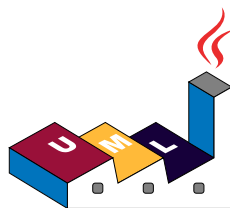


Построение диаграмм UML с использованием PlantUML



Справочное руководство по языку
(19 Сентябрь 2017 г. 8:38)

PlantUML - это проект с открытым кодом, позволяющий быстро создавать:

- Диаграммы последовательности (Sequence diagram),
- Диаграммы прецедентов (Usecase diagram),
- Диаграммы классов (Class diagram),
- Диаграммы активности (Activity diagram),
- Диаграммы компонентов (Component diagram),
- Диаграммы состояний (State diagram),
- Диаграммы объектов (Object diagram).

Для создания диаграмм применяется простой и интуитивно понятный язык.

1. Диаграммы последовательности

1.1. Основные примеры

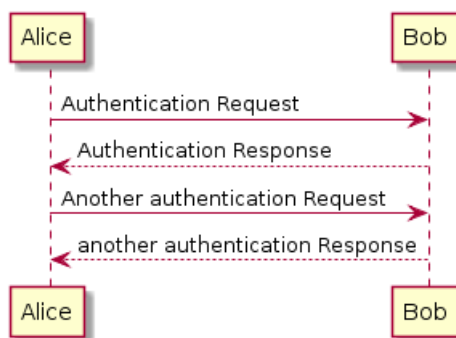
Последовательность ">" используется, чтобы нарисовать сообщение между двумя участниками. Участники не должны быть явно объявлены.

Для того, чтобы получить a dotted arrow, используйте "-->".

Также возможно использовать "<" и "<--". Это не изменит изображения, 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
```



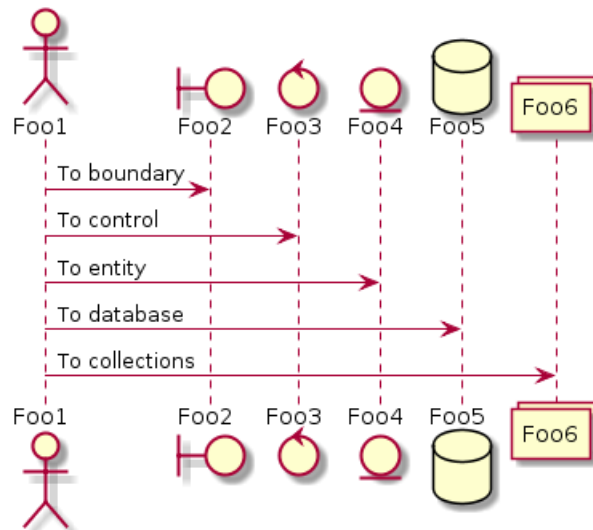
1.2. Объявление участников

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

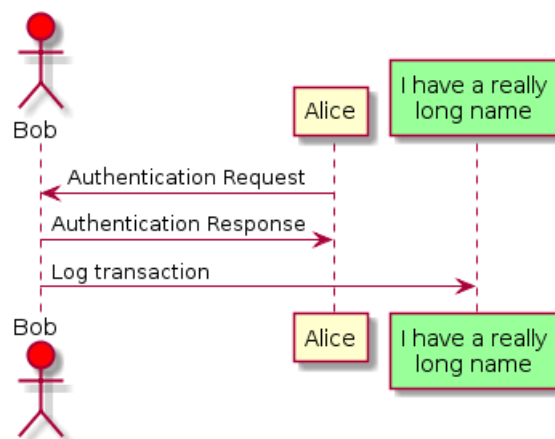
Также возможно изменить цвет фона actor-а или участника, используя имя цвета или его html-код

```

@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
    
```



1.3. Использование небуквенных символов в названиях участников

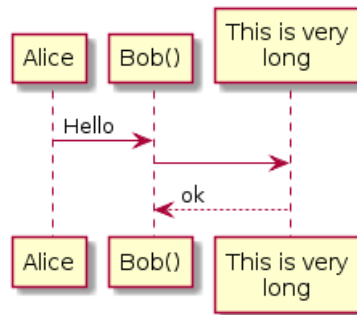
Вы можете использовать кавычки для задания участников. Также Вы можете использовать ключевое слово as для присвоения псевдонимов к этим участникам.

```

@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
```

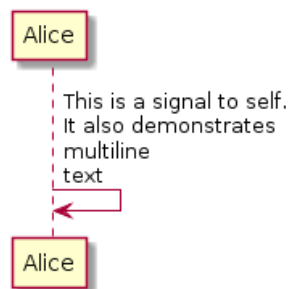


1.4. Сообщения к самому себе

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:

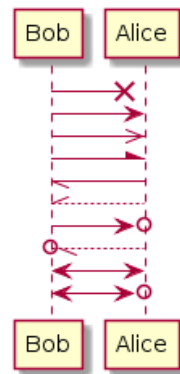
- закончить стрелку с помощью x для обозначения потерянного сообщения
- используя \ или / вместо < или > для создания только верхней или нижней части стрелки.
- повторите окончание стрелки (например, >> or //) для тонкой отрисовки.
- используйте -- вместо - для создания пунктирной стрелки
- заканчивать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

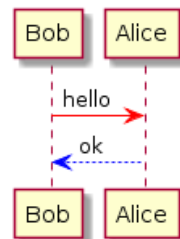




1.6. Изменить цвет стрелок

Вы можете изменить цвет отдельных стрелок, используя следующие правила:

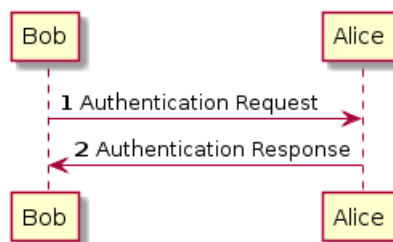
```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



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
```

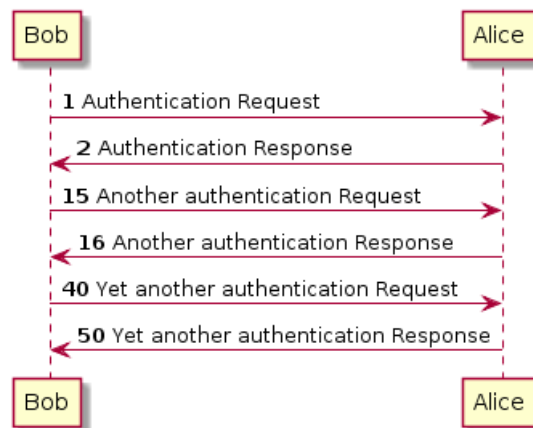


```

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.

```

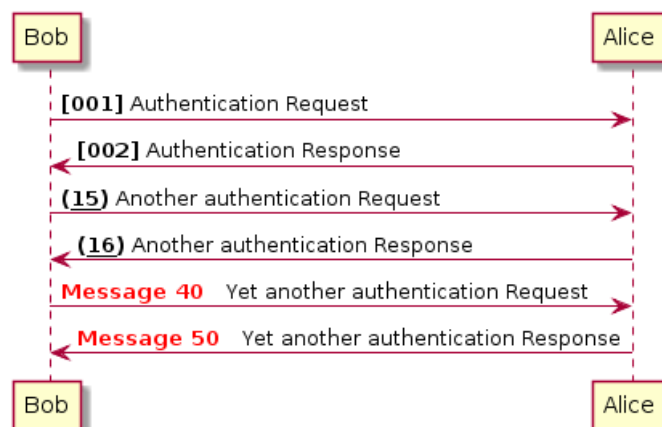
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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

@enduml

```



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

```



```

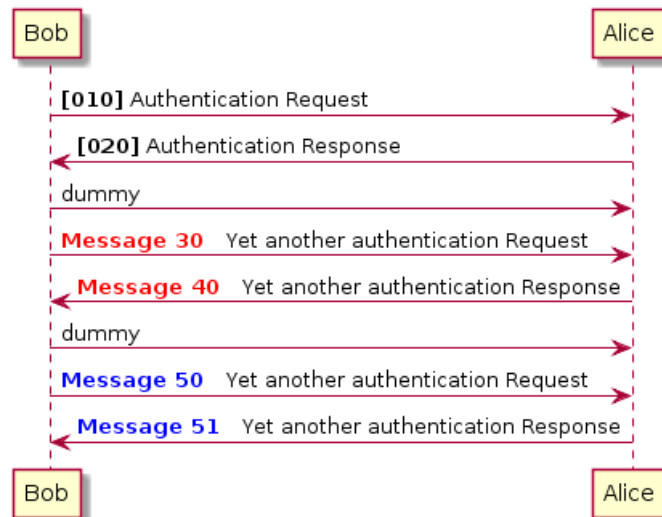
autonumber stop
Bob -> Alice : dummy

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. Разбиение диаграм

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

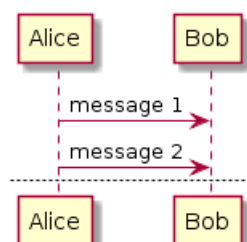
```

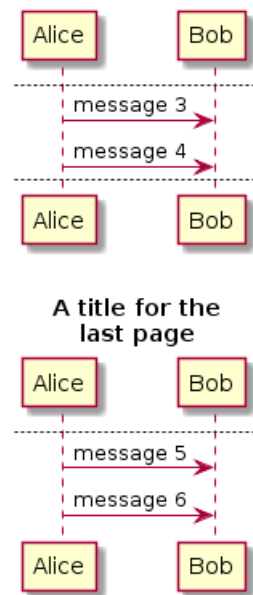
```
newpage A title for the\nlast page
```

```

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

```





1.9. 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.

```

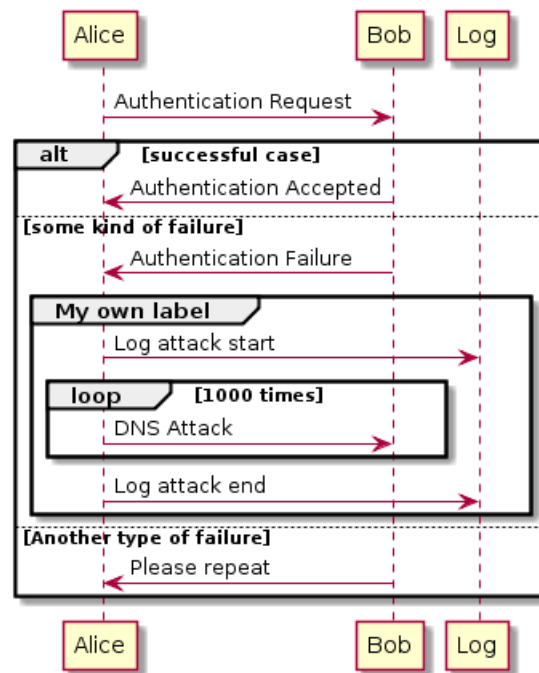
@startuml
Alice -> Bob: Authentication Request

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
  
```





1.10. 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.

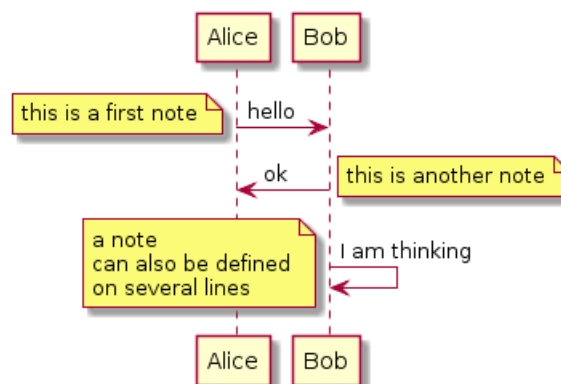
```

@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

```



1.11. Другие примечания

Так же возможно размещение примечаний относительно участников с использованием ключевых слов `note left of`, `note right of` или `note over`.



Возможно выделить примечание изменив цвет фона.

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

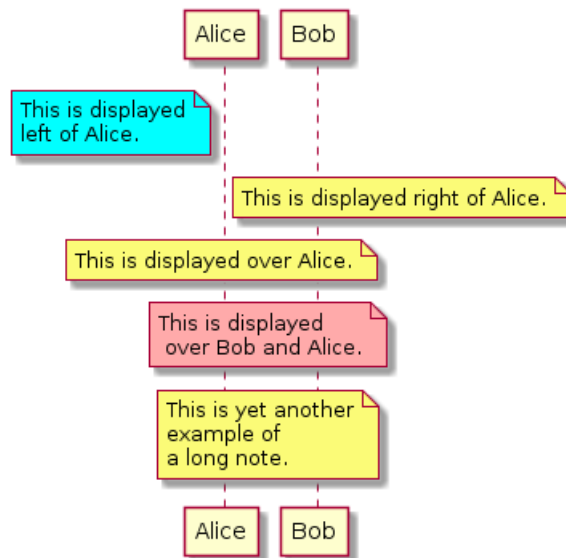
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note
```

```
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.
```

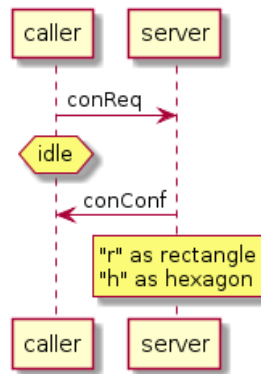
```
note over Bob, Alice
This is yet another
example of
a long note.
end note
@enduml
```



1.12. 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
```



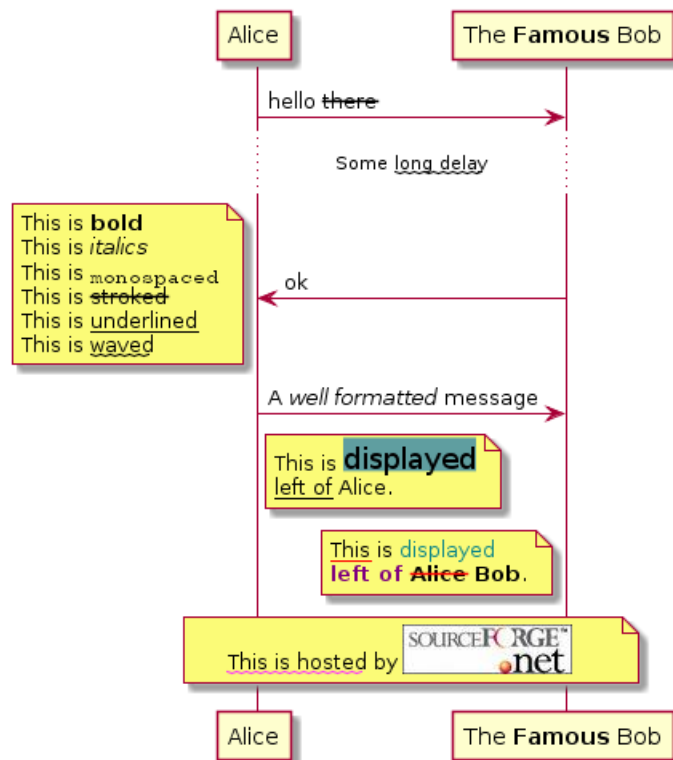
1.13. Creole и HTML

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
```



1.14. Разделитель

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

@startuml

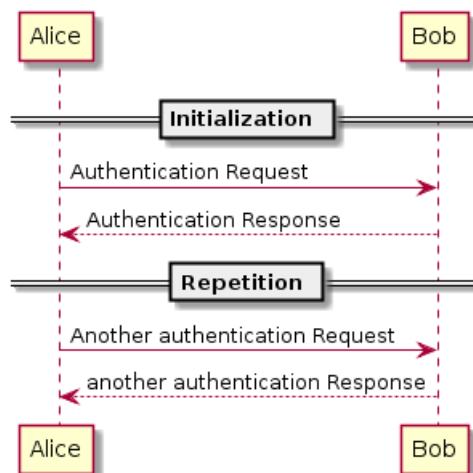
== Initialization ==

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

== Repetition ==

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

@enduml



1.15. Ссылки

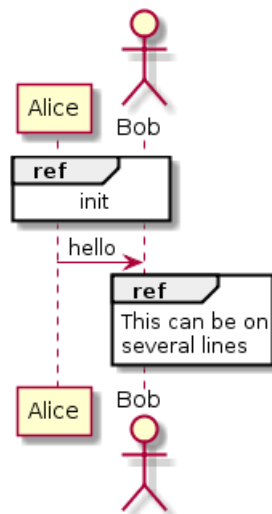
Вы можете использовать ссылки в диаграммах с помощью ключевого слова `ref over`.

```
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

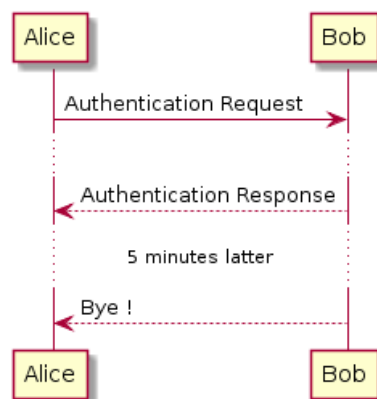
ref over Bob
This can be on
several lines
end ref
@enduml
```



1.16. 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 latter...
Bob -> Alice: Bye !
@enduml
```



1.17. 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
```

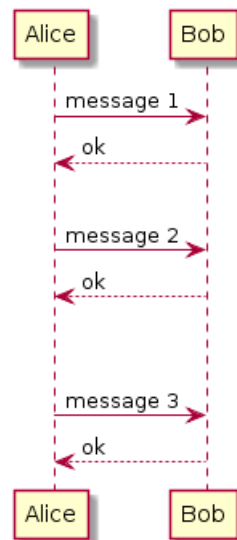
```
Bob --> Alice: ok
```

```
||45||
```

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

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

```
@enduml
```



1.18. 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
```

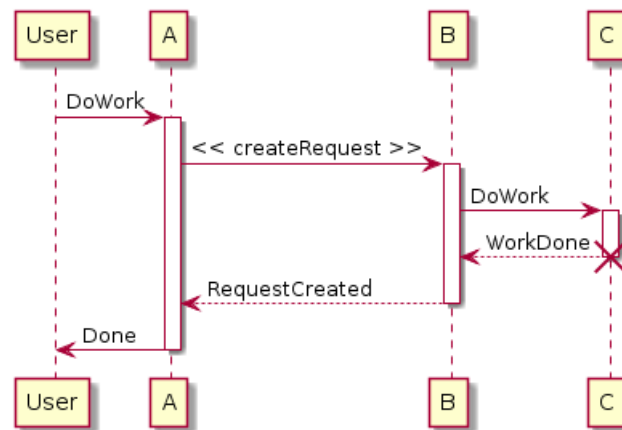
```
deactivate B
```

```
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

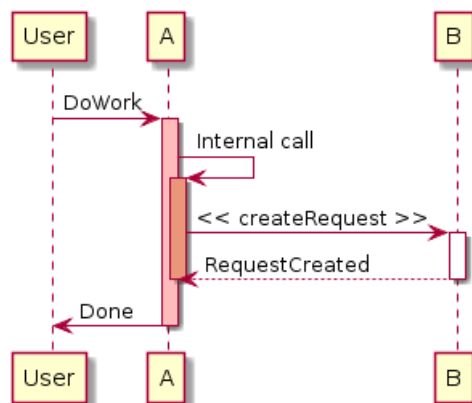
A -> A: Internal call
activate A #DarkSalmon

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

B -> A: RequestCreated
deactivate B
deactivate A

A -> User: Done
deactivate A

@enduml
```



1.19. 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

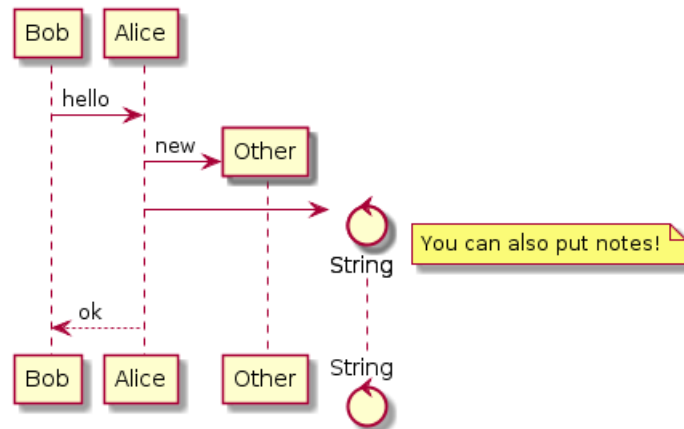
create control String
Alice -> String
```



note right : You can also put notes!

Alice → Bob : ok

@enduml



1.20. 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 >>

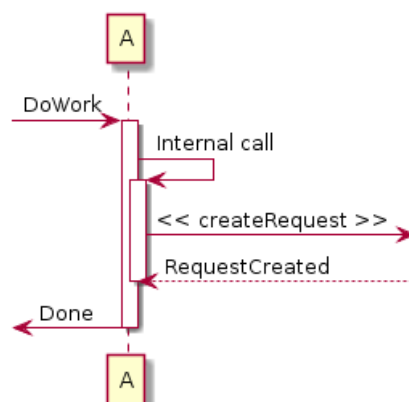
A<—] : RequestCreated

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

```



1.21. 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
```

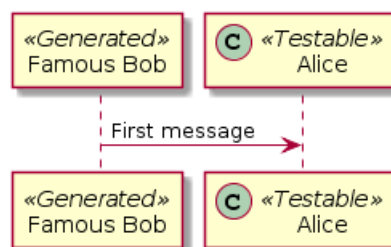
```

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
```

```

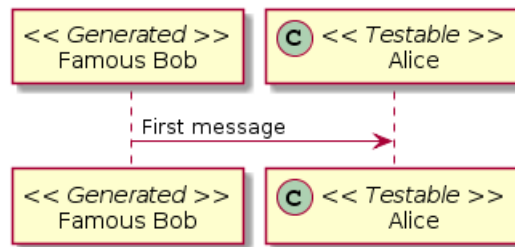
skinparam guillemet false
participant "Famous Bob" as Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

```

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

```
@enduml
```



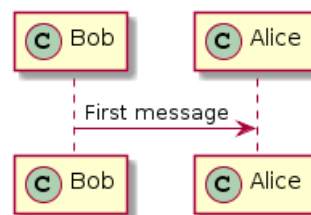


```
@startuml
```

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

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

```
@enduml
```



1.22. 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
```

Simple communication example



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

```
@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
```

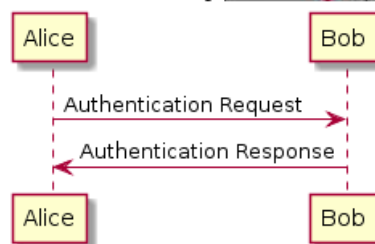
```
<u>Simple</u> communication example
on <i>several</i> lines and using <font color=red>html</font>
This is hosted by <img:sourceforge.jpg>
end title
```

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

```
@enduml
```

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

This is hosted by 

**1.23. Группировка участников**

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

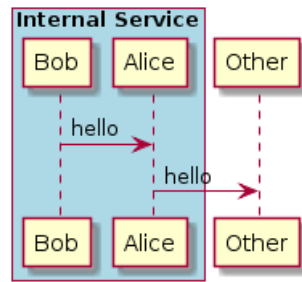
Вы можете задать опциональный заголовок и цвет фона, после команды `the box`.

```
@startuml
```

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

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

```
@enduml
```



1.24. Удаление футера

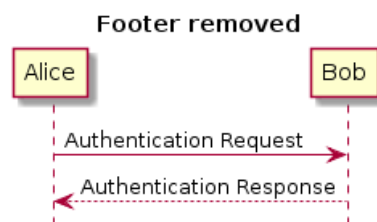
Вы можете использовать ключевое слово `hide footbox` для удаления футера из диаграммы.

```
@startuml
```

```
hide footbox
title Footer removed
```

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

```
@enduml
```



1.25. Skinparam

Вы можете использовать команду `skinparam` чтобы изменить цвет и шрифты рисования. Вы можете использовать эту команду:

- В определении диаграммы, как любые другие команды,
- В подключенном файле,
- В конфигурационном файле, заданном в командной строке или 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
```



```

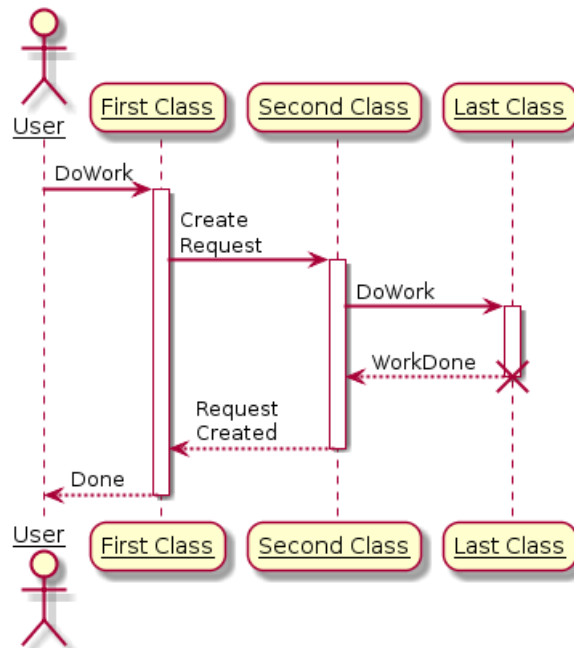
B → C: DoWork
activate C
C → B: WorkDone
destroy C

B → A: Request Created
deactivate B

A → User: Done
deactivate A

@enduml

```



```

@startuml
skinparam backgroundColor #EEEEBD
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

```

```

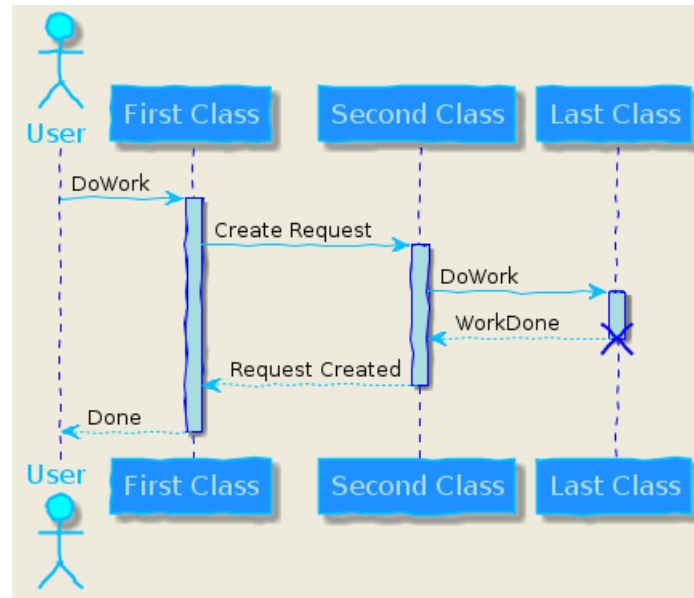
A -> User: Done
deactivate A

```

```

@enduml

```



1.26. Изменение отступов

Вы можете изменить некоторые настройки отступов

```

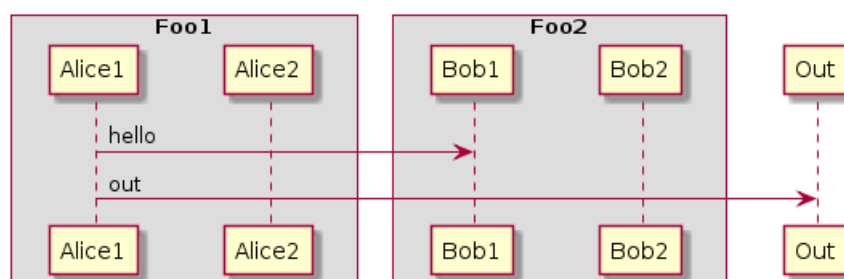
@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

```

```

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

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
```

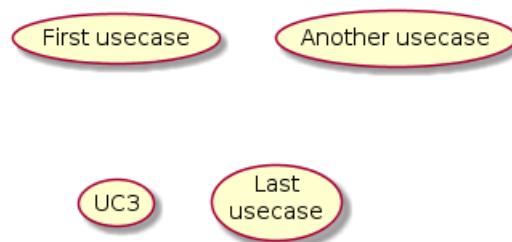
```
(First usecase)
```

```
(Another usecase) as (UC2)
```

```
usecase UC3
```

```
usecase (Last\nusecase) as UC4
```

```
@enduml
```



2.2. Актёры

Актёры 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 актёров не обязательны.

```
@startuml
```

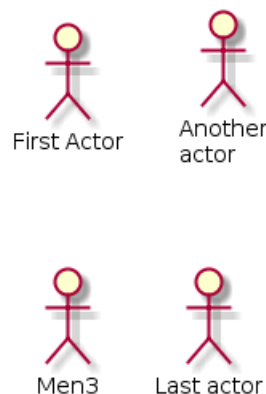
```
:First Actor:
```

```
: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.
```

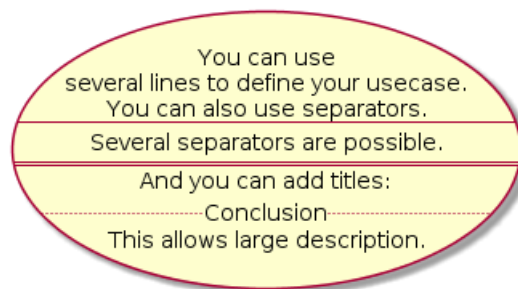
```
==
```

```
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.

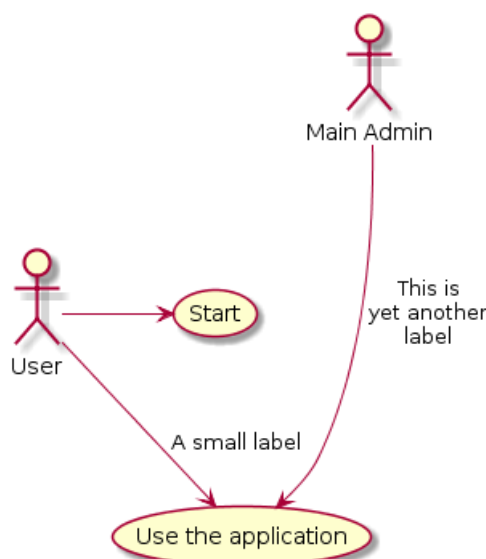
```
@startuml
```

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


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

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

```
@enduml
```



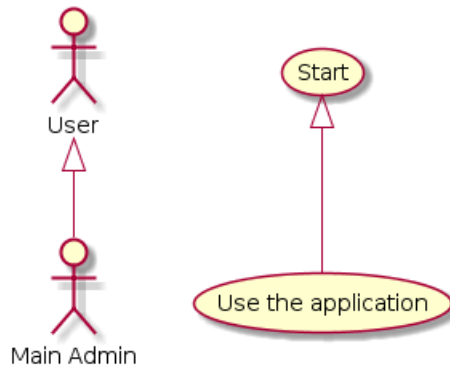
2.5. Расширение

Если один актёр/прецедент расширяют другой, вы можете использовать символ <|-- (который показывается как ).

```
@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)
```

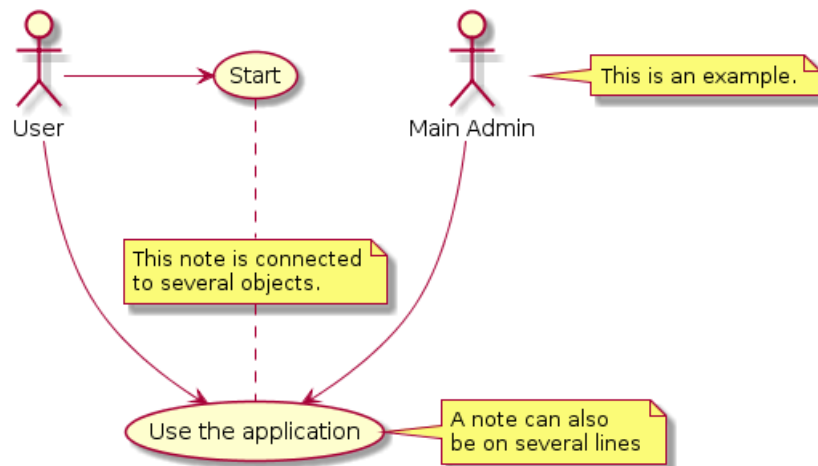
```
User -> (Start)
User --> (Use)
```

```
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 >>
  
```

```

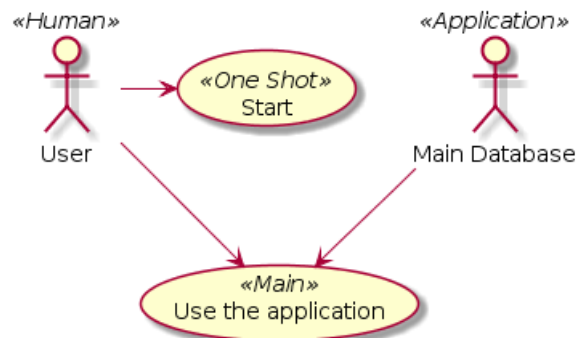
User -> (Start)
User --> (Use)
  
```

```

MySql --> (Use)
  
```

```

@enduml
  
```

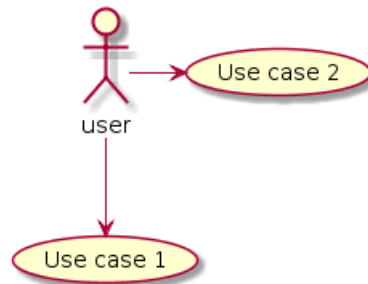


2.8. Смена направления стрелок

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

```

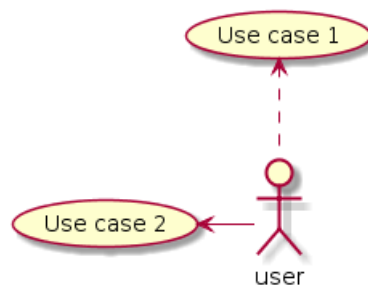
@startuml
:user: --> (Use case 1)
:user: --> (Use case 2)
@enduml
  
```



Вы так же можете изменить направление с помощью переворачивания связи:

```

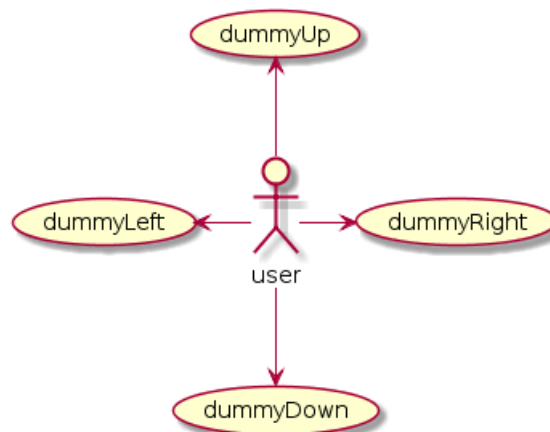
@startuml
(Use case 1) <.. :user:
(Use case 2) <- :user:
@enduml
  
```



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
  
```



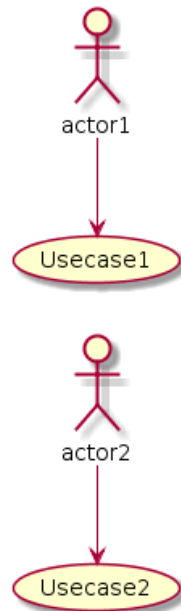
Вы можете записать короче, используя только первый символ названия направления (например, -d- вместо -down-) или первые два символа (-do-).

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.

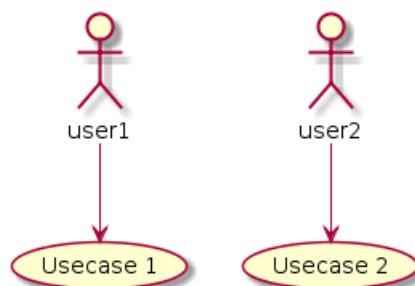
```
@startuml
:actor1: —> (Usecase1)
newpage
:actor2: —> (Usecase2)
@enduml
```



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
```



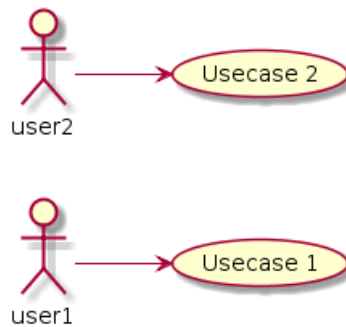
Вы можете изменить направление на слева направо используя команду `left to right direction`. Часто результат с таким направлением выглядит лучше.

```
@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

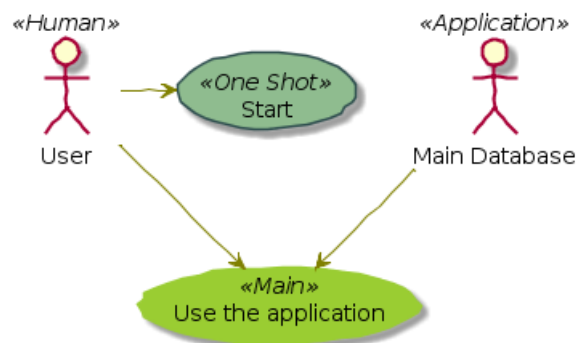
ActorBackgroundColor<< Human >> Gold
}

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

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

MySql --> (Use)

@enduml
  
```

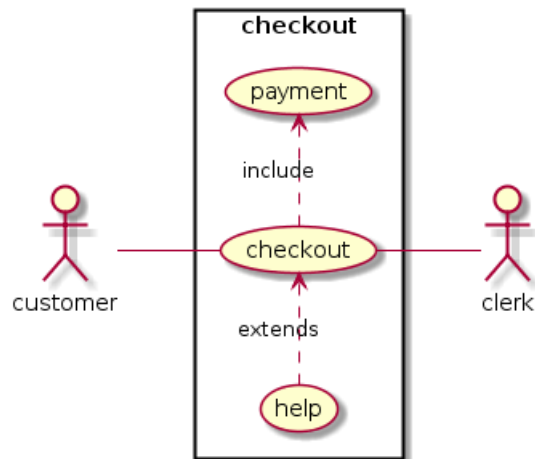


2.12. Полноценный пример

```

@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

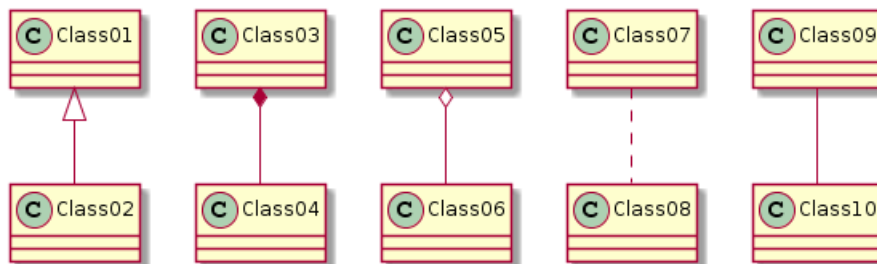
Отношения между классами определяются с помощью следующих символов:

расширение	< --	
композиция	*--	
агрегирование	o--	

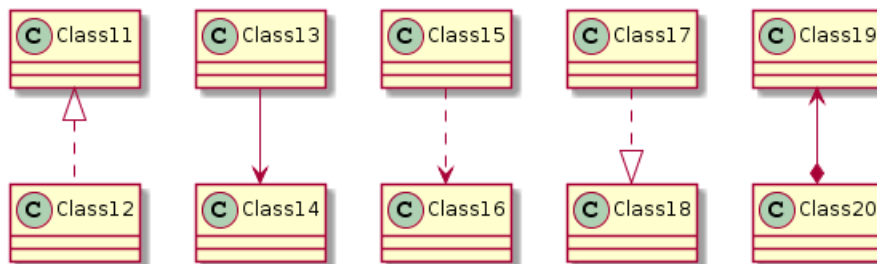
Можно заменить "-" на "..", чтобы создать пунктирную линию.

Зная эти правила можно нарисовать следующие изображения:

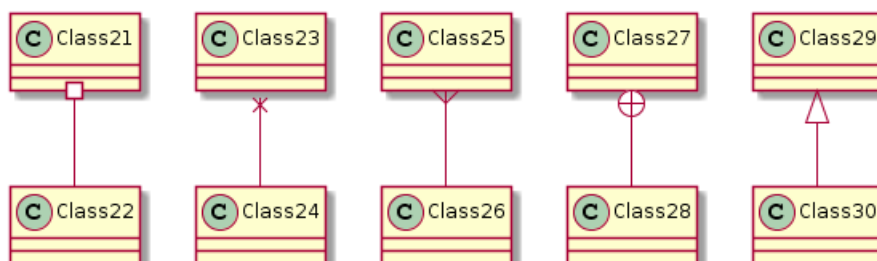
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



```
@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
```

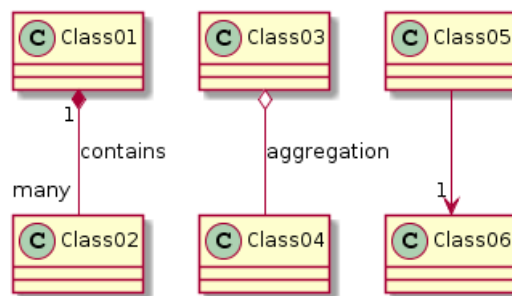


3.2. Метки на отношениях

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

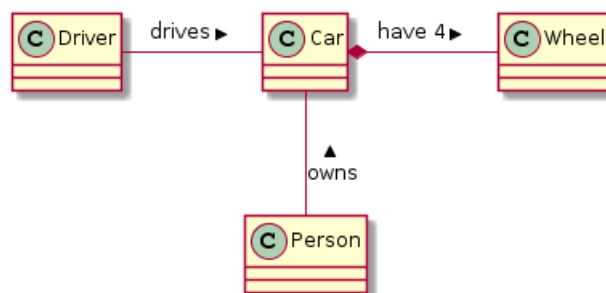
Для указания количества элементов на каждой стороне отношения можно использовать двойные кавычки "".

```
@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

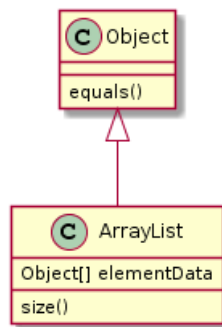
Для объявления полей и методов вы можете использовать символ ":", после которого указывается имя поля или метода.

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()
}

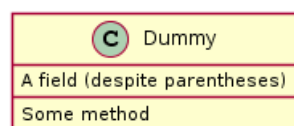
class Flight {
flightNumber : Integer
departureTime : Date
}
@enduml
```



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

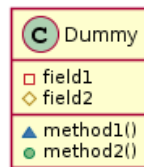
Определяя методы и поля, вы можете использовать символы указания видимости, приведённые в таблице ниже:

-	□	private
#	◇	protected
~	△	package private
+	○	public

@startuml

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

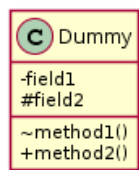
@enduml



Убрать значки можно командой `skinparam classAttributeIconSize 0`:

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

@enduml

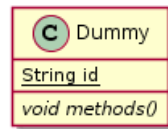


3.5. Абстрактные и статические

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 также можете использовать `{classifier}` как замену для `{static}`.

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



3.6. Расширенное тело класса

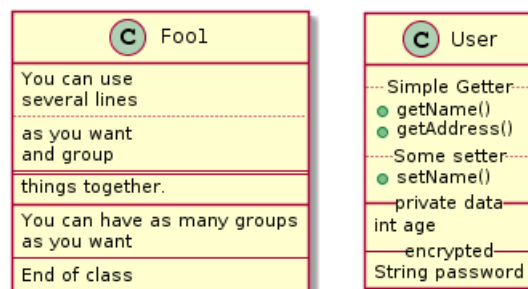
По умолчанию, методы и поля автоматически группируются PlantUML. Вы можете использовать разделители, чтобы определить собственный порядок полей и методов. Можно использовать следующие разделители: -- .. == __.

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 привязи, 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`

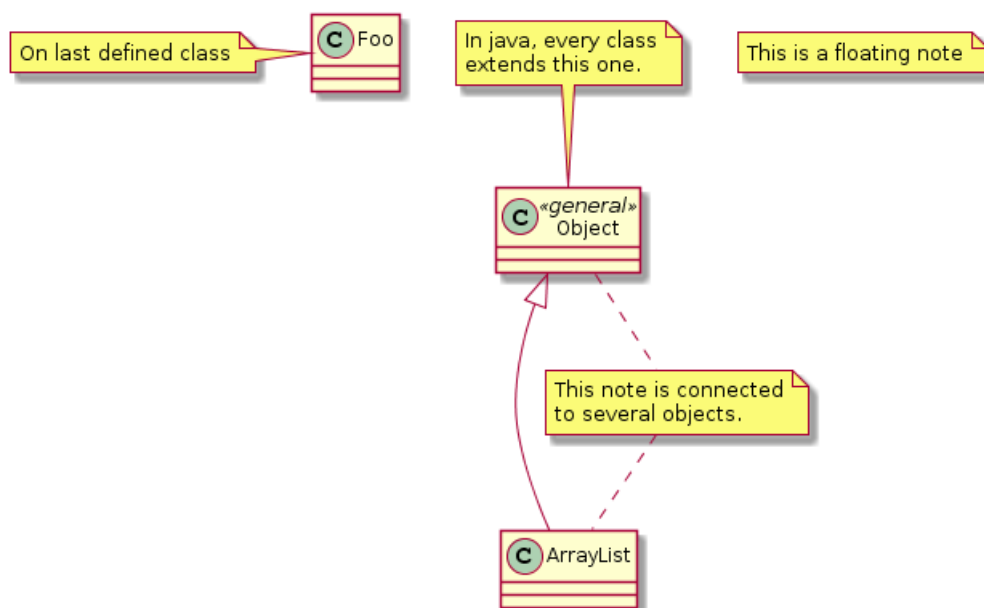
`Object .. N2`

`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`.

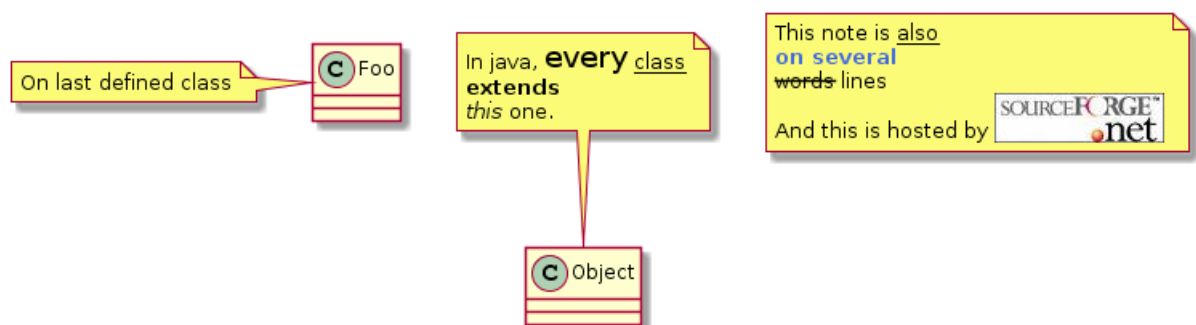
```
@startuml
```

```
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><del>color:royalBlue</del></b>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`.

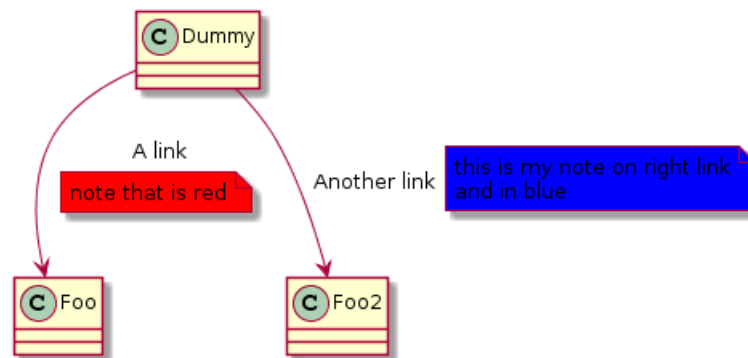
Вы также можете использовать `note left on link`, `note right on link`, `note top on link`, `note bottom on link` если вы хотите изменить относительную позицию заметки с надписью.

```
@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
```



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
```

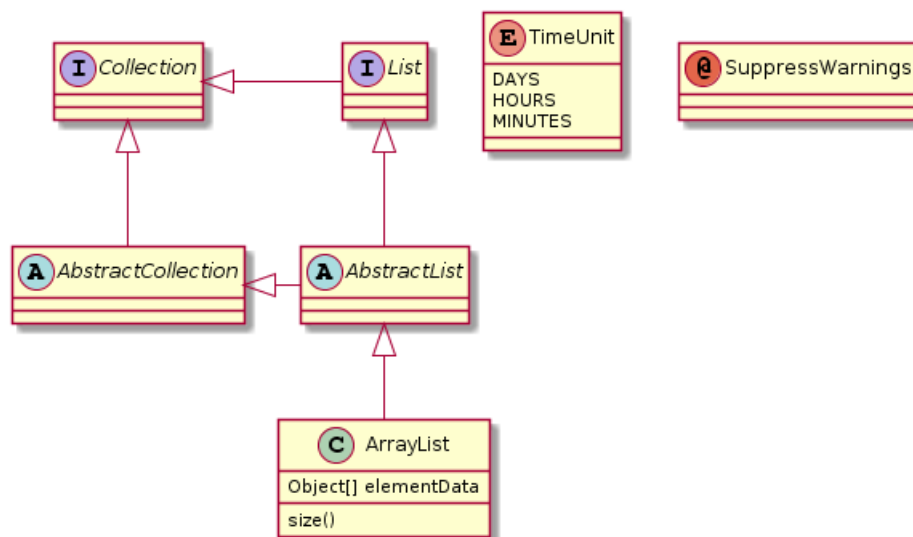
```
Collection <|-- List
AbstractCollection <|-- AbstractList
AbstractList <|-- ArrayList
```

```
class ArrayList {
Object[] elementData
size()
}
```

```
enum TimeUnit {
DAYS
HOURS
MINUTES
}
```

```
annotation SuppressWarnings
```

@enduml



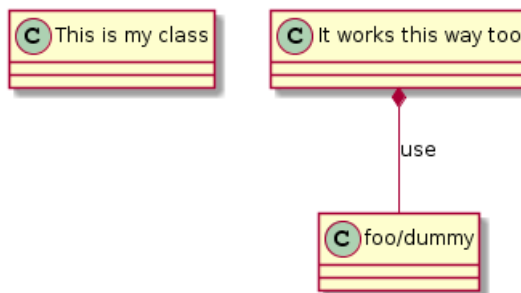
3.11. Использование не буквенных символов

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

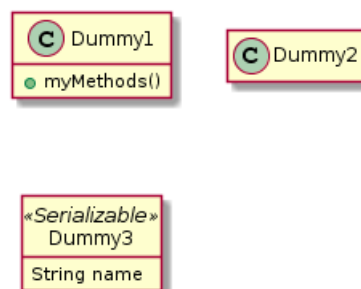
```
class Dummy1 {
+myMethods()
}
```

```
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.

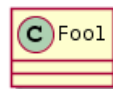
Это может быть полезно, если вы определяете большой !подключенный файл, и если вы хотите скрыть некоторые классы после включения.

```
@startuml
class Foo1
class Foo2

Foo2 *-- Foo1

hide Foo2

@enduml
```

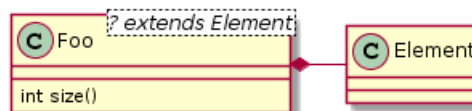


3.14. Использование дженериков

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`.

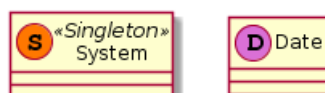
3.15. Определение метки

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).

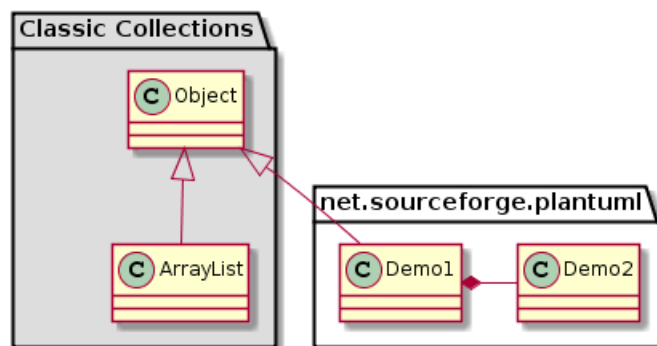
Обратите внимание, что определения пакета могут быть вложенными.

```
@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:

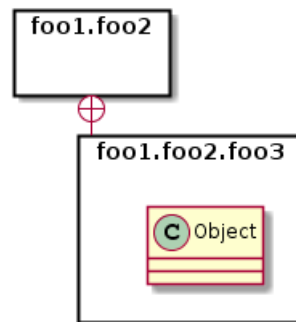
```
@startuml
skinparam packageStyle rectangle

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 {
```



```

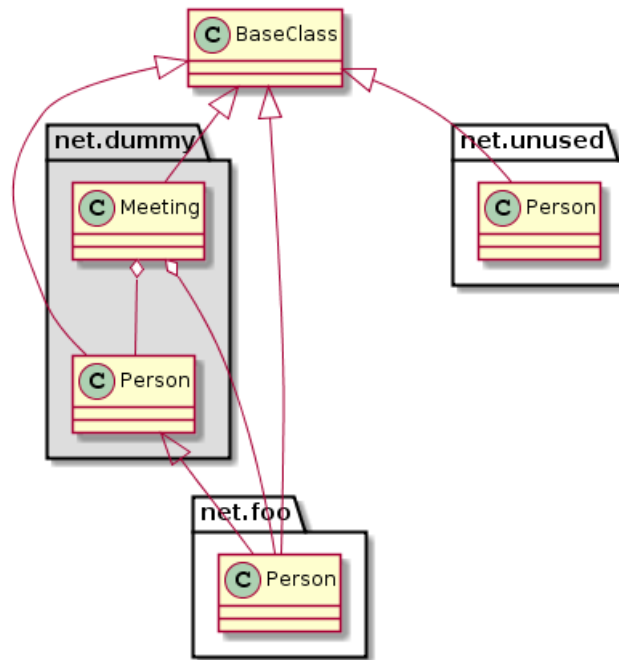
net.dummy.Person <|-- Person
.BaseClass <|-- Person

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 ???.

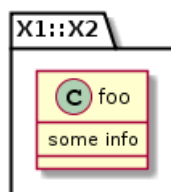
```

@startuml

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

@enduml

```



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

```

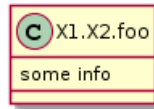
@startuml

set namespaceSeparator none

```

```
class X1.X2.foo {
some info
}

@enduml
```

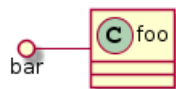


3.20. Lollipop интерфейс

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

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

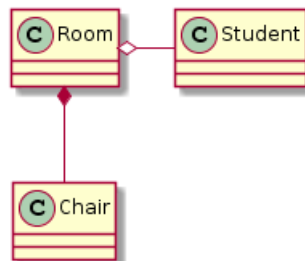
```
@startuml
class foo
bar ()- foo
@enduml
```



3.21. Изменение направления стрелок

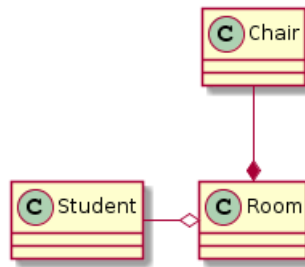
По умолчанию, связи между классами имеют два тире -- и вертикально ориентированны. Возможно создать горизонтальную связь, используя одно тире (or dot) вот так:

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



Вы можете изменить направление перевернув связь:

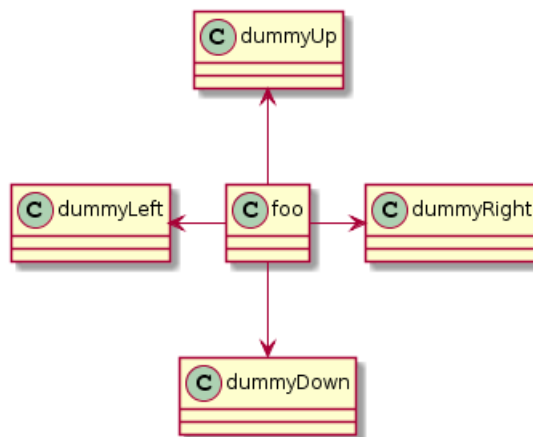
```
@startuml
Student -o Room
Chair --* Room
@enduml
```



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 (на-пример, -d- вместо -down-) или две первые буквы (-do-).

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

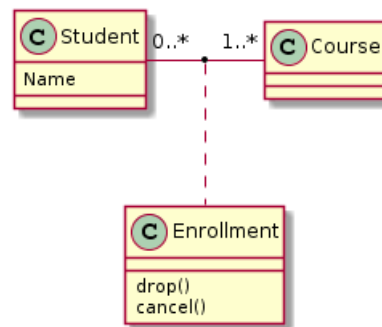
3.22. Ассоциация классов

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
  
```

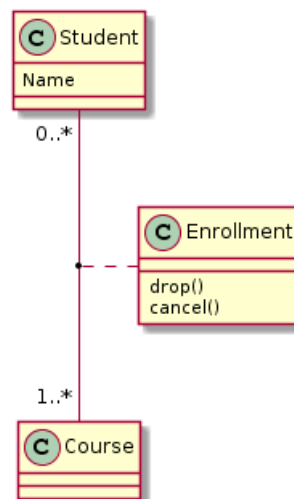



Вы можете задать это в другом направлении:

```

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

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



3.23. Skinparam

Вы можете использовать команду `skinparam` чтобы изменить цвета и шрифты рисунка.

Вы можете использовать команду :

- В определении диаграммы, как любая другая команда,
- В подключаемом файле,
- В конфигурационном файле, указываемом в командной строке или в задаче ANT:

```

@startuml
skinparam class {
BackgroundColor PaleGreen
ArrowColor SeaGreen
}
@enduml
  
```



```

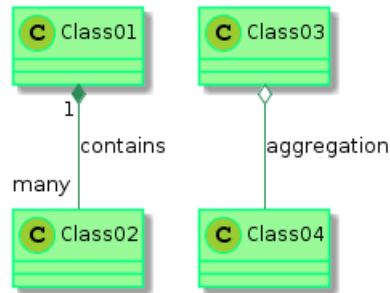
BorderColor SpringGreen
}
skinparam stereotypeCBackgroundColor YellowGreen

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

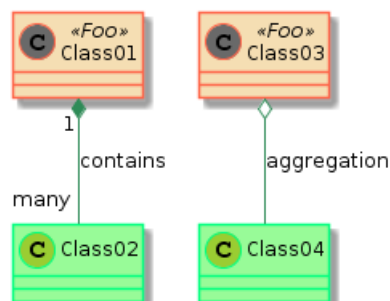
skinparam class {
  BackgroundColor PaleGreen
  ArrowColor SeaGreen
  BorderColor SpringGreen
  BackgroundColor<<Foo>> Wheat
  BorderColor<<Foo>> Tomato
}
skinparam stereotypeCBackgroundColor YellowGreen
skinparam stereotypeCBackgroundColor<< Foo >> DimGray

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

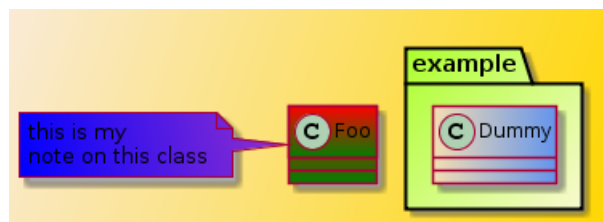
Например так :

```
@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

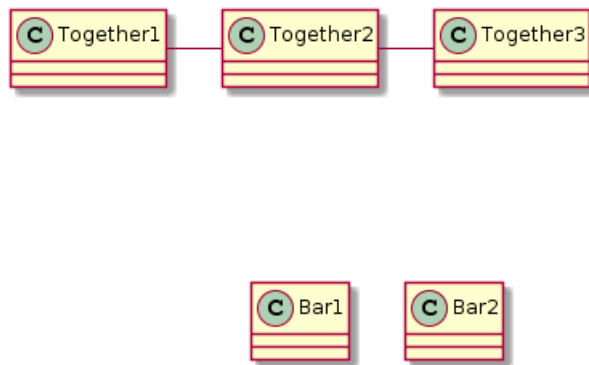
Sometimes, the default layout is not perfect...

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

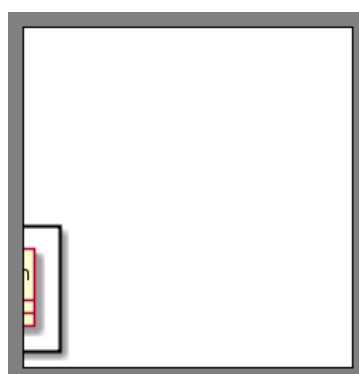
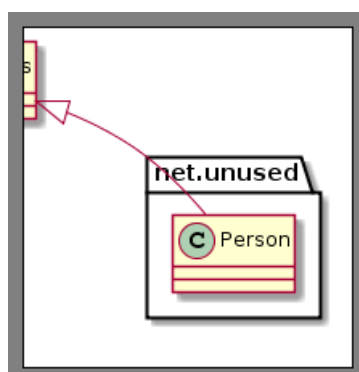
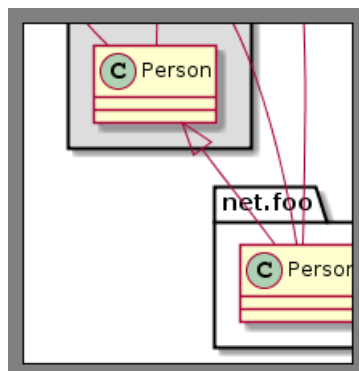
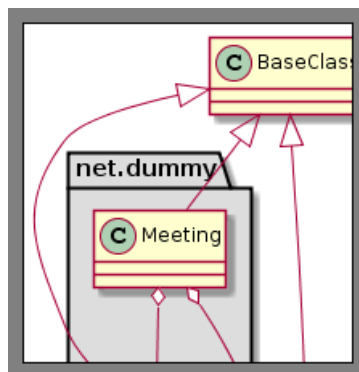
  .BaseClass <|-- Meeting
}

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.

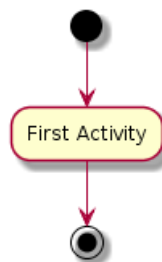
В некоторых случаях, вы можете использовать (*top) чтобы указать что начальная точка должна быть в верху диаграммы.

Используйте --> для стрелок.

```
@startuml
```

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

```
@enduml
```



4.2. Метка на стрелках

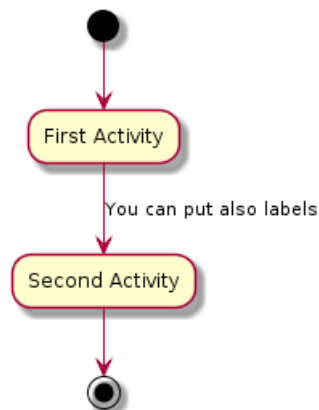
По умолчанию, стрелка начинается с последней использованной активности.

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

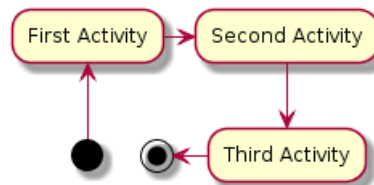
Вы можете использовать `->` для горизонтальных стрелок. Возможно задать направление стрелки используя следующий синтаксис:

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

```
@startuml
```

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

```
@enduml
```



4.4. Ветвления

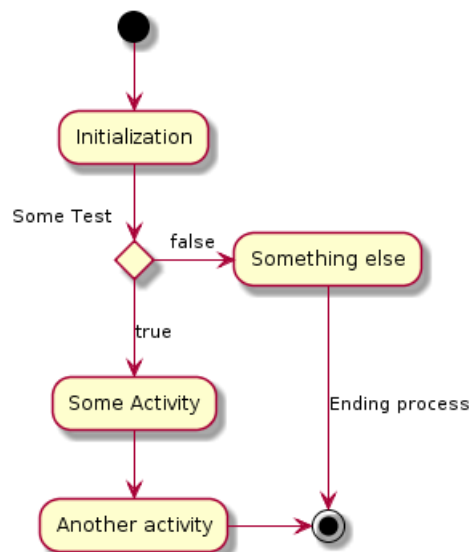
Вы можете использовать ключевые слова `if/then/else` чтобы определять ветки.

```
@startuml
```

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

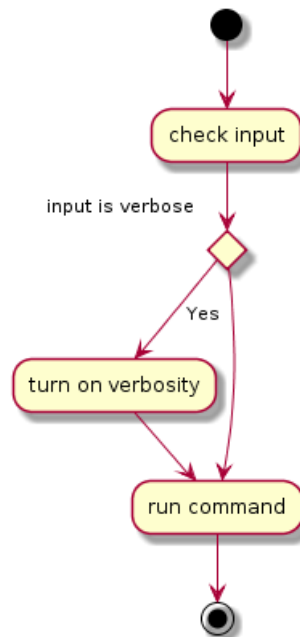
```
if "Some Test" then
-->[true] "Some Activity"
--> "Another activity"
-right-> (*)
else
->[false] "Something else"
-->[Ending process] (*)
endif
```

```
@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

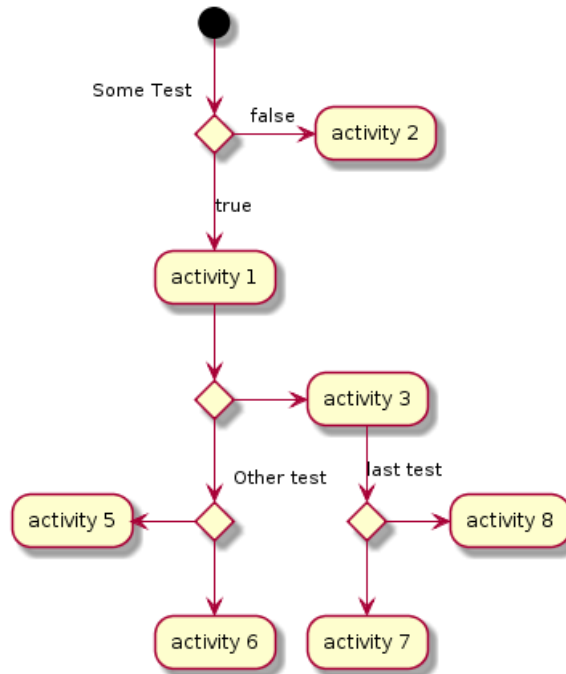
a3 --> if "last test" then
```



```

—> "activity 7"
else
—> "activity 8"
endif
@enduml

```



4.6. Синхронизация

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

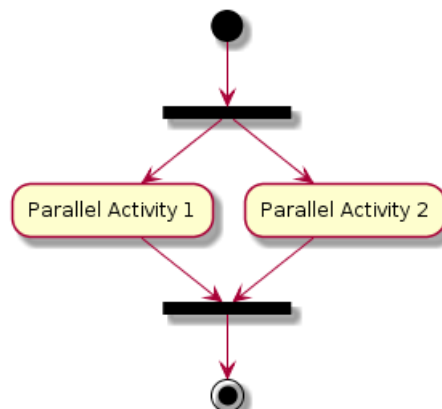
```

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

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

--> (*)
@enduml

```



4.7. Длинное описание активности

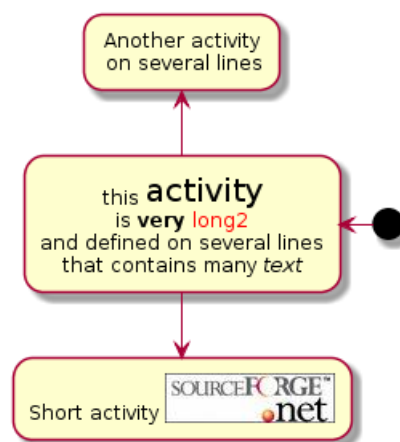
Когда вы задаёте активность, вы можете разделить её описание на несколько линий. Вы также можете добавить \n в описание.

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
```



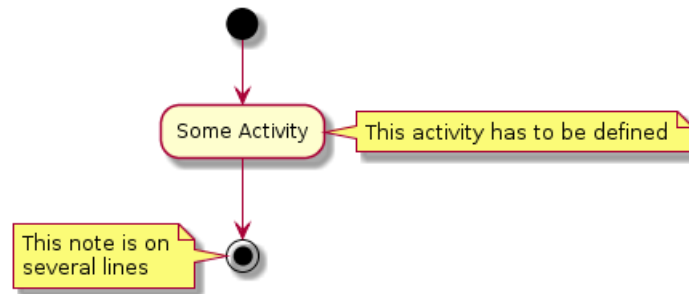
4.8. Заметки

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
```

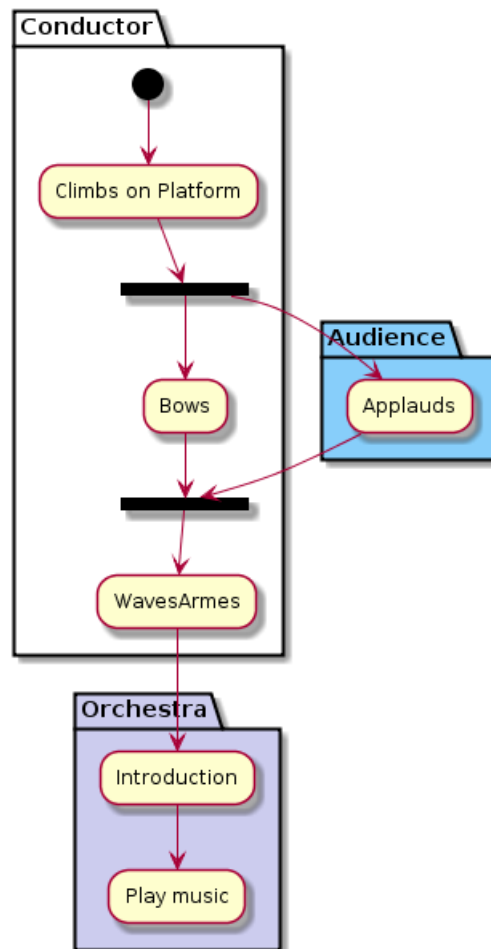
```
partition Conductor {
  (*) --> "Climbs on Platform"
  --> == S1 ==
  --> Bows
}
```

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

```

```

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

```

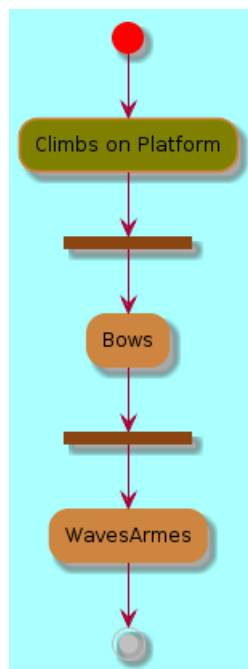


```

—> == 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

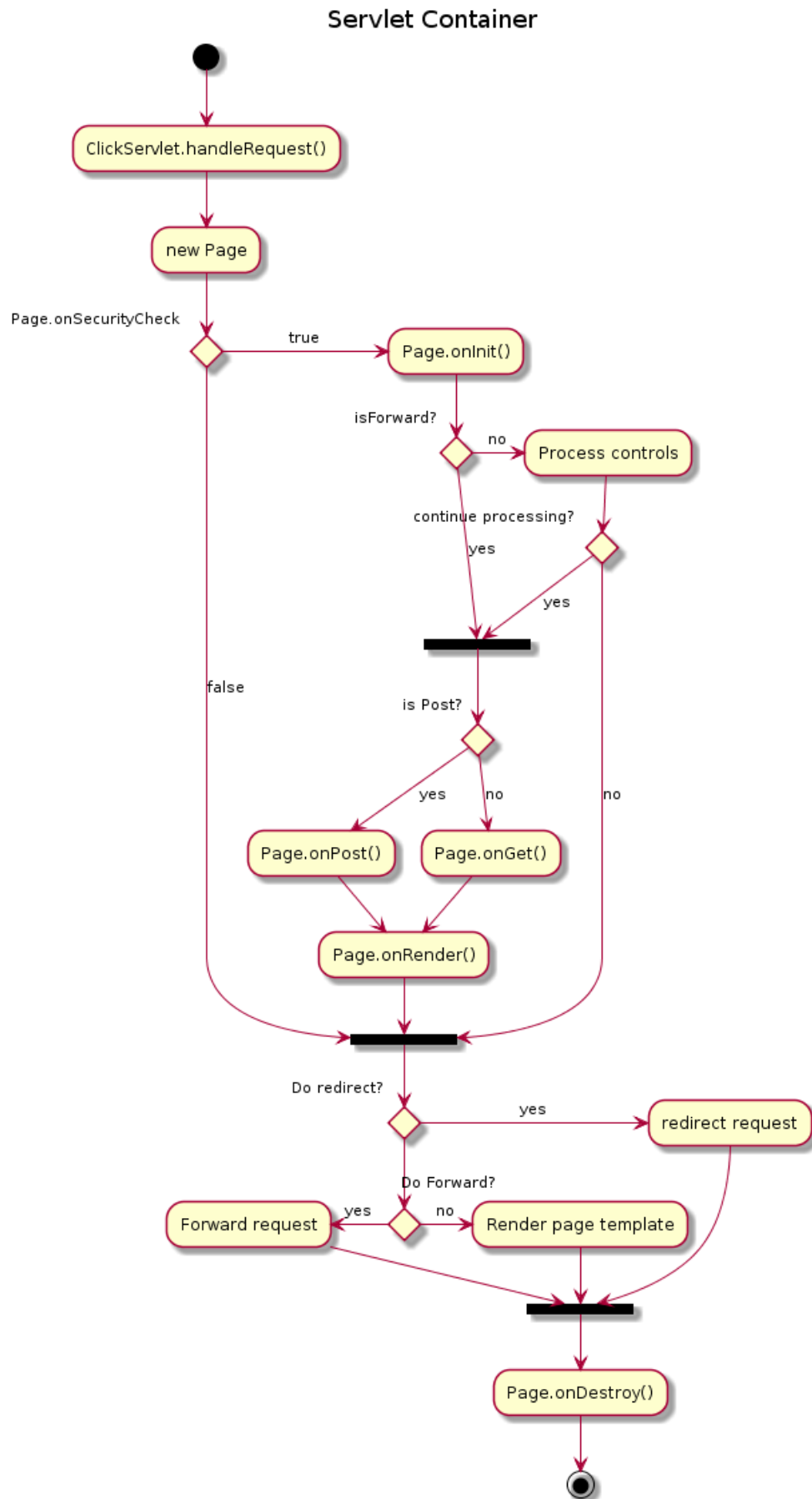
else
-->[false] ==REDIRECT_CHECK==
endif

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 мигрировании на новый синтаксис.

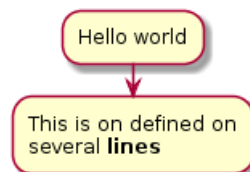
5.1. Простая активность

Описания активностей начинаются с : и заканчиваются с ;.

Форматировать текст возможно используя синтаксис creole.

Они косвенно связаны в порядке их определения.

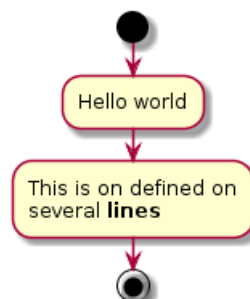
```
@startuml
:Hello world;
:This is on defined on
several lines;
@enduml
```



5.2. Старт/Стоп

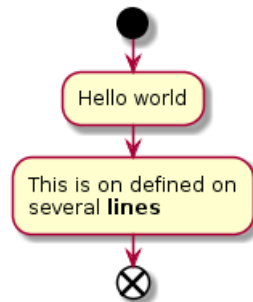
Вы можете использовать ключевые слова start и stop чтобы обозначать начало и конец диаграммы.

```
@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
```



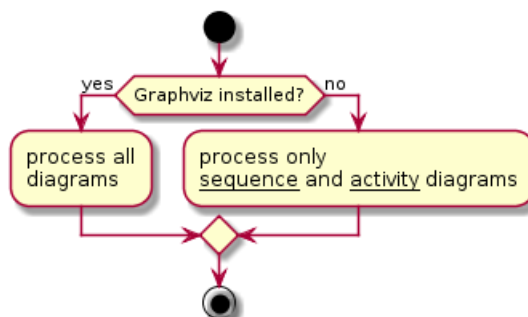
5.3. Условия

Вы можете использовать ключевые слова `if`, `then` и `else` чтобы добавить проверяющие если на вашу диаграмму. Описания могут быть добавлены используя круглые скобки.

```
@startuml
start

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

stop
@enduml
```



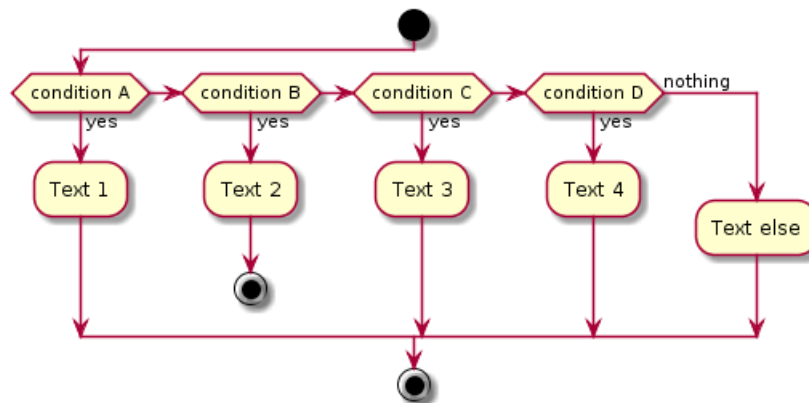
Вы можете использовать ключевые слова `elseif` чтобы создать несколько проверок:

```
@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

```



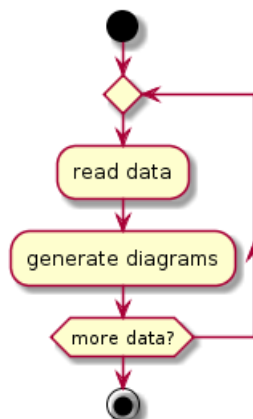
5.4. Повторяющийся цикл

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

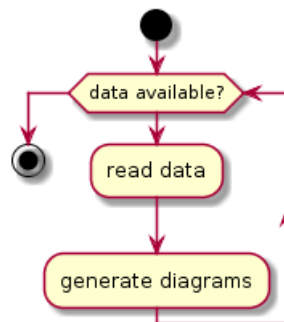
```



5.5. Цикл while

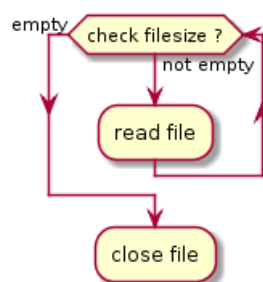
Вы можете использовать ключевые слова `while` и `end while` чтобы создавать повторяющиеся циклы.

```
@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. Параллельные процессы

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

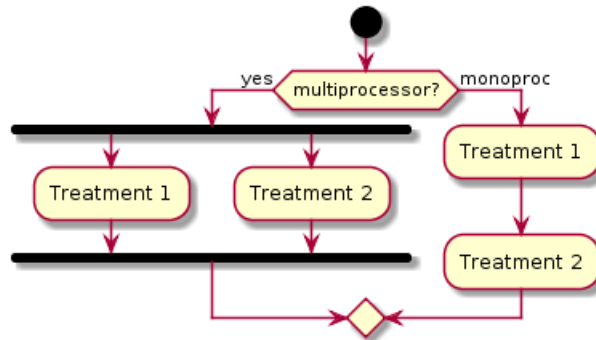
```
@startuml
start
if (multiprocessor?) then (yes)
fork
:Treatment 1;
fork again
end fork
stop
@enduml
```

```

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`.

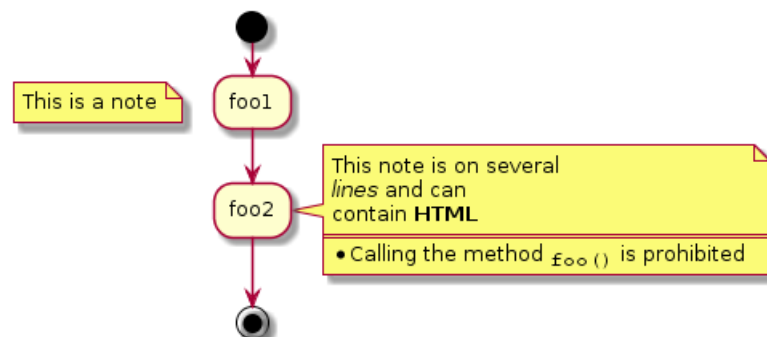
```
@startuml
```

```

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

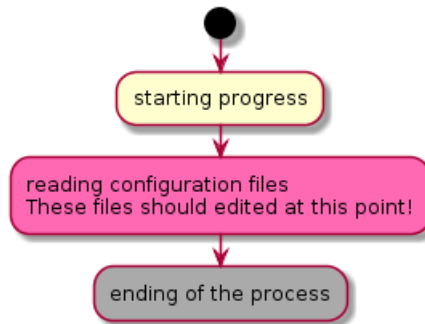
Вы можете задать цвет некоторым активностям.



```
@startuml
```

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

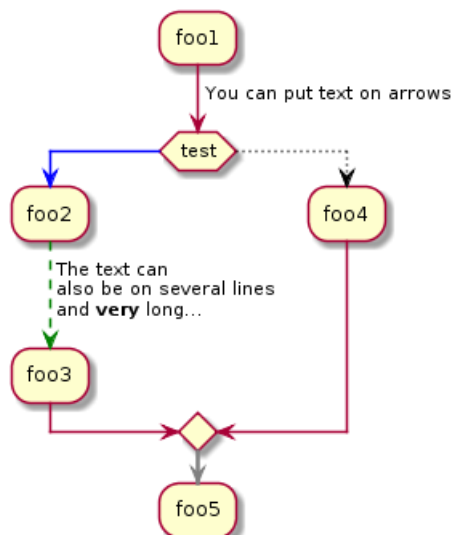
```
@enduml
```



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 спрятанные.

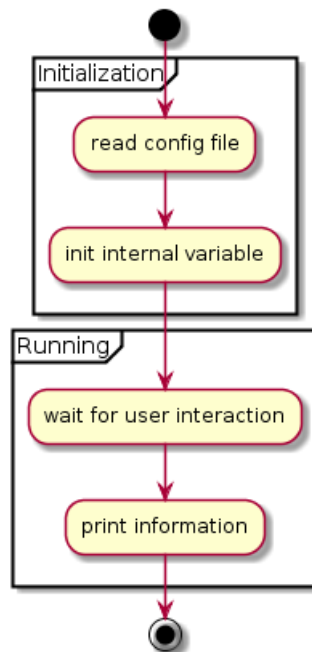
```
@startuml
:foo1;
-> You can put text on arrows;
if (test) then
-[#blue]->
:foo2;
-[#green,dashed]-> The text can
also be on several lines
and very long ...;
:foo3;
else
-[#black,dotted]->
:foo4;
endif
-[#gray,bold]->
:foo5;
@enduml
```



5.10. 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
```

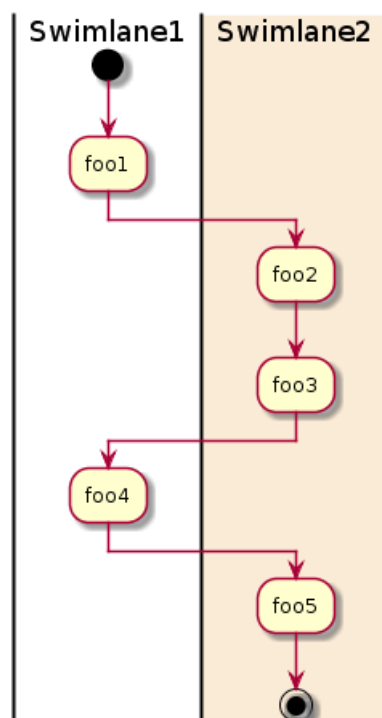


5.11. Дорожки

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
```

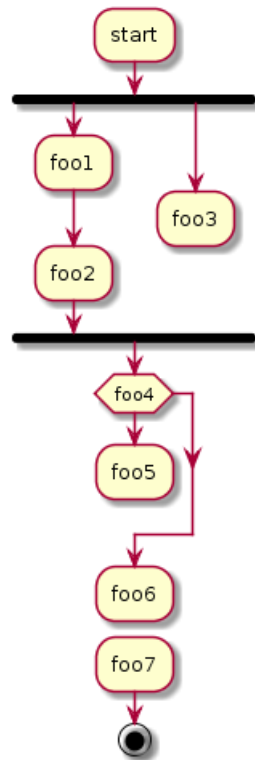


5.12. Отсоединение

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
  
```



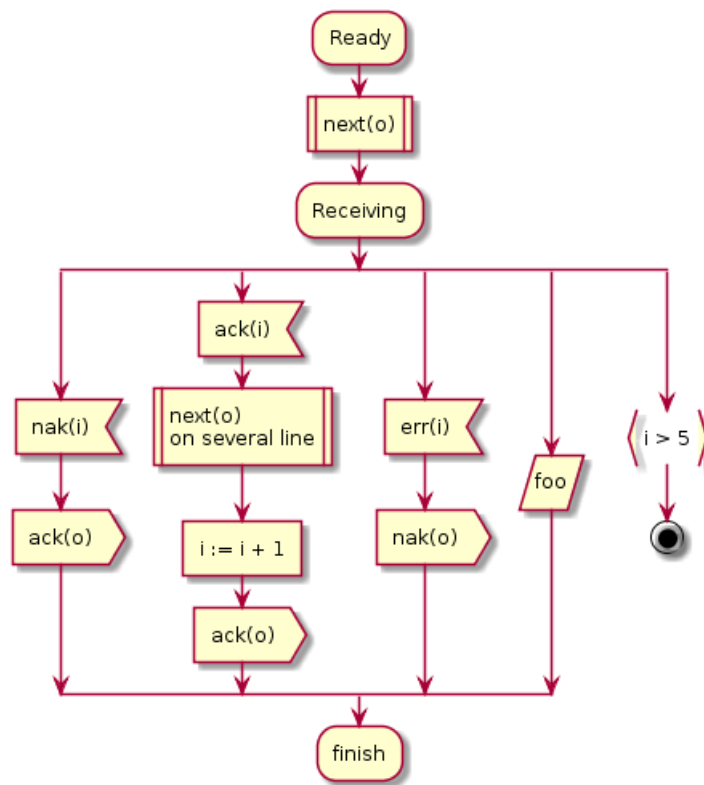
5.13. SDL

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

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

```

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

5.14. 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

```

```

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

```

```

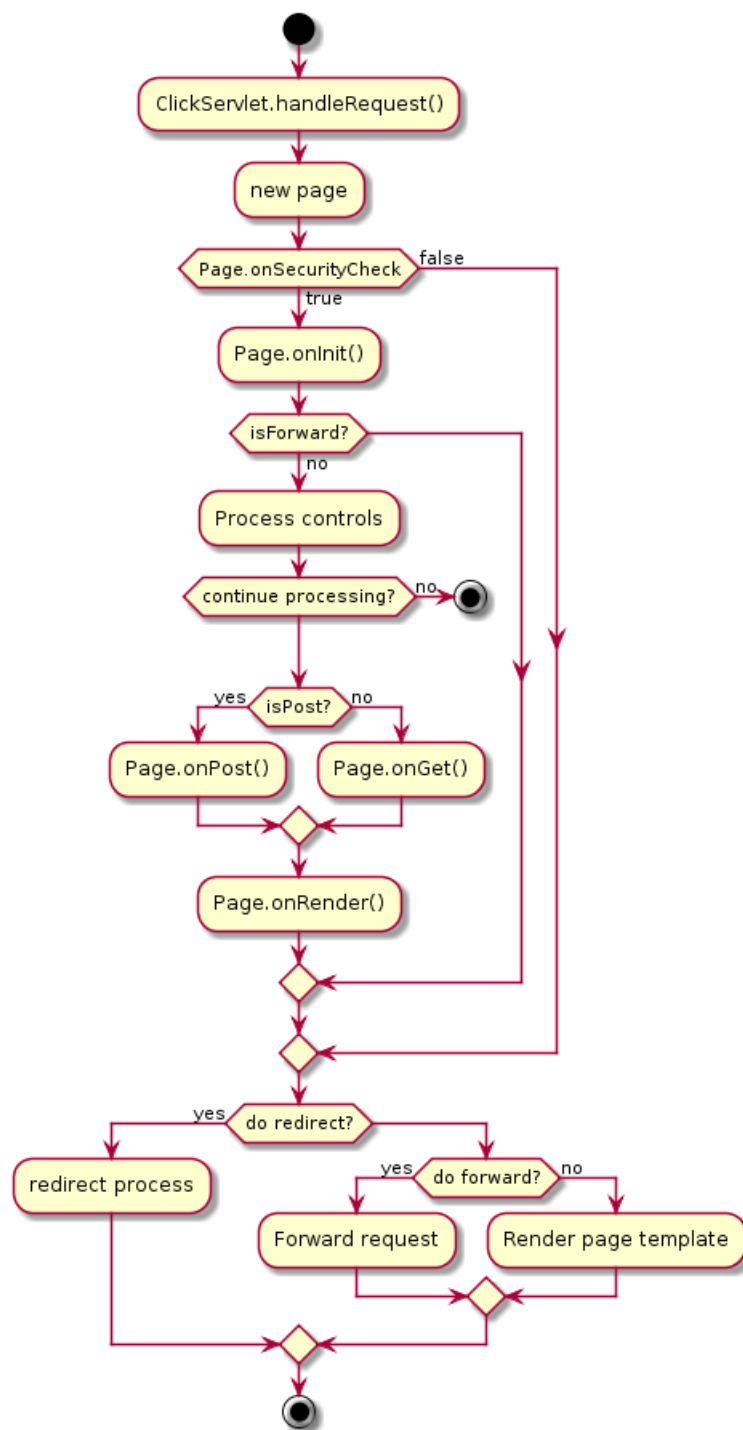
if (do redirect?) then (yes)
: redirect process;
else
if (do forward?) then (yes)
: Forward request;
else (no)
: Render page template;
endif
endif

```

stop

@enduml





6. Диаграмма компонентов

6.1. Компоненты

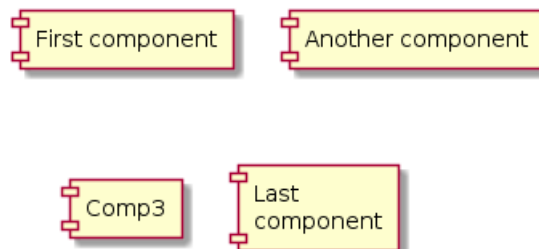
Обозначения компонентов должны быть заключены в квадратные скобки.

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).

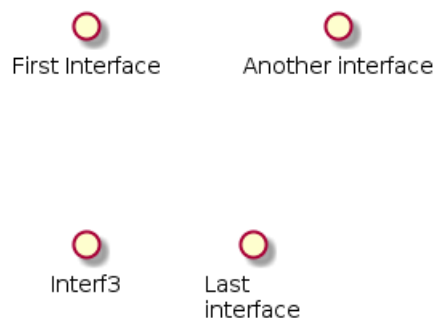
Также возможно использование ключевого слова `interface` для объявления интерфейса. Вы можете объявить алиас с помощью ключевого слова `as`. Этот алиас может быть использован позднее, когда будут задаваться связи.

Далее мы увидим, что задание интерфейсов опционально.

```
@startuml
```

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

```
@enduml
```



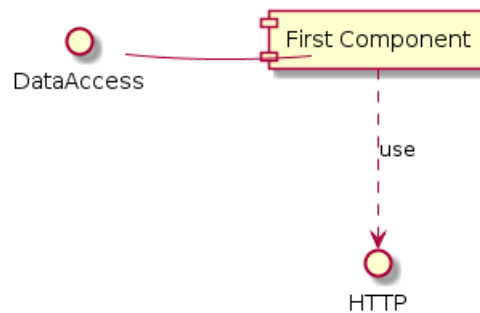
6.3. Простой пример

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 прикреплённой, используя ключевое слово `note`, а затем прикреплена к другим объектам, используя символ `..`.

```
@startuml
```

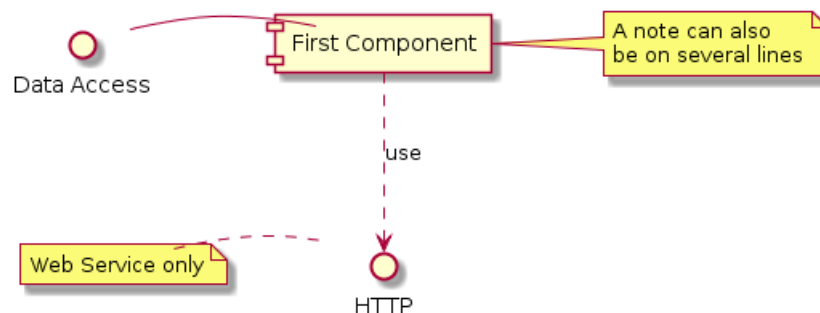
```
interface "Data Access" as DA
```

```
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]  
}
```

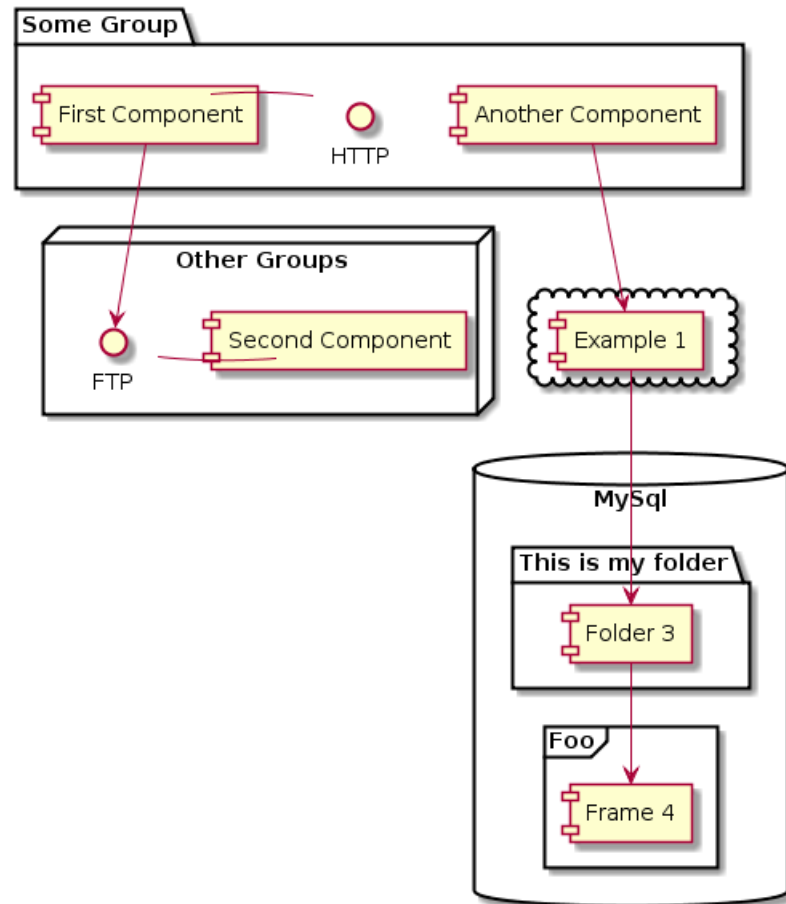
```
node "Other Groups" {  
  FTP -- [Second Component]  
  [First Component] --> FTP  
}
```

```
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]
```

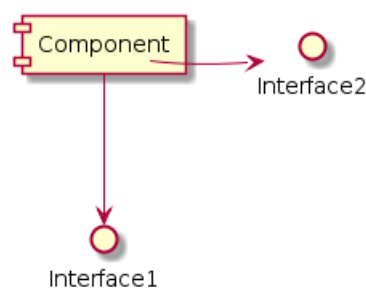
```
@enduml
```



6.6. Изменение направления стрелок

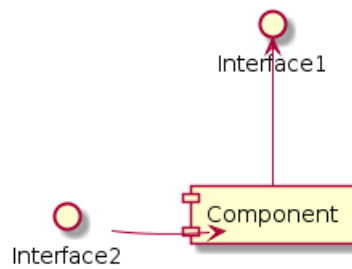
По умолчанию, связи между классами имеют два типа -- и ориентированы вертикально. Можно создавать горизонтальные связи с помощью одного типа (или точки), вот так:

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



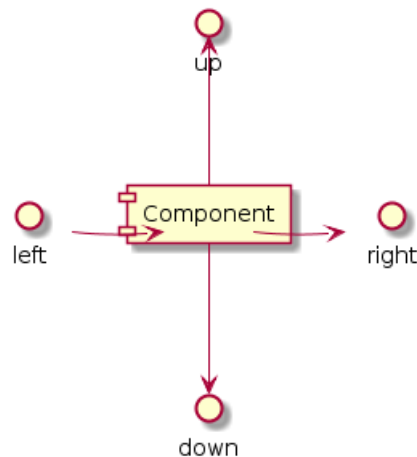
Вы также можете изменять направления, перевернув связь:

```
@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

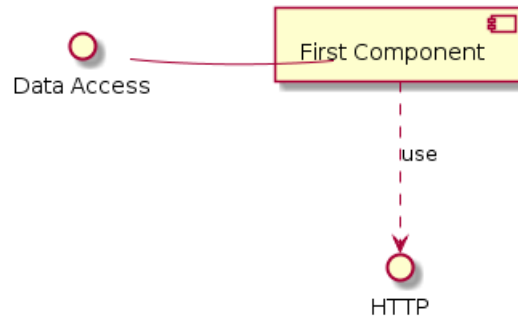
Команда `skinparam componentStyle uml2` используется, чтобы переключиться на нотацию UML2.

```
@startuml
skinparam componentStyle uml2

interface "Data Access" as DA

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

@enduml
```

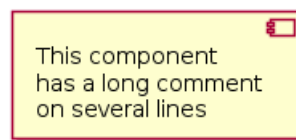


6.8. Длинное описание

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
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FF0000000000FF
FF0000000000FF
FF0000000000FF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FFFFFFFFFFFFFFFF
}
  
```

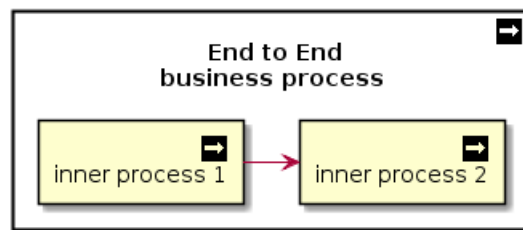


```

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

Чтобы изменить цвета и шрифты рисования, используйте команду `skinparam`.

Вы можете использовать эту команду :

- В определении диаграммы, как и любые другие команды,
- В подключаемом файле,
- В конфигурационном файле (указывается в командной строке или как параметр для задачи 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
```

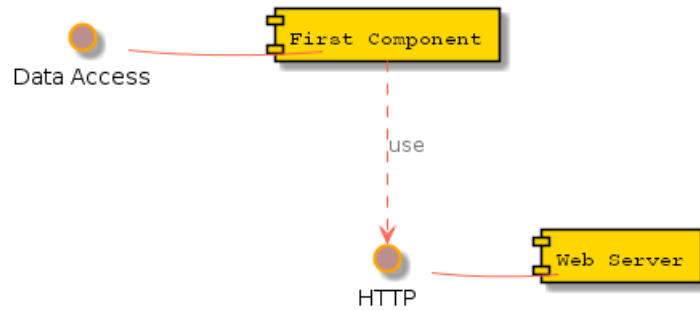
```

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

```

```
@enduml
```





```

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

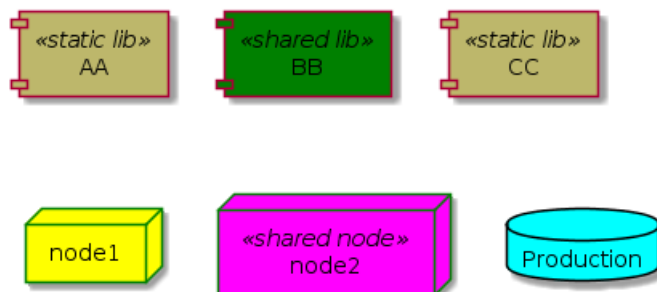
node node1
node node2 <<shared node>>
database Production

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

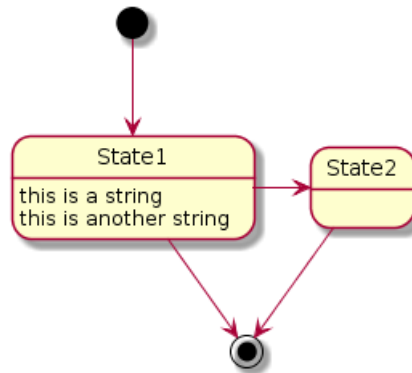
7.1. Простое состояние

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. Составное состояние

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

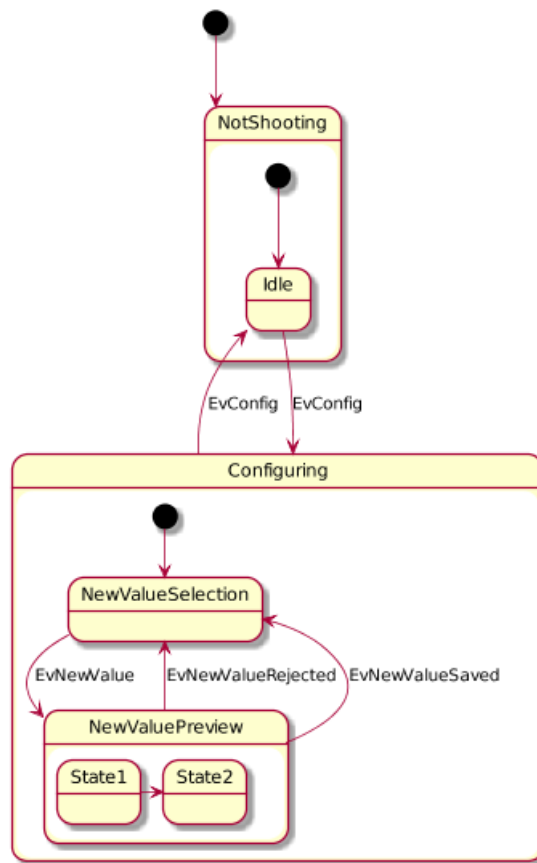
```
@startuml
scale 350 width
[*] --> NotShooting

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
```



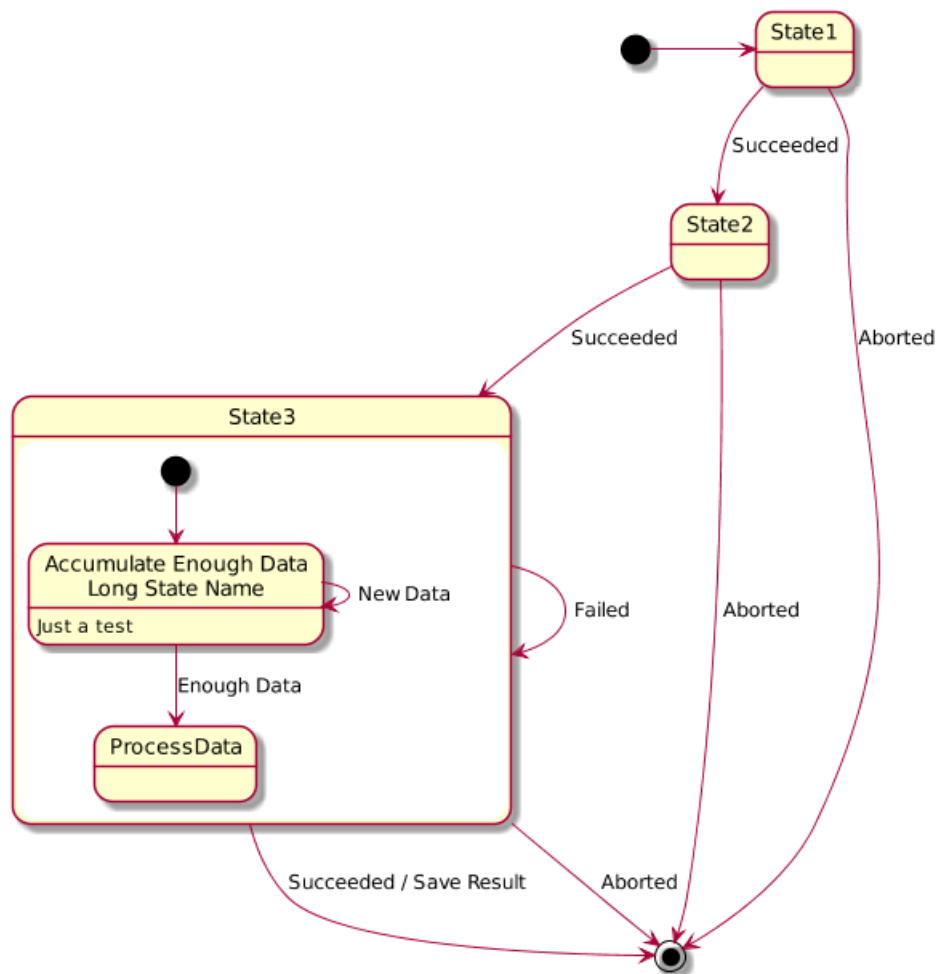
7.3. Длинные имена

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
```



7.4. Параллельные состояния

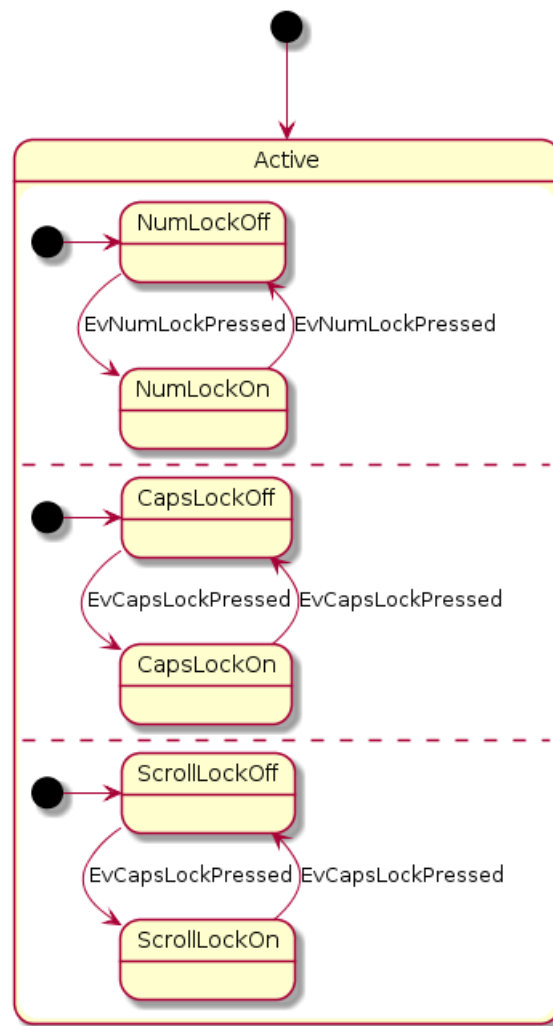
Используя оператор "--" или "||", вы можете объявлять параллельные подсостояния внутри составного состояния.

```

@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
  
```



7.5. Направления стрелок

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

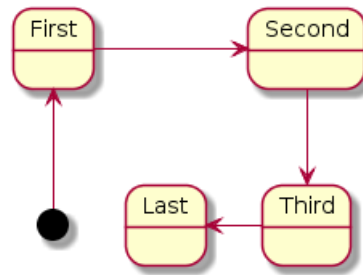
- `-down->` (default arrow)
- `-right->` or `->`
- `-left->`
- `-up->`

@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.

7.6. 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
```

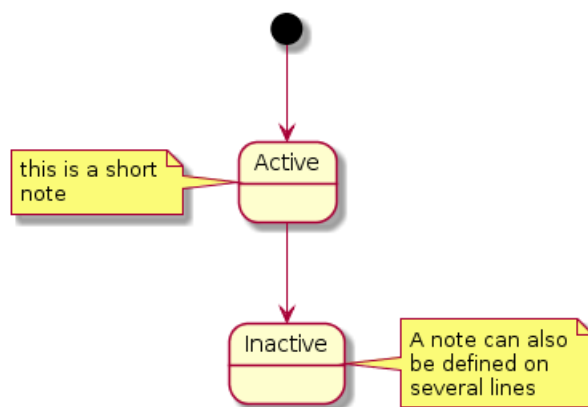
```
A note can also
```

```
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
```



7.7. Еще о заметках

Также заметки можно прикреплять к составным состояниям.

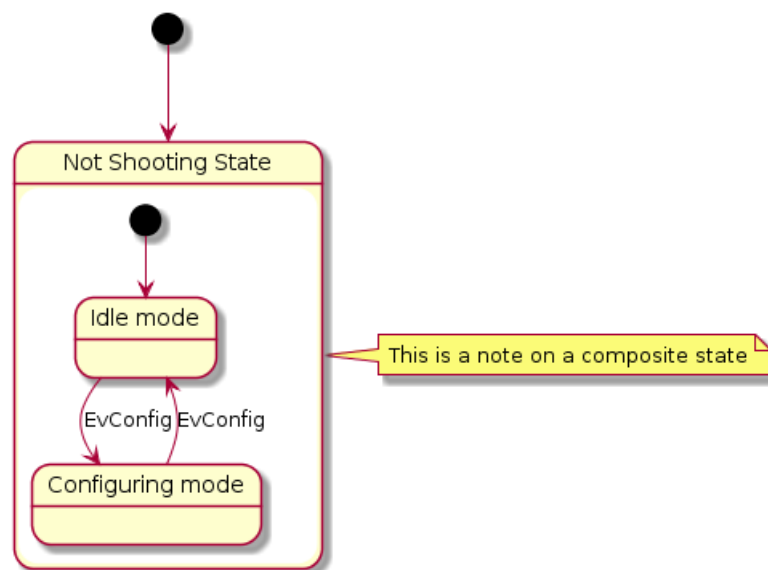
```
@startuml
```

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



7.8. Skinparam

Команда `skinparam` используется для указания шрифтов и настройки цвета элементов изображения.

Вы можете указать эту команду:

- в основном описании диаграммы - наряду с другими командами,
- во включаемом внешнем файле,
- в конфигурационном файле, передаваемом через командную строку или через ANT-задачу.

Вы можете задавать цвета и шрифты для именованных шаблонов состояний.

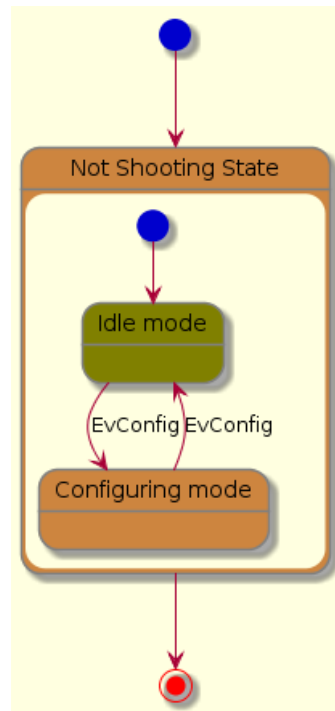
```
@startuml
skinparam backgroundColor LightYellow
skinparam state {
  StartColor MediumBlue
  EndColor Red
  BackgroundColor Peru
  BackgroundColor<<Warning>> Olive
  BorderColor Gray
  FontName Impact
}
```



```
[*] —> 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
```

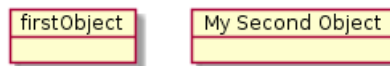


8. Диаграмма объектов

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 следующий символов :

расширение	< --	
композиция	*--	
агрегирование	o--	

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

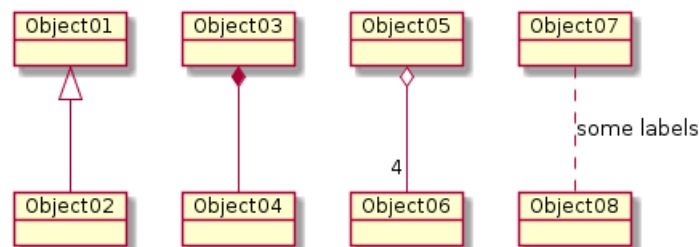
Зная данные правила, можно создать следующие картинки.

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
```

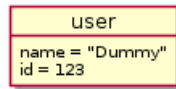


8.3. Добавление полей

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

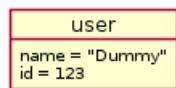


```
@startuml
object user
user : name = "Dummy"
user : id = 123
@enduml
```



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
- Стилизованный вывод

9. Общие команды

9.1. Комментарии

Все, что начинается с одинарной кавычки ' является комментарием.

Вы также можете поместить многострочный комментарий, используя '/' для открытия и '/' для закрытия комментария.

9.2. Заголовок и подвал

Для определения заголовка и подвала созданной диаграммы используйте команды `header` и `footer` соответственно.

Для выравнивания текста заголовка/подвала используйте префиксы `center`, `left` или `right`.

Заголовок и подвал могут содержать несколько строк. В этом случае их определение завершайте командами `endheader` и `endfooter` соответственно.

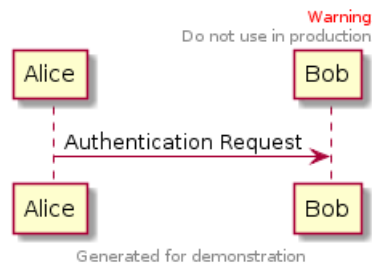
Заголовок и подвал поддерживают использование HTML разметки.

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

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

center footer Generated for demonstration

@enduml
```



9.3. Масштаб

Используйте команду `scale` для изменения размера генерируемого изображения.

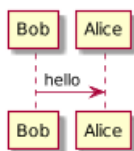
Используйте число или дробь для указания масштаба. Можно задавать ограничение отдельно только по ширине, либо ограничение отдельно только по высоте; значение указывается в пикселях. Можно указать оба ограничения (по высоте и ширине) - в этом случае изображение будет отмасштабировано так, чтобы вписаться в прямоугольник заданных размеров.

- `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
```



9.4. Заголовок

Ключевое слово `title` используется для задания заголовка. С помощью последовательности символов `\n` вы можете добавить перевод строки в заголовок.

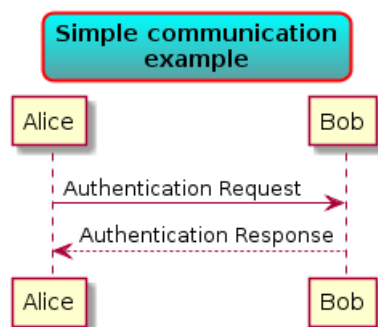
`skinparam` установками можно задать фон заголовка, цвет границы, толщину и окружность углов.

```
@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
```



Вы можете использовать форматирование на Creole для заголовков.

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 <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



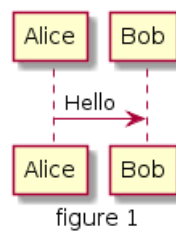
9.5. Надписи

Ключевым словом `caption` можно ставить надписи над диаграммой.

```
@startuml
```

```
caption figure 1
Alice -> Bob: Hello
```

```
@enduml
```



9.6. Легенда для диаграммы

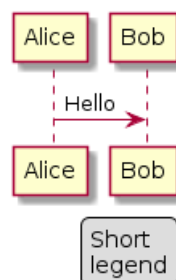
`legend` и `end legend` - пара ключевых слов для создания легенды на диаграмме.

Опционально можно указать выравнивание легенды диаграммы ключевым словом `left`, `right` или `center`.

```
@startuml
```

```
Alice -> Bob : Hello
legend right
Short
legend
endlegend
```

```
@enduml
```



10. Salt

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

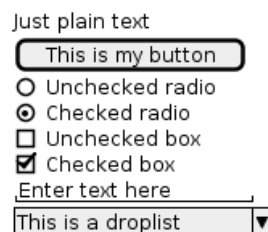
Вы можете использовать ключевое слово @startsalt, или @startuml, после которого стоит ключевое слово salt.

10.1. 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.

10.2. 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:

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
```

Login	MyName
Password	****
Cancel	OK

10.3. 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

10.4. 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

```



10.5. 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 Browse...

10.6. 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

```

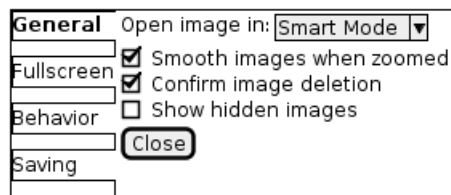
General
Fullscreen
Behavior
Saving

Open image in: Smart Mode ▼
☒ Smooth images when zoomed
☒ Confirm image deletion
☐ Show hidden images
Close



Вкладки также могут быть вертикально ориентированы:

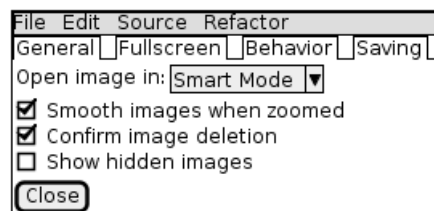
```
@startsalt
{+
{/ <b>General
Fullscreen
Behavior
Saving } |
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
[Close]
}
}
@endsalt
```



10.7. Использование меню

Вы можете добавить меню используя нотацию {*.

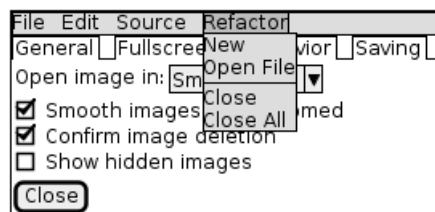
```
@startsalt
{+
{* File | Edit | Source | Refactor }
{/ General | Fullscreen | Behavior | Saving }
{
{ Open image in: | ^Smart Mode^ }
[X] Smooth images when zoomed
[X] Confirm image deletion
[ ] Show hidden images
}
[Close]
}
@endsalt
```



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
```





10.8. 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	

11. Creole

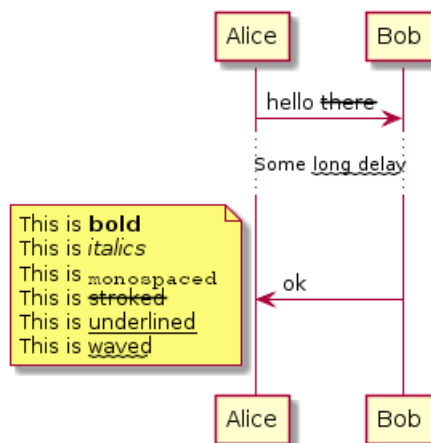
Легкий движок Creole интегрирован в PlantUML, чтобы предоставлять стандартный способ определения стиля текста.

На данный момент все диаграммы поддерживают данный синтаксис.

Обратите внимание, что сохраняется совместимость и с восходящей синтаксиса HTML.

11.1. Подчёркнутый текст

```
@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
```

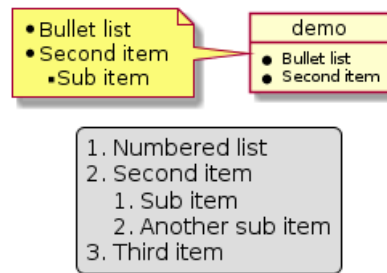


11.2. Список

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

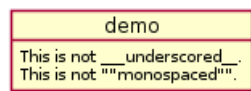




11.3. Символ экранирования

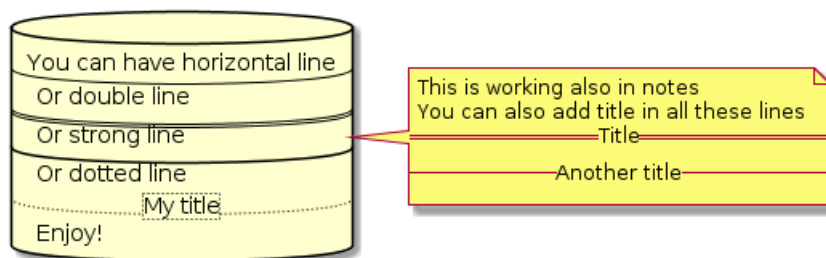
Вы можете использовать тильду ~, чтобы экранировать специальные символы creole.

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



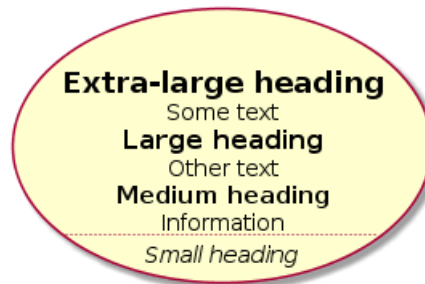
11.4. Горизонтальные линии

```
@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
```



11.5. Заголовки

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



11.6. Наследованный HTML

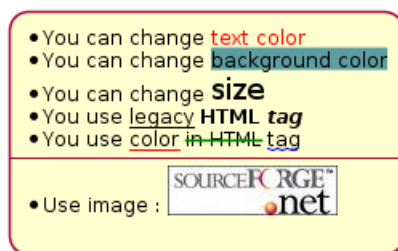
Some HTML tags are also working:

- `` для жирного текста
- `<u>` или `<u:#AAAAAA>` или `<u:colorName>` для подчёркивания
- `<i>` для курсива
- `<s>` или `<s:#AAAAAA>` или `<s:colorName>` для зачёркнутого текста
- `<w>` или `<w:#AAAAAA>` или `<w:colorName>` для подчёркнутого волной текста
- `<color:#AAAAAA>` или `<color:colorName>`
- `<back:#AAAAAA>` или `<back:colorName>` для цвета фона
- `<size:nn>` чтобы изменить размер шрифта
- `<img:file>` : файл должен быть доступен в файловой системе
- `<img:http://url>` : URL должен быть доступен через 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>HIML<i>tag</i></b>
* You use <u:red>color</u> <s:green>in HIML</s> <w:#0000FF>tag</w>

* Use image : <img:sourceforge.jpg>
;

@enduml
```



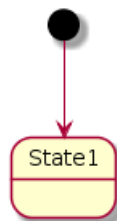
11.7. Таблица

Вы можете построить таблицу

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

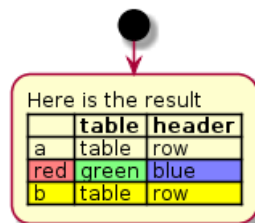
Example of simple table

	table	header
a	table	row
b	table	row



Вы можете указать цвета для линий и фона ячеек

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



11.8. Дерево

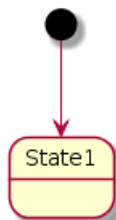
Вы можете использовать символы |_ чтобы создать дерево.

```
@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



11.9. Специальные символы

Есть возможность использовать символы Unicode с синтаксисом `&#` или `<U+XXXX>`

```

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



11.10. OpenIconic

OpenIconic - очень хороший open source набор иконок. Эти иконки могут быть интегрированы в парсер creole, так что вы можете пользоваться ими из коробки.

Вы можете использовать следующий синтаксис: `<&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♥



Полный список доступен на вебсайте OpenIconic, или в специальной диаграмме:

```

@startuml
listopeniconic
@enduml
  
```


List Open IconicCredit 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	coq	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	contrast	flag	lock-locked	power-standby	thumb-up
arrow-circle-right	bullhorn	copywriting	flash	lock-unlocked	print	timer
arrow-circle-top	calculator	credit-card	fork	loop-circular	project	transfer
arrow-left	calendar	crop	fullscreen-enter	loop-square	pulse	trash
arrow-right	camera-slr	dashboard	fullscreen-exit	loop	puzzle-piece	underline
arrow-thick-bottom	caret-bottom	data-transfer-download	globe	magnifying-glass	question-mark	vertical-align-bottom
arrow-thick-left	caret-left	data-transfer-upload	graph	map-marker	rain	vertical-align-center
arrow-thick-right	caret-right	delete	grid-four-up	map	random	vertical-align-top
arrow-thick-top	caret-top	dial	grid-three-up	media-pause	reload	video
arrow-top	cart	dollar	grid-two-up	media-play	resize-both	volume-high
audio-spectrum	chat	double-quote-sans-left	hard-drive	media-record	resize-height	volume-low
badge	check	double-quote-sans-right	header	media-skip-backward	resize-width	volume-off
ban	chevron-bottom	double-quote-serif-left	headphones	media-skip-forward	rss-alt	warning
bar-chart	chevron-left	double-quote-serif-right	heart	media-step-backward	rss	wifi
basket	chevron-right	droplet	home	media-step-forward	script	wrench
battery-empty	circle-check	eject	image	media-stop	share-boxed	x
battery-full	circle-x	elevator	inbox	medical-cross	share	yen
beaker	cloud-download	envelope-closed	infinity	menu	shield	zoom-in
	cloud-upload	envelope-open	info	microphone	signal	zoom-out
		euro	italic	minus	signpost	
			justify-center	monitor	sort-ascending	
			justify-left	moon	sort-descending	
				move	spreadsheet	



11.11. Задание и использование спрайтов

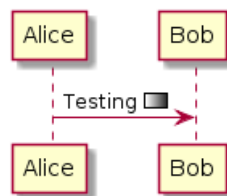
Спрайт - это маленький графический элемент, который может использоваться в диаграммах.

В PlantUML, спрайты монохромные, и могут иметь 4, 8 и 16 уровней серого.

Чтобы определить спрайт, вам нужно использовать шестнадцатитричную цифру между 0 и F для каждого пикселя.

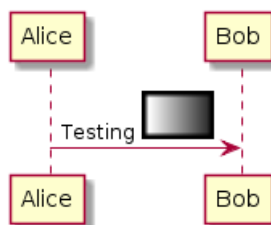
Затем вы можете использовать спрайт с помощью <\$XXX>, где XXX - это название спрайта.

```
@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
```



11.12. Кодирование спрайта

Чтобы кодировать спрайт, вы можете использовать командную строку команды так:



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

где `foo.png` - файл изображения, который вы хотите закодировать (он будет конвертирован в серый автоматически).

После `-encodesprite`, вам нужно задать формат: 4, 8, 16, 4z, 8z или 16z.

Число означает уровни серого и опциональный `z` используется для включения компрессии при задании спрайта.

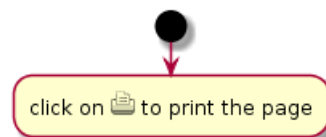
11.13. Ипортирование спрайта

Вы также можете запустить GUI чтобы сгенерировать спрайт из существующего изображения.

Нажмите на меню, а потом на `File/Open Sprite Window`.

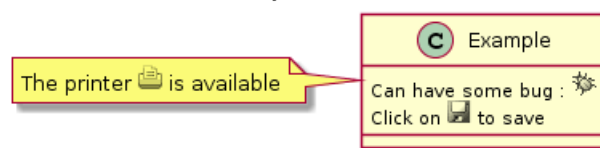
После копирования изображения в буфер обмена, несколько возможных вариантов спрайта будет показано, вы должны выбрать тот, который хотите.

11.14. Примеры



```
@startuml
sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHLuxP3L6kbTcizR8tAhzaqFvXwvFfPEq
start
:click on <$printer> to print the page;
@enduml
```

Use of sprites (🖨️, ⚙️...)



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p_FEjQEgB1z27aeqCqixa8S4OT7C53cKpsHpaYPDjY_12MHM-BLRyywPhrrlw3qum
sprite $printer [15x15/8z] NOtH3W0W208HxFz_kMAhj7lHWpa1XC716sz0Pq4MVPEWfBHLuxP3L6kbTcizR8tAhzaqFvXwvFfPEq
sprite $disk {
444445566677881
436000000009991
43600000000ACA1
53700000001A7A1
537000000012B8A1
538000000123B8A1
638000001233C9A1
634999AABBC99B1
744566778899AB1
7456AAAAA99AAB1
8566AFC228AABB1
8567AC8118BBBB1
867BD4433BBBB1
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
```

12. Изменение шрифта и цвета

12.1. Использование

Настройка шрифта и цвета производится с помощью команды `skinparam`. Пример:
`skinparam backgroundColor yellow`

Вы можете указать эту команду:

- В описании диаграммы - наряду с другими командами,
- В подключаемом файле (см. *Препроцессинг*),
- В конфигурационном файле, передаваемом через командную строку или через ANT-задачу.

12.2. Вложенные определения

Чтобы избежать повторений, возможно создавать вложенные определения. Так, как в следующем определении:

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

строго эквивалентно:

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

12.3. Цвет

Вы можете использовать стандартное название цвета или RGB код.

Название параметра	По умолчанию Значение	Цвет	Комментарий
backgroundColor	white		Задний фон страницы
activityArrowColor	#A80036		Цвет стрелок в диаграмме активности
activityBackgroundColor	#FEFECF		Задний фон активностей
activityBorderColor	#A80036		Цвет границ активностей
activityStartColor	black		Начальный круг в диаграмме активности
activityEndColor	black		Конечный круг в диаграмме активности
activityBarColor	black		Синхронизационная метка в диаграмме активности
usecaseArrowColor	#A80036		Цвет стрелок в диаграмме прецедентов
usecaseActorBackgroundColor	#FEFECF		Цвет головы актёра в диаграмме прецедентов
usecaseActorBorderColor	#A80036		Цвет границ актёра в диаграмме прецедентов
usecaseBackgroundColor	#FEFECF		Задний фон прецедентов
usecaseBorderColor	#A80036		Цвет границ прецедентов в диаграмме прецедентов
classArrowColor	#A80036		Цвет стрелок в диаграмме классов
classBackgroundColor	#FEFECF		Цвет фона классов/интерфейсов/перечислений
classBorderColor	#A80036		Цвет рамки классов/интерфейсов/перечислений
packageBackgroundColor	#FEFECF		Задний фон пакета в диаграмме классов
packageBorderColor	#A80036		Границы пакетов в диаграмме классов
stereotypeCBackgroundColor	#ADD1B2		Фон типа класса в диаграмме классов
stereotypeABackgroundColor	#A9DCDF		Фон индикаторов абстрактных классов
stereotypeIBackgroundColor	#B4A7E5		Фон типа интерфейсов в диаграмме классов
stereotypeEBackgroundColor	#EB937F		Фон типа перечисления в диаграмме классов
componentArrowColor	#A80036		Цвет стрелок в диаграмме компонентов
componentBackgroundColor	#FEFECF		Фон компонентов
componentBorderColor	#A80036		Границы компонентов
componentInterfaceBackgroundColor	#FEFECF		Фон интерфейсов в диаграмме компонентов
componentInterfaceBorderColor	#A80036		Границы интерфейса в диаграмме компонентов
noteBackgroundColor	#FBFB77		Фон заметок
noteBorderColor	#A80036		Граница заметок
stateBackgroundColor	#FEFECF		Фон состояний в диаграмме состояний
stateBorderColor	#A80036		Граница состояний в диаграмме состояний
stateArrowColor	#A80036		Цвета стрелок в диаграмме состояний
stateStartColor	black		Начальный круг в диаграмме состояний
stateEndColor	black		Конечный круг в диаграмме состояний
sequenceArrowColor	#A80036		Цвет стрелок в диаграмме последовательностей
sequenceActorBackgroundColor	#FEFECF		Цвет головы актёра в диаграмме последовательностей
sequenceActorBorderColor	#A80036		Границы актёра в диаграмме последовательностей
sequenceGroupBackgroundColor	#EEEEEE		Цвет заголовка alt/opt/loop в диаграмме последовательностей
sequenceLifeLineBackgroundColor	white		Фон линии активности в диаграмме последовательностей
sequenceLifeLineBorderColor	#A80036		Граница линии активности в диаграмме последовательностей
sequenceParticipantBackgroundColor	#FEFECF		Фон участника в диаграмме последовательностей
sequenceParticipantBorderColor	#A80036		Граница участника в диаграмме последовательностей



12.4. Имя, цвет и размер шрифта

Вы можете изменить шрифт изображения, используя `xxxFontColor`, `xxxFontSize` и `xxxFontName`.

Пример:

```
skinparam classFontColor red
skinparam classFontSize 10
skinparam classFontName Apex
```

Любой шрифт можно задать в качестве шрифта по-умолчанию, используя `skinparam defaultFontName`.

Пример:

```
skinparam defaultFontName Apex
```

Пожалуйста, заметьте, что название шрифта очень зависит от системы, так что не используйте это слишком активно, если вы хотите переносимости.

Параметр Имя	По умолчанию Значение	Комментарий
activityFontColor activityFontSize activityFontStyle activityFontName	black 14 plain	Для активности
activityArrowFontColor activityArrowFontSize activityArrowFontStyle activityArrowFontName	black 13 plain	Для текста в стрелках в диаграмме активности
circledCharacterFontColor circledCharacterFontSize circledCharacterFontStyle circledCharacterFontName circledCharacterRadius	black 17 bold Courier 11	Для текста в кругах для классов, перечислений
classArrowFontColor classArrowFontSize classArrowFontStyle classArrowFontName	black 10 plain	Используется для подписей стрелок в диаграмме
classAttributeFontColor classAttributeFontSize classAttributeIconSize classAttributeFontStyle classAttributeFontName	black 10 10 plain	Атрибуты класса и методы
classFontColor classFontSize classFontStyle classFontName	black 12 plain	Названия классов
classStereotypeFontColor classStereotypeFontSize classStereotypeFontStyle classStereotypeFontName	black 12 italic	Шаблоны классов
componentFontColor componentFontSize componentFontStyle componentFontName	black 14 plain	Название компонентов
componentStereotypeFontColor componentStereotypeFontSize componentStereotypeFontStyle componentStereotypeFontName	black 14 italic	Шаблоны в компонентах



componentArrowFontColor componentArrowFontSize componentArrowFontStyle componentArrowFontName	black 13 plain	Текст в стрелках в диаграмме компонентов
noteFontColor noteFontSize noteFontStyle noteFontName	black 13 plain	Для заметок во всех диаграммах, кроме после
packageFontColor packageFontSize packageFontStyle packageFontName	black 14 plain	Для названий пакетов и разделов
sequenceActorFontColor sequenceActorFontSize sequenceActorFontStyle sequenceActorFontName	black 13 plain	Для актёры в диаграмме последовательности
sequenceDividerFontColor sequenceDividerFontSize sequenceDividerFontStyle sequenceDividerFontName	black 13 bold	Для текста в разделителе в диаграмме послед
sequenceArrowFontColor sequenceArrowFontSize sequenceArrowFontStyle sequenceArrowFontName	black 13 plain	Для текста на стрелках в диаграмме последов
sequenceGroupingFontColor sequenceGroupingFontSize sequenceGroupingFontStyle sequenceGroupingFontName	black 11 plain	Для текста "else" в диаграмме последовательн
sequenceGroupingHeaderFontColor sequenceGroupingHeaderFontSize sequenceGroupingHeaderFontStyle sequenceGroupingHeaderFontName	black 13 plain	Для заголовков "alt/opt/loop" в диаграмме пос.
sequenceParticipantFontColor sequenceParticipantFontSize sequenceParticipantFontStyle sequenceParticipantFontName	black 13 plain	Для текста на участнике в диаграмме послед
sequenceTitleFontColor sequenceTitleFontSize sequenceTitleFontStyle sequenceTitleFontName	black 13 plain	Для заголовка в диаграмме последовательнос
titleFontColor titleFontSize titleFontStyle titleFontName	black 18 plain	Для заголовка во всех диаграммах, кроме пос.
stateFontColor stateFontSize stateFontStyle stateFontName	black 14 plain	Для состояния в диаграмме состояния
stateArrowFontColor stateArrowFontSize stateArrowFontStyle stateArrowFontName	black 13 plain	Для задания текста на стрелках в диаграммах
stateAttributeFontColor stateAttributeFontSize stateAttributeFontStyle stateAttributeFontName	black 12 plain	Для описания состояния в диаграмме состояни



usecaseFontColor usecaseFontSize usecaseFontStyle usecaseFontName	black 14 plain	Для меток прецедентов на диаграммах прецедентов
usecaseStereotypeFontColor usecaseStereotypeFontSize usecaseStereotypeFontStyle usecaseStereotypeFontName	black 14 italic	Для шаблонов в прецедентах
usecaseActorFontColor usecaseActorFontSize usecaseActorFontStyle usecaseActorFontName	black 14 plain	Для описания актёров в диаграммах прецедентов
usecaseActorStereotypeFontColor usecaseActorStereotypeFontSize usecaseActorStereotypeFontStyle usecaseActorStereotypeFontName	black 14 italic	Для шаблона для актёра
usecaseArrowFontColor usecaseArrowFontSize usecaseArrowFontStyle usecaseArrowFontName	black 13 plain	Для текста в стрелках в диаграмме прецедентов
footerFontColor footerFontSize footerFontStyle footerFontName	black 10 plain	Для подвала
headerFontColor headerFontSize headerFontStyle headerFontName	black 10 plain	Для заголовков



12.5. Черно-белый вывод

Вы можете принудительно задать использование черно-белого вывода используя команду `skinparam monochrome true`.

```
@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

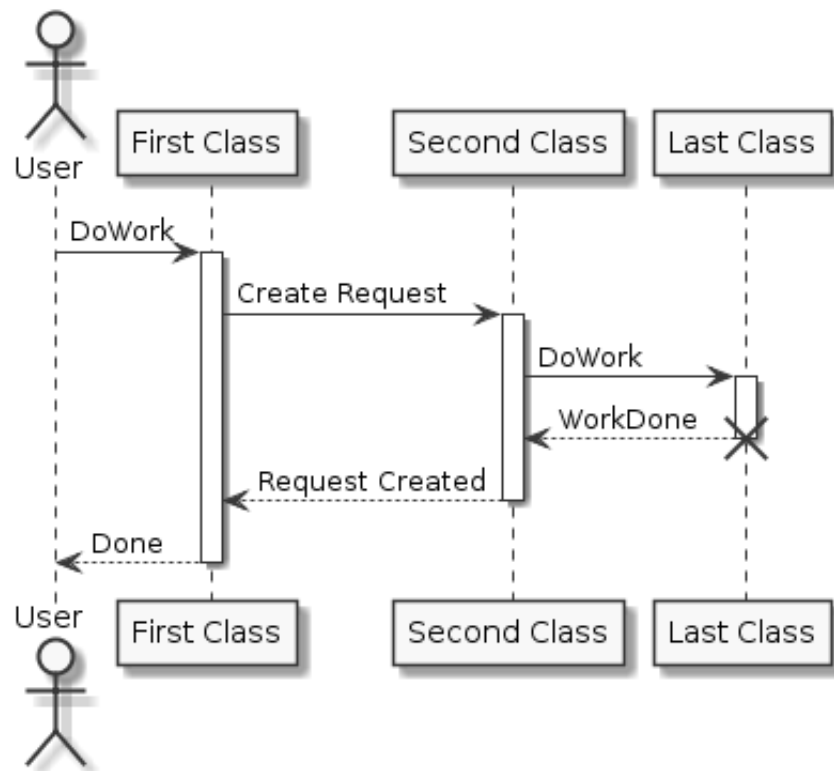
A -> B: Create Request
activate B

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

B -> A: Request Created
deactivate B

A -> User: Done
deactivate A

@enduml
```



13. Предварительная обработка

Некоторая небольшая возможность препроцессинга включена в PlantUML, и доступна для всех диаграмм.

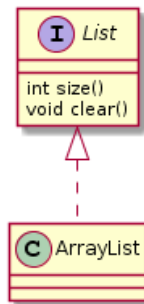
Эта функциональность очень схожа с препроцессором языка **C**, за исключением того что символ решетки "#" заменяется на символ восклицания "!".

13.1. Включение файлов

Вы можете использовать директиву `!include` для включения файлов в ваши диаграммы.

Представьте, что вы используете один класс во множестве диаграмм. Вместо того, чтобы дублировать описание этого класса, вы можете разместить его описание в отдельном файле.

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



File List.iuml: interface List List : int size() List : void clear()

Файл `List.iuml` может быть включен в несколько диаграмм, любые изменения в этом файле будут применяться ко всем диаграммам, включающим его.

Этот файл может быть включен только раз. Если вы хотите включить тот же самый файл несколько раз, вы должны использовать директиву `!include_many` вместо `!include`.

Вы также можете добавить несколько текстовых блоков `@startuml/@enduml` в подключаемом файле, а потом задавать конкретно добавляемый блок, указав `!0`, где `0` - это номер блока.

Например, если вы используете `!include foo.txt!1`, второй блок `@startuml/@enduml` внутри `foo.txt` будет подключен.

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`.

13.2. Включение URL

Используйте директиву `!includeurl`, чтобы подключить файлы из Internet/Intranet в свою диаграмму.

Вы также можете использовать `!includeurl http://someurl.com/mypath!0`, чтобы задать, какой `@startuml/@enduml` блок из `http://someurl.com/mypath` вы хотите подключить. Нотация `!0` notation обозначает первую диаграмму.



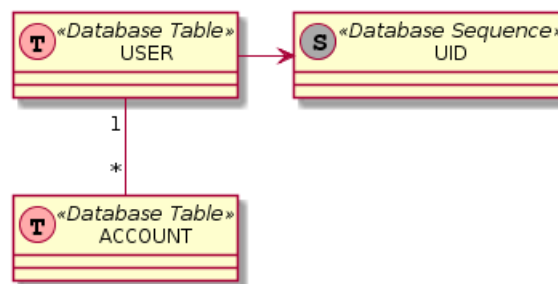
13.3. Константы

Вы можете определить константу, используя директиву `!define`. Как в языке **C**, название константы может содержать только буквы, цифры и символ подчёркивания, и не может начинаться с цифры.

```
@startuml
```

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

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



Естественно, вы можете использовать директиву `!include`, чтобы определить все константы в одном файле, который вы подключите в свои диаграммы.

Константы могут быть определены с помощью директивы `!undef XXX`.

Вы также можете задать константы из командной строки, флагами `-D`.

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

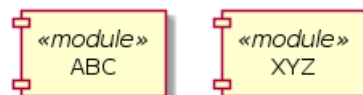
Заметьте, что флаг `-D` должен быть задан после секции `"-jar plantuml.jar"`.

13.4. Задание макроса

Вы также можете задать макрос с аргументами.

```
@startuml
```

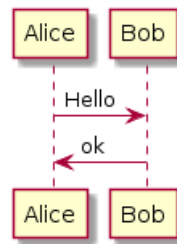
```
!define module(x) component x <<module>>
module(ABC)
module(XYZ)
@enduml
```



Макрос может иметь несколько аргументов.

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





13.5. Добавление даты и времени

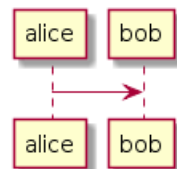
Вы можете добавить текущую дату и время, используя специальную переменную `%date%`.

Формат даты может быть преобразован с помощью форматирования, описанного в документации по SimpleDateFormat

```

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

Generated Sun Sep 03 17:09:54 UTC 2017 or 2017.09.03 at 17:09



13.6. Другие специальные переменные

Вы так же можете использовать следующие специальные переменные

`%dirpath%` Путь текущего файла

`%filename%` Имя текущего файла

13.7. Макросы на нескольких строках

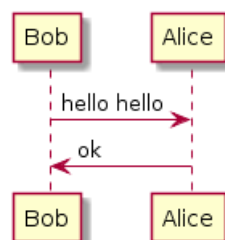
Вы также можете определить макрос на нескольких строках, используя `!definelong` и `!enddefinelong`.

```

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

```

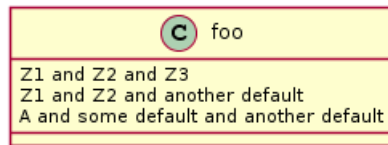
AUTHEN(Bob, Alice)
@enduml
  
```



13.8. Умолчания в параметрах макроса

Для параметров макроса можно назначить значения по умолчанию.

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



13.9. Условия

Вы можете использовать директивы `!ifdef XXX` и `!endif`, чтобы создавать условия рисования.

Строки между этими двумя директивами будут подключены, только если константа после директивы `!ifdef` была задана ранее.

Также можете использовать часть `!else`, которая будет подключена, если константа **не** была задана.

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

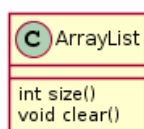


File ArrayList.iuml:

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

Потом можете использовать директиву `!define`, чтобы активировать условные части диаграммы.

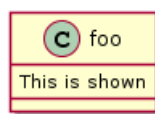
```
@startuml
!define SHOW_METHODS
!include ArrayList.iuml
@enduml
```



Вы также можете использовать директиву `!ifndef`, которая подключит строки, если данная константа **не** была задана.

Вы можете использовать логическое выражение со скобкой, операторы `&&` и `||` в тестировании.

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



13.10. Пути поиска

Вы можете явно задать `"plantuml.include.path"` в командной строке.

Например:

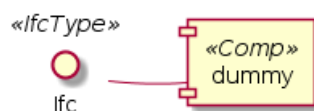
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

Заметьте, что опция `-D` должна быть поставлена до опции `-jar`. Опция `-D` после опции `-jar` будет использована для задания констант с помощью препроцессора PlantUML.

13.11. Расширенные функции

Есть возможность добавить текст к аргументу макроса, используя синтаксис `##`.

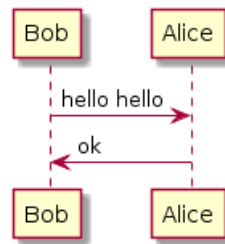
```
@startuml
!definelong COMP_TEXTGENCOMP(name)
[name] << Comp >>
interface Ifc << IfcType >> AS name##Ifc
name##Ifc - [name]
!enddefinelong
COMP_TEXTGENCOMP(dummy)
@enduml
```



Макрос может быть определён внутри другого макроса

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

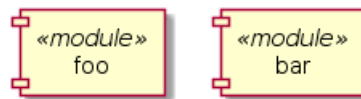




Макрос может быть полиморфичен с количеством аргументов.

```

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



Вы можете использовать переменную системного окружения или определение константы, когда подключаете файл:

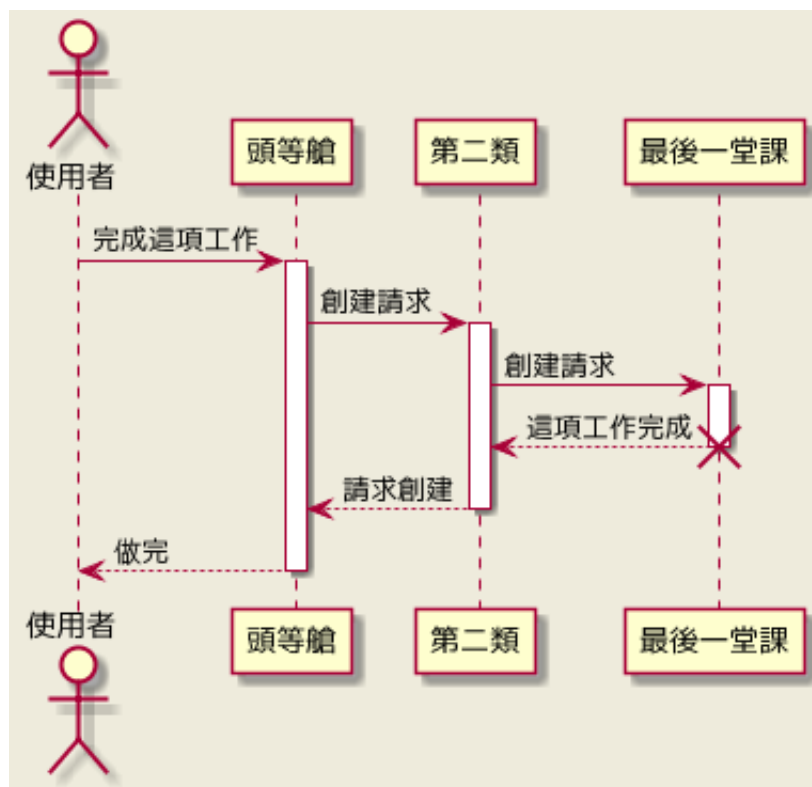
```

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


14. Интернационализация

Язык PlantUML использует буквы для задания актёра, прецедентов и так далее. Но буквы это не только латинские A-Z, это может быть *любая буква любого языка*.

```
@startuml
skinparam backgroundColor #EEEBDC
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
```



14.1. Кодировка

Кодировка по умолчанию используемая при чтении файла. содержащего текстовое описание UML зависит от системы. Обычно, всё должно быть хорошо, но в некоторых случаях, вы можете захотеть использовать другую кодировку. Для примера в командной строке:



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

Или в задании ANT:

```
<target name="main">  
<plantuml dir="./src" charset="UTF-8" />  
</target>
```

В зависимости от вашей установки Java, будут доступны следующие кодировки: ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16.

15. Названия цветов

Здесь перечислены названия цветов доступные в PlantUML. Названия цветов не чувствительны к регистру.

	AliceBlue		GhostWhite		NavajoWhite
	AntiqueWhite		GoldenRod		Navy
	Aquamarine		Gold		OldLace
	Aqua		Gray		OliveDrab
	Azure		GreenYellow		Olive
	Beige		Green		OrangeRed
	Bisque		HoneyDew		Orange
	Black		HotPink		Orchid
	BlanchedAlmond		IndianRed		PaleGoldenRod
	BlueViolet		Indigo		PaleGreen
	Blue		Ivory		PaleTurquoise
	Brown		Khaki		PaleVioletRed
	BurlyWood		LavenderBlush		PapayaWhip
	CadetBlue		Lavender		PeachPuff
	Chartreuse		LawnGreen		Peru
	Chocolate		LemonChiffon		Pink
	Coral		LightBlue		Plum
	CornflowerBlue		LightCoral		PowderBlue
	Cornsilk		LightCyan		Purple
	Crimson		LightGoldenRodYellow		Red
	Cyan		LightGreen		RosyBrown
	DarkBlue		LightGrey		RoyalBlue
	DarkCyan		LightPink		SaddleBrown
	DarkGoldenRod		LightSalmon		Salmon
	DarkGray		LightSeaGreen		SandyBrown
	DarkGreen		LightSkyBlue		SeaGreen
	DarkKhaki		LightSlateGray		SeaShell
	DarkMagenta		LightSteelBlue		Sienna
	DarkOliveGreen		LightYellow		Silver
	DarkOrchid		LimeGreen		SkyBlue
	DarkRed		Lime		SlateBlue
	DarkSalmon		Linen		SlateGray
	DarkSeaGreen		Magenta		Snow
	DarkSlateBlue		Maroon		SpringGreen
	DarkSlateGray		MediumAquaMarine		SteelBlue
	DarkTurquoise		MediumBlue		Tan
	DarkViolet		MediumOrchid		Teal
	Darkorange		MediumPurple		Thistle
	DeepPink		MediumSeaGreen		Tomato
	DeepSkyBlue		MediumSlateBlue		Turquoise
	DimGray		MediumSpringGreen		Violet
	DodgerBlue		MediumTurquoise		Wheat
	FireBrick		MediumVioletRed		WhiteSmoke
	FloralWhite		MidnightBlue		White
	ForestGreen		MintCream		YellowGreen
	Fuchsia		MistyRose		Yellow
	Gainsboro		Moccasin		



Содержание

1	Диаграммы последовательности	1
1.1	Основные примеры	1
1.2	Объявление участников	1
1.3	Использование небуквенных символов в названиях участников	2
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	Разбиение диаграмм	6
1.9	Группировка сообщений	7
1.10	Примечания в сообщениях	8
1.11	Другие примечания	8
1.12	Изменение формы примечаний	9
1.13	Creole и HTML	10
1.14	Разделитель	11
1.15	Ссылки	12
1.16	Задержка на диаграммах	12
1.17	Промежутки	13
1.18	Активация и деактивация линии существования	13
1.19	Отображение создания участника процессом	14
1.20	Входящие и исходящие сообщения	15
1.21	Шаблоны и отметки	16
1.22	Больше информации в заголовках	17
1.23	Группировка участников	18
1.24	Удаление футера	19
1.25	Skinparam	19
1.26	Изменение отступов	21
2	Диаграмма прецедентов	22
2.1	Прецеденты	22
2.2	Актёры	22
2.3	Описание прецедентов	22
2.4	Простой пример	23
2.5	Расширение	24
2.6	Использование заметок	24
2.7	Шаблоны	25
2.8	Смена направления стрелок	25
2.9	Разделение диаграмм	27
2.10	Направление слева направо	27
2.11	Skinparam	28
2.12	Полноценный пример	29



3	Диаграмма классов	30
3.1	Отношения между классами	30
3.2	Метки на отношениях	31
3.3	Добавление методов	32
3.4	Указание видимости	33
3.5	Абстрактные и статические	34
3.6	Расширенное тело класса	35
3.7	Заметки и шаблоны	36
3.8	Больше о заметках	37
3.9	Заметки на связях	38
3.10	Абстрактные классы и интерфейсы	39
3.11	Использование не буквенных символов	40
3.12	Скрытие атрибутов, методов...	41
3.13	Скрытие классов	42
3.14	Использование дженериков	42
3.15	Определение метки	42
3.16	Пакеты	43
3.17	Стили пакетов	43
3.18	Пространства имён	44
3.19	Автоматическое создание пространств имён	45
3.20	Lollipop интерфейс	46
3.21	Изменение направления стрелок	46
3.22	Ассоциация классов	47
3.23	Skinparam	48
3.24	Шаблоны со Skinparam	49
3.25	Цветовой градиент	49
3.26	Помощь в расположении классов	50
3.27	Разделение больших файлов	51
4	Диаграмма деятельности	53
4.1	Простая деятельность	53
4.2	Метка на стрелках	53
4.3	Изменение направления стрелки	54
4.4	Ветвления	54
4.5	Больше о ветках	55
4.6	Синхронизация	56
4.7	Длинное описание активности	57
4.8	Заметки	57
4.9	Разделы	58
4.10	Skinparam	59
4.11	Восьмиугольник	60
4.12	Полноценный пример	60



5	Диаграмма активности (бета)	63
5.1	Простая активность	63
5.2	Старт/Стоп	63
5.3	Условия	64
5.4	Повторяющийся цикл	65
5.5	Цикл while	66
5.6	Параллельные процессы	66
5.7	Заметки	67
5.8	Цвета	67
5.9	Стрелки	68
5.10	Группирование	69
5.11	Дорожки	69
5.12	Отсоединение	70
5.13	SDL	71
5.14	Полноценный пример	72
6	Диаграмма компонентов	74
6.1	Компоненты	74
6.2	Интерфейсы	74
6.3	Простой пример	75
6.4	Использование заметок	75
6.5	Группирование компонентов	76
6.6	Изменение направления стрелок	77
6.7	Использовании нотации UML2	78
6.8	Длинное описание	79
6.9	Индивидуальные цвета	79
6.10	Использование Sprite в стереотипах	79
6.11	Skinparam	80
7	Диаграмма состояний	82
7.1	Простое состояние	82
7.2	Составное состояние	82
7.3	Длинные имена	83
7.4	Параллельные состояния	84
7.5	Направления стрелок	85
7.6	Заметки	86
7.7	Еще о заметках	87
7.8	Skinparam	87
8	Диаграмма объектов	89
8.1	Определение объектов	89
8.2	Отношения между объектами	89
8.3	Добавление полей	89
8.4	Общие с диаграммами классов функции	90



9 Общие команды	91
9.1 Комментарии	91
9.2 Заголовок и подвал	91
9.3 Масштаб	91
9.4 Заголовок	92
9.5 Надписи	93
9.6 Легенда для диаграммы	93
10 Salt	94
10.1 Простые виджеты	94
10.2 Использование сетки	94
10.3 Использование разделителя	95
10.4 Древовидный виджет	95
10.5 Окружающие скобки	96
10.6 Добавление вкладок	96
10.7 Использование меню	97
10.8 Продвинутая таблица	98
11 Creole	99
11.1 Подчёркнутый текст	99
11.2 Список	99
11.3 Символ экранирования	100
11.4 Горизонтальные линии	100
11.5 Заголовки	101
11.6 Наследованный HTML	101
11.7 Таблица	102
11.8 Дерево	102
11.9 Специальные символы	103
11.1 OpenIconic	103
11.1 Задание и использование спрайтов	105
11.1 Кодирование спрайта	105
11.1 Ипортирование спрайта	106
11.1 Примеры	106
12 Изменение шрифта и цвета	108
12.1 Использование	108
12.2 Вложенные определения	108
12.3 Цвет	109
12.4 Имя, цвет и размер шрифта	110
12.5 Черно-белый вывод	113



13 Предварительная обработка	114
13.1 Включение файлов	114
13.2 Включение URL	114
13.3 Константы	115
13.4 Задание макроса	115
13.5 Добавление даты и времени	116
13.6 Другие специальные переменные	116
13.7 Макросы на нескольких строках	116
13.8 Умолчания в параметрах макроса	117
13.9 Условия	117
13.1 Пути поиска	118
13.1 Расширенные функции	118
14 Интернационализация	120
14.1 Кодировка	120
15 Названия цветов	122

