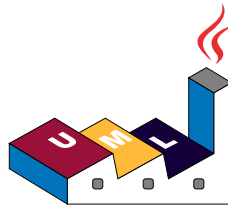


# Построение диаграмм UML с использованием PlantUML



Справочное руководство по языку  
(Версия 6085)

**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

Каждое UML description должно начинаться с @startuml и должно оканчиваться @enduml.

Последовательность "->" используется чтобы нарисовать сообщение между двумя участниками.

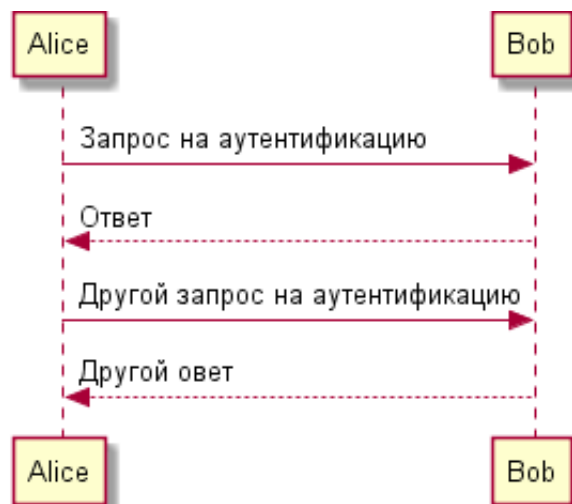
Участники не должны быть явно объявлены.

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

Также возможно использовать "<-" и "<--". Это не изменит изображения, но можете улучшить читабельность.

Пример:

```
@startuml
Alice -> Bob: Запроснааутентификацию
Bob --> Alice: Ответ
Alice -> Bob: Другойзапроснааутентификацию
Alice <-- Bob: Другойответ
@enduml
```



## 1.2 Объявление участников

Возможно изменить порядок участников, используя ключевое слово `participant`.

Также возможно использовать ключевое слово `actor` чтобы использовать фигурку человека вместо квадрата для изображения участника.

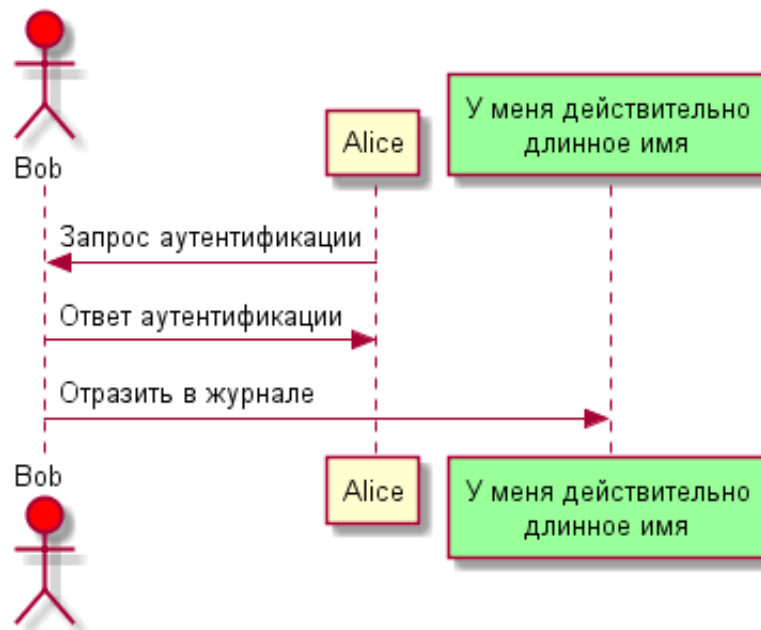
Можно переименовать участника используя ключевое слово `as`

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

Все, что начинается с одинарной кавычки ' является комментарием.

```
@startuml
actor Bob #red
' Единственная разница между actor и participant заключается в их изображении
participant Alice
participant Y "меня действительно \ длинное имя" as L #99FF99

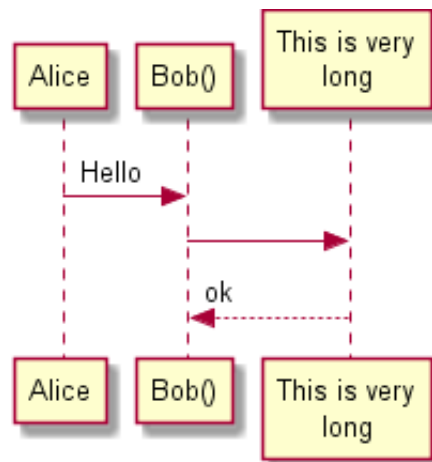
Alice->>Bob: Запрос аутентификации
Bob->>Alice: Ответ аутентификации
Bob->>L: Отобразить в журнале
@enduml
```



### 1.3 Use non-letters in participants

Вы можете использовать кавычки для задания участника. Также Вы можете использовать ключевые слова для присвоения псевдонимов к этим участникам.

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

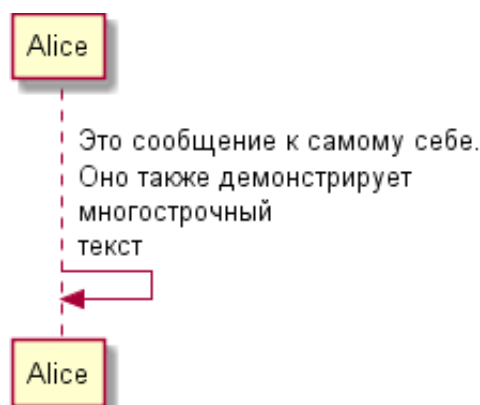


### 1.4 Сообщения к самому себе

Участник может посылать сообщения сам себе.

Также возможно создание многострочных используя \n.

```
@startuml
Alice->>Alice: Это сообщение к самому себе .\ Оно также демонстрирует \многострочный \текст
@enduml
```

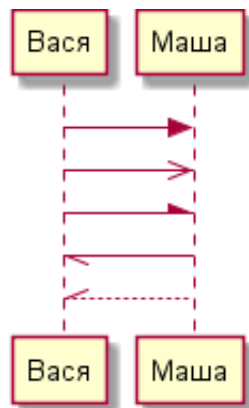


## 1.5 Изменить стиль стрелок

Вы можете изменить стиль стрелок следующими способами:

- используя \ или / вместо < или > для создания только верхней или нижней части стрелки.
- повторите окончание стрелки (например, >> or //) head to для тонкой отрисовки.
- используйте -- вместо - для создания пунктирной стрелки

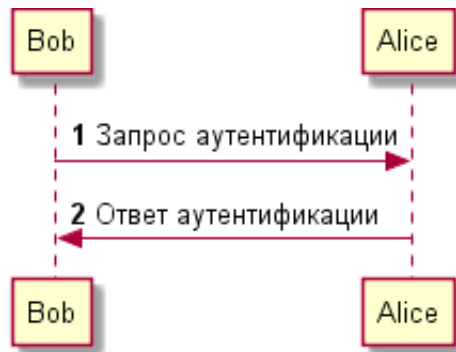
```
@startumlВася
-> МашаВася
->> МашаВася
-\ МашаВася
\\- МашаВася
//-- Маша
@enduml
```



## 1.6 Message sequence numbering

Ключевое слово `autonumber` используется для автоматической нумерации сообщений.

```
@startuml
autonumber
Bob -> Alice : Запросаутентификации
Bob <- Alice : Ответаутентификации
@enduml
```



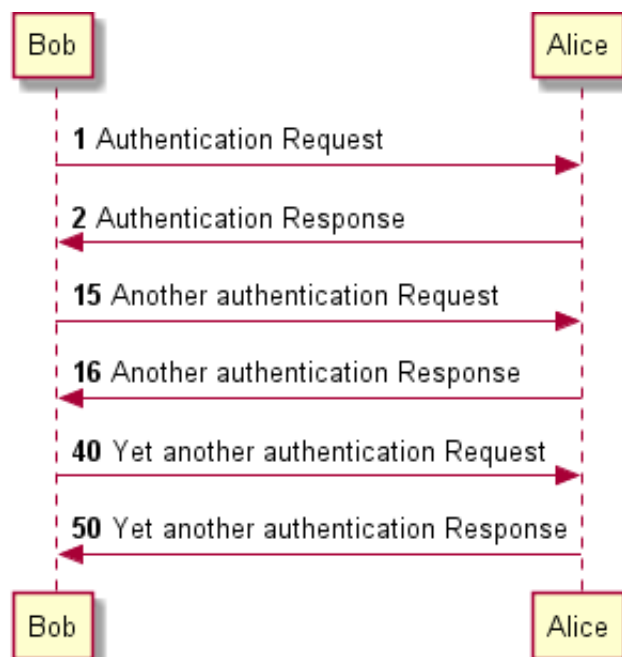
Можно указать

- начальный номер с помощью `"autonumber 'start'"`
- автоматическое увеличение номера используя `"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' means digit, '#' означает цифру или ноль, если не указан код символа ).

При форматировании также можно использовать теги html.

@startuml

```
autonumber "<b>[000]"Вася
```

```
→ Маша: ЗапросаутентификацииВася
```

```
<- Маша: Ответназапросаутентификации
```

```
autonumber 15 "<b>(<u>##</u>)"Вася
```

```
→ Маша: ДругойзапросаутентификацииВася
```

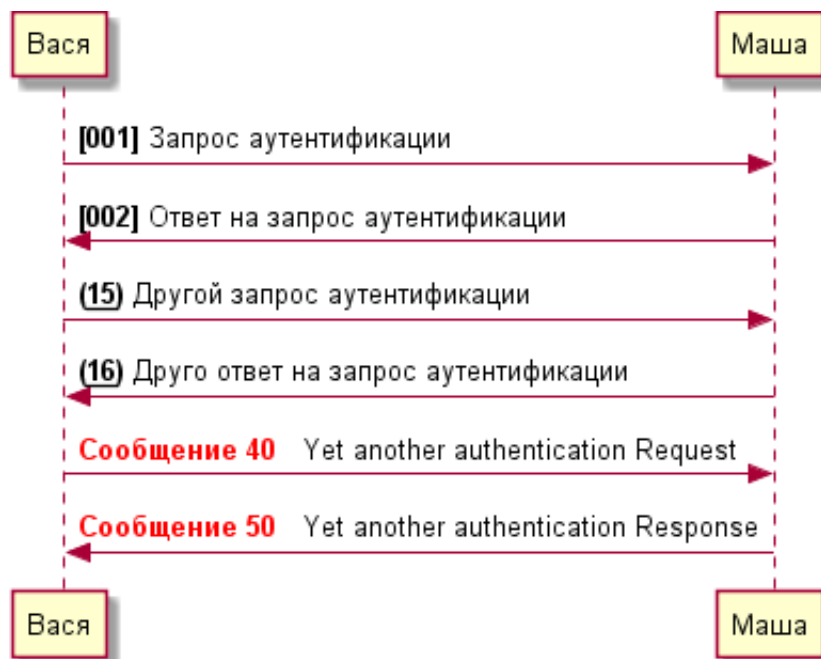
```
<- Маша: Другоответназапросаутентификации
```

```
autonumber 40 10 "<font color=red><b>Сообщение> 0 "Вася
```

```
→ Маша: Yet another authentication RequestВася
```

```
<- Маша: Yet another authentication Response
```

@enduml



## 1.7 Заголовок

Ключевое слово `title` используется для задания заголовка.

```
@startuml
```

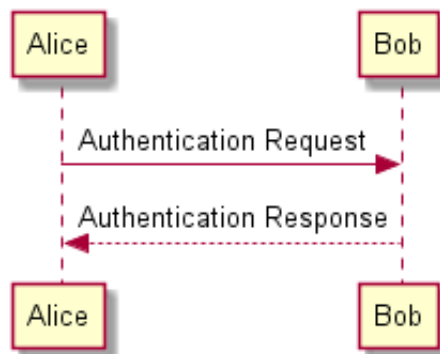
```
title Пример просток коммуникации
```

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

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

```
@enduml
```

### Пример просток коммуникации





## 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 : сообщение 1
```

```
Alice ->> Bob : сообщение 2
```

```
newpage
```

```
Alice ->> Bob : сообщение 3
```

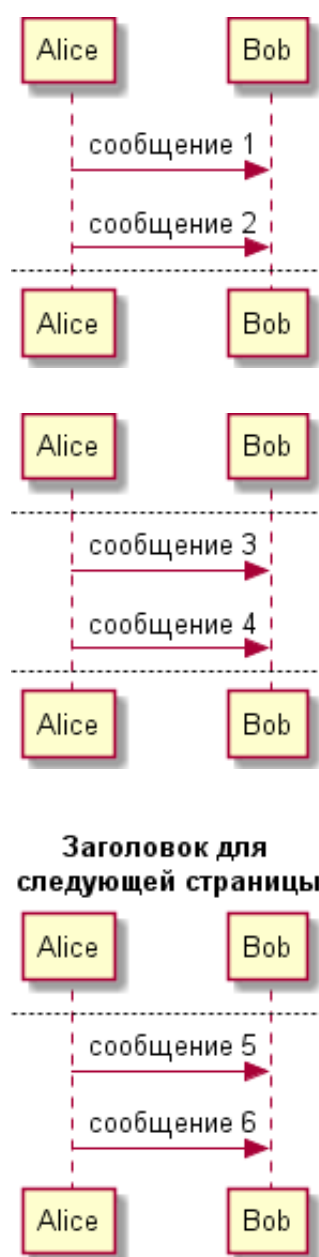
```
Alice ->> Bob : сообщение 4
```

```
newpage Заголовок для \следующей\ страницы
```

```
Alice ->> Bob : сообщение 5
```

```
Alice ->> Bob : сообщение 6
```

```
@enduml
```



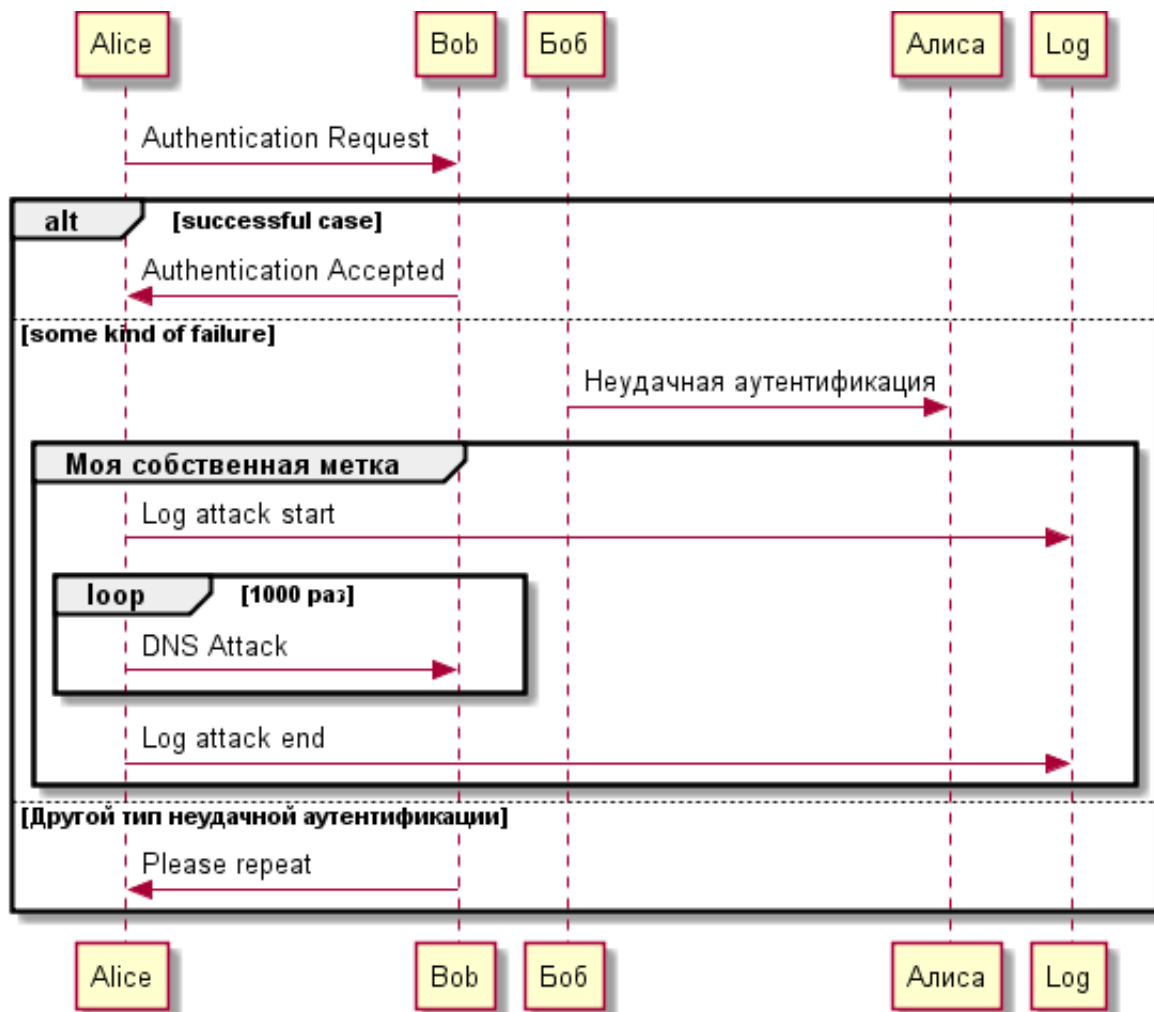
## 1.9 Группировка сообщений

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.

```
@startuml
Alice -> Bob: Authentication Request
alt successful case
    Bob -> Alice: Authentication Accepted
else some kind of failureБоб
    -> Алиса: Неудачнаяаутентификация
    group Моясобственнаяметка
        Alice -> Log : Log attack start
        loop 1000 раз
            Alice -> Bob: DNS Attack
        end
        Alice -> Log : Log attack end
    end
else Другойтипнеудачнойаутентификации
    Bob -> Alice: Please repeat
end
@enduml
```



## 1.10 Примечания в сообщениях

Можно размещать примечания в сообщениях используя :

- `note left of`
- `note right` keywords just after the message.

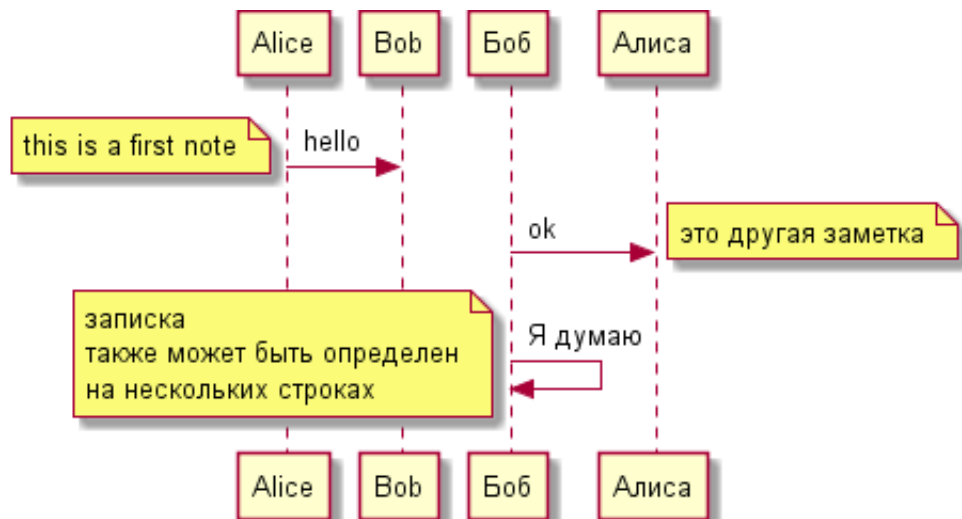
You can have multilines note using the `end note` keyword.

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

```
-> : ok
note right: этодругаязаметкаБобБоб
```

```
-> : Я думаю
note left: записка также может быть определен на нескольких строках
```

```
end note
@enduml
```



## 1.11 Some other notes

It is also possible to place notes relative to participant with:

- note left of,
- note right of or
- note over keywords.

It is possible to highlight a note by changing its background color. You can also have multiline note using the end note keywords.

```
@startuml
participant Alice
participant Боб
note left of Алиса : Это отображается слева от Алисы

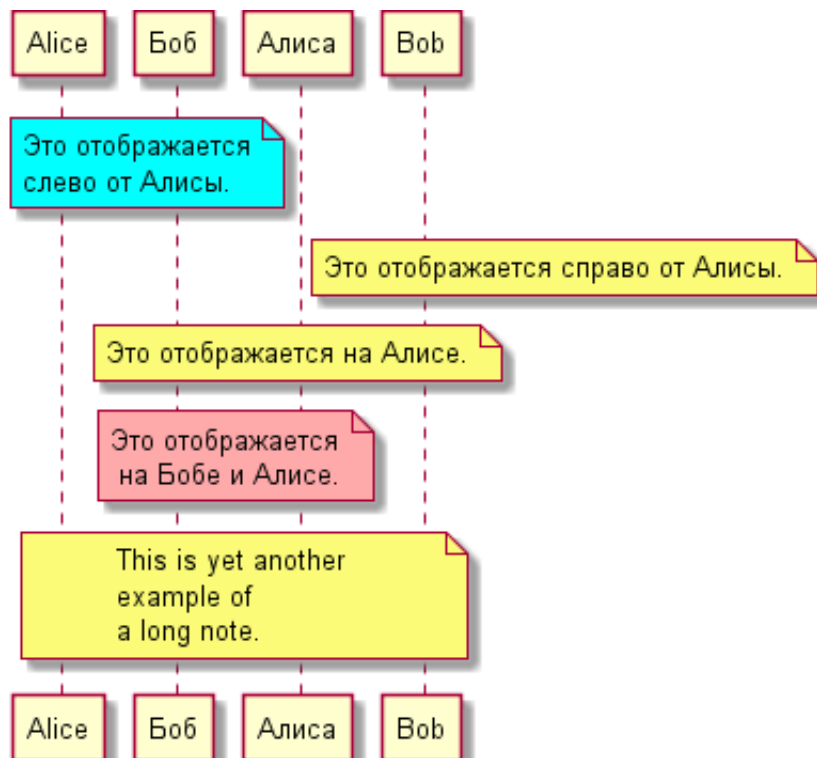
end note

note right of Алиса : Это отображается справа от Алисы

note over Алиса : Это отображается на Алисе

note over Алиса, Боб : #FFAAAA: Это отображается \n на Бобе и Алисе

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



## 1.12 Форматирование с использованием HTML

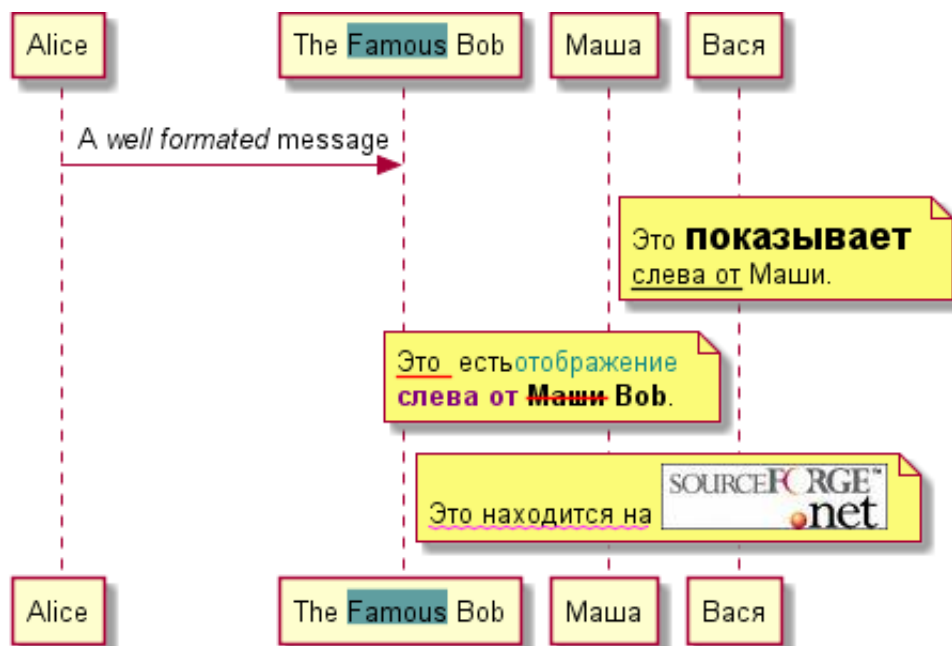
Так же можно использовать некоторые html теги :

- `<b>` для жирного текста
- `<u>` или `<u: >` для подчеркивания
- `<i>` для наклонного
- `<s>` или `<s: >` для зачеркнутого текста
- `<w>` или `<w:#AAAAAA>` или `<w: >` для текста подчеркнутого волнистой линией
- `<color:#AAAAAA>` или `<color:colorName>`
- `<back:#AAAAAA>` или `<back:colorName>` для цвета фона
- `<size:nn>` изменить размер шрифта
- `<img:file>` : файл должен быть доступен в файловой системе.

```
@startuml
participant Alice
participant "The <back:cadetblue>Famous</back> Bob" as Bob

Alice -> Bob : A <i>well formatted</i> message
note right of МашаЭто
  <b><size показывает:18></size></back>
  <услева> от</u> Маши.
end note
note left of Вася
  <u:red Это> </u> есть<colorотображение:#118888></color>
  <b><color:purple слева> от</color> <s:red Маши></s> Bob</b>.
end note
note over Маша, Вася
  <w:#FF33FFЭто> находитсяна </w> <img:img/sourceforge.jpg>
end note

@enduml
```



### 1.13 Divider

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

```
@startuml
```

```
== Инициализация==
```

```
Alice -> Bob: Запросаутентификации
```

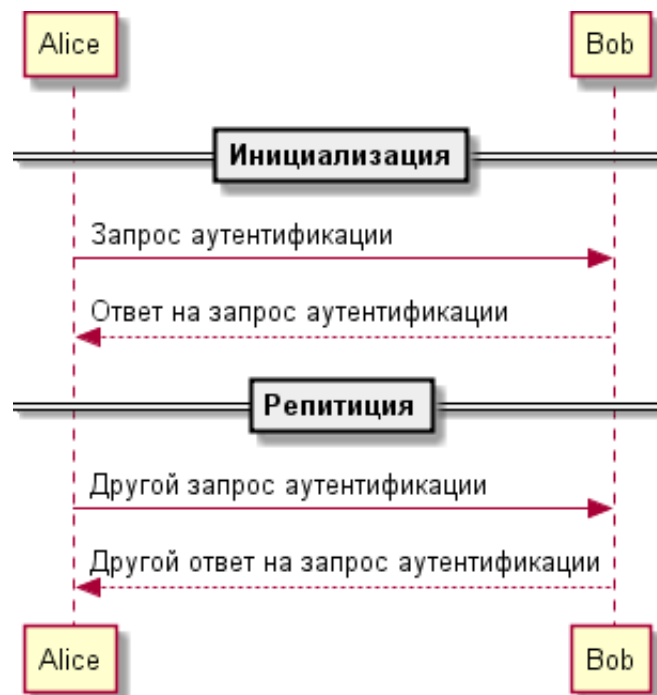
```
Bob --> Alice: Ответназапросаутентификации
```

```
== Репитиция==
```

```
Alice -> Bob: Другойзапросаутентификации
```

```
Alice <-- Bob: Другойответназапросаутентификации
```

```
@enduml
```



## 1.14 Строка активации и деактивации

The `activate` and `deactivate` are used to denote participant activation.

Once a participant is activated, its lifeline appears.

The `activate` and `deactivate` apply on the previous message.

The `destroy` denote the end of the lifeline of a participant.

```
@startuml
participant Пользователь
Пользователь
-> A: Выполнить
activate A

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

B -> C: Выполнить активация
C

B --> A: Запрос Создан деактивация
B

A -> Пользователь: Выполнено деактивация
A

@enduml
```

```
... (skipping 7 lines) ...
activate B

B -> C: Выполнить
активация C
Syntax Error?
```



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

```
@startuml
participant User
```

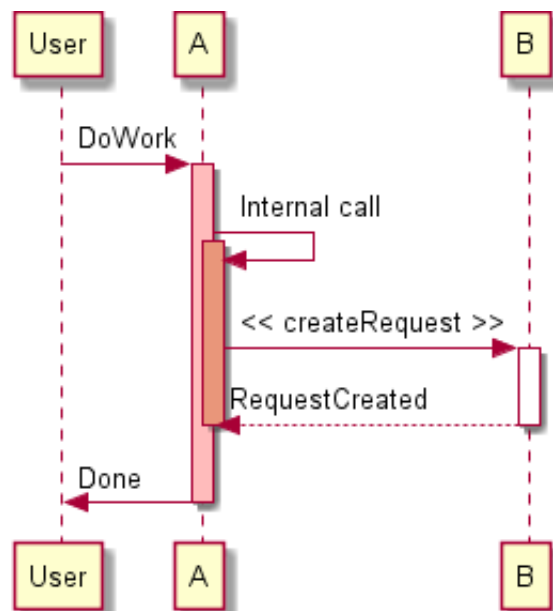
```
User -> A: DoWork
activate A #FFBBBB
```

```
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
```



## 1.15 Participant creation

You can use the `create` keyword just before the first reception of a message to emphasize the fact that this message is actually *creating* this new object.

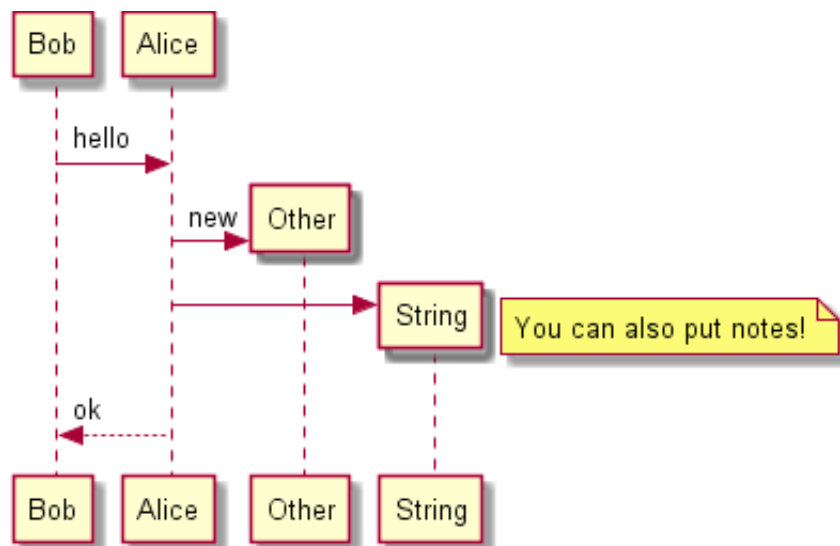
```
@startuml
Bob ->> Alice : hello

create Other
Alice ->> Other : new

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

Alice -->> Bob : ok

@enduml
```

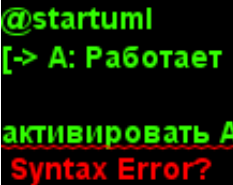


## 1.16 Incoming and outgoing messages

You can use incoming or outgoing arrows if you want to focus on a part of the diagram.

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

```
@startuml
[-> A: Работаетактивировать
A
A -> A: Внутреннийвызовактивировать
A
A ->] : << createRequest >>
A<--] : RequestCreated
deactivate A
[<- A: Done
deactivate A
@enduml
```



## 1.17 Stereotypes and Spots

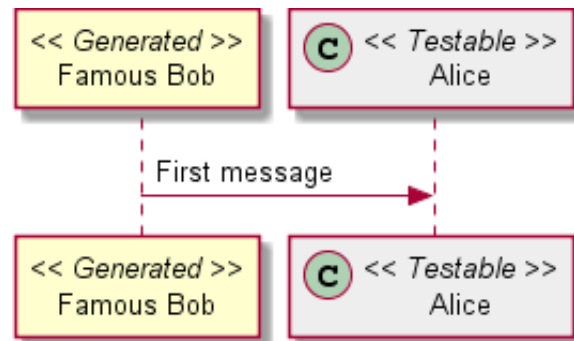
It is possible to add stereotypes to participants using "<<" and ">>". In the stereotype, you can add a spotted character in a colored circle using the syntax "(X,color)".

```
@startuml
```

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

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

```
@enduml
```

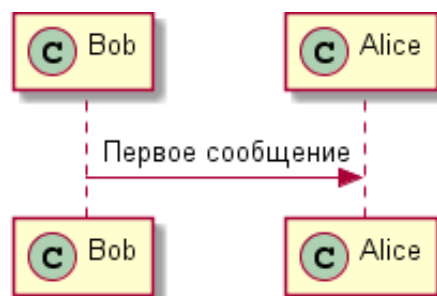


```
@startuml
```

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

```
Bob->>Alice: Первое сообщение
```

```
@enduml
```



## 1.18 More information on titles

В заголовке можно использовать некоторые HTML теги.

```
@startumlЗаголовок
```

```
<u>простого</u> примеравзаимодействияАлиса
```

```
-> Боб: ЗапросаутентификацииБоб
```

```
—> Алиса: Ответаутентификации
```

```
@enduml
```

```
@startuml
Заголовок <u>простого</u> примера взаимодействия
Syntax Error?
```

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

```
@startumlПримерзаголовка
```

```
<u>Simple</u> communication example\nна несколькихстроках
```

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

```
Bob —> Alice: Authentication Response
```

```
@enduml
```

```
@startuml
Пример заголовка <u>Simple</u> communication example\nна нескольких строках
Syntax Error?
```

You can also define title on several lines using `title` and `end title` ключевые слова.

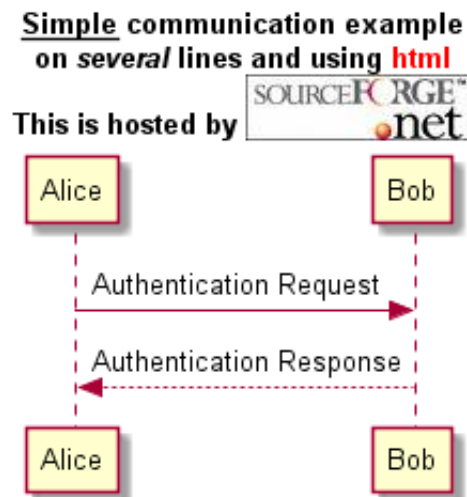
```
@startuml
```

```
title
```

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

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

```
@enduml
```



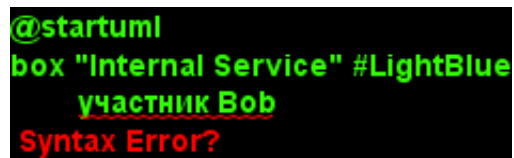
## 1.19 Participants englobed

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 участник
    Bob
    Alice
end box участник
    Other
```

```
Bob -> Алиса : привет
Alice -> Other : привет
@enduml
```



```
@startuml
box "Internal Service" #LightBlue
    участник Bob
Syntax Error?
```

## 1.20 Удаление футера

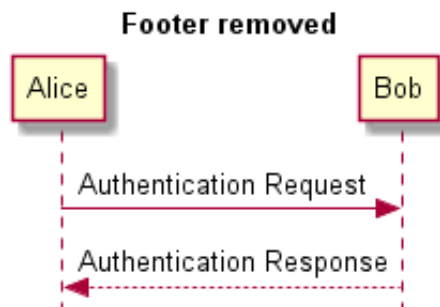
Вы можете использовать ключевое слово `hide footbox` для удаления футера из диаграммы.

```
@startuml
```

```
hide footbox
title Footer removed
```

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

```
@enduml
```





## 1.21 Skinparam

Вы можете использовать команду `skinparam` для изменения цвета и шрифта при отрисовке. You can use this command :

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

```
@startuml
skinparam backgroundColor #EEEEBD

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

    ParticipantBorderColor DeepSkyBlue
    ParticipantBackgroundColor DodgerBlue
    ParticipantFontName Impact
    ParticipantFontSize 17
    ParticipantFontColor #A9DCDF

    ActorBackgroundColor aqua
    ActorFontColor DeepSkyBlue
    ActorFontSize 17
    ActorFontName Apex
}

actor User
participant "First Class" as A
participant "Second Class" as B
participant "Last Class" as C
User -> A: DoWork
activate A
A -> B: Create Request
activate B
B -> C: DoWork
activate C
C --> B: WorkDoneдеактивация
C
B --> A: Request Createdдеактивация
B
A --> User: Doneдеактивация
A
@enduml
```

```
... (skipping 29 lines) ...
B -> C: DoWork
activate C
C --> B: WorkDone
деактивация C
Syntax Error?
```

## 1.22 Skin

Используйте ключевое слово `skin` для изменения представления generated diagram. На данный момент доступно только две шкурки (*Rose*, используемая по умолчанию, и *BlueModern*), но Вы можете создать свою собственную шкурку.

```
@startuml
skin BlueModern

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

User -> A: DoWorkактивация
A
A -> B: << createRequest >>активация
B
B -> C: DoWorkактивация
C
C -> B: WorkDoneдеактивация
C
B -> A: Request <u>Created </u>деактивация
B
A -> User: Выполненодеактивация
A

@enduml
```

```
... (skipping 6 lines) ...
participant "Last Class" as C

User -> A: DoWork
активация A
Syntax Error?
```

## 2 Диаграмма прецедентов

### 2.1 Прецеденты

Use cases are enclosed using between parentheses (because two parentheses выглядит 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

```
@startumlПервый
```

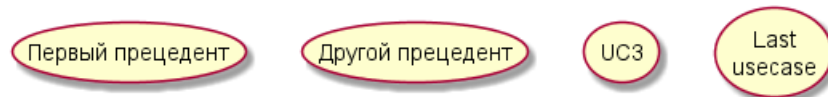
```
( прецедент)Другой
```

```
( прецедент) as (UC2)
```

```
usecase UC3
```

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

```
@enduml
```



## 2.2 Актёры

Актёры обозначаются заданием значения между двумя двоеточиями.

Также Вы можете использовать ключевое слово `actor` для определения актёра.

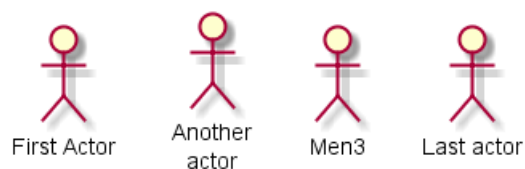
И вы можете создать псевдоним, используя ключевое слово `as`. Этот псевдоним будет использован позднее, при определении отношений.

Мы увидим, что определения `Actor` не обязательны.

```
@startuml
```

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

```
@enduml
```



## 2.3 Основной пример

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

The more dashes "-" in the arrow, the longer the arrow.

You can add a label on the arrow, by adding a ":" character in the arrow definition.

In this example, you see that *User* has not been defined before, and is implicitly defined as an actor.

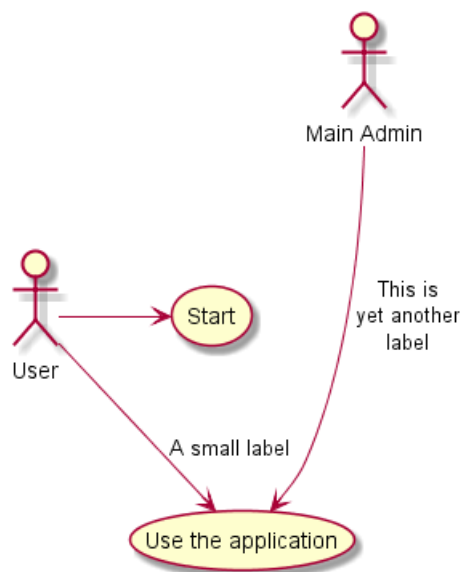
```
@startuml
```

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


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


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

```
@enduml
```



## 2.4 Расширение

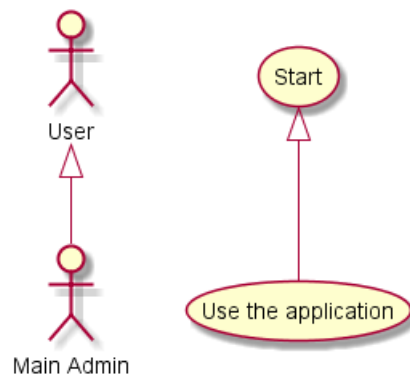
If one actor/use case extends another one, you can use the symbol <|-- (which stands for .

As for smiley, when you turn your head, you will see the symbol 

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

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

```
@enduml
```



## 2.5 Использование заметок

Вы можете использовать:

- note left of,
- note right of,
- note top of,
- note bottom of

keywords to define notes related to a single object. A note can be also define alone with the note keywords, then linked to other objects using the .. symbol.

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

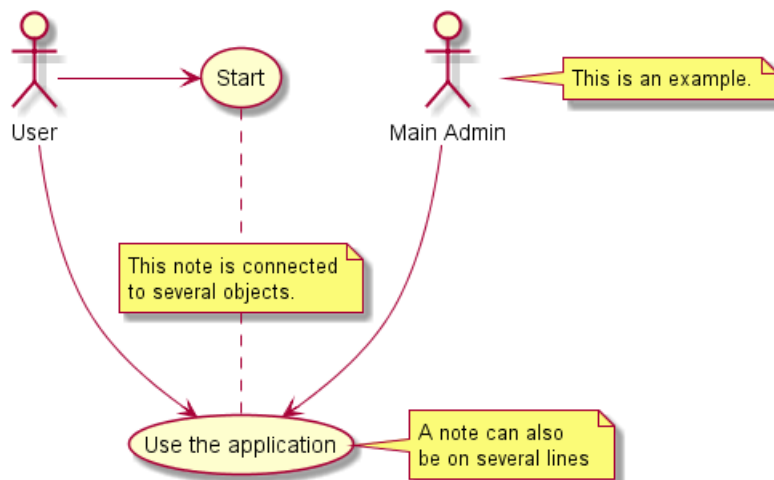
```
User -> (Start)
User --> (Use)
```

```
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
```



## 2.6 Stereotypes

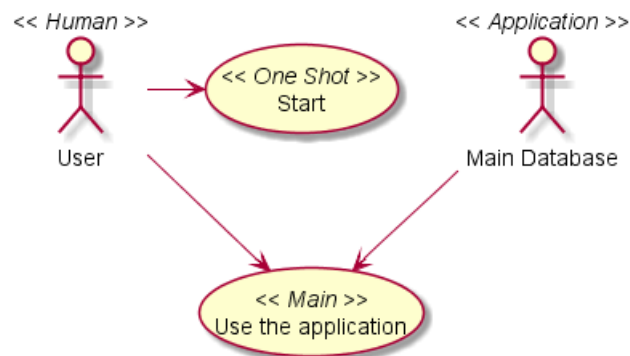
You can add stereotypes while defining actors and use cases using "<<" and ">>".

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

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

MySql -> (Use)

@enduml
```

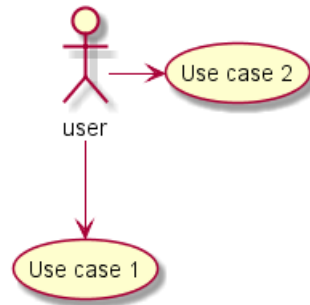




## 2.7 Changing arrows direction

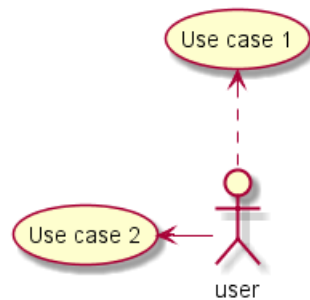
By default, links between classes have two dashes -- and are vertically oriented. It is possible to use horizontal link by putting a single dash (or dot) like this:

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



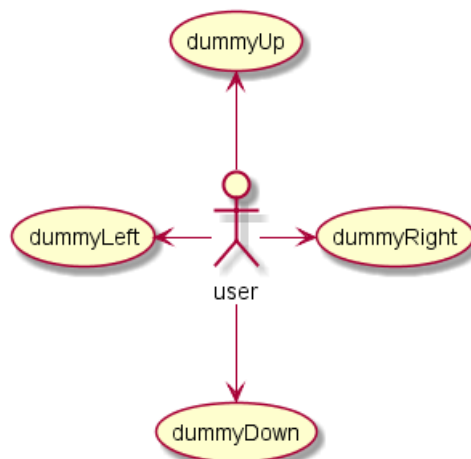
You can also change directions by reversing the link:

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



It is also possible to change arrow direction by adding left, right, up or down keywords inside the arrow:

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



You can shorten the arrow by using only the first character of the direction Вы можете записать короче, используя только первый символ названия направления (например, -d- вместо -down-) или первые два символа (-do-).

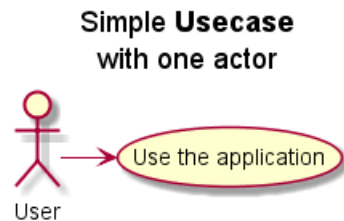
Пожалуйста, помните, что Вы не должны использовать эту функциональность без реальной необходимости: *GraphViz* обычно даёт хороший результат без дополнительных настроек.

## 2.8 Title the diagram

The `title` keywords is used to put a title.

You can use `title` and `end title` keywords for a longer title, as in sequence diagrams.

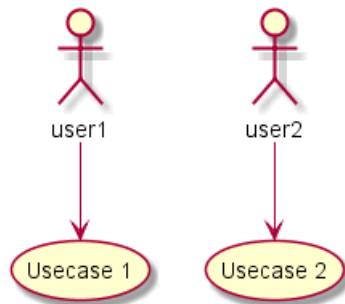
```
@startuml
title Simple <b>Usecase</b>\nwith one actor
usecase (Use the application) as (Use)
User -> (Use)
@enduml
```



## 2.9 Направление слева направо

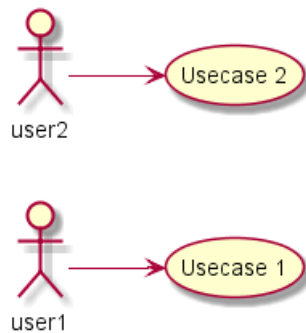
Общее поведение по умолчанию - построение диаграмм сверху вниз.

```
@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
```



## 2.10 Skinparam

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

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

```
@startuml
skinparam usecase {
    BackgroundColor DarkSeaGreen
    BorderColor DarkSlateGray

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

    ArrowColor Olive
    ActorBorderColor black
    ActorFontName Courier

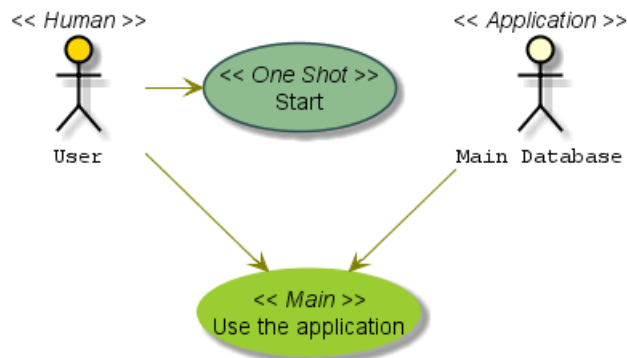
    ActorBackgroundColor<< Human >> Gold
}

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

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

MySql --> ( Use )




@enduml
```



### 3 Диаграмма классов

#### 3.1 Отношения между классами

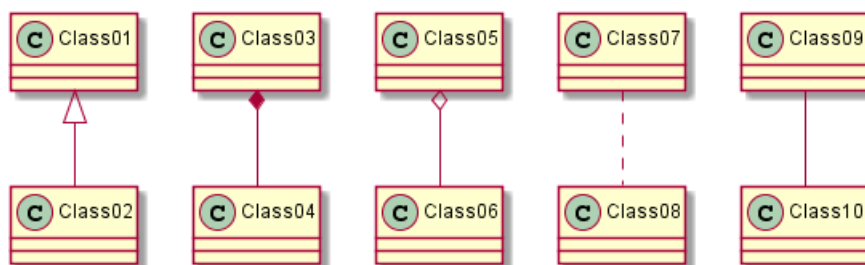
Отношения между классами определяется с помощью следующих символов:

расширение	< --	
состав	*--	
агрегирования	o--	

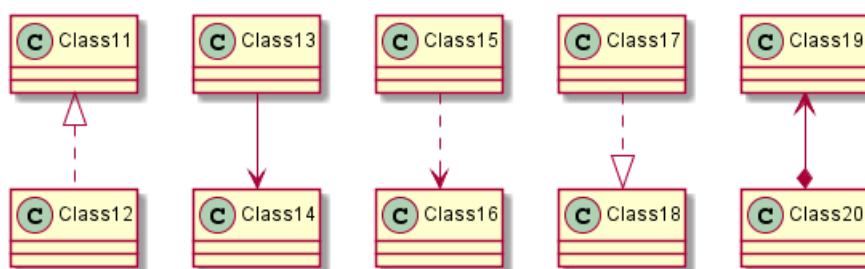
Можно заменить "texttt -"на " чтобы иметь пунктирную line.

Knowing thoses rules, it is possible to draw the following drawings:

```
@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
```



### 3.2 Label on relations

It is possible to add a label on the relation, using ":" followed by the text of the label.

For cardinality, you can use double-quotes on each side of the relation.

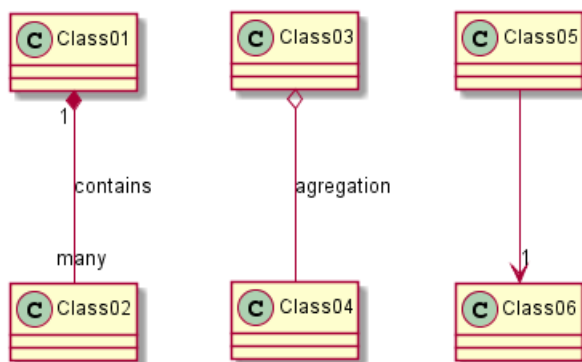
```
@startuml
```

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

```
Class03 o— Class04 : agregation
```

```
Class05 —> "1" Class06
```

```
@enduml
```



### 3.3 Adding methods

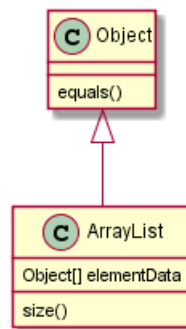
To declare fields and methods, you can use the symbol ":" followed by the field's or method's name.

The system checks for parenthesis to choose between methods and fields.

```
@startuml
Object <|-- ArrayList

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

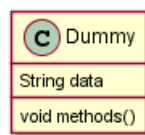
@enduml
```



It is also possible to group between brackets {} all fields and methods.

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

@enduml
```



### 3.4 Defining visibility

When you define methods or fields, you can use characters to define the visibility of the corresponding item:

-	□	■	private
#	◇	◆	protected
~	△	▲	package private
+	○	●	public

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



You can turn off this feature using the `skinparam classAttributeIconSize 0` command :

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





### 3.5 Notes and stereotypes

Stereotypes are defined with the class keyword, "<<" and ">>".

You can also define notes using note left of, note right of, note top of, note bottom of keywords.

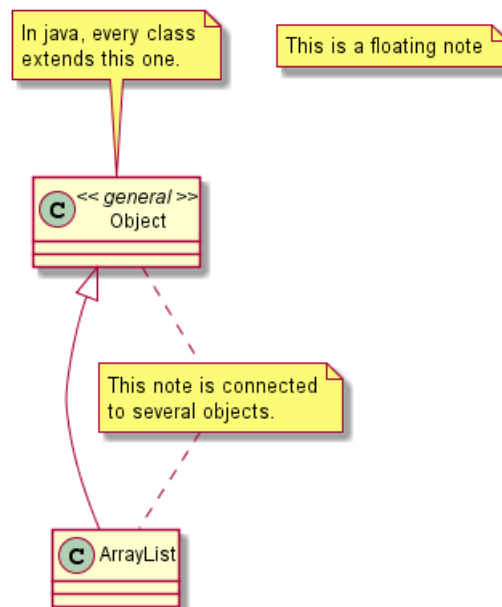
A note can be also define alone with the note keywords, then linked to other objects using the ".." symbol.

```
@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
```

```
@enduml
```



### 3.6 More on notes

Также допускается использование некоторых HTML-тегов, таких как:

- `<b>`
- `<u>`
- `<i>`
- `<s>`, `<del>`, `<strike>`
- `<font color="#AAAAAA">` или `<font color="colorName">`
- `<color:#AAAAAA>` или `<color:colorName>`
- `<size:nn>` задать размер шрифта
- `` или `<img:file>` : файл должен быть доступен файловой системой

You can also have a note on several lines.

```
@startuml
```

```
note top of Object
```

```
  In java , every class
```

```
  extends
```

```
  this one.
```

```
end note
```

```
note as N1
```

```
  This <size:10>note</size> is also
```

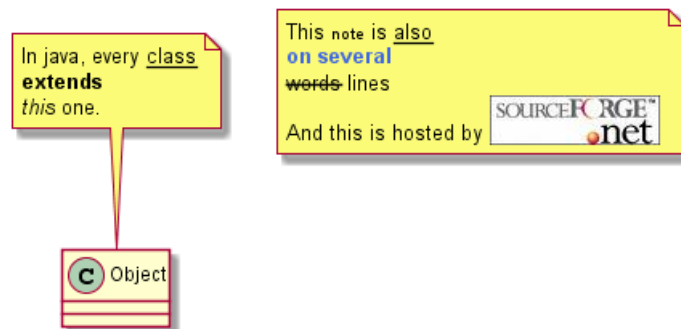
```
  <color:royalBlue>on several</color>
```

```
  words lines
```

```
  And this is hosted by <img:img/sourceforge.jpg>
```

```
end note
```

```
@enduml
```



### 3.7 Abstract class and interface

You can declare a class as abstract using "abstract" or "abstract class" keywords. The class will be printed in italic.

You can use the interface and enum keywords too.

@startuml

```
abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection
```

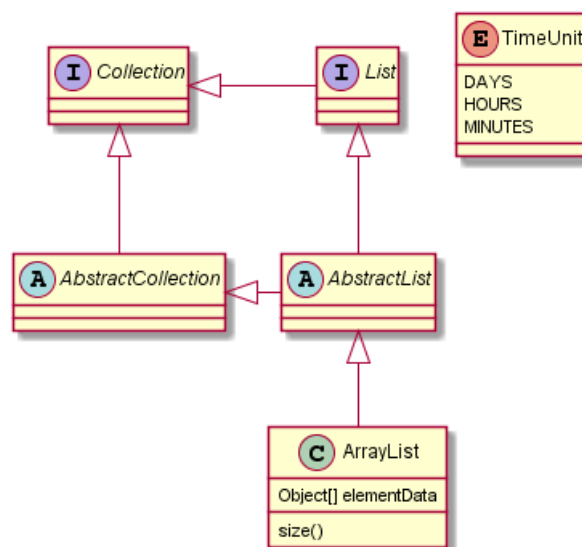
```
List <|-- AbstractList
Collection <|-- AbstractCollection
```

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

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

```
enum TimeUnit
TimeUnit : DAYS
TimeUnit : HOURS
TimeUnit : MINUTES
```

@enduml



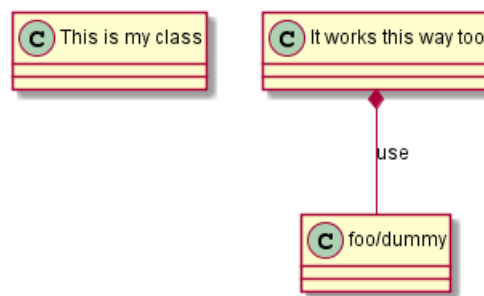
### 3.8 Using non-letters

If you want to use non-letters in the class (or enum...) display, you can either :

- Use the `as` keyword in the class definition
- Поставьте 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
```



### 3.9 методы скрытия атрибут...

You can parameterize the display of classes using the `hide/show` command.

The basic command is: `hide empty members`. Эта команда была скрыта атрибуты или методы, если они пусты

Вместо `empty members`, вы можете использовать:

- `empty fields or 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` for the circled character in front of class name,
- `stereotype` for the stereotype.

You can also provide, just after the `hide` or `show` keyword:

- `class` для всех классов,
- `interface` для всех интерфейсов,
- `enum` for all enums,
- `<<foo1>>` for classes which are stereotyped with *foo1*,
- существующее имя класса.

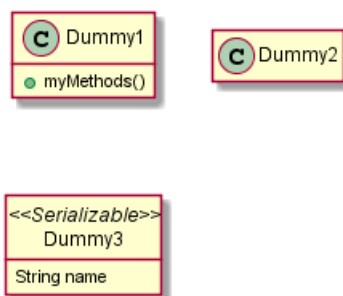
You can use several `show/hide` commands to define rules and exceptions.

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

class Dummy2 {
+hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

hide members
hide <<Serializable>> circle
show Dummy1 method
show <<Serializable>> fields
@enduml
```



### 3.10 Specific Spot

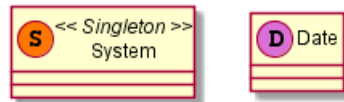
Usually, a spotted character (C, I, E or A) is used for classes, interface, enum and abstract classes. But you can define your own spot for a class when you define the stereotype, adding a single character и цвет как в этом примере:

```
@startuml
```

```
class System << (S,#FF7700) Singleton >>
```

```
class Date << (D,orchid) >>
```

```
@enduml
```



### 3.11 Пакеты

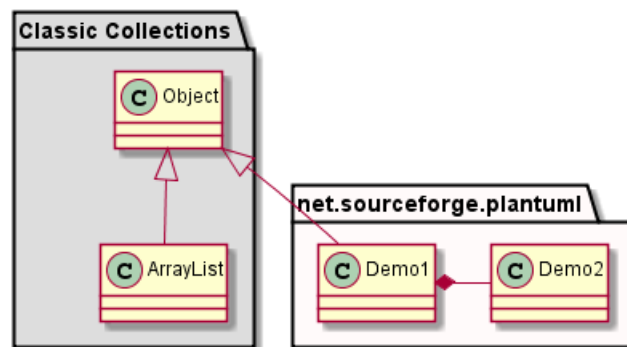
Вы можете определить пакет используя кодовое слово `package`, и если хотите, определите цвет фона своего пакета (Используя код цвета html или его имя). После определения класса, он автоматически помещается в последний использованный вами пакет. а также вы можете закрыть определение пакета, используя ключевое слово `end package`. Вы также можете использовать фигурные скобки `{ }`.

Обратите внимание, что определения пакета могут быть вложенными.

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

package net.sourceforge.plantuml #Snow
    Object <|-- Demo1
    Demo1 *-- Demo2
end package

@enduml
```



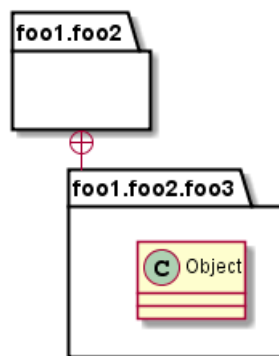
You can also define links between packages, like in the following example:

```
@startuml
package foo1.foo2
end package

package foo1.foo2.foo3 {
    class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



### 3.12 Пространства имен

В блоках, 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.

You can refer to classes from other namespaces by fully qualify them. Classes from the default namespace are qualified with a starting dot.

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
```

```
end namespace
```

```
namespace net.foo {
```

```
  net.dummy.Person <|-- Person
```

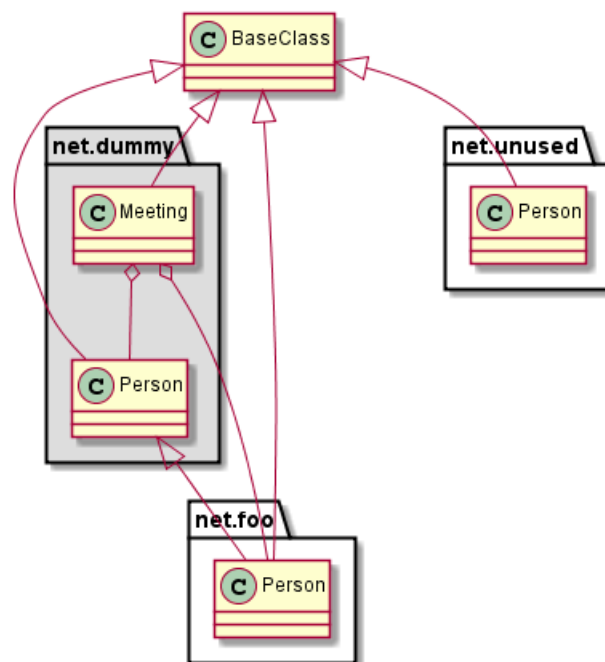
```
  .BaseClass <|-- Person
```

```
  net.dummy.Meeting o-- Person
```

```
}
```

```
BaseClass <|-- net.unused.Person
```

@enduml

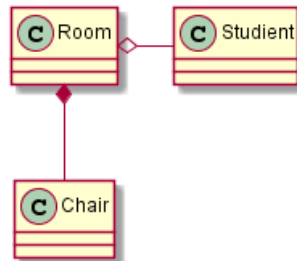




### 3.13 Изменение направления стрелок

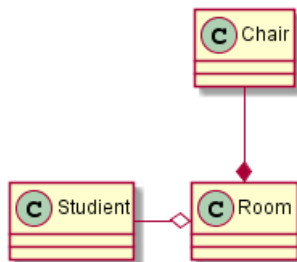
By default, links between classes have two dashes -- and are vertically oriented. It is possible to use horizontal link by putting a single dash (or dot) like this:

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



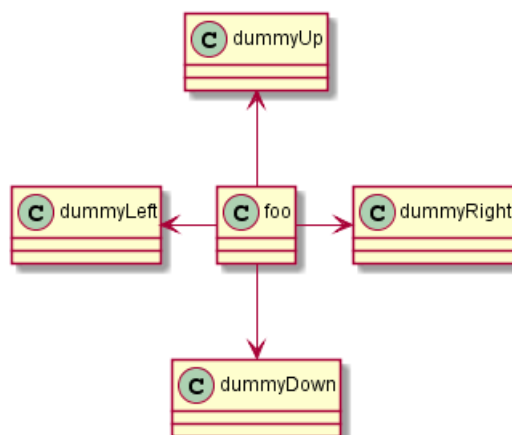
You can also change directions by reversing the link:

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



Также возможно изменять направление стрелок, добавляя left, right, up or down keywords inside the arrow:

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



You can shorten the arrow by using only the first character of the direction (for example, -d- instead of -down-) or the two first characters (-do-)

Please note that you should not abuse this fonctionnality : *GraphViz* gives usually good results without tweaking.

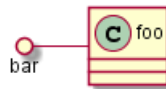


### 3.14 Lollipop интерфейс

You can also define lollipops interface on classes, using the following syntax:

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

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

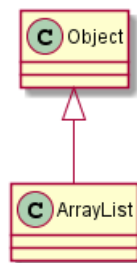


### 3.15 Заголовок диаграммы

`title` используется для задания заголовков. You can use `title` and `end title` keywords for a longer title, as in sequence diagrams.

```
@startuml
title Simple <b>example</b>\nof title
Object <|-- ArrayList
@enduml
```

**Simple example  
of title**

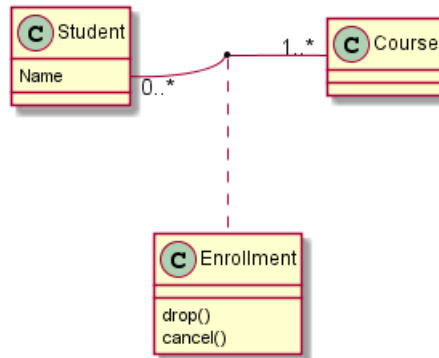


### 3.16 Ассоциация классов

You can define *association class* after that a relation has been defined between two classes, like in this example:

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

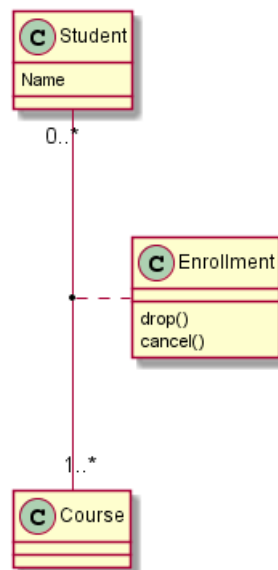
Enrollment : drop()
Enrollment : cancel()
@enduml
```



You can define it in another direction:

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

Enrollment : drop()
Enrollment : cancel()
@enduml
```



### 3.17 Skinparam

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

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

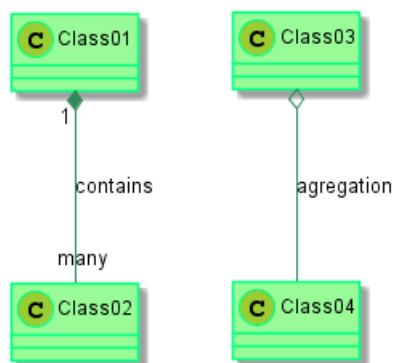
```
@startuml
```

```
skinparam classBackgroundColor PaleGreen
skinparam classArrowColor SeaGreen
skinparam classBorderColor SpringGreen
skinparam stereotypeCBackgroundColor YellowGreen
```

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

```
Class03 o— Class04 : agregation
```

```
@enduml
```



### 3.18 Skinned Stereotypes

You can define specific color and fonts for stereotyped classes.

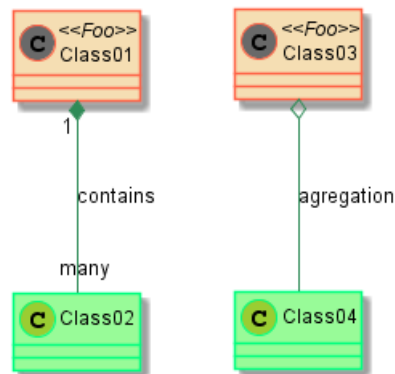
@startuml

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

```
Class01 <<Foo>>
Class01 "1" *— "many" Class02 : contains
```

```
Class03 <<Foo>> o— Class04 : agregation
```

@enduml



### 3.19 Splitting large files

Иногда могут получиться очень большие файлы изображений. Вы можете использовать команду "page (hpages)x(vpages)" чтобы разделить создаваемое изображение на несколько файлов (страниц) :

- *hpages* - число страниц по горизонтали ,
- *vpages* - число страниц по вертикали

```
@startuml
' Split into 4 pages
page 2x2

class BaseClass

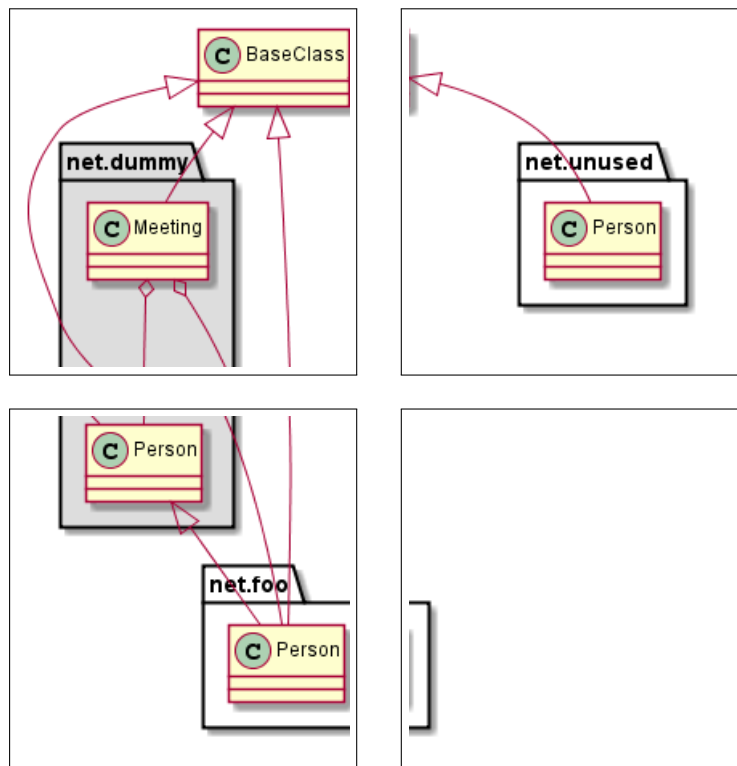
namespace net.dummy #DDDDDD
    BaseClass <|-- Person
    Meeting o-- Person

    BaseClass <|-- Meeting
end namespace

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.

Используйте --> для стрелок.

Example:

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

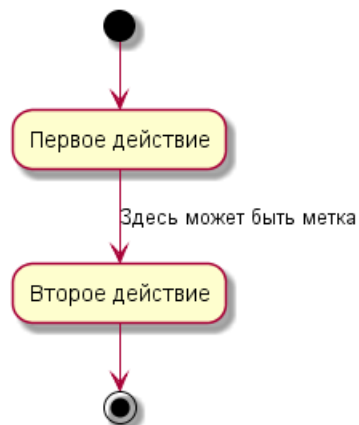


### 4.2 Метка на стрелках

По умолчанию, стрелка начинается с последней использованной активности.

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

```
@startuml
(*) --> "Первое" действие"Здесь
-->[ можетбытьметка ] Второе" действие"
--> (*)
@enduml
```

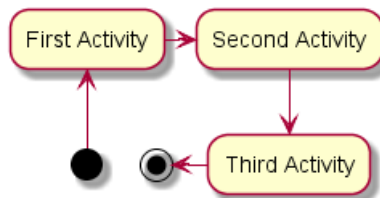


### 4.3 Изменение направления стрелки

Вы можете использовать `->` для горизонтальных стрелок. Если это возможно то для явного указания направления используйте следующий синтаксис:

- `-down->` (стрелка по умолчанию)
- `-right->` или `->`
- `-left->`
- `-up->`

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



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

Пожалуйста, обратите внимание, что вы не должны злоупотреблять этой функциональностью : *GraphViz* обычно дает хорошие результаты без настройки.

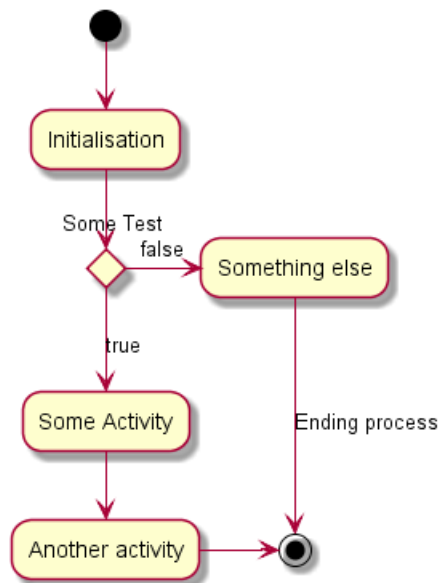


## 4.4 Ветвления

You can use if/then/else keywords to define branches.

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

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



## 4.5 More on Branches

By default, a branch is connected to the last defined activity, but it is possible to override this and to define a link with the `if` keywords.

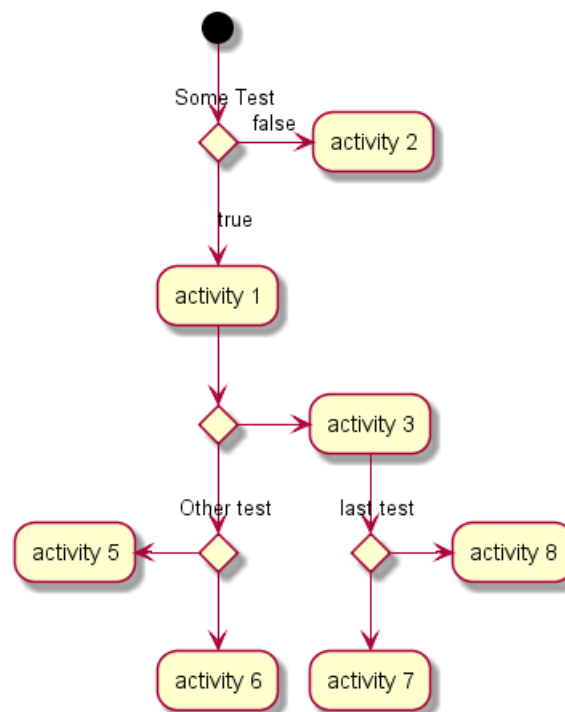
It is also possible to nest branches.

```
@startuml
(*) --> if "Some Test" then
    -->[true] "activity 1"

    if "" then
        -> "activity 3" as a3
    else
        if "Other test" then
            -left-> "activity 5"
        else
            --> "activity 6"
        endif
    endif
endif

else
    ->[false] "activity 2"
endif

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



## 4.6 Синхронизация

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

@startuml

(\*) --> ===B1===

--> "Параллельное" действие 1"

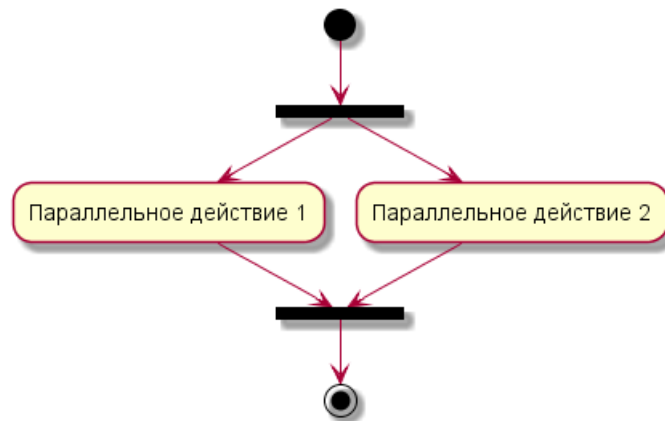
--> ===B2===

===B1=== --> "Параллельное" действие 2"

--> ===B2===

--> (\*)

@enduml



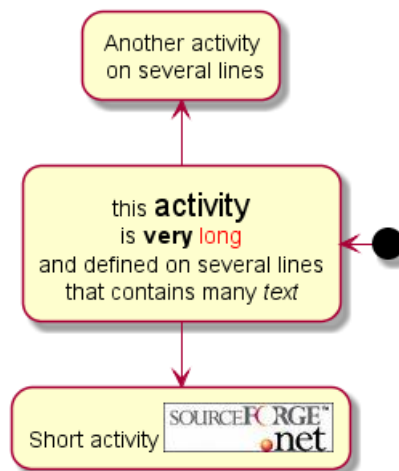
## 4.7 Long activity description

When you declare activities, you can span on several lines the description text. You can also add \n in the description. It is also possible to use few html tags like :

- `<b>`
- `<i>`
- `<font size="nn">` or `<size:nn>` to change font size
- `<font color="#AAAAAA">` or `<font color="colorName">`
- `<color:AAAAAA>` or `<color:colorName>`
- `<img:file.png>` to include an image

You can also give a short code to the activity with the `as` keyword. This code can be used latter in the diagram description.

```
@startuml
(*) -left-> "this <size:20>activity </size>
    is <b>very </b> <color:red>long </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:img/sourceforge.jpg>"
@enduml
```



## 4.8 Заметки

You can add notes on a activity using the commands:

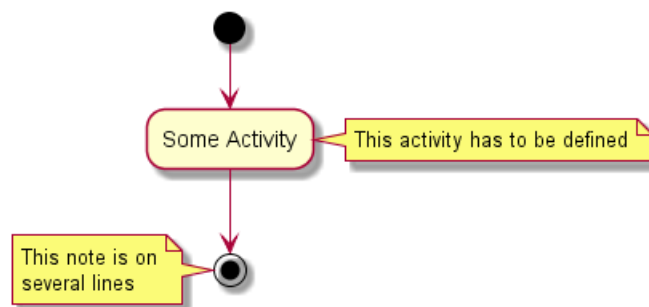
- `note left,`
- `note right,`
- `note top,`
- `note bottom,`

just after the description of the activity you want to note. If you want to put a note on the starting point, define the note at the very beginning of the diagram description.

You can also have a note on several lines, using the `end note` keywords.

```
@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 Partition

You can define a partition using the `partition` keyword, and optionally declare a background color for your partition (using a html color code or name).

When you declare activities, they are automatically put in the last used partition.

You can close the partition definition using the `end partition` keyword.

```
@startuml
```

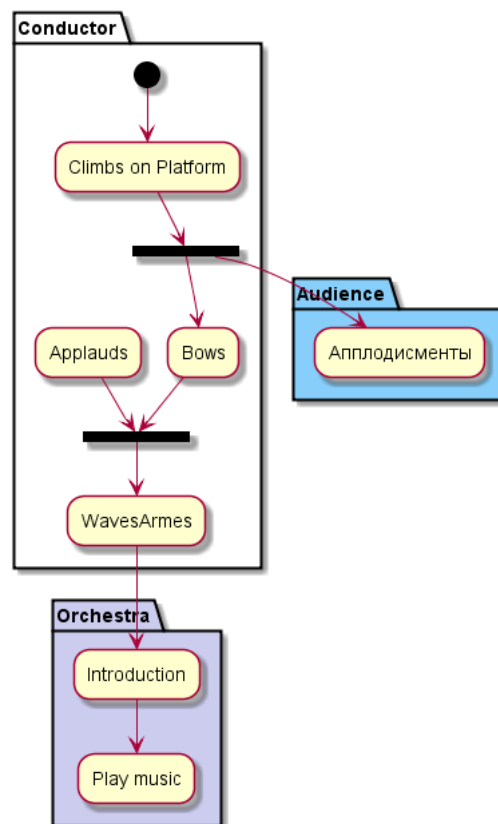
```
partition Conductor
(*) --> "Climbs on Platform"
--> === S1 ===
--> Bows
end partition
```

```
partition Audience #LightSkyBlue
=== S1 === --> Applaudissements
```

```
partition Conductor
Bows --> === S2 ===
--> WavesArmes
Applauds --> === S2 ===
end partition
```

```
partition Orchestra #CCCCEE
WavesArmes --> Introduction
--> "Play music"
end partition
```

```
@enduml
```



### 4.10 Заголовок диаграммы

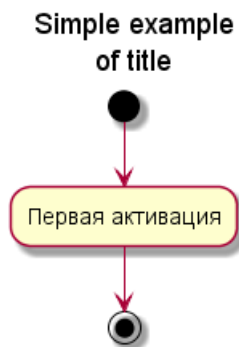
The `title` keywords is used to put a title. You can use `title` and `end title` keywords for a longer title, as in sequence diagrams.

```
@startuml
title Simple example\nof title
```

```
(*) --> Первая" активация"
```

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

```
@enduml
```



## 4.11 Skinparam

You can use the `skinparam` command to change colors and fonts for the drawing. You can use this command :

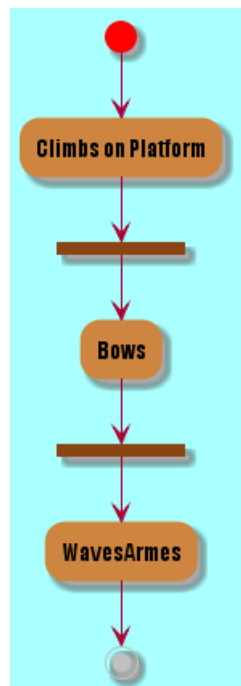
- In the diagram definition, like any other commands,
- In an included file,
- In a configuration file, provided in the command line or the ANT task.

@startuml

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

```
(*) --> "Climbs on Platform"
--> == S1 ==
--> Bows
--> == S2 ==
--> WavesArmes
--> (*)
```

@enduml





## 4.12 Complete example

```

@startuml
' http://click.sourceforge.net/images/activity-diagram-small.png
title Servlet Container

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

if "Page.onSecurityCheck" then
  -->[true] "Page.onInit()"

  if этоПервый?" then
    -->[no] Управление процессом"

    if продолжить" процесс?" then
      -->[yes] ===RENDERING===
    else
      -->[no] ===REDIRECT.CHECK===
    endif

  elseда
    -->[] ===RENDERING===
  endif

  if "is Post?" thenда
    -->[] "Page.onPost()"
    --> "Page.onRender()" as render
    --> ===REDIRECT.CHECK===
  elseнет
    -->[] "Page.onGet()"
    --> render
  endif

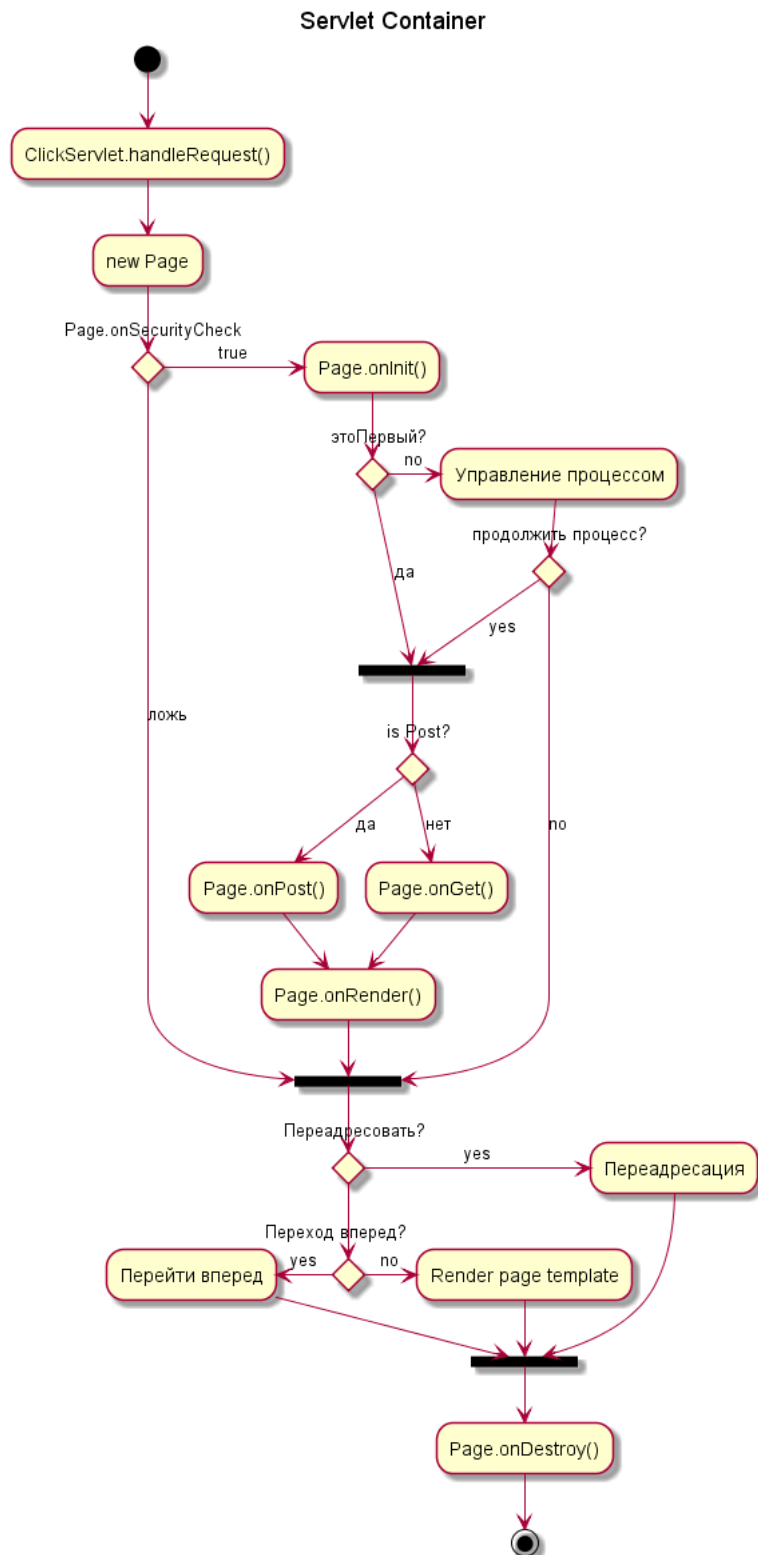
elseложь
  -->[] ===REDIRECT.CHECK===
endif

if Переадресовать?" then
  -->[yes] Переадресация""
  --> ==BEFORE_DESTROY==
else
  if Переход" вперед?" then
    -left-->[yes] Перейти" вперед"
    --> ==BEFORE_DESTROY==
  else
    -right-->[no] "Render page template"
    --> ==BEFORE_DESTROY==
  endif
endif

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

@enduml

```



## 5 Component Diagram

### 5.1 Компоненты

Components must be bracketed.

You can also use the `component` keyword to defines a component.

And you can define an alias, using the `as` keyword.

This alias will be used latter, when defining relations.

```
@startuml
```

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

```
@enduml
```



## 5.2 Interfaces

Interface can be defined using the "()" symbol (because this looks like a circle).

You can also use the `interface` keyword to defines an interface.

And you can define an alias, using the `as` keyword.

This alias will be used latter, when defining relations.

We will see latter that interface declaration is optional.

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



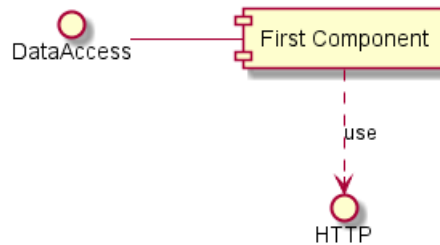
### 5.3 Basic example

Links between elements are made using combinations of dotted line "--", straight line "--" and arrows "-->" symbols.

```
@startuml
```

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

```
@enduml
```



## 5.4 Using notes

You can use the

- note left of,
- note right of,
- note top of,
- note bottom of,

keywords to define notes related to a single object.

A note can also be defined alone with the `note` keywords, then linked to other objects using the `..` symbol.

```
@startuml
```

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

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

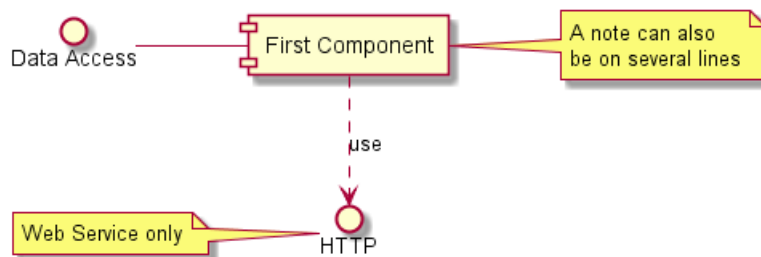
```
[ First Component] ..> HTTP : use
```

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

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

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

```
@enduml
```



## 5.5 Grouping Components

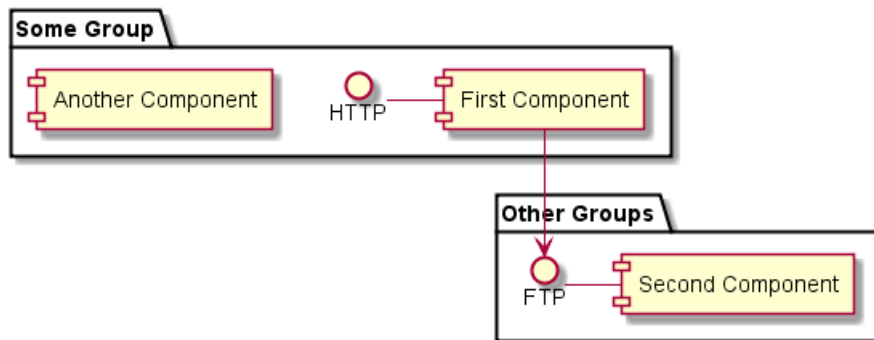
Вы можете использовать ключевое слово `package` чтобы сгруппировать компоненты и интерфейсы вместе.

@startuml

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

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

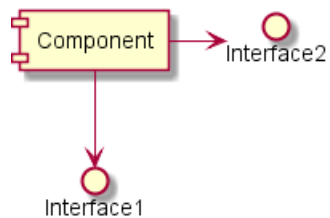
@enduml



## 5.6 Изменение направления связи

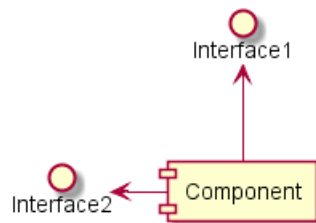
By default, links between classes have two dashes -- and are vertically oriented. It is possible to use horizontal link by putting a single dash (or dot) like this:

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



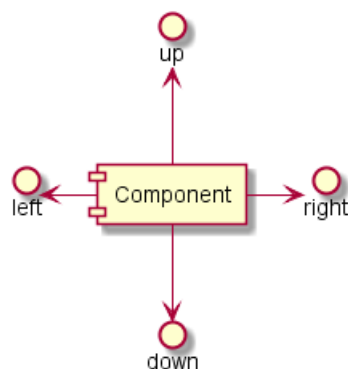
You can also change directions by reversing the link:

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



It is also possible to change arrow direction by adding left, right, up or down keywords inside the arrow:

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



You can shorten the arrow by using only the first character of the direction (for example, -d- instead of -down-) or the two first characters (-do-).

Please note that you should not abuse this functionality : *GraphViz* gives usually good results without tweaking.





## 5.7 Title the diagram

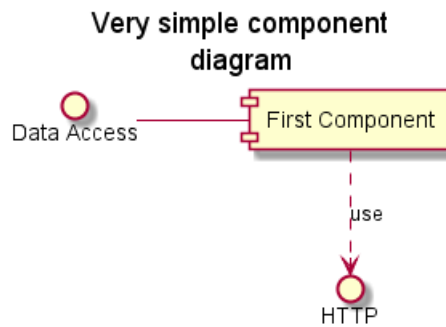
The `title` keywords is used to put a title. You can use `title` and `end title` keywords for a longer title, as in sequence diagrams.

```
@startuml
title Very simple component\ndiagram

interface "Data Access" as DA

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

@enduml
```



## 5.8 Skinparam

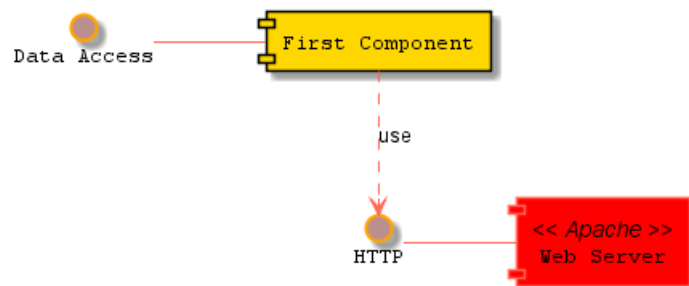
Чтобы изменить цвета и шрифт, используйте команду `skinparam`. Также вы можете применить эту команду :

- В определении диаграммы, как и любые другие команды,
- В вспомогательном (include) файле,
- В конфигурационном файле (указывается в командной строке или как параметр для задачи ANT).

```
@startuml
skinparam component {
  FontSize 13
  InterfaceBackgroundColor RosyBrown
  InterfaceBorderColor orange
  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
```



## 6 Диаграмма состояний

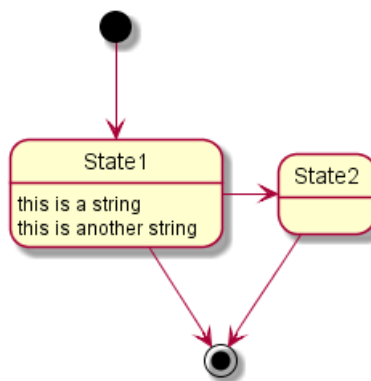
### 6.1 Простое состояние

Для изображения начального и конечного псевдосостояний используется [\*].

Пишите --> для изображения переходов.

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

State1 --> State2
State2 --> [*]
@enduml
```



## 6.2 Составное состояние

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

```
@startuml
```

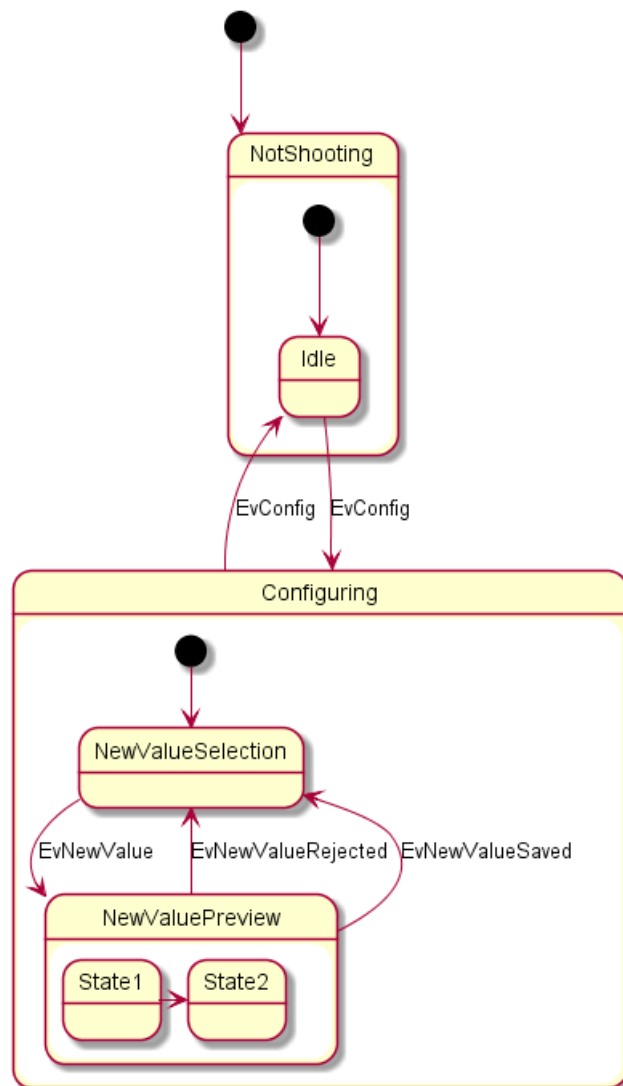
```
[*] --> NotShooting
```

```
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
```



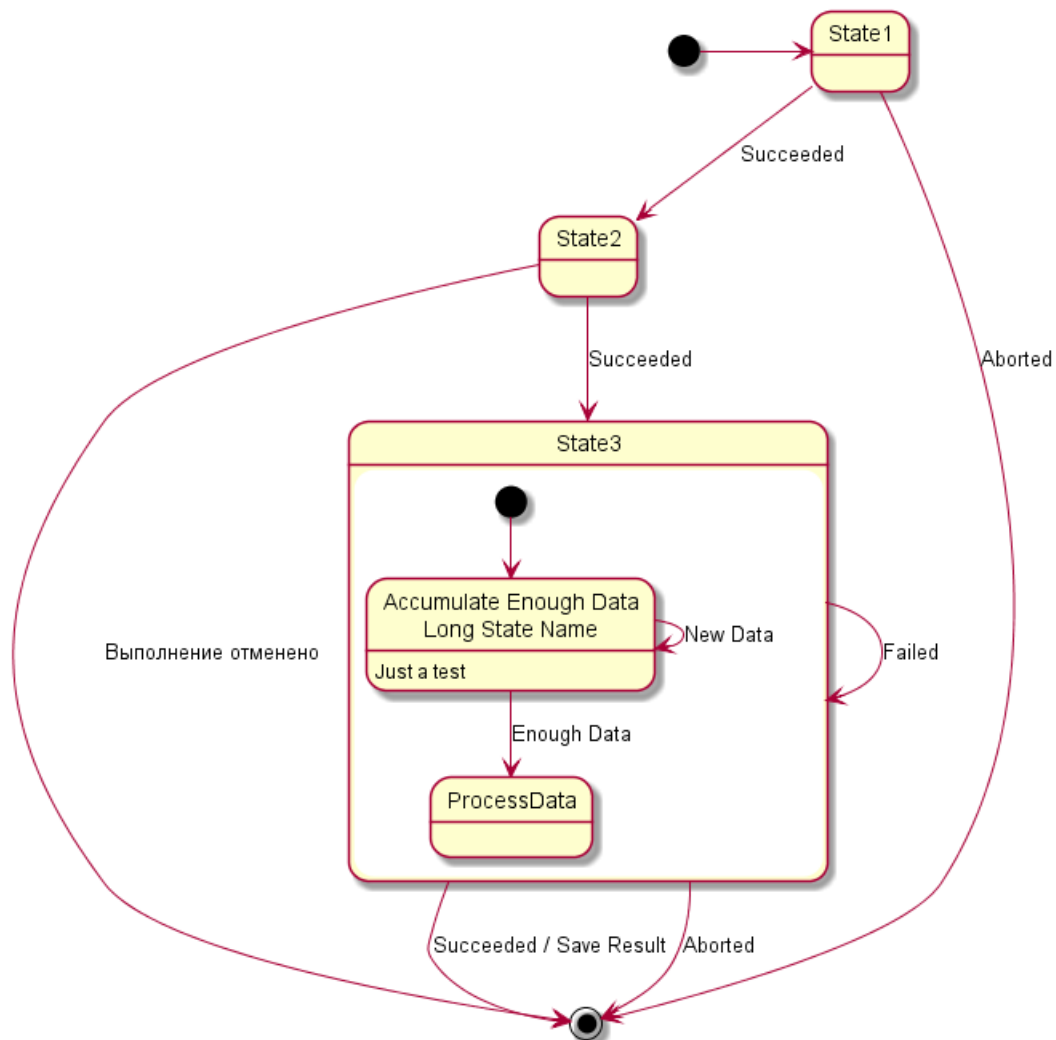
### 6.3 Длинные имена

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

@startuml

```
[*] -> State1
State1 -> State2 : Succeeded
State1 -> [*] : Aborted
State2 -> State3 : Succeeded
State2 -> [*] : Выполнениеотменено
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



## 6.4 Параллельные состояния

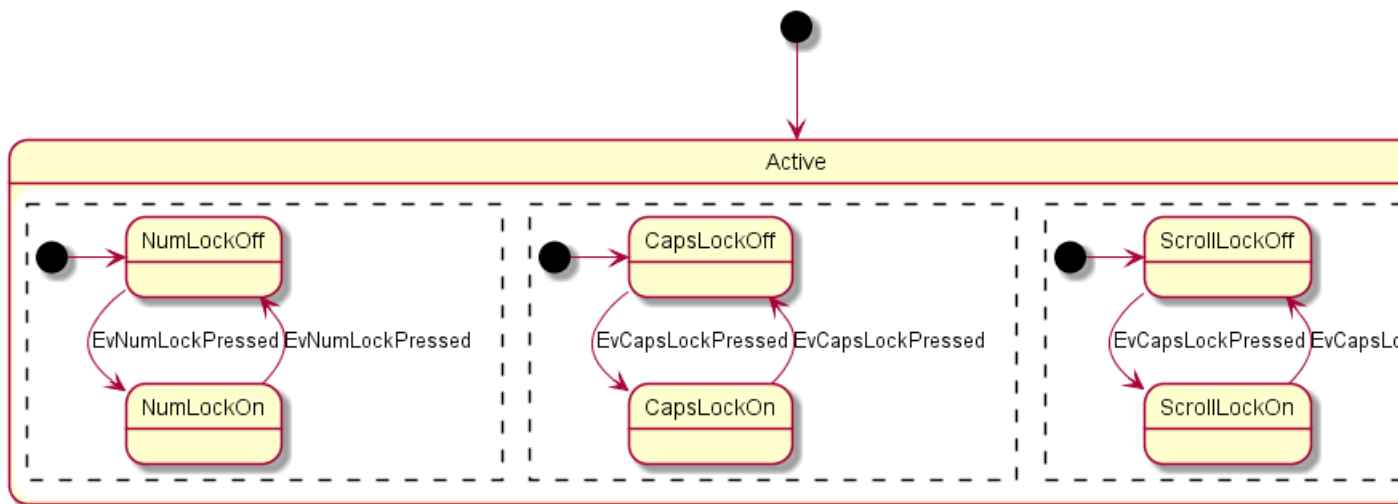
Используя оператор "-- вы можете объявлять параллельные подсостояния внутри составного состояния.

```
@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
```



## 6.5 Направления стрелок

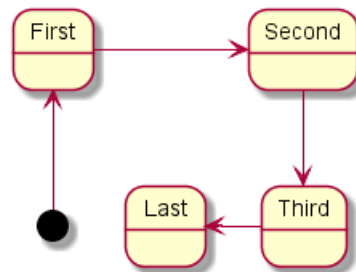
Для изображения стрелок перехода горизонтально используется оператор `->`. Следующий синтаксис позволяет задать другое направление.

- `-down->` или `-->`
- `-right->` или `->`
- `-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.

## 6.6 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.

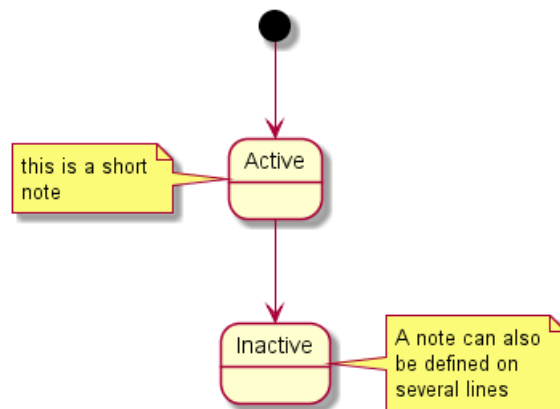
```
@startuml
```

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





## 6.7 Еще о заметках

Также заметки можно прикреплять к составным состояниям.

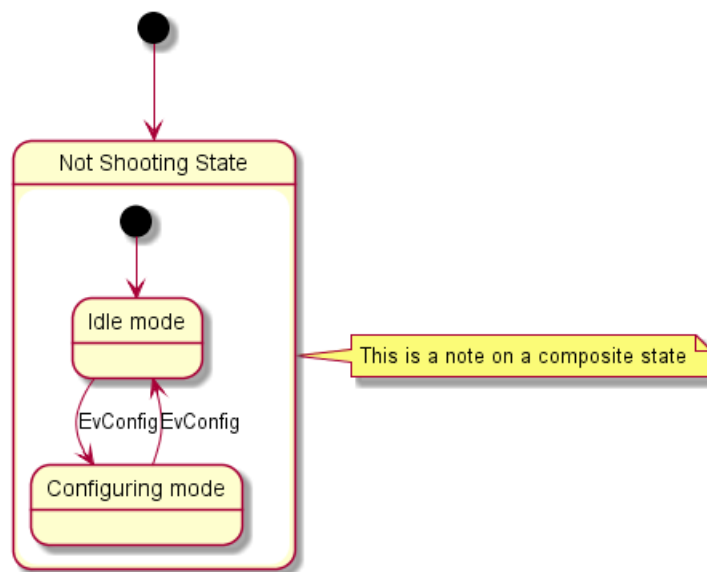
@startuml

[\*] --> 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 Диаграмма объектов

### 7.1 Определение объектов

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

```
@startuml
объект первыйОБъектСледующий
```

```
" объект" as o2
```

```
@enduml
```

```
@startuml
объект первыйОБъект
Syntax Error?
Did you mean:
объект- первыйОБъект
объектпервыйОБъект
```

### 7.2 Отношения между объектами

Отношения между объектами определяются с использованием следующих символов :

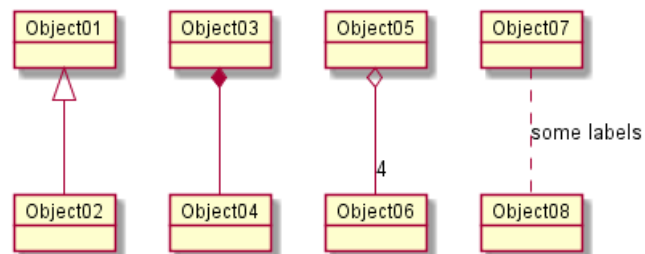
Extension	< --	
Composition	*--	
Agregation	o--	

It is possible to replace "--" by "." to have a dotted line.

Knowing thoses rules, it is possible to draw the following drawings:

```
@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
```



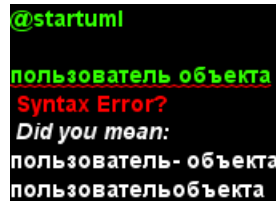
### 7.3 Добавление полей

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

@startumlпользовательобъекта

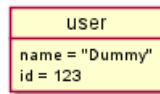
```
user : name = "Dummy"
user : id = 123
```

@enduml



It is also possible to ground between brackets {} all fields.

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



## 8 Common commands

### 8.1 Заголовок и подвал

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

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

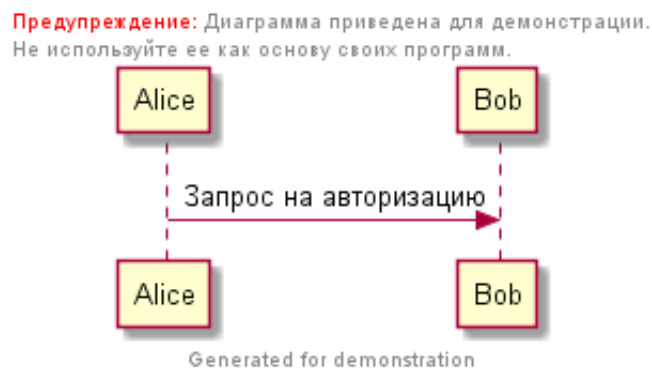
As for title, it is possible to define a header or a footer on several lines.

Также возможно поместить HTML-текст в заголовок или подвал.

```
@startuml
Alice -> Bob: Запрос на авторизацию

header
<font color=red Предупреждение>:</font> Диаграмма приведена для демонстрации . Не используйте ее как основу своих программ
endheader

center footer Generated for demonstration
@enduml
```

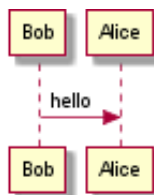


## 8.2 Масштаб

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

- scale 1.5,
- scale 2/3,
- scale 200 width,
- scale 200 height,
- scale 200\*100

```
@startuml
scale 180*90
Bob->Alice : hello
@enduml
```



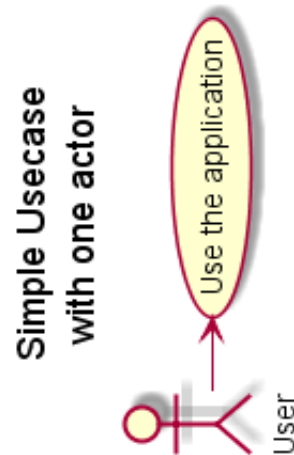
### 8.3 Вращения

Иногда, особенно для печати, вы можете захотеть повернуть сгенерированное изображение, чтобы оно лучше умещалось на странице. Для этого вы можете использовать команду `rotate`.

```
@startuml
rotate

title Simple Usecase\nwith one actor
"Use the application" as (Use)
User -> (Use)

@enduml
```



## 9 Изменить шрифты и цвета

### 9.1 Использование

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

```
skinparam backgroundColor yellow
```

Вы можете использовать команду :

- In the diagram definition, like any other commands,
- В подключаемом файле (см. *Preprocessing*),
- In a configuration file, provided in the command line or the ANT task.

### 9.2 Вложенный

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 эквивалент:

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



## 9.3 Color

You can use either standard color name or RGB code.

Parameter name	Default Value	Color	Comment
backgroundColor	white		Background of the page
activityArrowColor	#A80036		Color of arrows in activity diagrams
activityBackgroundColor	#FEFCE		Background of activities
activityBorderColor	#A80036		Color of activity borders
activityStartColor	black		Starting circle in activity diagrams
activityEndColor	black		Ending circle in activity diagrams
activityBarColor	black		Synchronization bar in activity diagrams
usecaseArrowColor	#A80036		Color of arrows in usecase diagrams
usecaseActorBackgroundColor	#FEFCE		Head's color of actor in usecase diagrams
usecaseActorBorderColor	#A80036		Color of actor borders in usecase diagrams
usecaseBackgroundColor	#FEFCE		Background of usecases
usecaseBorderColor	#A80036		Color of usecase borders in usecase diagrams
classArrowColor	#A80036		Color of arrows in class diagrams
classBackgroundColor	#FEFCE		Цвет фона классов/интерфейсов/перечислений в диаграммах классов
classBorderColor	#A80036		Цвет рамки классов/интерфейсов/перечислений в диаграммах классов
packageBackgroundColor	#FEFCE		Background of packages in class diagrams
packageBorderColor	#A80036		Границы пакетов в диаграмме классов
stereotypeCBackgroundColor	#ADD1B2		Фон типа класса в диаграмме классов
stereotypeABackgroundColor	#A9DCDF		Фон индикаторов абстрактных классов в диаграмме классов
stereotypeIBackgroundColor	#B4A7E5		Background of interface spots in class diagrams
stereotypeEBackgroundColor	#EB937F		Background of enum spots in class diagrams
componentArrowColor	#A80036		Color of arrows in component diagrams
componentBackgroundColor	#FEFCE		Background of components
componentBorderColor	#A80036		Borders of components
componentInterfaceBackgroundColor	#FEFCE		Background of interface in component diagrams
componentInterfaceBorderColor	#A80036		Border of interface in component diagrams
noteBackgroundColor	#FBFB77		Фон заметок
noteBorderColor	#A80036		Граница заметок
stateBackgroundColor	#FEFCE		Background of states in state diagrams
stateBorderColor	#A80036		Граница состояний в диаграмме состояний
stateArrowColor	#A80036		Цвета стрелок в диаграмме состоянийColors of arrows in state diagrams
stateStartColor	black		Starting circle in state diagrams
stateEndColor	black		Ending circle in state diagrams
sequenceArrowColor	#A80036		Color of arrows in sequence diagrams
sequenceActorBackgroundColor	#FEFCE		Head's color of actor in sequence diagrams
sequenceActorBorderColor	#A80036		Border of actor in sequence diagrams
sequenceGroupBackgroundColor	#EEEEEE		Header color of alt/opt/loop in sequence diagrams
sequenceLifeLineBackgroundColor	white		Background of life line in sequence diagrams
sequenceLifeLineBorderColor	#A80036		Border of life line in sequence diagrams
sequenceParticipantBackgroundColor	#FEFCE		Background of participant in sequence diagrams
sequenceParticipantBorderColor	#A80036		Border of participant in sequence diagrams



## 9.4 Имя, цвет и размер шрифта

Вы можете изменить шрифт изображения, используя параметры `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
```

Please note the fontname is highly system dependant, so do not over use it, if you look for portability.

Parameter Name	Default Value	Comment
activityFontColor activityFontSize activityFontStyle activityFontName	black 14 plain	Used for activity box
activityArrowFontColor activityArrowFontSize activityArrowFontStyle activityArrowFontName	black 13 plain	Used for text on arrows in activity diagrams
circledCharacterFontColor circledCharacterFontSize circledCharacterFontStyle circledCharacterFontName circledCharacterRadius	black 17 bold Courier 11	Used for text in circle for class, enum and others
classArrowFontColor classArrowFontSize classArrowFontStyle classArrowFontName	black 10 plain	Используется для подписей стрелок в диаграмме классов
classAttributeFontColor classAttributeFontSize classAttributeIconSize classAttributeFontStyle classAttributeFontName	black 10 10 plain	Class attributes and methods
classFontColor classFontSize classFontStyle classFontName	black 12 plain	Used for classes name
classStereotypeFontColor classStereotypeFontSize classStereotypeFontStyle classStereotypeFontName	black 12 italic	Used for stereotype in classes
componentFontColor componentFontSize componentFontStyle componentFontName	black 14 plain	Used for components name
componentStereotypeFontColor componentStereotypeFontSize componentStereotypeFontStyle componentStereotypeFontName	black 14 italic	Used for stereotype in components
componentArrowFontColor componentArrowFontSize componentArrowFontStyle componentArrowFontName	black 13 plain	Used for text on arrows in component diagrams



noteFontColor noteFontSize noteFontStyle noteFontName	black 13 plain	Used for notes in all diagrams but sequence diagrams
packageFontColor packageFontSize packageFontStyle packageFontName	black 14 plain	Used for package and partition names
sequenceActorFontColor sequenceActorFontSize sequenceActorFontStyle sequenceActorFontName	black 13 plain	Used for actor in sequence diagrams
sequenceDividerFontColor sequenceDividerFontSize sequenceDividerFontStyle sequenceDividerFontName	black 13 bold	Used for text on dividers in sequence diagrams
sequenceArrowFontColor sequenceArrowFontSize sequenceArrowFontStyle sequenceArrowFontName	black 13 plain	Used for text on arrows in sequence diagrams
sequenceGroupingFontColor sequenceGroupingFontSize sequenceGroupingFontStyle sequenceGroupingFontName	black 11 plain	Used for text for "else" in sequence diagrams
sequenceGroupingHeaderFontColor sequenceGroupingHeaderFontSize sequenceGroupingHeaderFontStyle sequenceGroupingHeaderFontName	black 13 plain	Used for text for "alt/opt/loop" headers in sequence diagrams
sequenceParticipantFontColor sequenceParticipantFontSize sequenceParticipantFontStyle sequenceParticipantFontName	black 13 plain	Used for text on participant in sequence diagrams
sequenceTitleFontColor sequenceTitleFontSize sequenceTitleFontStyle sequenceTitleFontName	black 13 plain	Used for titles in sequence diagrams
titleFontColor titleFontSize titleFontStyle titleFontName	black 18 plain	Used for titles in all diagrams but sequence diagrams
stateFontColor stateFontSize stateFontStyle stateFontName	black 14 plain	Used for states in state diagrams
stateArrowFontColor stateArrowFontSize stateArrowFontStyle stateArrowFontName	black 13 plain	Используется для задания текста на стрелках в диаграммах состояния
stateAttributeFontColor stateAttributeFontSize stateAttributeFontStyle stateAttributeFontName	black 12 plain	Used for states description in state diagrams
usecaseFontColor usecaseFontSize usecaseFontStyle usecaseFontName	black 14 plain	Используется для меток usecase'ов на usecase диаграммах

usecaseStereotypeFontColor usecaseStereotypeFontSize usecaseStereotypeFontStyle usecaseStereotypeFontName	black 14 italic	Used for stereotype in usecase
usecaseActorFontColor usecaseActorFontSize usecaseActorFontStyle usecaseActorFontName	black 14 plain	Used for actor labels in usecase diagrams
usecaseActorStereotypeFontColor usecaseActorStereotypeFontSize usecaseActorStereotypeFontStyle usecaseActorStereotypeFontName	black 14 italic	Used for stereotype for actor
usecaseArrowFontColor usecaseArrowFontSize usecaseArrowFontStyle usecaseArrowFontName	black 13 plain	Used for text on arrows in usecase diagrams
footerFontColor footerFontSize footerFontStyle footerFontName	black 10 plain	
headerFontColor headerFontSize headerFontStyle headerFontName	black 10 plain	Используется для заголовков

## 9.5 Черно-белый вывод

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

```
@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

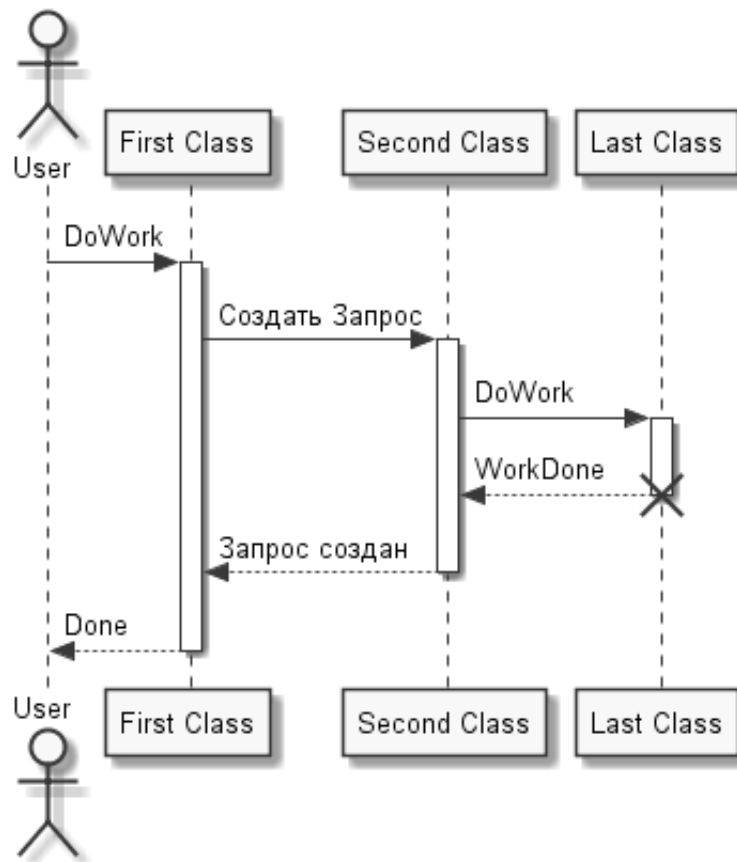
A -> B: СоздатьЗапрос
activate B

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

B --> A: Запроссоздан
deactivate B

A --> User: Done
deactivate A

@enduml
```



## 10 Preprocessing

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

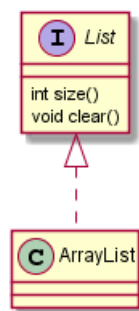
Эта функциональность очень схожа с препроцессором языка C, за исключением того что символ решетки "#" заменяется на символ восклицания "!".

### 10.1 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 которые его включают.

## 10.2 Константы

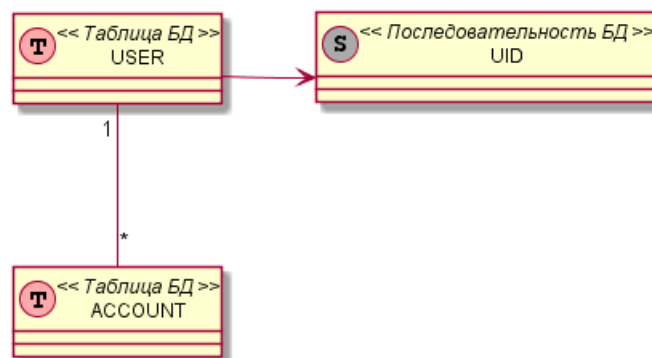
Вы можете определить константу используя директиву `!define`. Как в C language, a constant name can only use alphanumeric and underscore characters, and cannot start with a digit.

```
@startuml
```

```
!define SEQUENCE (S,#AAAAAA) ПоследовательностьБД
!define TABLE (T,#FFAAAA) ТаблицаБД
```

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

```
@enduml
```



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

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

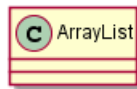
## 10.3 Conditions

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

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

You can also provide a `!else` part which will be included if the константа не определена.

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

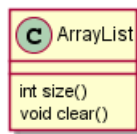


File `ArrayList.iuml`:

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

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

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

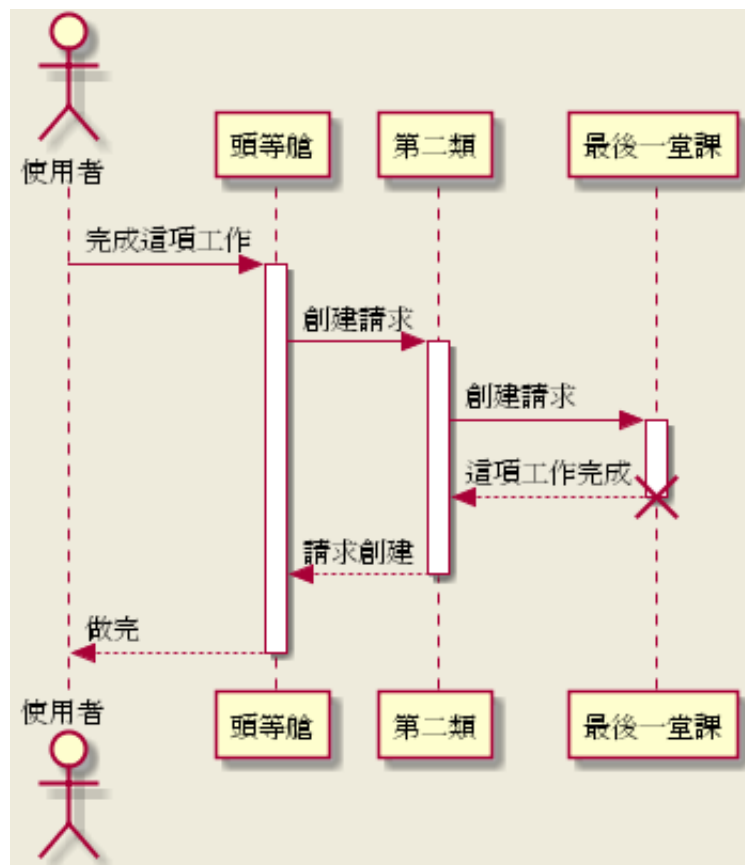


You can also use the `!ifndef` directive that includes lines if the provided constant has *NOT* been defined.

## 11 Internationalization

The PlantUML language use *letters* to define actor, usecase and so on. But *letters* are not only A-Z latin characters, it could be *any kind of letter from any language*.

```
@startuml
skinparam backgroundColor #EEEEBD
actor 使用者
participant 頭等艙" as A
participant 第二類" as B
participant 最後一堂課" as 別的東西使用者
-> A: 完成這項工作
activate A
A -> B: 創建請求
activate B
B -> 別的東西: 創建請求
activate 別的東西別的東西
-> B: 這項工作完成
destroy 別的東西
B -> A: 請求創建
deactivate B
A -> 使用者: 做完
deactivate A
@enduml
```



### 11.1 Кодировка

The default charset used when reading the text files containing the UML text description is system dependant. Normally, it should just be fine, but in some case, you may want to the use another charset. For example, with the command line:

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





Or, with the ant task:

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

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



## 12 Цветные имена

Here is the list of colors recognized by PlantUML. Note that color names are case insensitive.

	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		

<b>1</b>	<b>Диаграммы последовательности</b>	<b>1</b>
1.1	Основные примеры	1
1.2	Объявление участников	2
1.3	Use non-letters in participants	3
1.4	Сообщения к самому себе	3
1.5	Изменить стиль стрелок	4
1.6	Message sequence numbering	5
1.7	Заголовок	7
1.8	Разбиение диаграмм	8
1.9	Группировка сообщений	9
1.10	Примечания в сообщениях	11
1.11	Some other notes	12
1.12	Форматирование с использованием HTML	13
1.13	Divider	14
1.14	Строка активации и деактивации	15
1.15	Participant creation	17
1.16	Incoming and outgoing messages	18
1.17	Stereotypes and Spots	19
1.18	More information on titles	20
1.19	Participants englobes	22
1.20	Удаление футера	23
1.21	Skinparam	24
1.22	Skin	25
<b>2</b>	<b>Диаграмма прецедентов</b>	<b>26</b>
2.1	Прецеденты	26
2.2	Актёры	27
2.3	Основной пример	28
2.4	Расширение	29
2.5	Использование заметок	30
2.6	Stereotypes	31
2.7	Changing arrows direction	32
2.8	Title the diagram	33
2.9	Направление слева направо	34
2.10	Skinparam	35
<b>3</b>	<b>Диаграмма классов</b>	<b>36</b>
3.1	Отношения между классами	36
3.2	Label on relations	37
3.3	Adding methods	38
3.4	Defining visibility	39
3.5	Notes and stereotypes	40
3.6	More on notes	41
3.7	Abstract class and interface	42
3.8	Using non-letters	43
3.9	методы скрытия атрибутов	44
3.10	Specific Spot	45
3.11	Пакеты	46
3.12	Пространства имен	47
3.13	Изменение направления стрелок	48
3.14	Lollipop интерфейс	49
3.15	Title the diagram	49
3.16	Ассоциация классов	50
3.17	Skinparam	51
3.18	Skinned Stereotypes	52
3.19	Splitting large files	53

<b>4</b>	<b>Диаграмма деятельности</b>	<b>54</b>
4.1	Простая деятельность . . . . .	54
4.2	Метка на стрелках . . . . .	54
4.3	Изменение направления стрелки . . . . .	55
4.4	Ветвления . . . . .	56
4.5	More on Branches . . . . .	57
4.6	Синхронизация . . . . .	58
4.7	Long activity description . . . . .	59
4.8	Заметки . . . . .	60
4.9	Partition . . . . .	61
4.10	Заголовок диаграммы . . . . .	62
4.11	Skinparam . . . . .	63
4.12	Complete example . . . . .	64
<b>5</b>	<b>Component Diagram</b>	<b>66</b>
5.1	Компоненты . . . . .	66
5.2	Interfaces . . . . .	67
5.3	Basic example . . . . .	68
5.4	Using notes . . . . .	69
5.5	Grouping Components . . . . .	70
5.6	Изменение направления связи . . . . .	71
5.7	Title the diagram . . . . .	72
5.8	Skinparam . . . . .	73
<b>6</b>	<b>Диаграмма состояний</b>	<b>74</b>
6.1	Простое состояние . . . . .	74
6.2	Составное состояние . . . . .	75
6.3	Длинные имена . . . . .	76
6.4	Параллельные состояния . . . . .	77
6.5	Направления стрелок . . . . .	78
6.6	Заметки . . . . .	79
6.7	Еще о заметках . . . . .	80
<b>7</b>	<b>Диаграмма объектов</b>	<b>81</b>
7.1	Определение объектов . . . . .	81
7.2	Отношения между объектами . . . . .	81
7.3	Добавление полей . . . . .	82
<b>8</b>	<b>Common commands</b>	<b>83</b>
8.1	Заголовок и подвал . . . . .	83
8.2	Масштаб . . . . .	84
8.3	Вращения . . . . .	85
<b>9</b>	<b>Изменить шрифты и цвета</b>	<b>86</b>
9.1	Использование . . . . .	86
9.2	Вложенный . . . . .	86
9.3	Color . . . . .	87
9.4	Имя, цвет и размер шрифта . . . . .	88
9.5	Черно-белый вывод . . . . .	91
<b>10</b>	<b>Preprocessing</b>	<b>92</b>
10.1	Включение файлов . . . . .	92
10.2	Константы . . . . .	93
10.3	Conditions . . . . .	94
<b>11</b>	<b>Internationalization</b>	<b>95</b>
11.1	Кодировка . . . . .	95
<b>12</b>	<b>Цветные имена</b>	<b>97</b>

