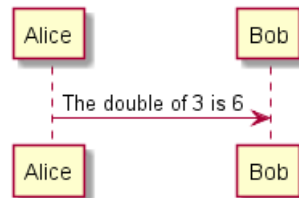
@startuml
!function $double($a)
!return $a + $a
!endfunction

```

```

Alice -> Bob : The double of 3 is $double(3)
@enduml

```



It is possible to shorten simple function definition in one line:

```

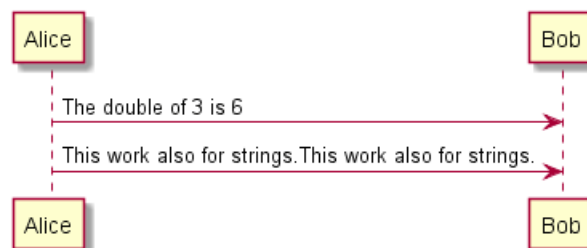
@startuml
!function $double($a) return $a + $a

```

```

Alice -> Bob : The double of 3 is $double(3)
Alice -> Bob : $double("This work also for strings.")
@enduml

```

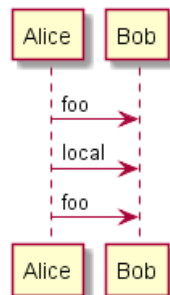


As in void function, variables are local by default (they are destroyed when the function is exited). However, you can access global variables from a function. However, you can use the `local` keyword to create a local variable if ever a global variable exists with the same name.

```
@startuml
!function $dummy()
!local $ijk = "local"
Alice -> Bob : $ijk
!endfunction

!global $ijk = "foo"

Alice -> Bob : $ijk
$dummy()
Alice -> Bob : $ijk
@enduml
```

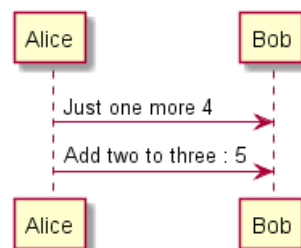


19.6 Default argument value

In both return and void functions, you can define default values for arguments.

```
@startuml
!function $inc($value, $step=1)
!return $value + $step
!endfunction

Alice -> Bob : Just one more $inc(3)
Alice -> Bob : Add two to three : $inc(3, 2)
@enduml
```

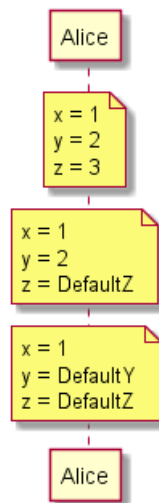


Only arguments at the end of the parameter list can have default values.

```
@startuml
!function defaulttest($x, $y="DefaultY", $z="DefaultZ")
note over Alice
    x = $x
    y = $y
    z = $z
end note
!endfunction
```



```
defaulttest(1, 2, 3)
defaulttest(1, 2)
defaulttest(1)
@enduml
```

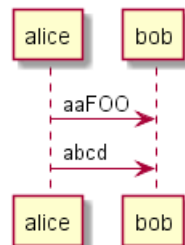


19.7 Unquoted function

By default, you have to put quotes when you call a function. It is possible to use the `unquoted` keyword to indicate that a function does not require quotes for its arguments.

```
@startuml
!unquoted function id($text1, $text2="FOO") return $text1 + $text2

alice -> bob : id(aa)
alice -> bob : id(ab,cd)
@enduml
```



19.8 Including files or URL

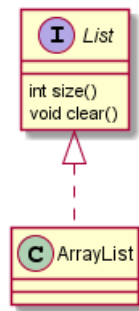
Use the `!include` directive to include file in your diagram. Using URL, you can also include file from Internet/Intranet.

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.

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. The `!0` notation denotes the first diagram.

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`.

By default, a file can only be included once. You can use `!include_many` instead of `!include` if you want to include some file several times. Note that there is also a `!include_once` directive that raises an error if a file is included several times.

19.9 Including Subpart

You can also use `!startsub NAME` and `!endsub` to indicate sections of text to include from other files using `!includesub`. For example:

file1.puml:

```

@startuml

A -> A : stuff1
!startsub BASIC
B -> B : stuff2
!endsub
C -> C : stuff3
!startsub BASIC
D -> D : stuff4
!endsub
@enduml
  
```

`file1.puml` would be rendered exactly as if it were:

```

@startuml

A -> A : stuff1
B -> B : stuff2
C -> C : stuff3
D -> D : stuff4
@enduml
  
```



However, this would also allow you to have another file2.puml like this:

file2.puml

```
@startuml
```

```
title this contains only B and D
!includesub file1.puml!BASIC
@enduml
```

This file would be rendered exactly as if:

```
@startuml
```

```
title this contains only B and D
B -> B : stuff2
D -> D : stuff4
@enduml
```

19.10 Builtin functions

Some functions are defined by default. Their name starts by %

| Name | Description | |
|---------------------|--|---------------------|
| %strlen | Calculate the length of a String | % |
| %substr | Extract a substring. Takes 2 or 3 arguments %substr("abcdef", 3, 2) | "d |
| %strpos | Search a substring in a string | %strp |
| %intval | Convert a String to Int | % |
| %file_exists | Check if a file exists on the local filesystem | %file_exis |
| %function_exists | Check if a function exists | %function_e |
| %variable_exists | Check if a variable exists | %variable_ |
| %set_variable_value | Set a global variable | %set_variable_value |
| %get_variable_value | Retrieve some variable value | %get_variab. |
| %getenv | Retrieve environment variable value | % |
| %dirpath | Retrieve current dirpath | |
| %filename | Retrieve current filename | |
| %date | Retrieve current date. You can provide an optional format for the date | %date("y |
| %true | Return always true | |
| %false | Return always false | |
| %not | Return the logical negation of an expression | |

19.11 Logging

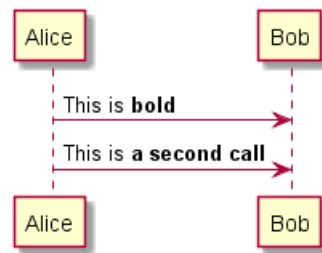
You can use !log to add some log output when generating the diagram. This has no impact at all on the diagram itself. However, those logs are printed in the command line's output stream. This could be useful for debug purpose.

```
@startuml
```

```
!function bold($text)
!$result = "<b>"+ $text + "</b>"
!log Calling bold function with $text. The result is $result
!return $result
!endfunction
```

```
Alice -> Bob : This is bold("bold")
Alice -> Bob : This is bold("a second call")
@enduml
```





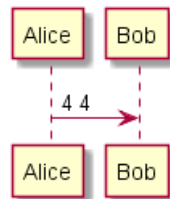
19.12 Memory dump

You can use `!memory_dump` to dump the full content of the memory when generating the diagram. An optional string can be put after `!memory_dump`. This has no impact at all on the diagram itself. This could be useful for debug purpose.

```

@startuml
!function $inc($string)
!$val = %intval($string)
!log value is $val
!dump_memory
!return $val+1
!endfunction

Alice -> Bob : 4 $inc("3")
!unused = "foo"
!dump_memory EOF
@enduml
  
```



19.13 Assertion

You can put assertion in your diagram.

```

@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fail"
@enduml
  
```



Welcome to PlantUML!

If you use this software, you accept its license.
(details by typing license keyword)

You can start with a simple UML Diagram like:

Bob->Alice: Hello

Or

```
class Example
```

You will find more information about PlantUML syntax on <http://plantuml.com>



[From string (line 3)]

```
@startuml
Alice -> Bob : Hello
!assert %strpos("abcdef", "cd")==3 : "This always fail"
Assertion error : This always fail
```

19.14 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

...
```

19.15 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.

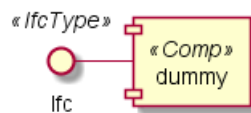
19.16 Argument concatenation

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

```
@startuml
!unquoted function COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!endfunction
```



```
COMP_TEXTGENCOMP(dummy)
@enduml
```



19.17 Dynamic function invocation

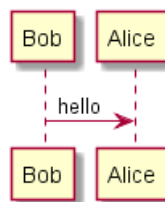
You can dynamically invoke a void function using the special `%invoke_void_func()` void function. This function takes as first argument the name of the actual void function to be called. The following argument are copied to the called function.

For example, you can have:

```
@startuml
!function $go()
  Bob -> Alice : hello
!endfunction

!$wrapper = "$go"

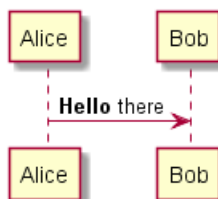
%invoke_void_func($wrapper)
@enduml
```



For return functions, you can use the corresponding special function `%call_user_func()` :

```
@startuml
!function bold($text)
!return "<b>"+ $text + "</b>"
!endfunction
```

```
Alice -> Bob : %call_user_func("bold", "Hello") there
@enduml
```



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
```

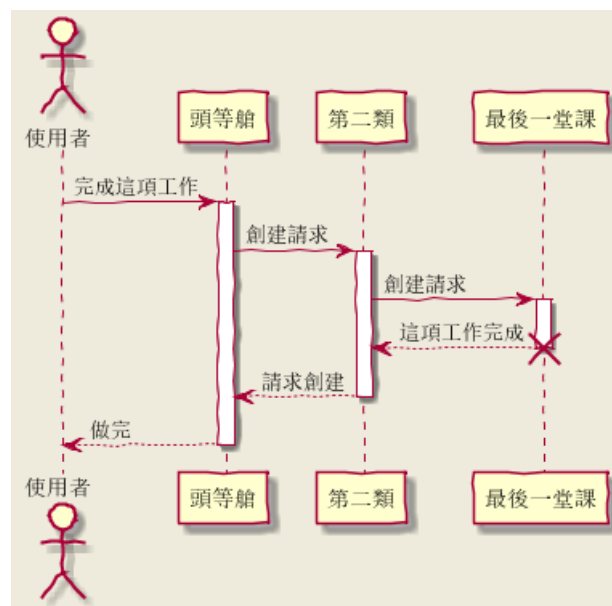
```
使用者 -> A: 完成這項工作
activate A
```

```
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
```



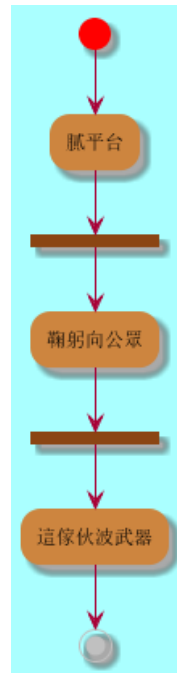
```
@startuml

(*) --> "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
```



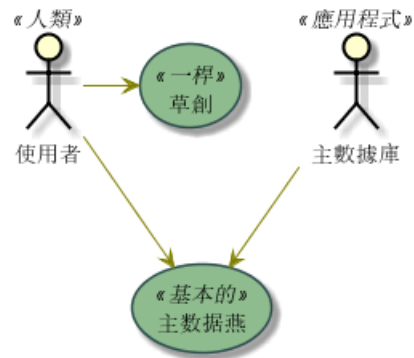
```
@startuml
skinparam usecaseBackgroundColor DarkSeaGreen
skinparam usecaseArrowColor Olive
skinparam actorBorderColor black
skinparam usecaseBorderColor DarkSlateGray
```

```
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
```





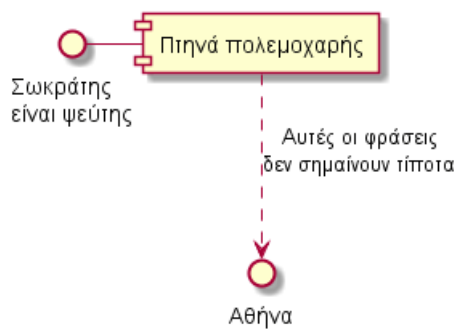
@startuml

() "Σωκράτης ψεύτης" as Σωκράτης

Σωκράτης - [Πτηνά πολεμοχαρές]

[Πτηνά πολεμοχαρές] ..> () Αθήνα : Αυτές οι φράσεις σημαίνουν 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 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.

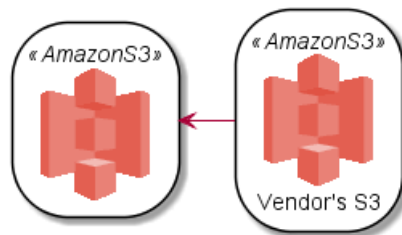
Use it by including the file that contains the sprite, eg: `!include <aws/Storage/AmazonS3/AmazonS3>`. When imported, you can use the sprite as normally you would, using `<$sprite_name>`.

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>
```

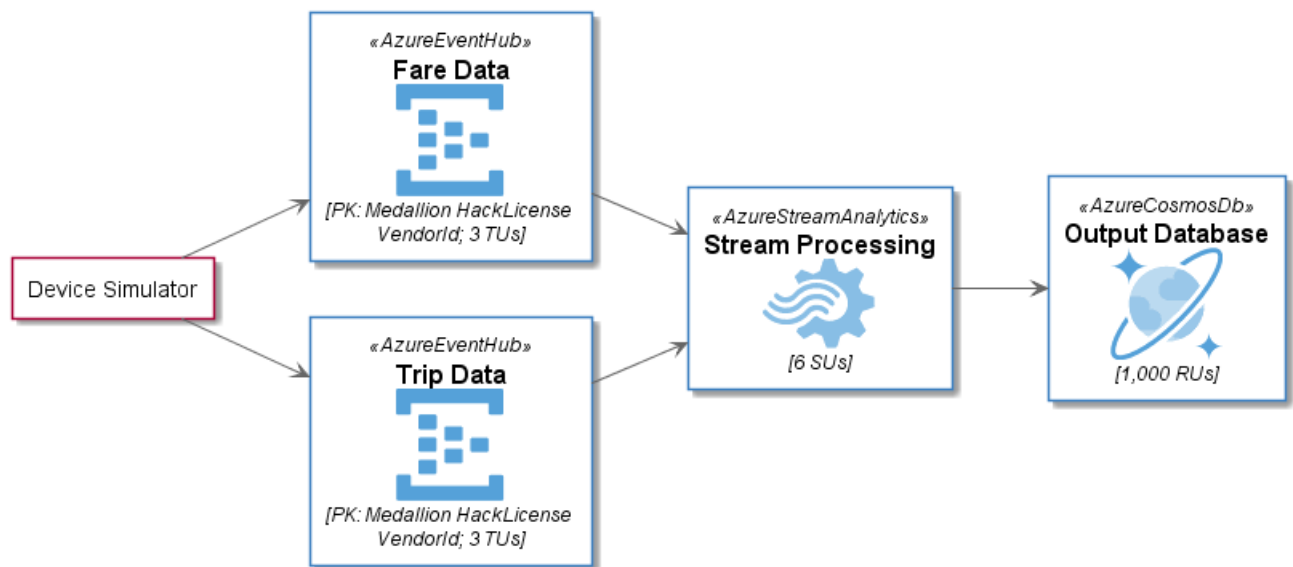
left to right direction




```
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

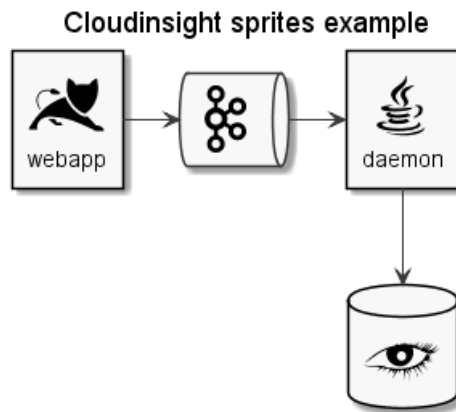
skinparam monochrome true

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

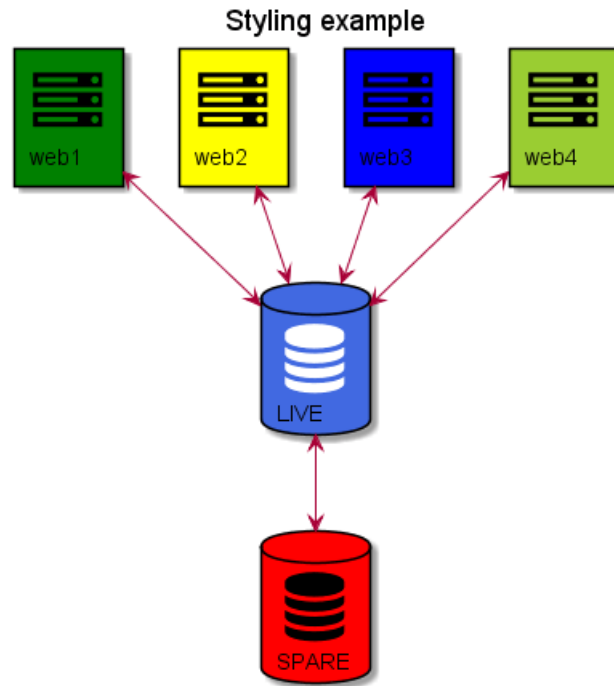
FA_DATABASE(db1,LIVE,database,white) #RoyalBlue
FA_DATABASE(db2,SPARE,database) #Red

db1 <--> db2

web1 <--> db1
web2 <--> db1
web3 <--> db1
web4 <--> db1
@enduml

```

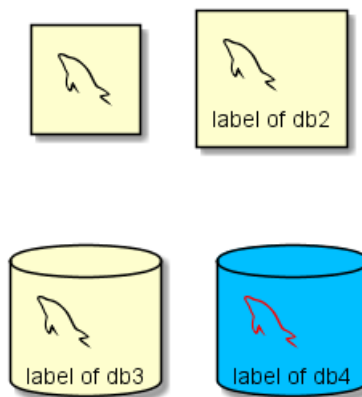




```

@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

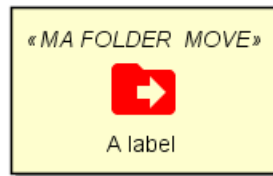


again the use of the prefix MA_.

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

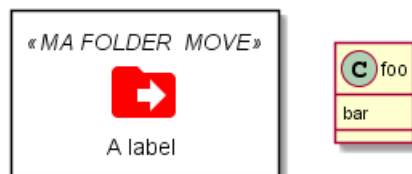
When mixing sprites macros with other elements you may get a syntax error if, for example, trying to add a rectangle along with classes. In those cases, add { and } after the macro to create the empty rectangle.

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>
```

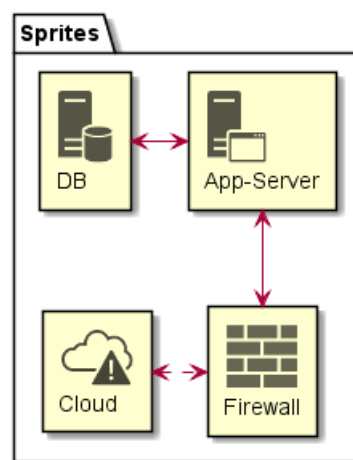


```
title Office Icons Example
```

```
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
```

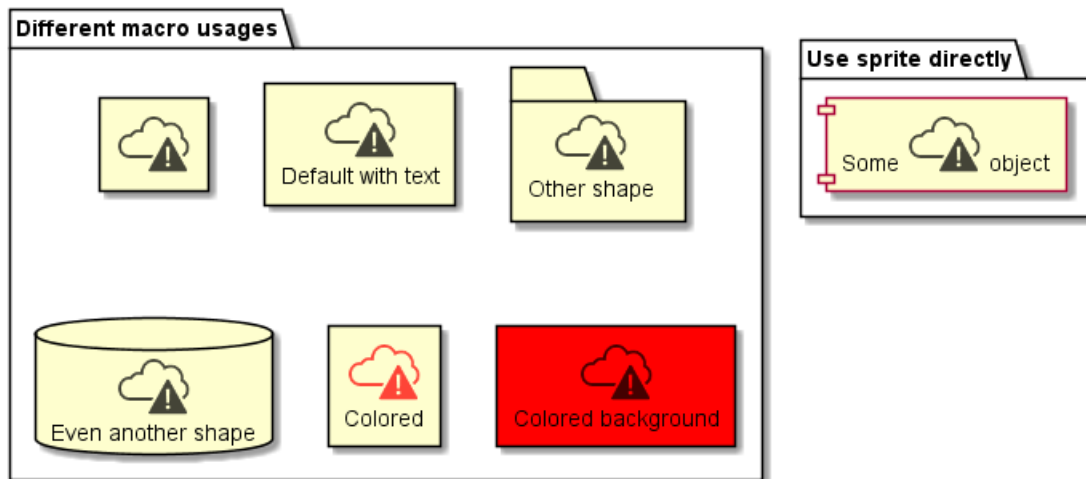
```
title Extended Office Icons Example
```

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

```
package "Different macro usages" {
  OFF_CLOUD_DISASTER_RED(cloud1)
  OFF_CLOUD_DISASTER_RED(cloud2,Default with text)
  OFF_CLOUD_DISASTER_RED(cloud3,Other shape,Folder)
  OFF_CLOUD_DISASTER_RED(cloud4,Even another shape,Database)
  OFF_CLOUD_DISASTER_RED(cloud5,Colored,Rectangle, red)
  OFF_CLOUD_DISASTER_RED(cloud6,Colored background) #red
}
@enduml
```



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 consistently.

```
@startuml
!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")

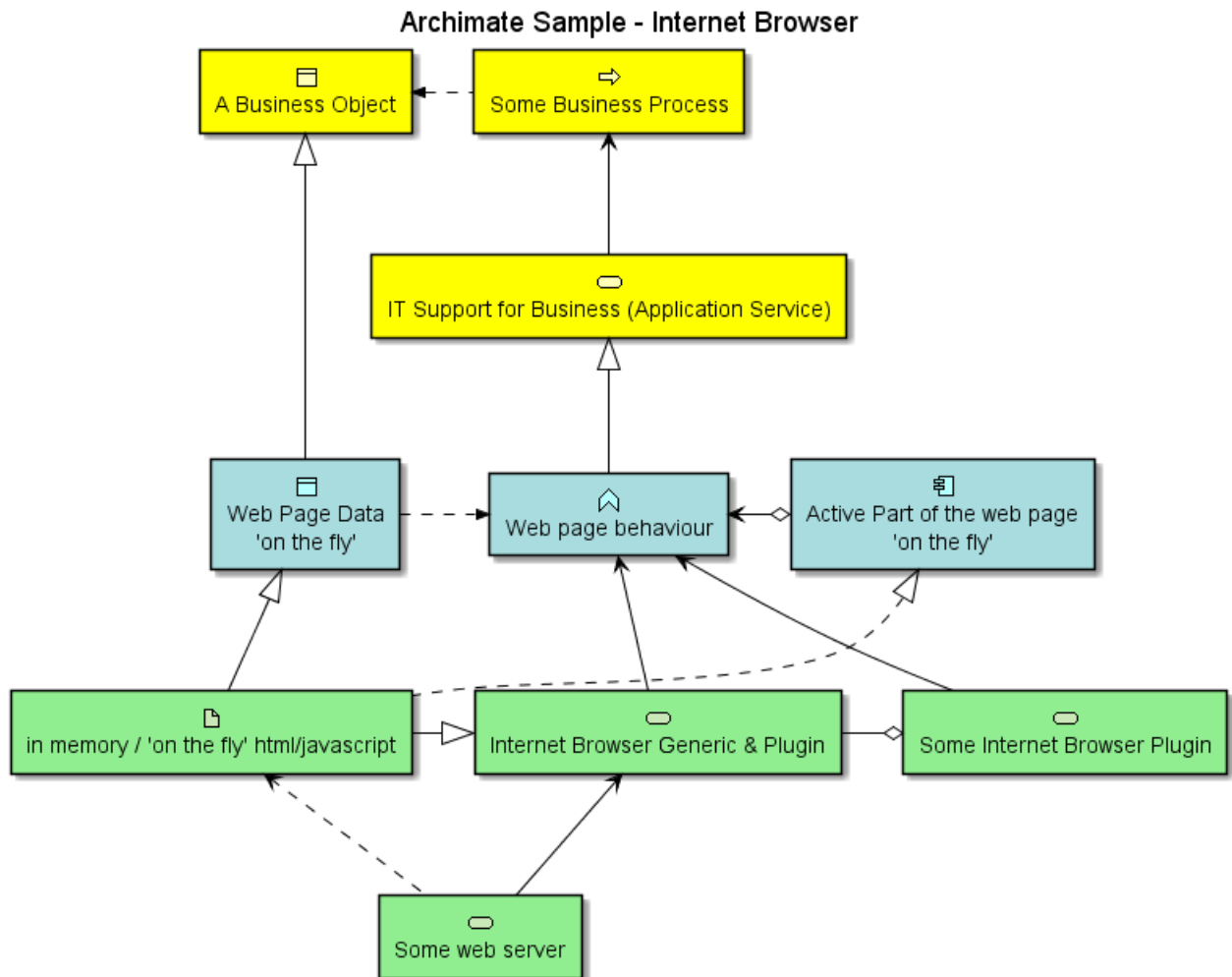
'Relationships
Rel_Flow_Left(someBusinessProcess, businessObject, "")
Rel_Serving_Up(itSupportService, someBusinessProcess, "")
Rel_Specialization_Up(webpageBehaviour, itSupportService, "")
Rel_Flow_Right(dataObject, webpageBehaviour, "")
Rel_Specialization_Up(dataObject, businessObject, "")
Rel_Assignment_Left(ActivePartWebPage, webpageBehaviour, "")
Rel_Specialization_Up(inMemoryItem, dataObject, "")
Rel_Realization_Up(inMemoryItem, ActivePartWebPage, "")
Rel_Specialization_Right(inMemoryItem,internetBrowser, "")
Rel_Serving_Up(internetBrowser, webpageBehaviour, "")
Rel_Serving_Up(internetBrowserPlugin, webpageBehaviour, "")
Rel_Aggregation_Right(internetBrowser, internetBrowserPlugin, "")
Rel_Access_Up(webServer, inMemoryItem, "")
```



```

Rel_Serving_Up(webServer, internetBrowser, "")
@enduml

```



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>**awslib**

Version 3.0.0

Delivered by <https://github.com/aws-labs/aws-icons-for-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>**osa**

Version 0.0.1

Delivered by <https://github.com/Crashedmind/PlantUML-opensecurityarchitecture-icons>**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 | Диаграмма последовательности | 1 |
| 1.1 | Основные примеры | 1 |
| 1.2 | Объявление участников | 1 |
| 1.3 | Использование небуквенных символов в названиях участников | 3 |
| 1.4 | Сообщения к самому себе | 3 |
| 1.5 | Изменить стиль стрелок | 3 |
| 1.6 | Изменить цвет стрелок | 4 |
| 1.7 | Нумерация сообщений в последовательностях | 4 |
| 1.8 | Page Title, Header and Footer | 7 |
| 1.9 | Разбиение диаграмм | 7 |
| 1.10 | Группировка сообщений | 8 |
| 1.11 | Примечания в сообщениях | 9 |
| 1.12 | Другие примечания | 10 |
| 1.13 | Изменение формы примечаний | 10 |
| 1.14 | Creole и HTML | 11 |
| 1.15 | Разделитель | 12 |
| 1.16 | Ссылки | 13 |
| 1.17 | Задержка на диаграммах | 13 |
| 1.18 | Промежутки | 14 |
| 1.19 | Активация и деактивация линии существования | 15 |
| 1.20 | Return | 16 |
| 1.21 | Отображение создания участника процессом | 16 |
| 1.22 | Входящие и исходящие сообщения | 17 |
| 1.23 | Шаблоны и отметки | 18 |
| 1.24 | Больше информации в заголовках | 19 |
| 1.25 | Группировка участников | 20 |
| 1.26 | Удаление футера | 21 |
| 1.27 | Skinparam | 21 |
| 1.28 | Изменение отступов | 23 |
| 2 | Диаграмма прецедентов | 25 |
| 2.1 | Прецеденты | 25 |
| 2.2 | Актёры | 25 |
| 2.3 | Описание прецедентов | 26 |
| 2.4 | Простой пример | 26 |
| 2.5 | Расширение | 27 |
| 2.6 | Использование заметок | 27 |
| 2.7 | Шаблоны | 28 |
| 2.8 | Смена направления стрелок | 29 |
| 2.9 | Разделение диаграмм | 30 |
| 2.10 | Направление слева направо | 30 |
| 2.11 | Skinparam | 31 |
| 2.12 | Полноценный пример | 32 |
| 3 | Диаграмма классов | 33 |
| 3.1 | Отношения между классами | 33 |
| 3.2 | Метки на отношениях | 34 |
| 3.3 | Добавление методов | 35 |
| 3.4 | Указание видимости | 36 |
| 3.5 | Абстрактные и статические | 36 |
| 3.6 | Расширенное тело класса | 37 |
| 3.7 | Заметки и шаблоны | 38 |
| 3.8 | Больше о заметках | 38 |
| 3.9 | Заметки на связях | 39 |
| 3.10 | Абстрактные классы и интерфейсы | 40 |
| 3.11 | Использование не буквенных символов | 41 |
| 3.12 | Скрытие атрибутов, методов... | 41 |



| | | |
|----------|--|-----------|
| 3.13 | Скрытие классов | 42 |
| 3.14 | Использование дженериков | 43 |
| 3.15 | Определение метки | 43 |
| 3.16 | Пакеты | 43 |
| 3.17 | Стили пакетов | 44 |
| 3.18 | Пространства имён | 45 |
| 3.19 | Автоматическое создание пространств имён | 46 |
| 3.20 | Lollipop интерфейс | 47 |
| 3.21 | Изменение направления стрелок | 47 |
| 3.22 | Ассоциация классов | 48 |
| 3.23 | Skinparam | 49 |
| 3.24 | Шаблоны со Skinparam | 50 |
| 3.25 | Цветовой градиент | 50 |
| 3.26 | Помощь в расположении классов | 51 |
| 3.27 | Разделение больших файлов | 52 |
| 4 | Диаграмма деятельности | 54 |
| 4.1 | Простая деятельность | 54 |
| 4.2 | Метка на стрелках | 54 |
| 4.3 | Изменение направления стрелки | 54 |
| 4.4 | Ветвления | 55 |
| 4.5 | Больше о ветках | 56 |
| 4.6 | Синхронизация | 57 |
| 4.7 | Длинное описание активности | 58 |
| 4.8 | Заметки | 58 |
| 4.9 | Разделы | 59 |
| 4.10 | Skinparam | 60 |
| 4.11 | Восьмиугольник | 61 |
| 4.12 | Полноценный пример | 61 |
| 5 | Диаграмма активности (бета) | 64 |
| 5.1 | Простая активность | 64 |
| 5.2 | Старт/Стоп | 64 |
| 5.3 | Условия | 65 |
| 5.4 | Повторяющийся цикл | 66 |
| 5.5 | Цикл while | 67 |
| 5.6 | Параллельные процессы | 67 |
| 5.7 | Заметки | 68 |
| 5.8 | Цвета | 69 |
| 5.9 | Стрелки | 69 |
| 5.10 | Connector | 70 |
| 5.11 | Группирование | 70 |
| 5.12 | Дорожки | 71 |
| 5.13 | Отсоединение | 72 |
| 5.14 | SDL | 73 |
| 5.15 | Полноценный пример | 74 |
| 6 | Диаграмма компонентов | 76 |
| 6.1 | Компоненты | 76 |
| 6.2 | Интерфейсы | 76 |
| 6.3 | Простой пример | 77 |
| 6.4 | Использование заметок | 77 |
| 6.5 | Группирование компонентов | 78 |
| 6.6 | Изменение направления стрелок | 79 |
| 6.7 | Использовании нотации UML2 | 80 |
| 6.8 | Длинное описание | 81 |
| 6.9 | Индивидуальные цвета | 81 |
| 6.10 | Использование Sprite в стереотипах | 81 |
| 6.11 | Skinparam | 82 |



| | | |
|-----------|---|------------|
| 7 | Диаграмма состояний | 84 |
| 7.1 | Простое состояние | 84 |
| 7.2 | Change state rendering | 84 |
| 7.3 | Составное состояние | 85 |
| 7.4 | Длинные имена | 86 |
| 7.5 | Параллельные состояния | 87 |
| 7.6 | Направления стрелок | 88 |
| 7.7 | Заметки | 89 |
| 7.8 | Еще о заметках | 90 |
| 7.9 | Skinparam | 90 |
| 8 | Диаграмма объектов | 92 |
| 8.1 | Определение объектов | 92 |
| 8.2 | Отношения между объектами | 92 |
| 8.3 | Добавление полей | 92 |
| 8.4 | Общие с диаграммами классов функции | 93 |
| 9 | Timing Diagram | 94 |
| 9.1 | Declaring participant | 94 |
| 9.2 | Adding message | 94 |
| 9.3 | Relative time | 95 |
| 9.4 | Participant oriented | 96 |
| 9.5 | Setting scale | 96 |
| 9.6 | Initial state | 96 |
| 9.7 | Intricated state | 97 |
| 9.8 | Hidden state | 98 |
| 9.9 | Adding constraint | 98 |
| 9.10 | Adding texts | 99 |
| 10 | Gantt Diagram | 100 |
| 10.1 | Declaring tasks | 100 |
| 10.2 | Adding constraints | 100 |
| 10.3 | Short names | 100 |
| 10.4 | Customize colors | 101 |
| 10.5 | Milestone | 101 |
| 10.6 | Calendar | 101 |
| 10.7 | Close day | 101 |
| 10.8 | Simplified task succession | 102 |
| 10.9 | Separator | 102 |
| 10.10 | Working with resources | 102 |
| 10.11 | Complex example | 103 |
| 11 | MindMap | 104 |
| 11.1 | OrgMode syntax | 104 |
| 11.2 | Removing box | 104 |
| 11.3 | Arithmetic notation | 105 |
| 11.4 | Markdown syntax | 105 |
| 11.5 | Changing diagram direction | 106 |
| 11.6 | Complete example | 106 |
| 12 | Work Breakdown Structure | 108 |
| 12.1 | OrgMode syntax | 108 |
| 12.2 | Change direction | 108 |
| 12.3 | Arithmetic notation | 109 |
| 13 | Maths | 111 |
| 13.1 | Standalone diagram | 111 |
| 13.2 | How is this working ? | 112 |



| | |
|---|------------|
| 14 Common commands | 113 |
| 14.1 Comments | 113 |
| 14.2 Footer and header | 113 |
| 14.3 Zoom | 113 |
| 14.4 Title | 114 |
| 14.5 Caption | 115 |
| 14.6 Legend the diagram | 115 |
| 15 Salt | 117 |
| 15.1 Простые виджеты | 117 |
| 15.2 Использование сетки | 117 |
| 15.3 Group box | 118 |
| 15.4 Использование разделителя | 118 |
| 15.5 Древовидный виджет | 119 |
| 15.6 Окружающие скобки | 119 |
| 15.7 Добавление вкладок | 120 |
| 15.8 Использование меню | 120 |
| 15.9 Продвинутая таблица | 121 |
| 15.10OpenIconic | 122 |
| 15.11Include Salt | 122 |
| 15.12Scroll Bars | 125 |
| 16 Creole | 127 |
| 16.1 Emphasized text | 127 |
| 16.2 List | 127 |
| 16.3 Escape character | 128 |
| 16.4 Horizontal lines | 128 |
| 16.5 Headings | 129 |
| 16.6 Legacy HTML | 129 |
| 16.7 Table | 130 |
| 16.8 Tree | 131 |
| 16.9 Special characters | 131 |
| 16.10OpenIconic | 131 |
| 17 Defining and using sprites | 133 |
| 17.1 Encoding Sprite | 134 |
| 17.2 Importing Sprite | 134 |
| 17.3 Examples | 134 |
| 18 Skinparam command | 136 |
| 18.1 Usage | 136 |
| 18.2 Nested | 136 |
| 18.3 Black and White | 136 |
| 18.4 Shadowing | 137 |
| 18.5 Reverse colors | 138 |
| 18.6 Colors | 138 |
| 18.7 Font color, name and size | 139 |
| 18.8 Text Alignment | 139 |
| 18.9 Examples | 140 |
| 18.10List of all skinparam parameters | 143 |
| 19 Preprocessing | 146 |
| 19.1 Migration notes | 146 |
| 19.2 Variable definition | 146 |
| 19.3 Conditions | 147 |
| 19.4 Void function | 147 |
| 19.5 Return function | 148 |
| 19.6 Default argument value | 149 |
| 19.7 Unquoted function | 150 |



| | |
|--|------------|
| 19.8 Including files or URL | 150 |
| 19.9 Including Subpart | 151 |
| 19.10 Builtin functions | 152 |
| 19.11 Logging | 152 |
| 19.12 Memory dump | 153 |
| 19.13 Assertion | 153 |
| 19.14 Building custom library | 154 |
| 19.15 Search path | 154 |
| 19.16 Argument concatenation | 154 |
| 19.17 Dynamic function invocation | 155 |
| 20 Unicode | 156 |
| 20.1 Examples | 156 |
| 20.2 Charset | 158 |
| 21 Standard Library | 159 |
| 21.1 AWS library | 159 |
| 21.2 Azure library | 159 |
| 21.3 Cloud Insight | 160 |
| 21.4 Devicons and Font Awesome library | 161 |
| 21.5 Google Material Icons | 162 |
| 21.6 Office | 163 |
| 21.7 ArchiMate | 165 |
| 21.8 Miscellaneous | 166 |

