

Drawing UML with PlantUML

Language Reference Guide

PlantUML is an Open Source project that allows to quickly write:

- sequence diagram,
- use case diagram,
- class diagram,
- activity diagram,
- component diagram

Diagrams are defined using a simple and intuitive language.

1- Sequence Diagram

Basic examples

Every UML description must start by @startuml and must finish by @enduml.

The sequence ">" is used to draw a message between two participants. Participants do not have to be explicitly declared.

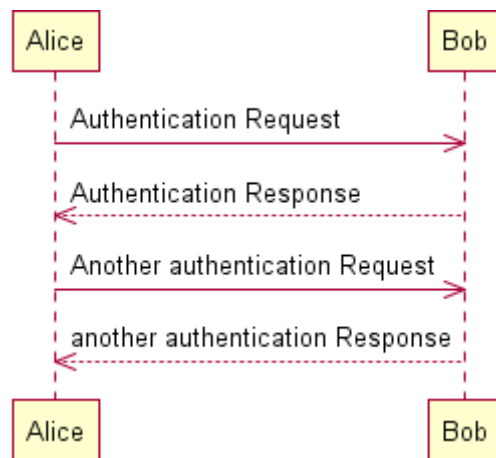
To have a dotted arrow, you use -->

It is also possible to use <- et <--. That does not change the drawing, but may improve lisibility.

Example:

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

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



Declaring participant

It is possible to change participant order using the `participant` keyword. .

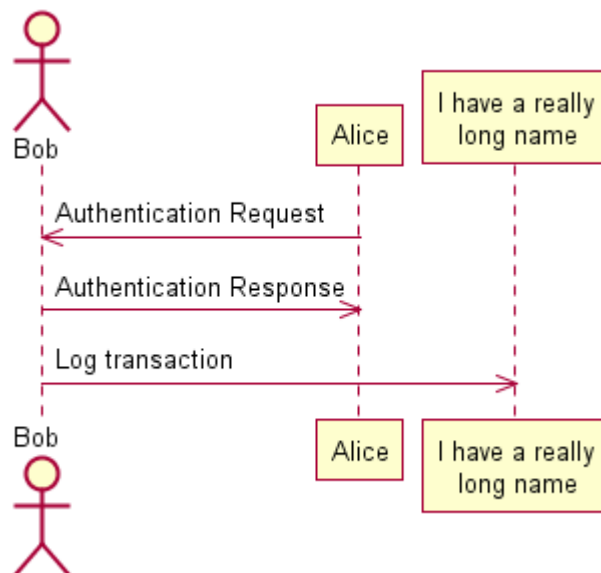
It is also possible to use the `actor` keyword to use a stickman instead of a box for the participant.

You can rename a participant using the `as` keyword.

Everything that starts with simple quote ' is a comment.

```
@startuml
actor Bob
' The only difference between actor and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L

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

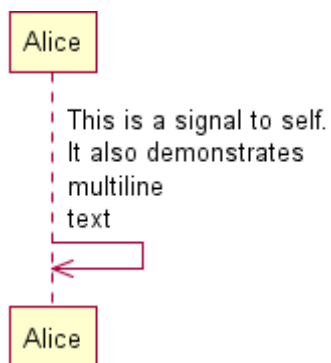


Message to Self

A participant can send a message to itself.

It is also possible to have multilines using \n.

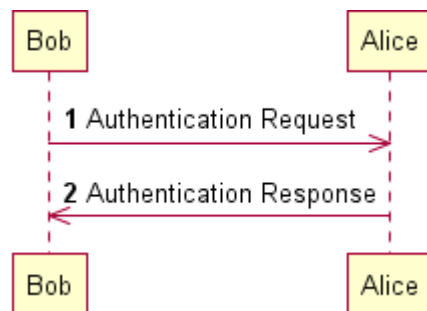
```
@startuml
Alice->Alice: This is a signal to self.\nIt also
demonstrates\nmultiline \ntext
@enduml
```



Message sequence numbering

The keyword `autonumber` is used to automatically add number to messages.

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



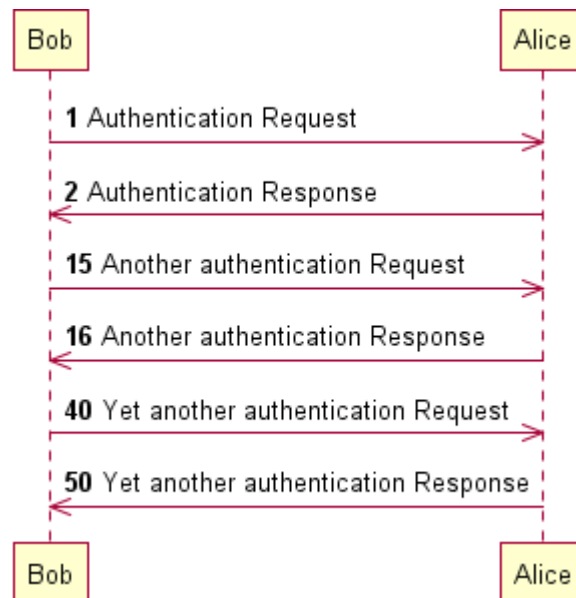
You can specify a startnumber with autonumber 'start' , and also an increment with autonumber 'start' 'increment'

```
@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
```



You can specify a format for your number by using between double-quote.

The formatting is done with the Java class `DecimalFormat` ('0' means digit, '#' means digit and zero if absent).

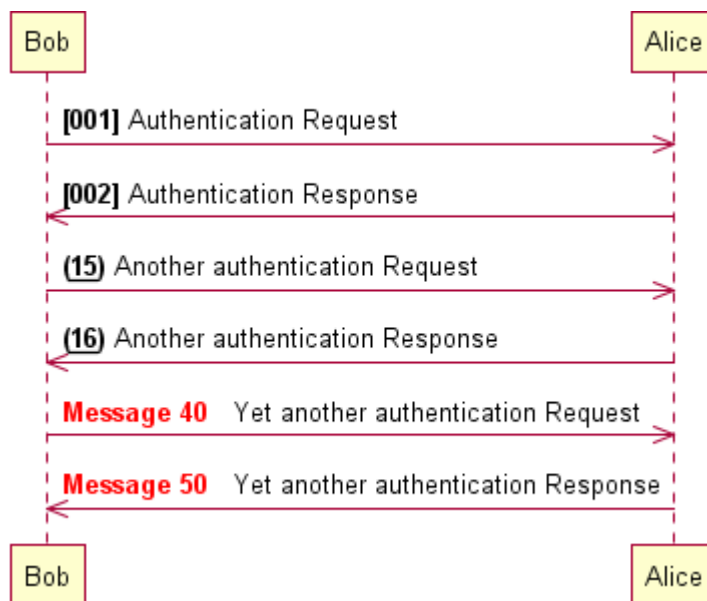
You can also use some html tags in the format.

```
@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
```

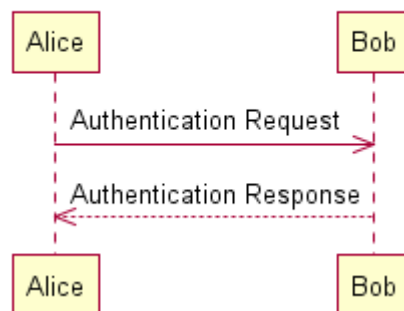


Title

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

```
@startuml
title Simple communication example
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
@enduml
```

Simple communication example



Splitting diagrams

The `newpage` keyword is used to split a diagram into several images.

You can put a title for the new page just after the `newpage` keyword.

```
@startuml
Alice -> Bob : message 1
Alice -> Bob : message 2

newpage

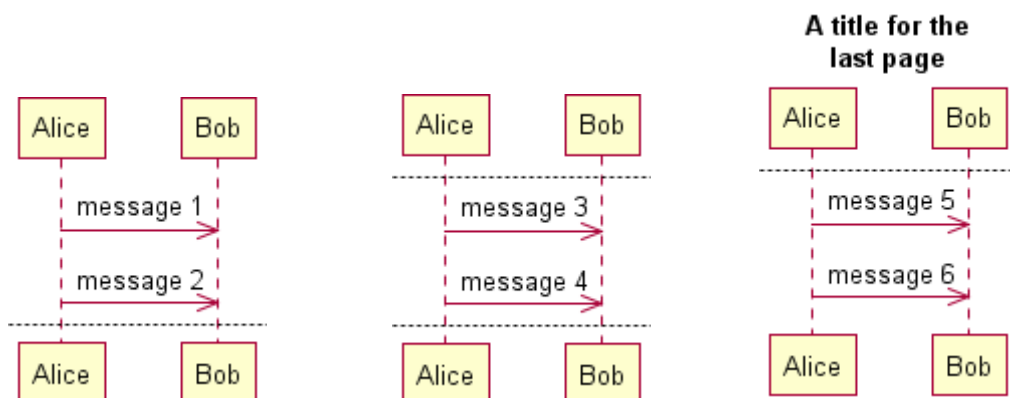
Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page

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

@enduml
```

This is very handy to print long diagram on several pages.



Grouping message

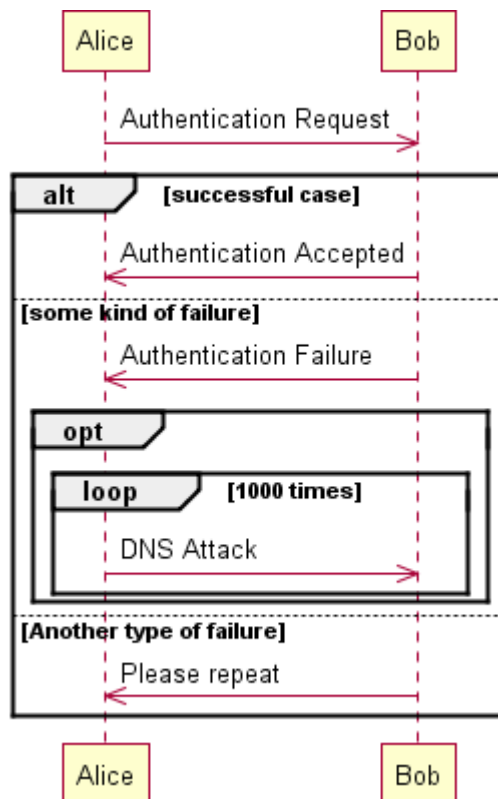
It is possible to group messages together using `alt/else`, `opt` or `loop` keywords.

It is possible to add a text that will be displayed into the header.

The `end` keyword is used to close the group.

Note that it is possible to nest groups.

```
@startuml
Alice -> Bob: Authentication Request
alt successful case
    Bob -> Alice: Authentication Accepted
else some kind of failure
    Bob -> Alice: Authentication Failure
    opt
        loop 1000 times
            Alice -> Bob: DNS Attack
        end
    end
else Another type of failure
    Bob -> Alice: Please repeat
end
@enduml
```



Notes on messages

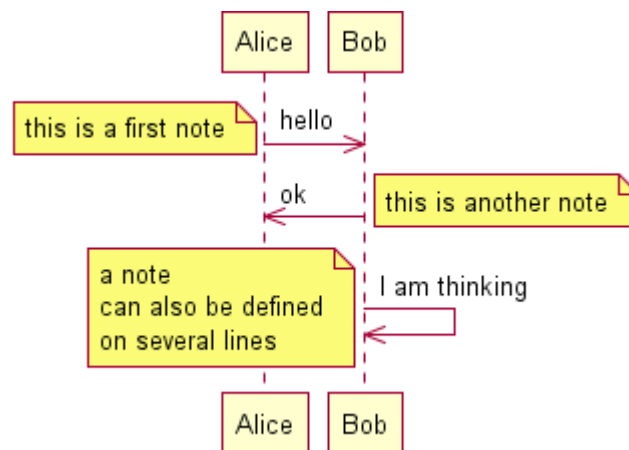
It is possible to put notes on message using the `note left` or `note right` keywords *just after the message*.

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

```
@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
```



Some other notes

It is also possible to place notes relative to participant with `note left of`, `note right of` or `note over` keywords.

You can also have multilines note using the `end note` keywords.

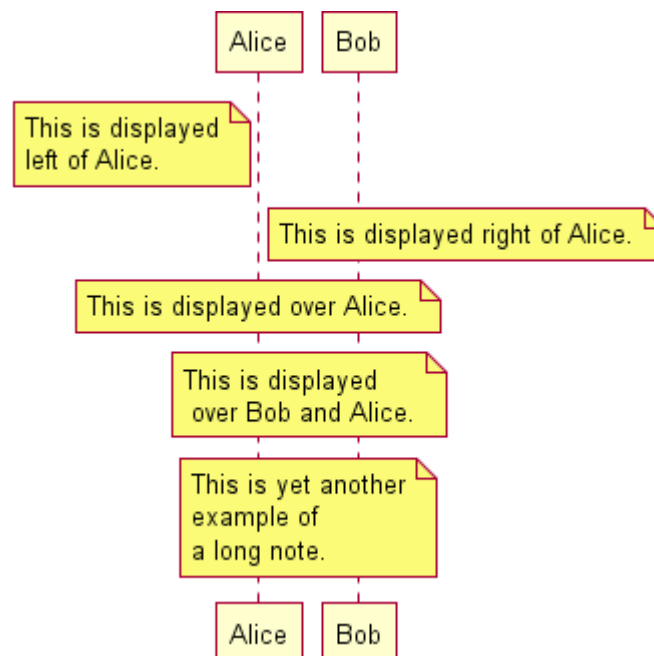
```
@startuml
participant Alice
participant Bob
note left of Alice
  This is displayed
  left of Alice.
end note

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

note over Alice: This is displayed over Alice.

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

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



Formatting using HTML

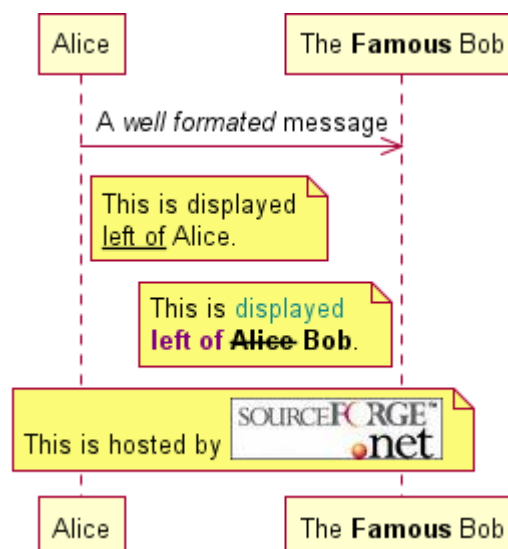
It is also possible to use few html tags like :

- ``
- `<u>`
- `<i>`
- `<strike>`
- `` or ``
- `` : the file must be accessible by the filesystem

```
@startuml
participant Alice
participant "The <b>Famous</b> Bob" as Bob

Alice -> Bob : A <i>well formatted</i> message
note right of Alice
    This is displayed
    <u>left of</u> Alice.
end note
note left of Bob
    This is <font color=#118888>displayed</font>
    <b><font color=purple>left of</font> <strike>Alice</strike>
Bob</b>.
end note
note over Alice, Bob
    This is hosted by <img src=sourceforge.jpg>
end note

@enduml
```



Lifeline Activation and Destruction

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

Once a participant is activated, its lifeline appears.

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

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

```
@startuml
participant User

User -> A: DoWork
activate A

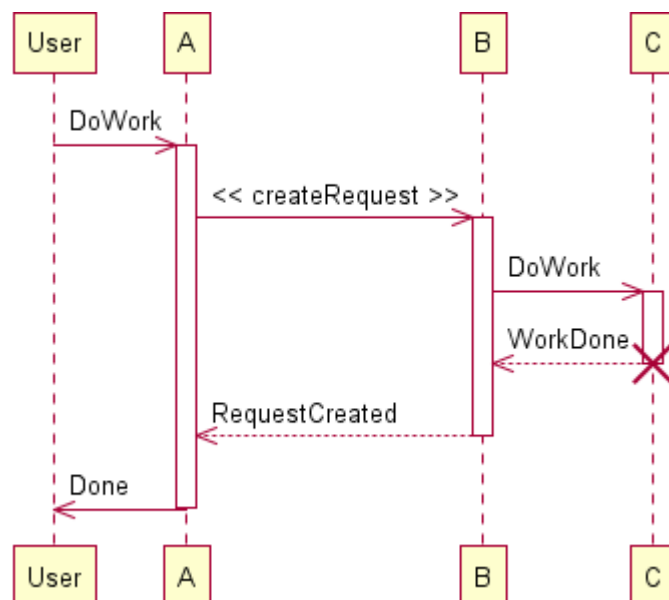
A -> B: << createRequest >>
activate B

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

B --> A: RequestCreated
deactivate B

A -> User: Done
deactivate A

@enduml
```



Nested lifeline can be used

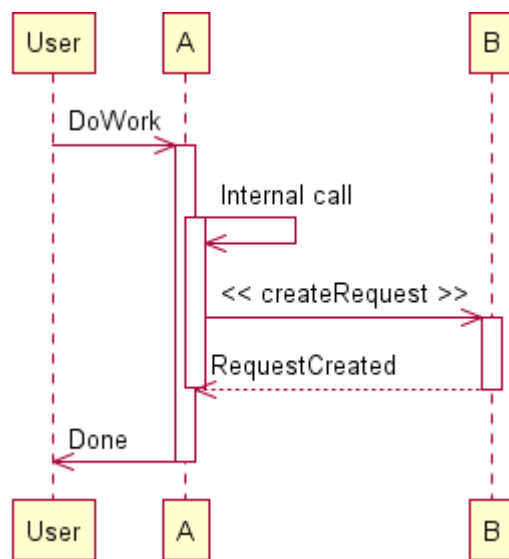
```
@startuml
participant User
User -> A: DoWork
activate A

A -> A: Internal call
activate A

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

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

@enduml
```



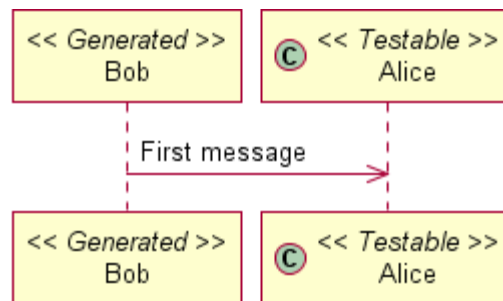
Stereotypes and Spots

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)`.

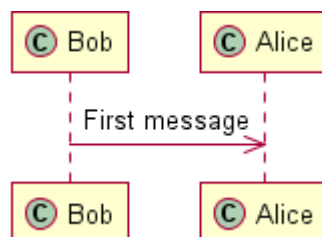
```
@startuml
participant Bob << Generated >>
participant Alice << (C,#ADD1B2) Testable >>

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



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

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



More information on titles

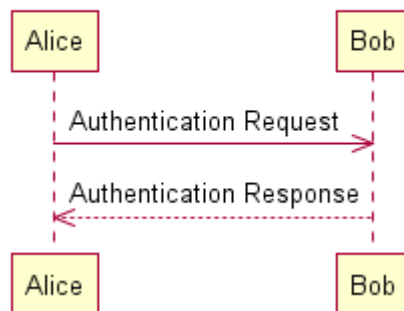
You can use some HTML tags in the title like :

- ``
- `<u>`
- `<i>`
- `<strike>`
- `` or ``
- ``: the file must be accessible by the filesystem

```
@startuml
title <u>Simple</u> communication example
```

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

Simple communication example

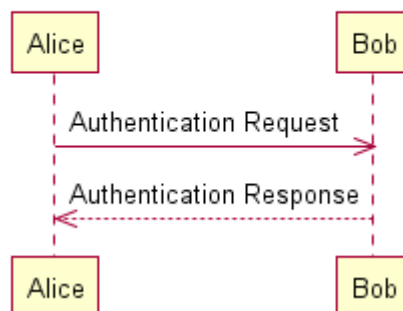


You can add newline using `\n` in the title description.

```
@startuml
title <u>Simple</u> communication example\nnon several lines
```

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

Simple communication example on several lines

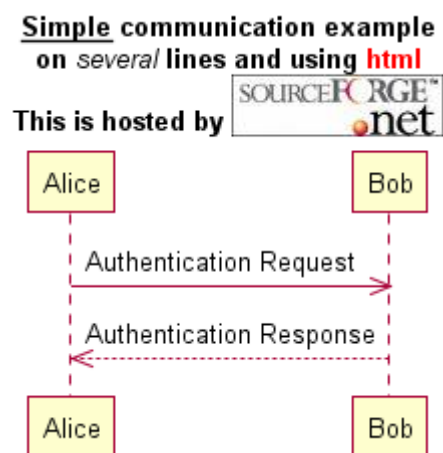


You can also define title on several lines using `title` and `end title` keywords.

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

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

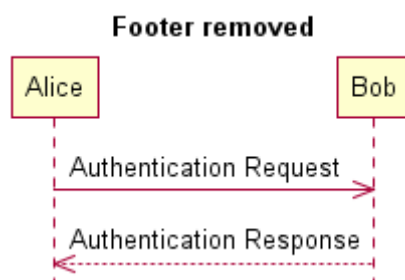
```
@enduml
```



Removing Footer

You can use the `footbox off` keywords to remove the footer of the diagram.

```
@startuml
footbox off
title Footer removed
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
@enduml
```



Skin

Use the keyword `skin` to change the look of the generated diagram.

There are only two skins available today (Rose, which is the default, and BlueModern), but it is possible to write your own skin.

```
@startuml
skin BlueModern

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

User -> A: DoWork
activate A

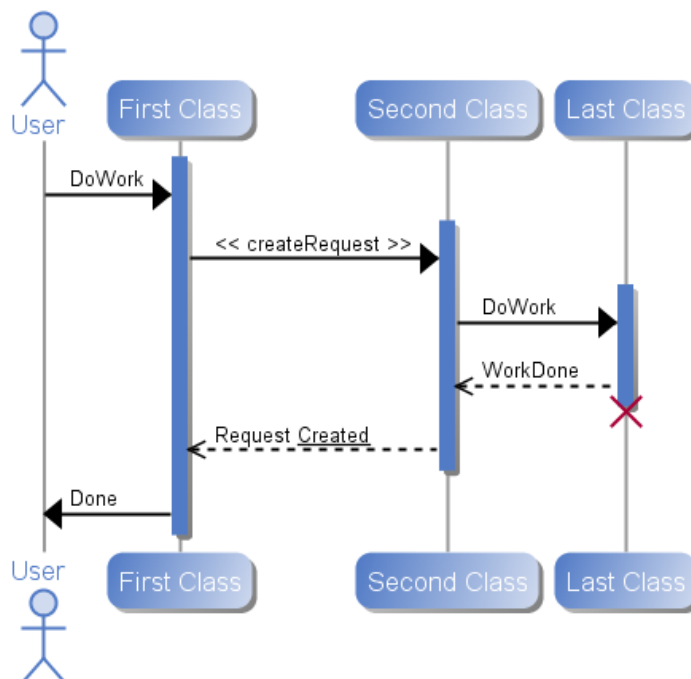
A -> B: << createRequest >>
activate B

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

B --> A: Request <u>Created</u>
deactivate B

A -> User: Done
deactivate A

@enduml
```



2- Use Case Diagram

Actor and Use Case definitions

To define actors and use cases, you can use the `actor` and `usecase` keywords.
It is possible to have long descriptions using the `as` keywords.

```
@startuml
actor User
actor "Main Admin" as Admin
usecase Start
usecase "Use the application" as Use
@enduml
```



Links between actors and use cases

To link actors and use cases, the arrow `-->` is used.

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

If there is only one dashes in the arrow, it means that the arrow is horizontal.

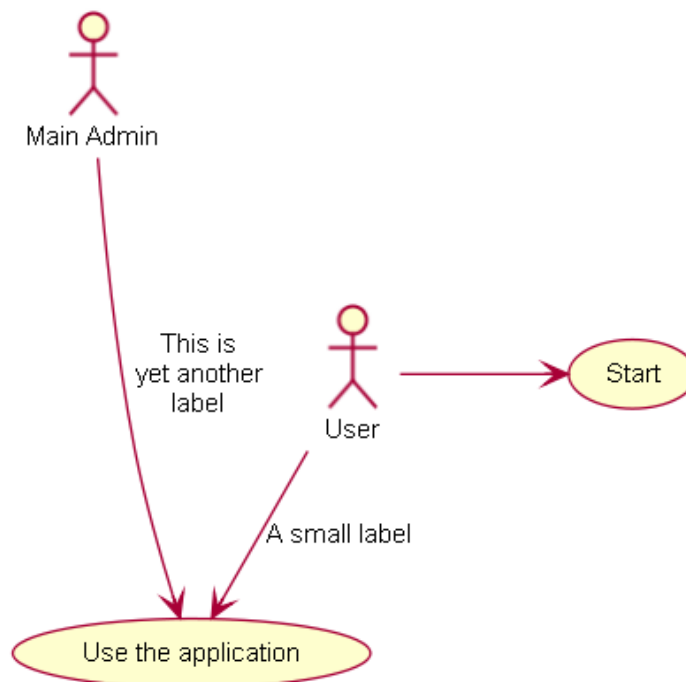
You can add a label after the symbol `:`.

```
@startuml
actor User
actor "Main Admin" as Admin
usecase Start
usecase "Use the application" as Use

User -> Start
User --> Use : A small label


Admin ---> Use : This is\
yet another\
label

@enduml
```



Extension

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

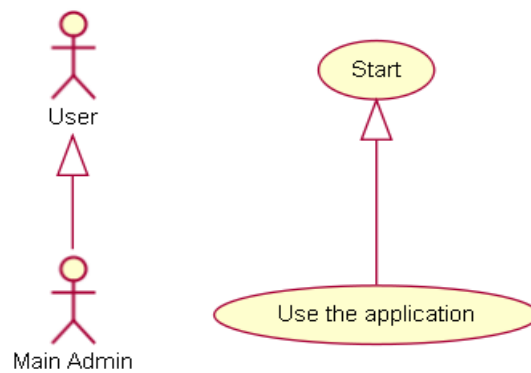
This symbol stands for .

Use case can also have extensions.

```
@startuml
actor User
actor "Main Admin" as Admin
usecase Start
usecase "Use the application" as Use

User <|-- Admin
Start <|-- Use

@enduml
```



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

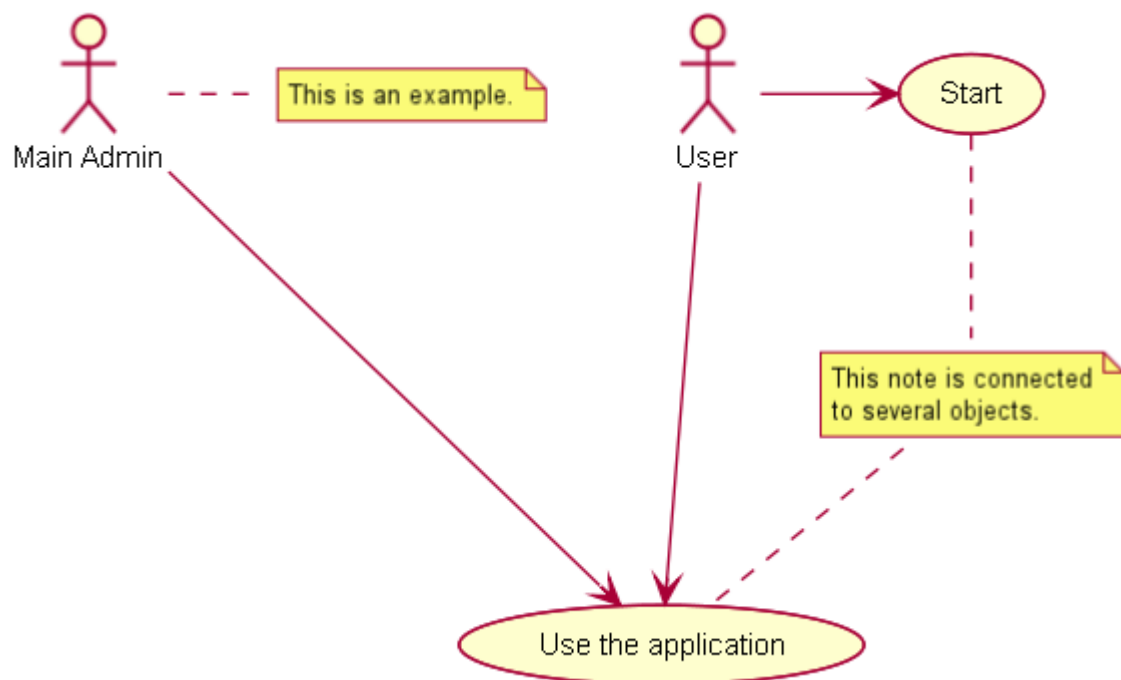
```
@startuml
actor User
actor "Main Admin" as Admin
usecase Start
usecase "Use the application" as Use

User -> Start
User --> Use

Admin ---> Use

note right of Admin : This is an example.

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



Stereotypes

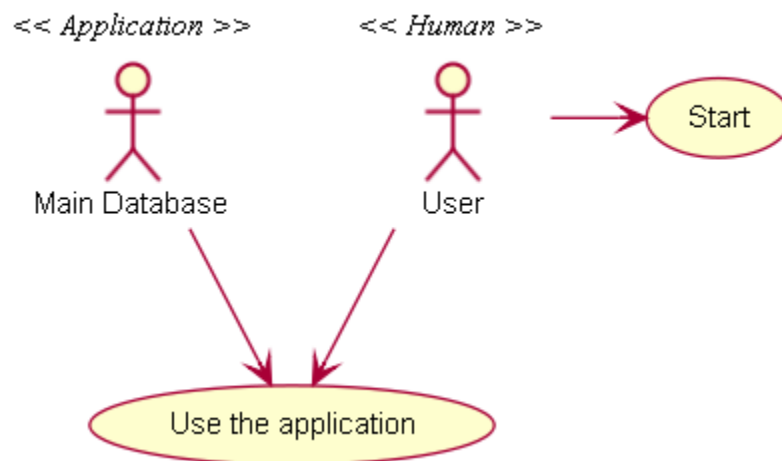
You can add stereotypes while defining actors and use cases using "<< " and ">> "

```
@startuml
actor User << Human >>
actor "Main Database" as Base << Application >>
usecase Start << One Shot >>
usecase "Use the application" as Use

User -> Start
User --> Use

Base --> Use

@enduml
```



Title the diagram

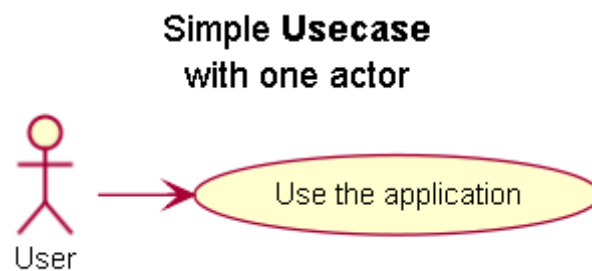
The `title` keywords is used to put a title.

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

actor User
usecase "Use the application" as Use
User -> Use

@enduml
```



A complete example

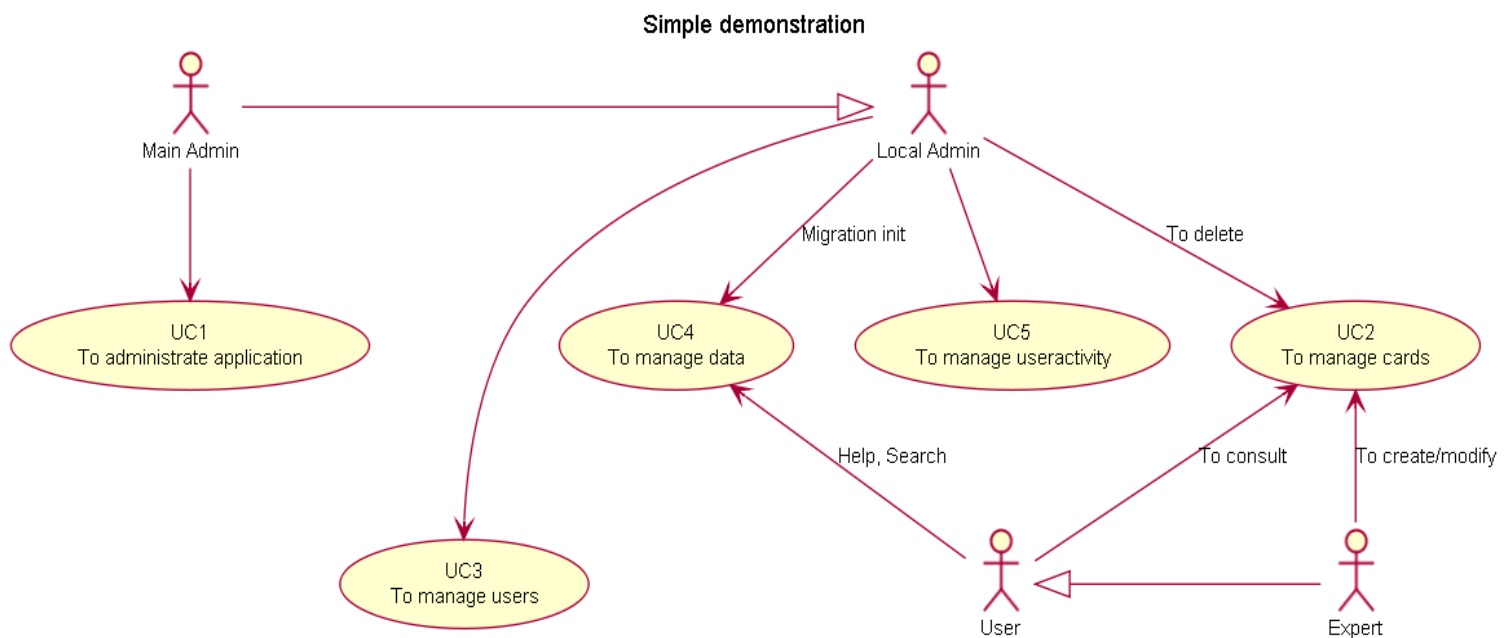
You can define the direction of any arrow as <|-- or --|> .

```
@startuml
actor "Main Admin" as Admin
actor "Local Admin" as LAdmin
actor Expert
actor User
Admin --|> LAdmin
User <|-- Expert

usecase "UC1\nTo administrate application" as UC1
usecase "UC2\nTo manage cards" as UC2
usecase "UC3\nTo manage users" as UC3
usecase "UC4\nTo manage data" as UC4
usecase "UC4\nTo manage useractivity" as UC5

Admin --> UC1
LAdmin --> UC2 : To delete
LAdmin ---> UC3
LAdmin --> UC4 : Migration init
LAdmin --> UC5




UC2 <-- Expert : To create/modify
UC2 <-- User : To consult
UC4 <-- User : Help, Search
@enduml
```



3- Classes Diagram

Relations between classes

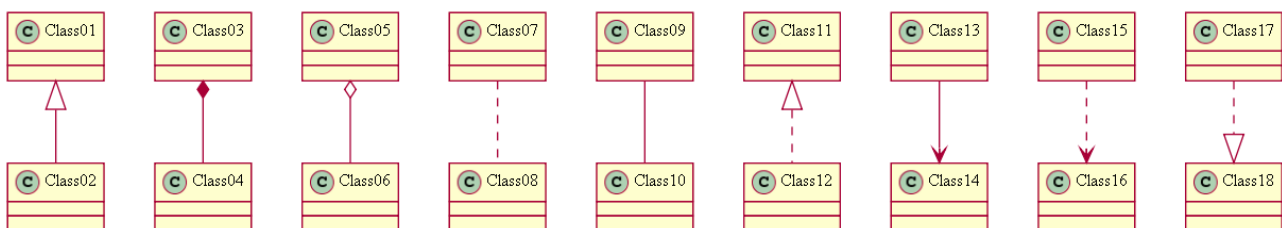
Relations between classes are defined using the following symbols :

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
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
Class11 <|.. Class12
Class13 --> Class14
Class15 ..> Class16
Class17 ..|> Class18
@enduml
```

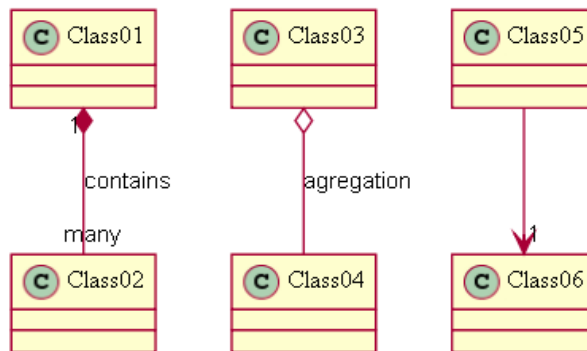


Label on relations

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
```



Adding methods

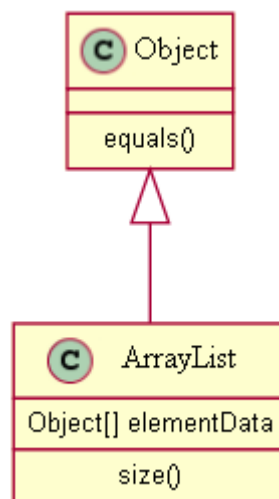
To declare fields and methods, you can use the symbol ":" followed by the field's or method's name.

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

```
@startuml
Object <|-- ArrayList

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

@enduml
```



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.

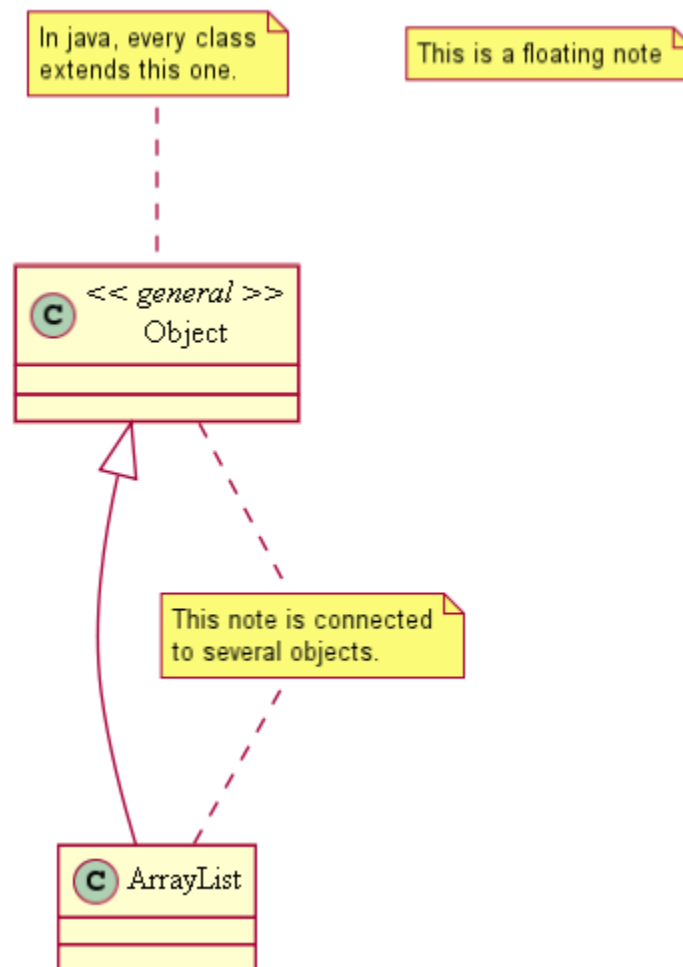
A note can be also define alone with the `note` keywords, then linked to other objects using the `..` symbol.

```
@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
```



More on notes

It is also possible to use few html tags like :

- ``
- `<u>`
- `<i>`
- `<strike>`
- `` or ``
- `` : the file must be accessible by the filesystem

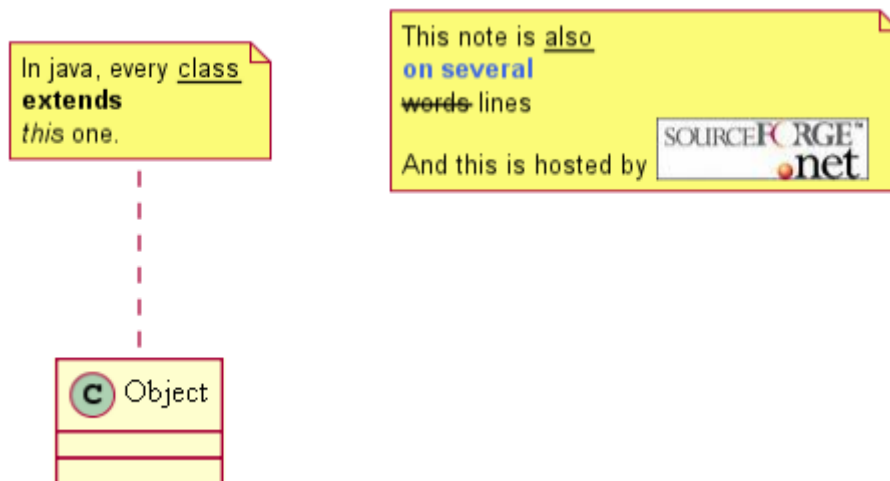
You can also have a note on several lines.

```
@startuml

note top of Object
    In java, every <u>class</u>
    <b>extends</b>
    <i>this</i> one.
end note

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

@enduml
```

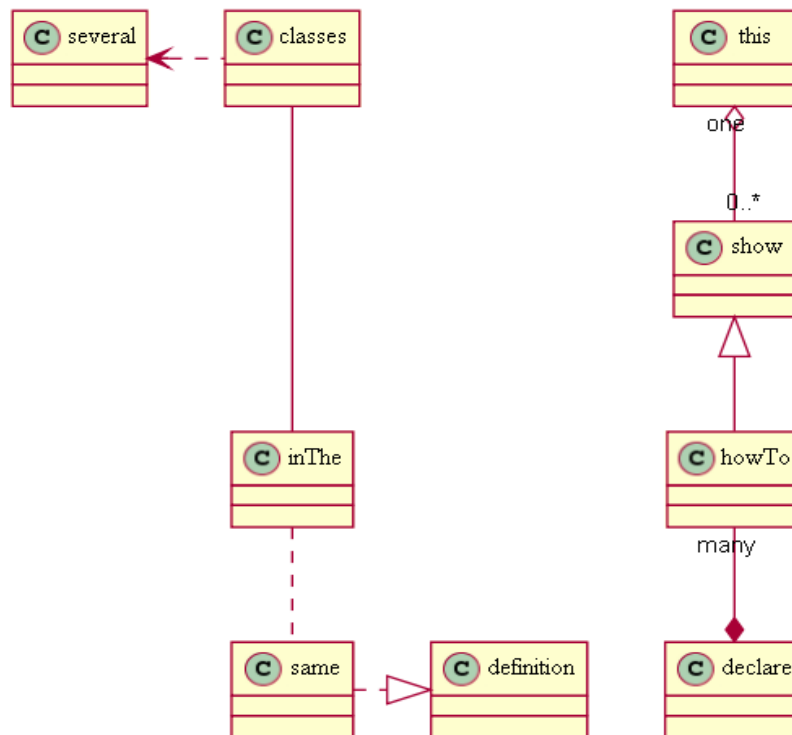


Chained relations

It is possible to chain relation definition on the same line.

Please note that the number of dashes – or dot . in a relation does change arrow's length.

```
@startuml
this "one" o-- "0..*" show <|-- howTo "many" --* declare
several <.. classes --- inThe .. same .|> definition
@enduml
```



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

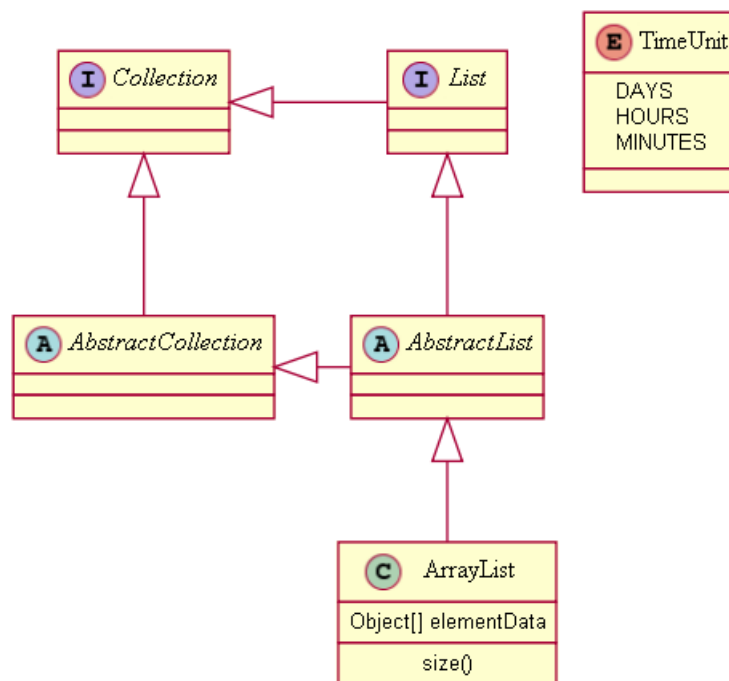
List <|-- AbstractList
Collection <|-- AbstractCollection

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

ArrayList : Object[] elementData
ArrayList : size()

enum TimeUnit
TimeUnit : DAYS
TimeUnit : HOURS
TimeUnit : MINUTES

@enduml
```

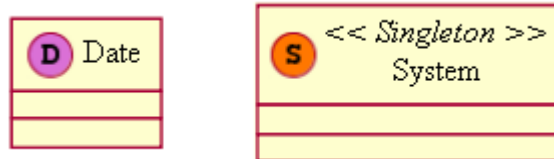


Specific Spot

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 and a color, like in this example:

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



Using packages

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

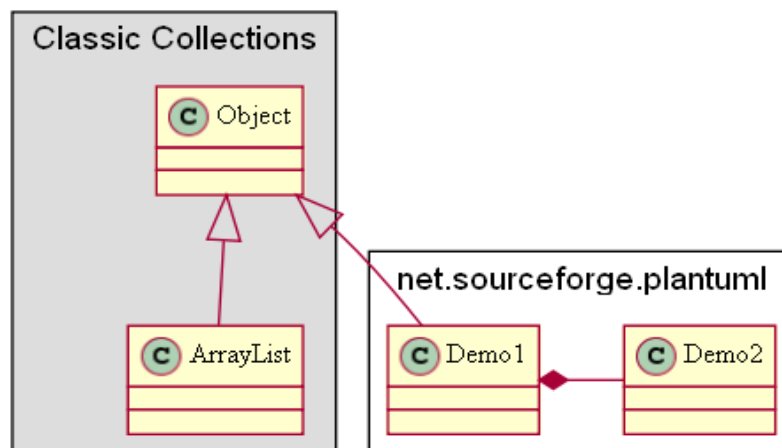
When you declare classes, they are automatically put in the last used package.

```
@startuml

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

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

@enduml
```



Title the diagram

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

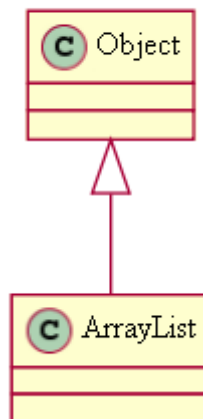
You can use `title` and `end title` keywords for a longer title, as in sequence diagrams.

```
@startuml img/classes15.png
title Simple <b>example</b>\nof title

Object <|-- ArrayList

@enduml
```

Simple example of title

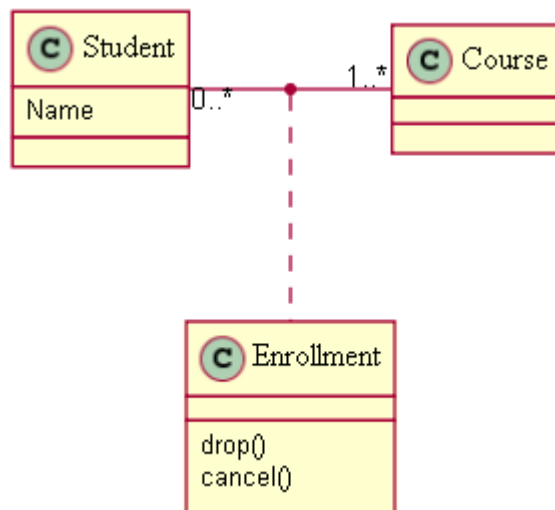


Association classes

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

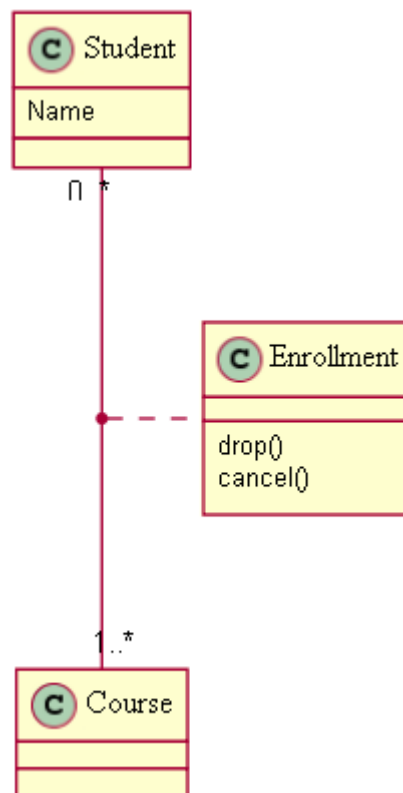
Enrollment : drop()
Enrollment : cancel()
@enduml
```



You can define it in another direction:

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

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



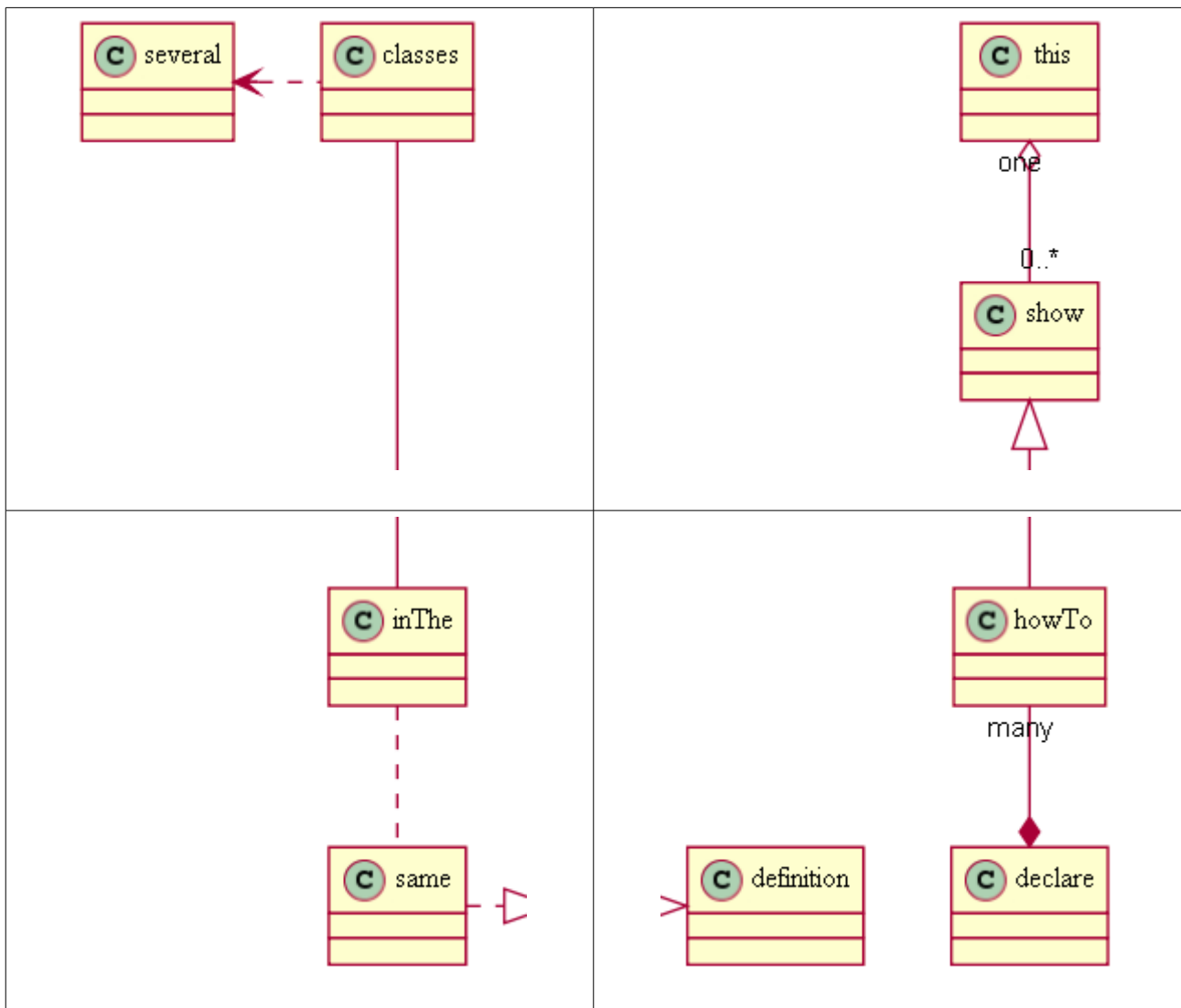
Splitting large files

Sometimes, you will get some very large image files.

You can use the "page (hpages) x (vpages) " command to split the generated image into several files :

hpages is a number that indicated the number of horizontal pages, and vpages is a number that indicated the number of vertical pages.

```
@startuml
' Split into 4 pages
page 2x2
this "one" o-- "0..*" show <|-- howTo "many" --* declare
several <. classes --- inThe .. same .|> definition
@enduml
```



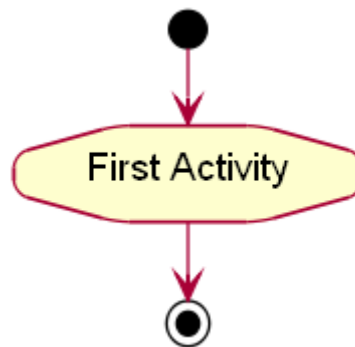
4- Activity Diagram

Simple Activity

You can use `$start` for the starting point and `$end` for the ending point of the activity diagram.

Use `-->` for arrows.

```
@startuml
$start --> "First Activity"
"First Activity" --> $end
@enduml
```



Arrows

You can use `->` for horizontal arrows.

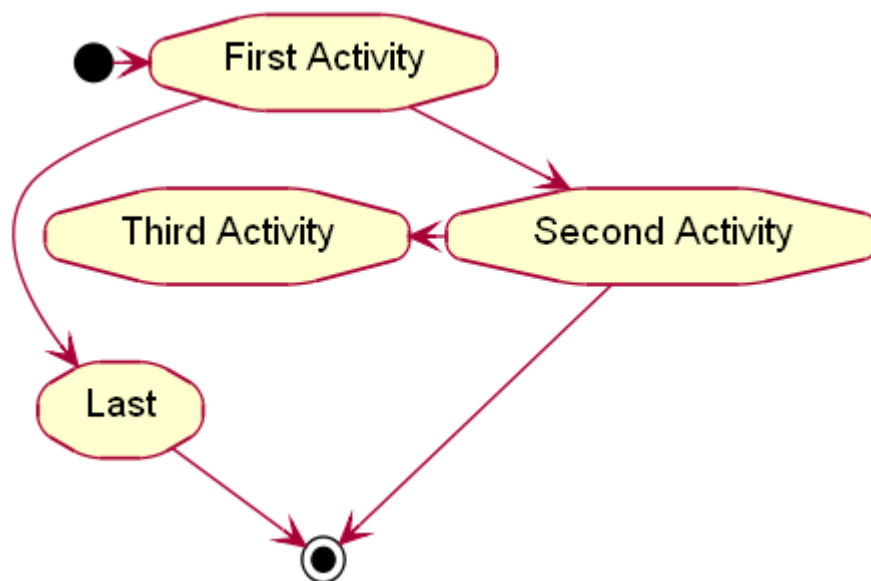
You can use `--->` for longer arrows.

By default, an arrow starts at the last used activity.

```
@startuml
$start -> "First Activity"
--> "Second Activity"
"Third Activity" <- "Second Activity"
--> *end

"First Activity" ---> Last
--> $end

@enduml
```



Branches

You can use <> followed with an internal label for branches.

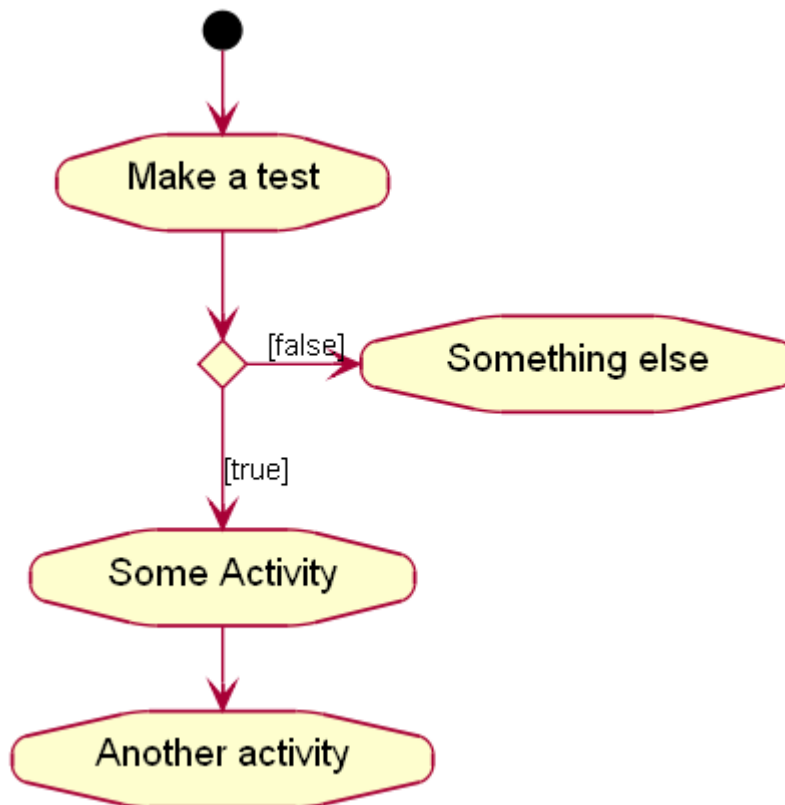
The notation [some label] always refers to the last used branche.

```
@startuml
$start --> "Make a test"
--> <> B1

--> [true] "Some Activity"
--> "Another activity"

-> [false] "Something else"

@enduml
```



Synchronization

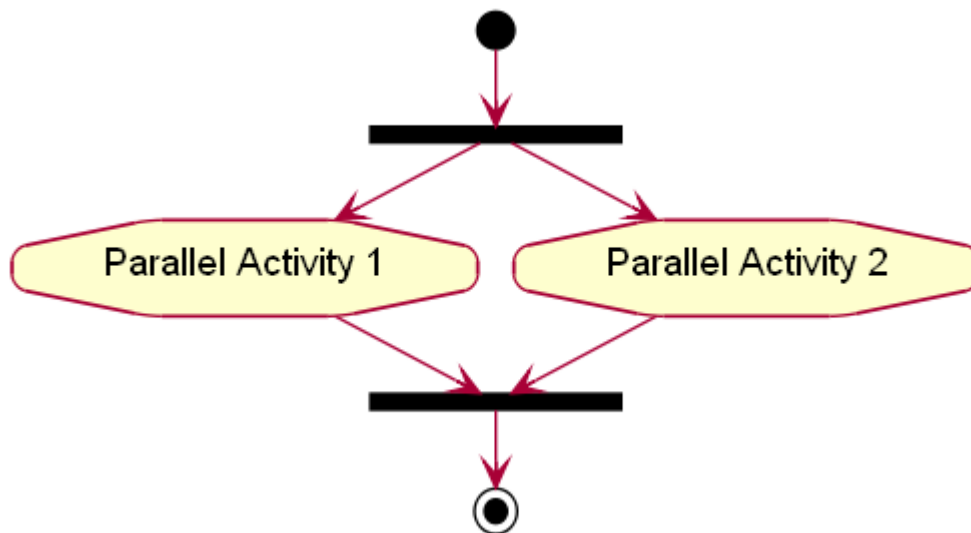
You can use `=== code ===` to display synchronization bars.

```
@startuml
$start --> ===B1===
--> "Parallel Activity 1"
--> ===B2===

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

--> $end

@enduml
```



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 :

- ``
- `<i>`
- ``
- `` or ``

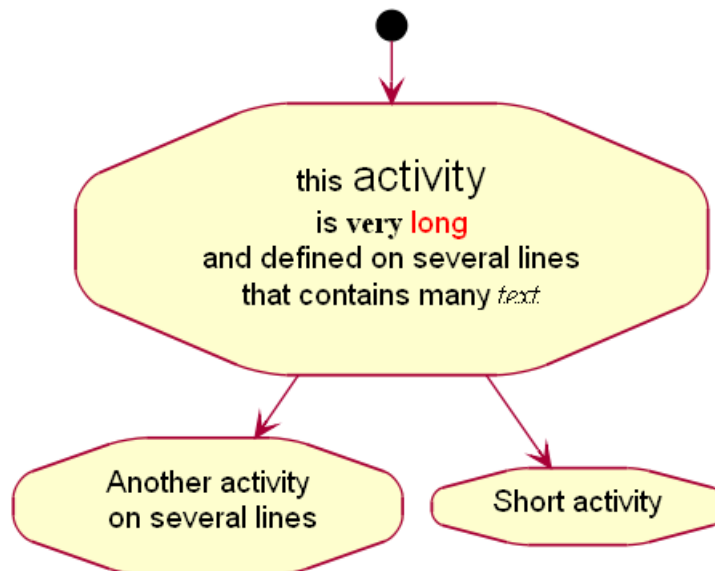
You can also give a short code to the activity with the `as` keyword. This code can be used latter in the diagram description.

```
@startuml
$start --> "this <font size=20>activity</font>
           is <b>very</b> <font color=red>long</font>
           and defined on several lines
           that contains many <i>text</i>" as A1

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

A1 --> "Short activity"

@enduml
```



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.

```
@startuml

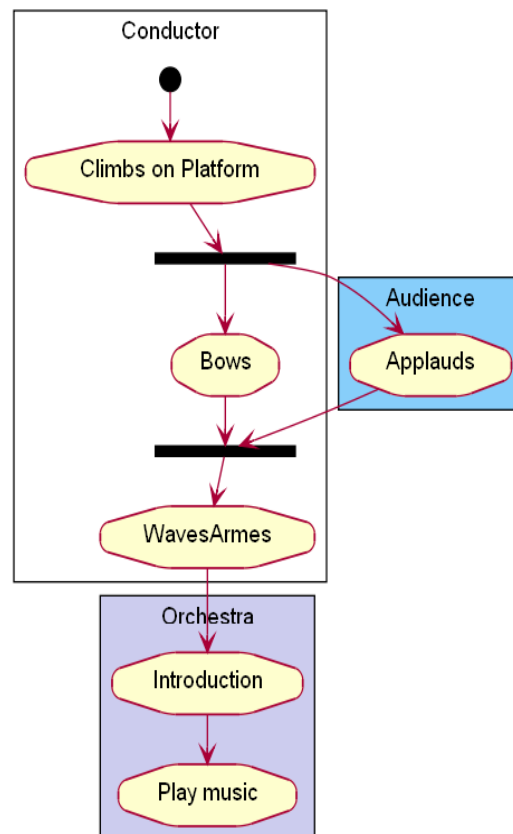
partition Conductor
$start --> "Climbs on Platform"
--> === S1 ===
--> Bows

partition Audience LightSkyBlue
=== S1 === --> Applauds

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

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

@enduml
```

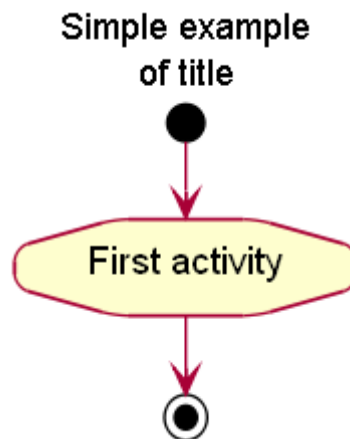


Title the diagram

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 example\nof title
$start --> "First activity"
--> $end
@enduml
```



Content

1- Sequence Diagram.....	2
Declaring participant.....	3
Message to Self.....	4
Message sequence numbering.....	5
Title.....	8
Splitting diagrams.....	9
Grouping message.....	10
Notes on messages.....	11
Some other notes.....	12
Formatting using HTML.....	13
Lifeline Activation and Destruction.....	14
Stereotypes and Spots.....	16
More information on titles.....	17
Removing Footer.....	19
Skin.....	20
2- Use Case Diagram.....	21
Links between actors and use cases.....	22
Extension.....	23
Using notes.....	24
Stereotypes.....	25
Title the diagram.....	26
A complete example.....	27
3- Classes Diagram.....	28
Label on relations.....	29
Adding methods.....	30
Notes and stereotypes.....	31
More on notes.....	32
Chained relations.....	33
Abstract class and interface.....	34
Specific Spot.....	35
Using packages.....	36
Title the diagram.....	37
Association classes.....	38
Splitting large files.....	40
4- Activity Diagram.....	41
Arrows.....	42
Branches.....	43
Synchronization.....	44
Long activity description.....	45
Partition.....	46
Title the diagram.....	47