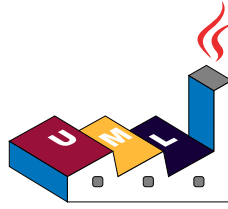


# Diagramando UML con PlantUML



## Guía de Referencia del Lenguaje (martes 17 de octubre de 2017 20:09)

**PlantUML** es un proyecto Open Source (código abierto) que permite escribir rápidamente:

- Diagramas de Secuencia,
- Diagramas de Casos de uso,
- Diagramas de Clases,
- Diagramas de Actividades,
- Diagramas de Componentes,
- Diagramas de Estados,
- Diagramas de Objetos.

Los diagramas son definidos usando un lenguaje simple e intuitivo.

# 1 Diagrama de Secuencia

## 1.1 Ejemplo básico

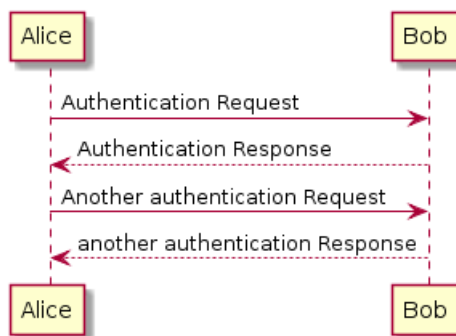
La secuencia `->` es usada para dibujar un mensaje entre dos participantes. Los participantes tienen que ser declarados explícitamente.

Para definir una flecha punteada, se debe usar `-->`

También se puede usar `<-` y `<--`. No provoca cambios en el dibujo, pero puede mejorar la legibilidad. Tenga en cuenta que esto sólo es posible en diagramas de secuencia, las reglas son diferentes para otros diagramas.

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

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



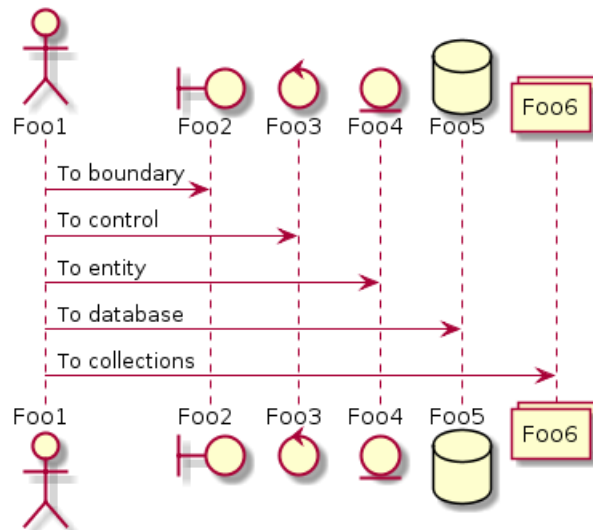
## 1.2 Declarando participantes

Es posible cambiar el orden de los participantes usando la palabra reservada `participant`.

También es posible el uso de otras palabras reservadas para declarar un participante:

- actor
- boundary
- control
- entity
- database

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



Se puede renombrar un participante usando la palabra reservada **as**.

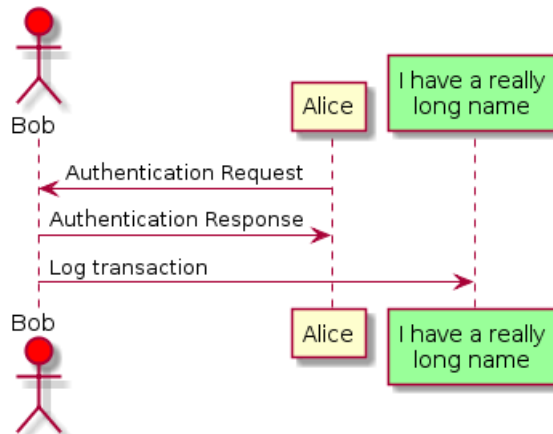
También es posible cambiar el color de fondo de los actores o participantes.

```

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

```

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

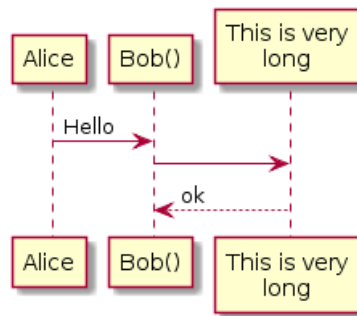


### 1.3 Sin usar letras en participantes

Puedes usar comillas para definir participantes. Y puedes usar la palabra reservada **as** para asignar un alias a esos participantes.

```

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



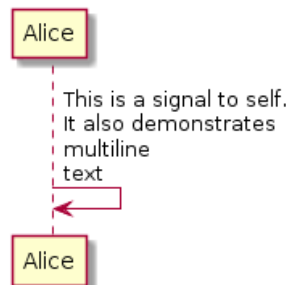
## 1.4 Auto-Mensaje

Un participante puede enviar mensajes así mismo.

También es posible tener un mensaje multi-líneas usando \n.

```

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



## 1.5 Cambiar estilo de la flecha

Puede cambiar el estilo de la flecha de diferentes formas:

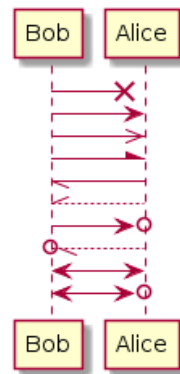
- añade una x al final para indicar un mensaje perdido
- utilice \ o / en lugar de < o > para tener solo la parte inferior o superior de la flecha
- repite la cabeza de la flecha (por ejemplo, >> o //) para tener un trazo más fino
- Utilice -- en lugar de - para obtener una flecha punteada.
- añade una "o" al final de la cabeza de una flecha
- utilice flechas bidireccionales

```

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

Bob ->o Alice
Bob o\-- Alice

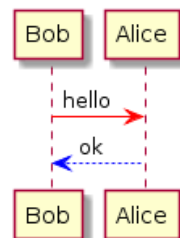
Bob <-> Alice
Bob <->o Alice
@enduml
  
```



## 1.6 Cambiar el color de la flecha

Puede cambiar el color de flechas individuales usando la siguiente notación:

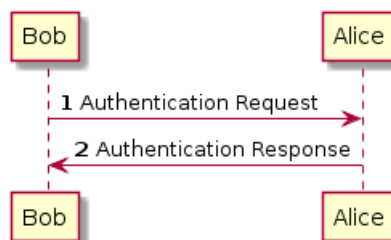
```
@startuml
Bob -[#red]> Alice : hello
Alice -[#0000FF]->Bob : ok
@enduml
```



## 1.7 Numeración de la secuencia de mensajes

La palabra clave `autonumber` es usada para añadir automáticamente números a los mensajes.

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



Puedes especificar un número de comienzo con `autonumber 'número inicial'`, y también un incremento con `autonumber 'número inicial' 'incremento'`.

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

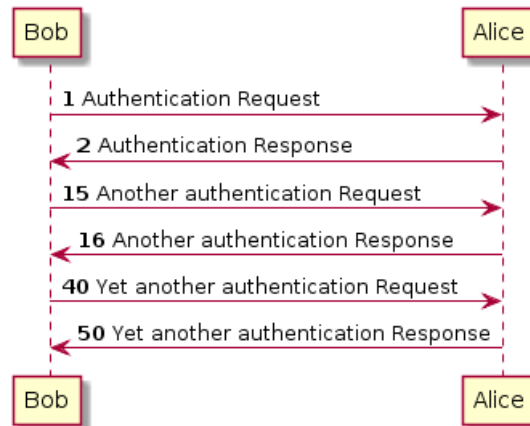
autonumber 15
Bob -> Alice : Another authentication Request
Bob <- Alice : Another authentication Response

autonumber 40 10
Bob -> Alice : Yet another authentication Request
```

```

Bob <- Alice : Yet another authentication Response
@enduml

```



Puedes especificar un formato para su número usándolo entre comillas dobles.

El formateo se hace mediante la clase Java `DecimalFormat` ('0' denota un dígito, '#' denota un dígito y cero si está ausente).

Puedes usar alguna etiqueta HTML en el formato.

```

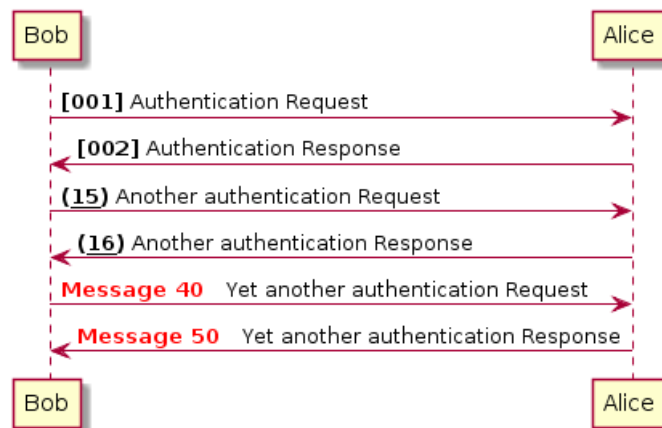
@startuml
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response

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

```



También puedes usar `autonumber stop` y `autonumber resume 'increment' 'format'` para pausar y continuar la numeración automática, respectivamente.

```

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

autonumber stop
Bob -> Alice : dummy

```

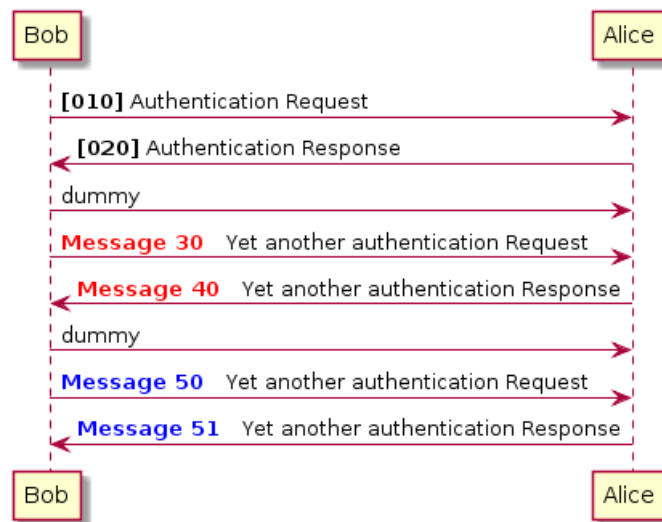
```

autonumber resume "<font color=red><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response

autonumber stop
Bob -> Alice : dummy

autonumber resume 1 "<font color=blue><b>Message 0  "
Bob -> Alice : Yet another authentication Request
Bob <- Alice : Yet another authentication Response
@enduml

```



## 1.8 Dividiendo diagramas

La palabra reservada `newpage` es empleada para dividir un diagrama en varias imágenes.

Puedes colocar un título para la página nueva justo después de la palabra reservada `newpage`.

Esto es bastante práctico con *Word* para devolver diagramas grandes en varias páginas.

```

@startuml

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

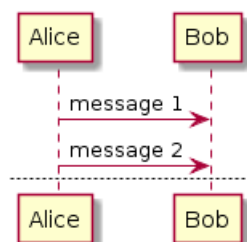
newpage

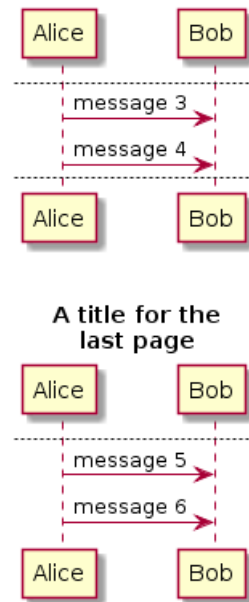
Alice -> Bob : message 3
Alice -> Bob : message 4

newpage A title for the\nlast page

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

```





## 1.9 Agrupando mensajes

Es posible agrupar mensajes usando las siguientes palabras reservadas:

- alt/else
- opt
- loop
- par
- break
- critical
- group, seguida de un texto para mostrar

Es posible añadir un texto que será mostrado en el encabezado (excepto para **group**).

La palabra reservada **end** es usada para cerrar el grupo.

Tenga en cuenta que es posible anidar grupos.

```

@startuml
Alice -> Bob: Authentication Request

alt successful case

Bob -> Alice: Authentication Accepted

else some kind of failure

Bob -> Alice: Authentication Failure
group My own label
Alice -> Log : Log attack start
loop 1000 times
Alice -> Bob: DNS Attack
end
Alice -> Log : Log attack end
end

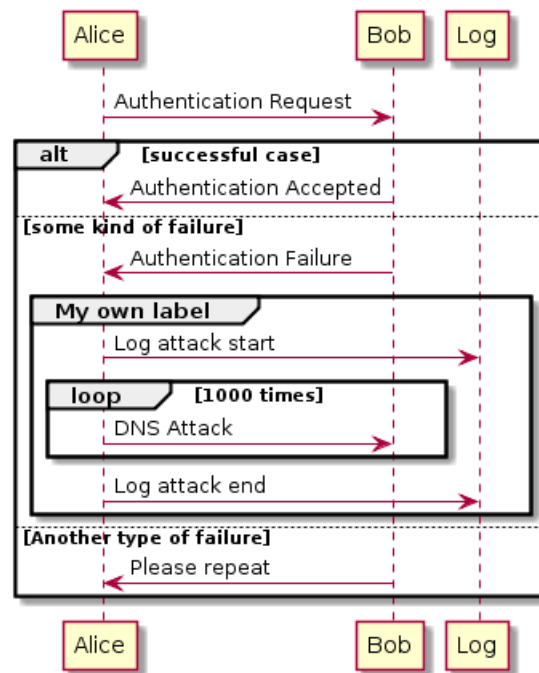
else Another type of failure

Bob -> Alice: Please repeat

end
@enduml
  
```







## 1.10 Notas en mensajes

Es posible colocar notas en mensajes usando las palabras reservadas `note left` o `note right` inmediatamente después del mensaje.

Puedes tener una nota multi-líneas usando la palabra reservada `end note`.

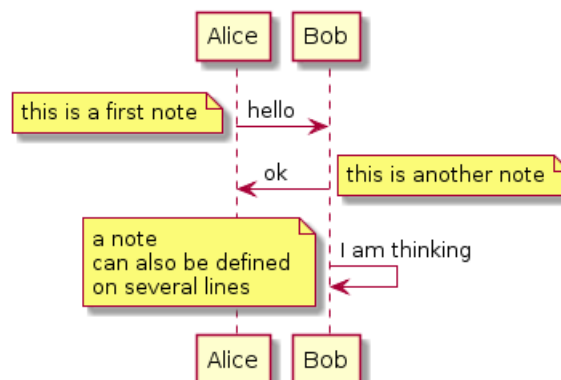
```

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

Bob->>Alice : ok
note right: this is another note

Bob->>Bob : I am thinking
note left
a note
can also be defined
on several lines
end note
@enduml

```



## 1.11 Algunas otras notas

También es posible colocar notas relativas al participante con las palabras reservadas `note left of`, `note right of` o `note over`.

Es posible resaltar una nota cambiando su color de fondo.



También puedes tener una nota multi-líneas usando la palabra reservada `end note`.

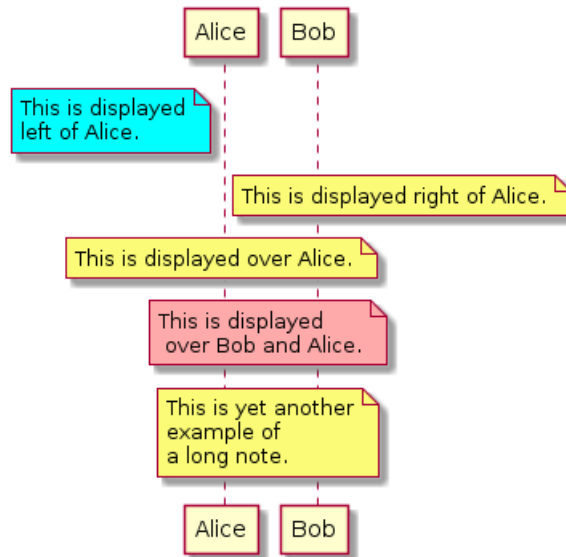
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

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

note over Alice: This is displayed over Alice.

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

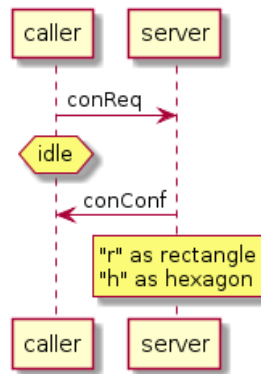
note over Bob, Alice
This is yet another
example of
a long note.
end note
@enduml
```



## 1.12 Cambiando el aspecto de las notas

Puedes usar las palabras reservadas `hnote` y `rnote` para cambiar el aspecto de las notas.

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



## 1.13 Creole y HTML

También es posible usar sintaxis de WikiCreole:

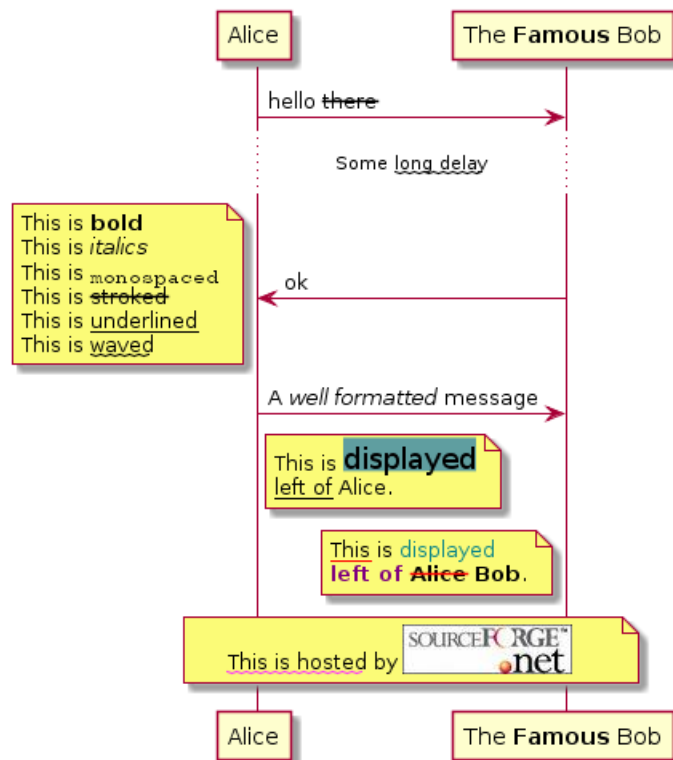
```

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

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

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

```



## 1.14 Divisor

Si quieres, puedes dividir un diagrama usando el separador == para separar su diagrama en pasos lógicos.

```
@startuml
```

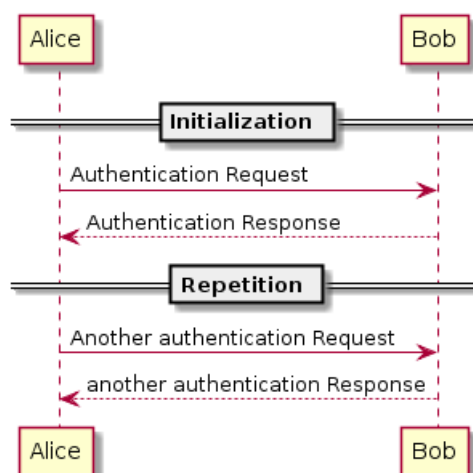
```
== Initialization ==
```

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

```
== Repetition ==
```

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

```
@enduml
```



## 1.15 Referencia

Puedes referenciar en un diagrama utilizando la palabra clave **ref** over.



```

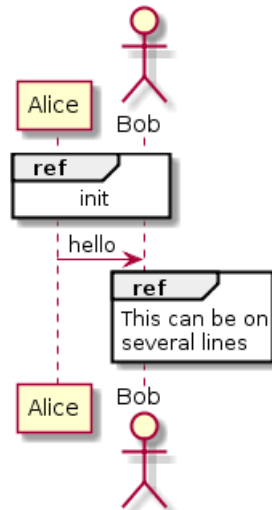
@startuml
participant Alice
actor Bob

ref over Alice, Bob : init

Alice -> Bob : hello

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

```



## 1.16 Retardo

Puedes usar ... para indicar un retardo en el diagrama. Y también es posible colocar un mensaje con ese retardo.

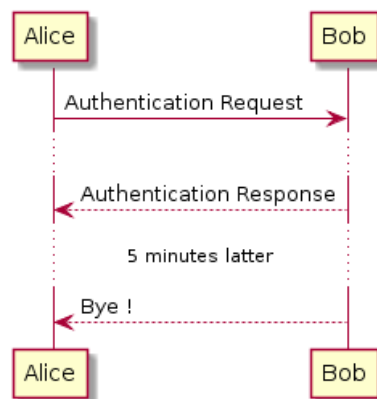
```

@startuml

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

@enduml

```



## 1.17 Espaciado

Puedes usar ||| para indicar espaciado en el diagrama.



También es posible especificar un número de píxel para ser usado.

```
@startuml
```

```
Alice -> Bob: message 1
Bob --> Alice: ok
|||
Alice -> Bob: message 2
Bob --> Alice: ok
||45||
Alice -> Bob: message 3
Bob --> Alice: ok
```

```
@enduml
```



## 1.18 Activación y Destrucción de la Línea de vida

`activate` y `deactivate` son usados para denotar la activación de un participante.

Una vez que un participante es activado, su línea de vida aparece.

`activate` y `deactivate` aplica en el mensaje anterior.

`destroy` denota el final de la línea de vida de un participante.

```
@startuml
```

```
participant User
```

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

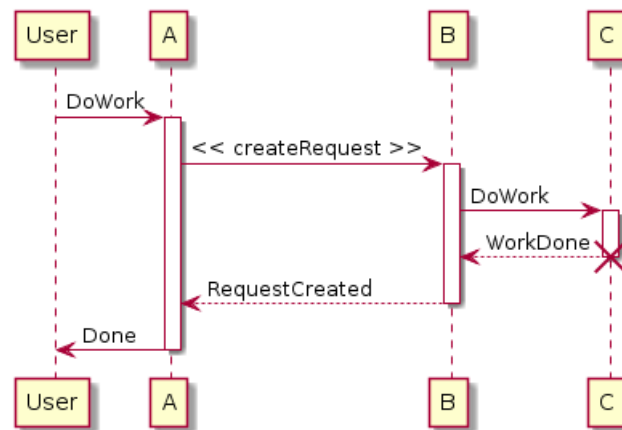
```
A -> B: << createRequest >>
activate B
```

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

```
B --> A: RequestCreated
deactivate B
```

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

```
@enduml
```



Puede usarse anidamiento de líneas de vida, y es posible agregar un color a dicha línea de vida.

```

@startuml
participant User

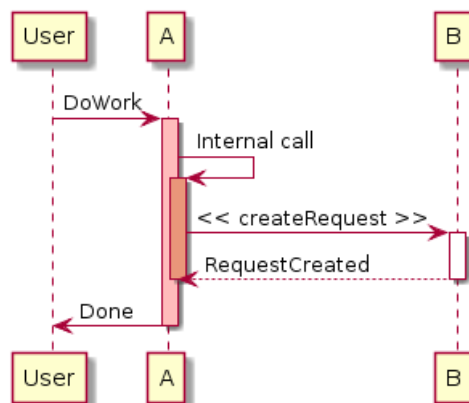
User -> A: DoWork
activate A #FFBBBB

A -> A: Internal call
activate A #DarkSalmon

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

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

@enduml
  
```



## 1.19 Creación de participante

Puedes usar la palabra reservada **create** justo antes de la primera recepción de un mensaje para recalcar el hecho de que ese mensaje se encuentra *creando* ese nuevo objeto.

```

@startuml
Bob -> Alice : hello

create Other
Alice -> Other : new

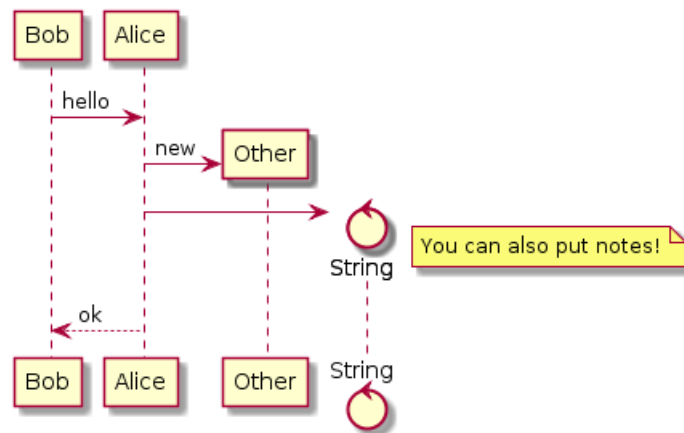
create control String
Alice -> String
note right : You can also put notes!
  
```



```

Alice --> Bob : ok
@enduml

```



## 1.20 Mensajes entrantes y salientes

Puedes usar flechas entrantes y salientes si quieres centrarte en una parte del diagrama.

Utilice corchetes para denotar el lado izquierdo "[" o el lado derecho "]" del diagrama.

```

@startuml
[-> A: DoWork

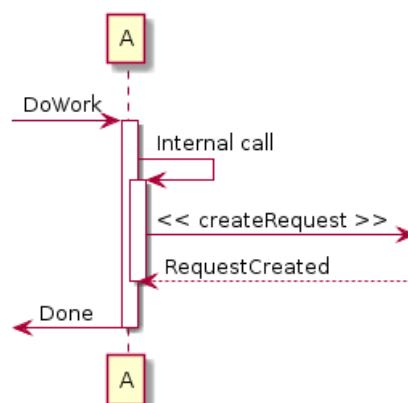
activate A

A -> A: Internal call
activate A

A -> ] : << createRequest >>

A<-- ] : RequestCreated
deactivate A
[<- A: Done
deactivate A
@enduml

```



También puedes tener la siguiente sintaxis:

```

@startuml
[-> Bob
[o-> Bob
[o->> Bob
[x-> Bob

[<- Bob
[x<- Bob

```





```

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

```

```

Bob <-]
Bob x<-]
@enduml

```



## 1.21 Estereotipos y marcas

Es posible añadir estereotipos a participantes usando << y >>.

En el estereotipo, puedes añadir un carácter marcado en un círculo coloreado usando la sintaxis (X,color).

```

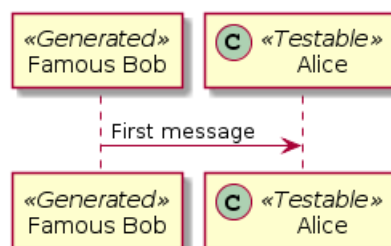
@startuml

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

Bob->>Alice: First message

@enduml

```



Por defecto, *guillemet* (comillas) son usadas para mostrar el estereotipo. Puedes cambiar este comportamiento usando `skinparam guillemet`:

```

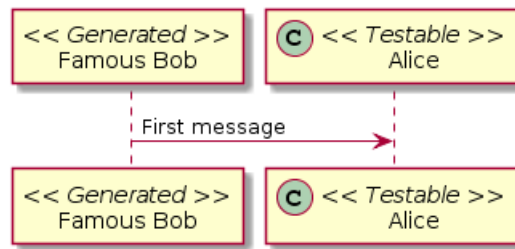
@startuml

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

Bob->>Alice: First message

@enduml

```



```

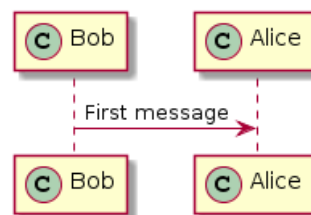
@startuml

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

Bob->>Alice: First message

@enduml

```



## 1.22 Mayor información en los títulos

Puedes usar sintaxis de Creole en el título.

```

@startuml

title __Simple__ **communication** example

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

@enduml

```

**Simple communication example**



Puedes añadir una nueva línea usando \n en la descripción del título.

```

@startuml

title __Simple__ communication example\nnon several lines

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

@enduml

```

**Simple communication example  
on several lines**

Además puedes definir un título en varias líneas usando las palabras reservadas `title` y `end title`.

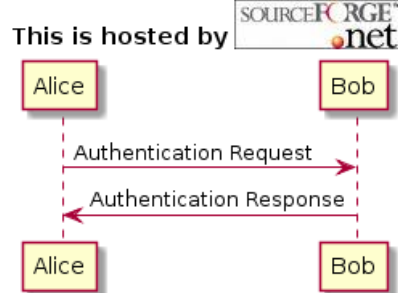
```

@startuml

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

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

@enduml
  
```

**Simple communication example  
on several lines and using `html`****1.23 Entorno de participante**

Es posible dibujar una caja alrededor de algunos participantes, usando los comandos `box` y `end box`.

Puedes añadir un título opcional o un color de fondo opcional, después de la palabra reservada `box`.

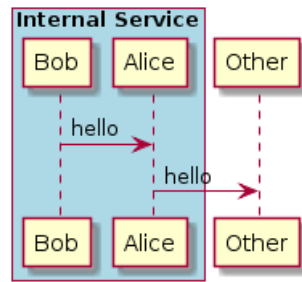
```

@startuml

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

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

@enduml
  
```



## 1.24 Removiendo pie de página

Puedes usar las palabras reservadas `hide footbox` para remover el pie de página del diagrama.

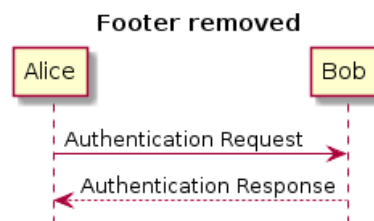
```

@startuml

hide footbox
title Footer removed

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

@enduml
  
```



## 1.25 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y fuentes en el dibujo.

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

También puedes cambiar otros parámetros de renderización, como se ve en los siguientes ejemplos

```

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

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

User -> A: DoWork
activate A

A -> B: Create Request
activate B

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

```

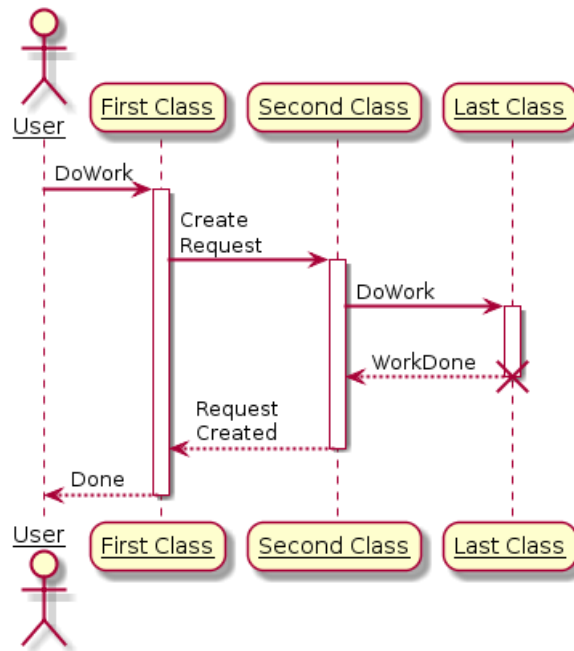
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



```

@startuml
skinparam backgroundColor #EEEBDC
skinparam handwritten true

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

    ParticipantBorderColor DeepSkyBlue
    ParticipantBackgroundColor DodgerBlue
    ParticipantFontName Impact
    ParticipantFontSize 17
    ParticipantFontColor #A9DCDF

    ActorBackgroundColor aqua
    ActorFontColor DeepSkyBlue
    ActorFontSize 17
    ActorFontName Aapex
}

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

User -> A: DoWork
activate A

A -> B: Create Request
activate B

B -> C: DoWork
activate C

C --> B: WorkDone
deactivate C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



```

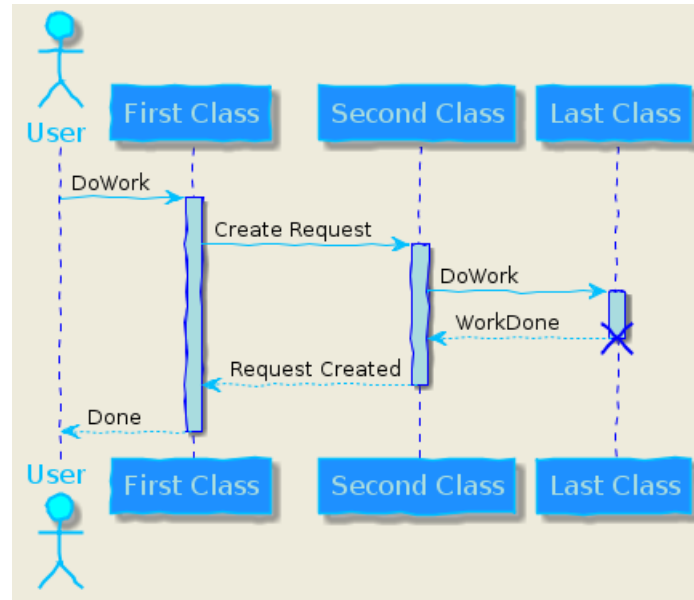
C --> B: WorkDone
destroy C

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml

```



## 1.26 Cambiando el relleno

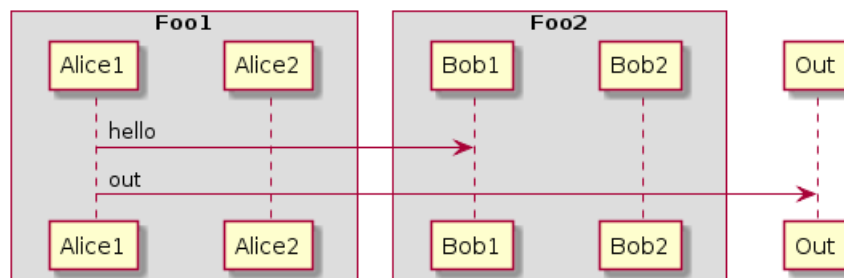
Es posible ajustar algunos parámetros de relleno

```

@startuml
skinparam ParticipantPadding 20
skinparam BoxPadding 10

box "Foo1"
participant Alice1
participant Alice2
end box
box "Foo2"
participant Bob1
participant Bob2
end box
Alice1 -> Bob1 : hello
Alice1 -> Out : out
@enduml

```



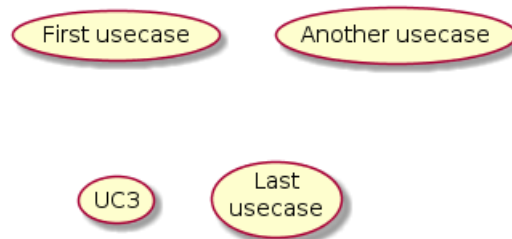
## 2 Diagrama de Casos de Uso

### 2.1 Casos de uso

Los casos de uso estan encerrados entre paréntesis (los paréntesis tienen un aspecto similar a un óvalo).

También puede usar la palabra **usecase** para crear un caso de uso. Además puede crear un alias, usando la palabra **as**. Este alias será usado mas adelante, cuando defina relaciones.

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



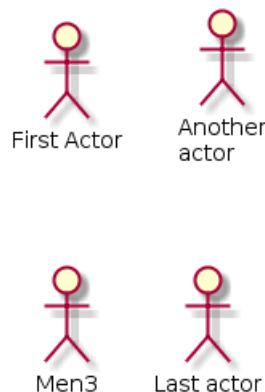
### 2.2 Actores

Los actores se encierran entre dos puntos.

También puedes usar la palabra reservada **actor** para definir un actor. Además puedes definir un alias, usando la palabra reservada **as**. Este alias será usado más adelante, cuando definamos relaciones.

Veremos más adelante que las declaraciones de los actores son opcionales.

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



### 2.3 Descripción de Casos de uso

Si quiere realizar una descripción en varias líneas, puede usar citas ( " " ).

También puede usar los siguientes separadores: -- .. == \_\_. Y puede introducir títulos dentro de los separadores.



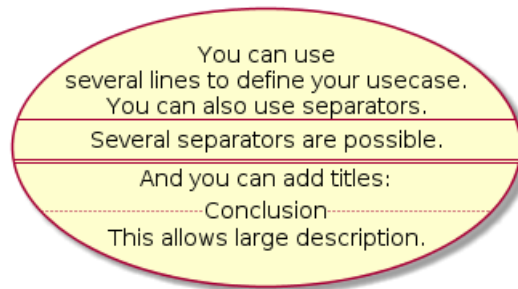
```

@startuml

usecase UC1 as "You can use
several lines to define your usecase.
You can also use separators.
--
Several separators are possible.
==
And you can add titles:
..Conclusion..
This allows large description."

@enduml

```



## 2.4 Ejemplo básico

Para relacionar actores y casos de uso, la flecha --> es usada.

Cuanto más guiones "-" en la flecha, más larga será la misma. Puedes añadir una etiqueta en la flecha, añadiendo el carácter ":" en la definición de la flecha.

En este ejemplo, puedes ver que *User* no ha sido definido, y es usado como un actor.

```

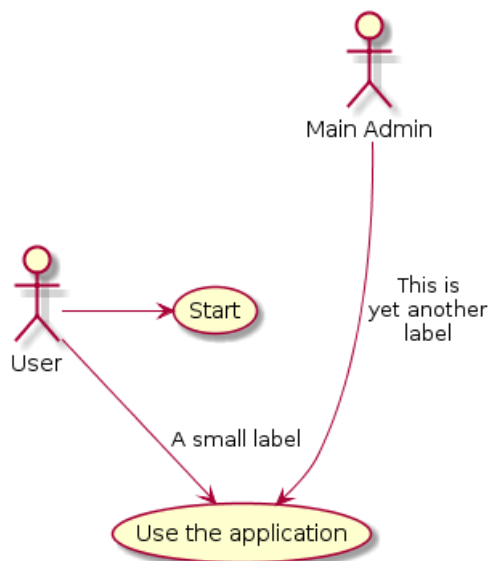
@startuml

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

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

@enduml

```



## 2.5 Extensión

Si un actor/caso de uso extiende a otro, puedes usar el símbolo <|-- (que significa ).





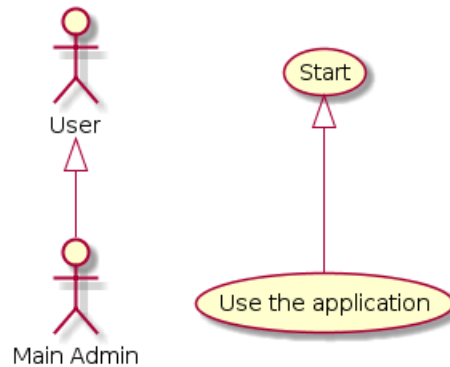
```

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

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

@enduml

```



## 2.6 Usando notas

Puedes usar las palabras claves: `note left of` , `note right of` , `note top of` , `note bottom of` , para añadir notas relacionadas a un objeto en particular.

También se puede añadir una nota solitaria con la palabra clave `note`, y después relacionarla con otro objeto usando el símbolo `..` .

```

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

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

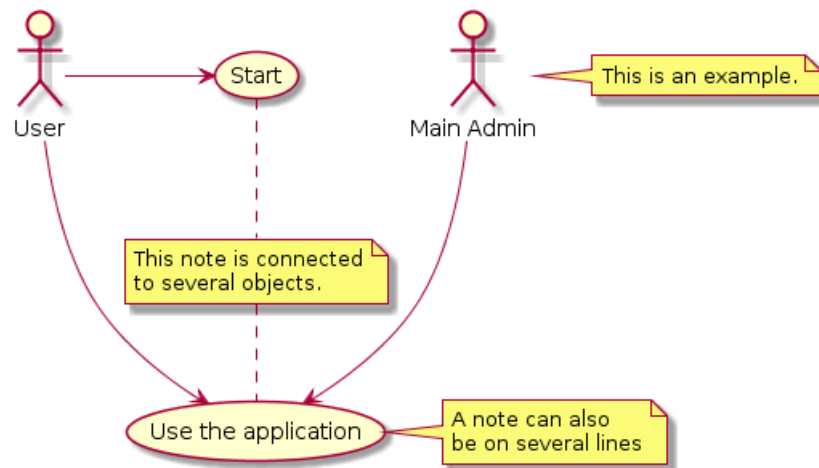
Admin ---> (Use)

note right of Admin : This is an example.

note right of (Use)
A note can also
be on several lines
end note

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

```



## 2.7 Estereotipos

Puedes añadir estereotipos mientras defines actores y casos de uso, usando " << " y " >> ".

```

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

```

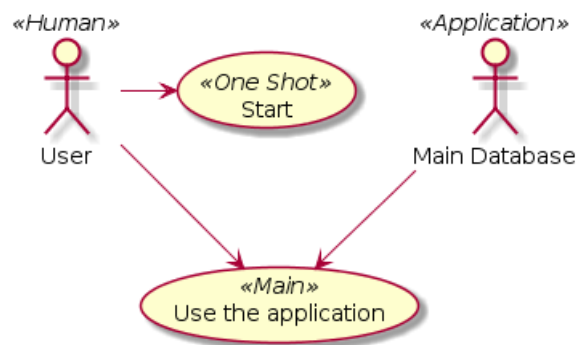
User -> (Start)
User --> (Use)
  
```

```

MySql --> (Use)
  
```

```

@enduml
  
```

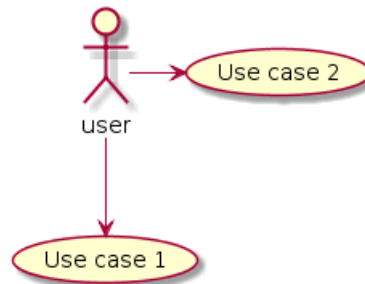


## 2.8 Cambiar dirección a las flechas

Por defecto, conexiones entre clases tiene dos guiones -- y son verticalmente orientadas. Es posible usar una conexión horizontal, colocando un solo guión (o punto) de esta forma:

```

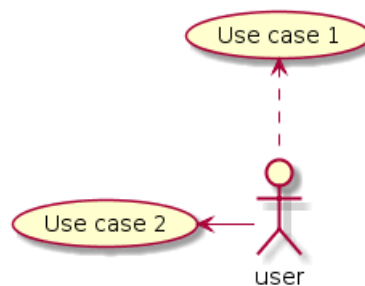
@startuml
:user: --> (Use case 1)
:user: -> (Use case 2)
@enduml
  
```



También puedes cambiar de dirección revirtiendo la conexión:

```

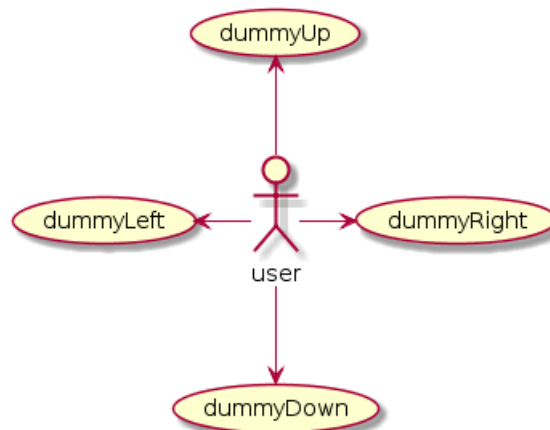
@startuml
(Use case 1) <.. :user:
(Use case 2) <- :user:
@enduml
  
```



También es posible cambiar la dirección de una flecha añadiendo las palabras clave **left**, **right**, **up** or **down**, dentro de la misma:

```

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



Puedes acortar la flecha usando sólo el primer carácter de la dirección (por ejemplo, **-d-** en lugar de **-down-**) o los primeros dos caracteres (**-do-**).

Por favor tenga en cuenta que no debería abusar de esta funcionalidad : *Graphviz* usualmente devuelve buenos resultados sin realizar muchos ajustes.

## 2.9 Dividiendo los diagramas

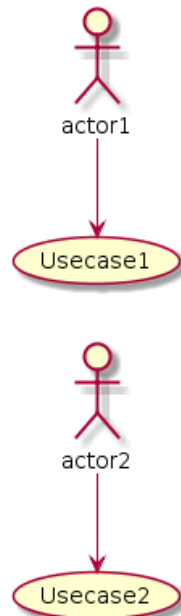
La palabra clave **newpage** divide su diagrama en varias páginas o imágenes.



```

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

```



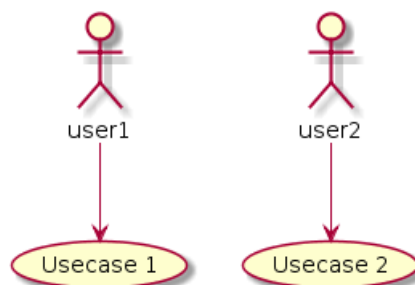
## 2.10 Dirección: de izquierda a derecha

El comportamiento general cuando se construye un diagrama, es **top to bottom**.

```

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

```

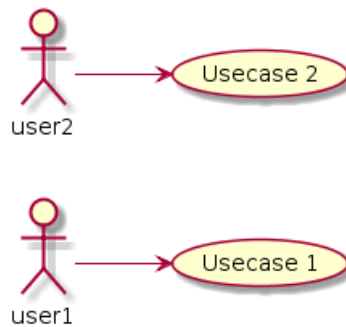


Puede cambiar a **left to right** usando el comando **left to right direction**. En ocasiones, el resultado es mejor con esta dirección.

```

@startuml
left to right direction
user1 --> (Usecase 1)
user2 --> (Usecase 2)
@enduml

```



## 2.11 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y las fuentes de los dibujos

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para los actores y casos de uso estereotipados.

```

@startuml
skinparam handwritten true

skinparam usecase {
  BackgroundColor DarkSeaGreen
  BorderColor DarkSlateGray
}

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

ArrowColor Olive
ActorBorderColor black
ActorFontName Courier

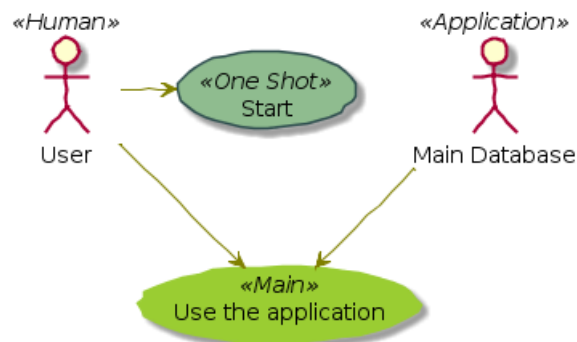
ActorBackgroundColor<< Human >> Gold
}

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

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

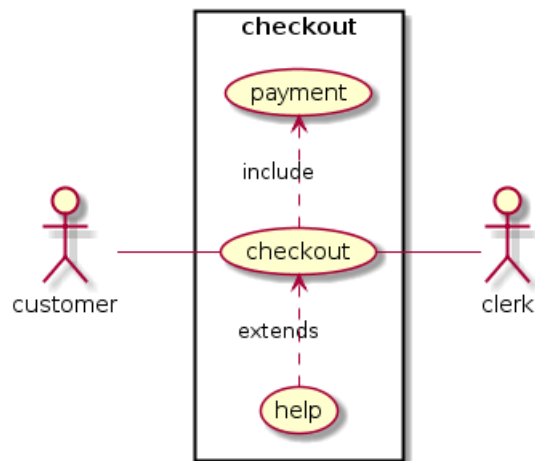
MySql --> (Use)

@enduml
  
```



## 2.12 Un ejemplo completo




```
@startuml
left to right direction
skinparam packageStyle rectangle
actor customer
actor clerk
rectangle checkout {
customer -- (checkout)
(checkout) .> (payment) : include
(help) .> (checkout) : extends
(checkout) -- clerk
}
@enduml
```



## 3 Diagrama de Clases

### 3.1 Relación entre clases

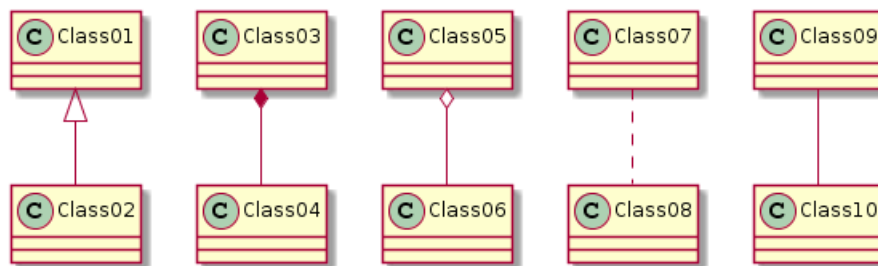
Las relaciones entre clases se definen usando los siguientes símbolos:

Extensión	< --	
Composición	*--	
Agregación	o--	

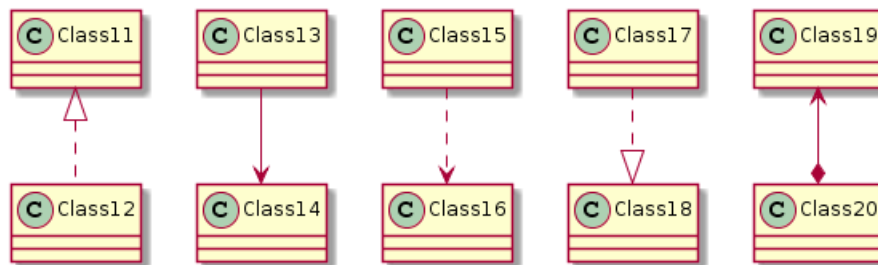
Es posible intercambiar -- por .. para tener líneas punteadas.

Sabiendo esas reglas, es posible sacar los siguientes dibujos:

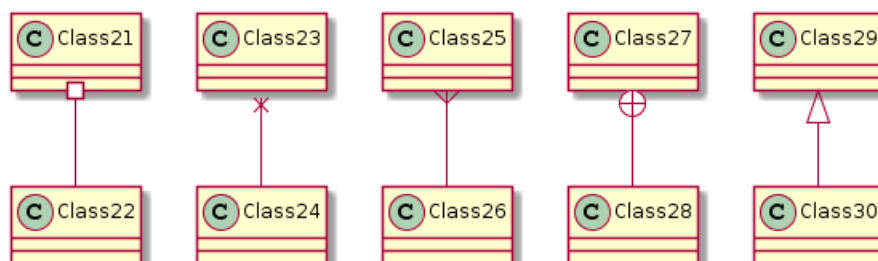
```
@startuml
Class01 <|-- Class02
Class03 *-- Class04
Class05 o-- Class06
Class07 .. Class08
Class09 -- Class10
@enduml
```



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



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

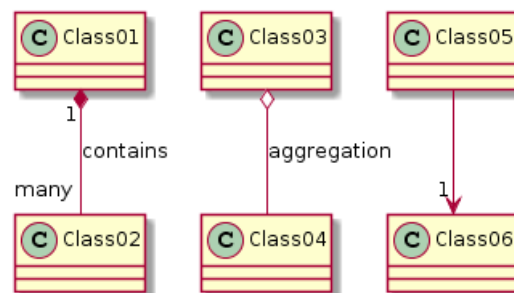


### 3.2 Etiquetas en las relaciones

Es posible añadir etiquetas en las relaciones, usando ":", seguido del texto de la etiqueta.

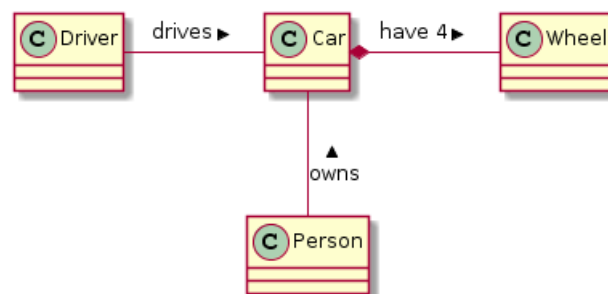
Para la cardinalidad, puede usar comillas dobles "" en cada lado de la relación.

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



Se puede añadir una flecha extra apuntando a un objeto, mostrando que objeto actúa sobre el otro objeto, usando < o > al inicio o al final de la etiqueta.

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





### 3.3 Añadiendo métodos

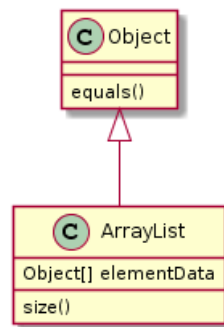
Para declarar las propiedades y métodos, se puede usar el símbolo ":" seguido del nombre de la propiedad o el método.

El sistema busca por paréntesis para elegir entre métodos y propiedades.

```
@startuml
Object <|-- ArrayList

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

@enduml
```



También es posible agrupar entre llaves { todos las propiedades y métodos.

Tenga en cuenta que la sintaxis es muy flexible acerca del orden tipo/nombre.

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

class Flight {
flightNumber : Integer
departureTime : Date
}

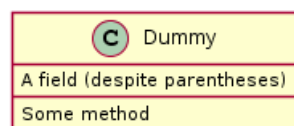
@enduml
```



Puede usar los modificadores {field} y {method} para modificar el comportamiento por defecto del parse sobre los campos y métodos.

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

@enduml
```

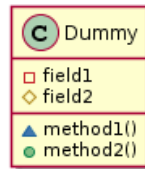


### 3.4 Definiendo la visibilidad

Cuando defines propiedades o métodos, puedes usar caracteres para establecer la visibilidad que les correspondan:

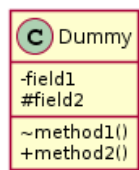
-	□	■	privado
#	◇	◆	protegido
~	△	▲	paquete privado
+	○	●	público

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



Puedes desactivar esta característica usando el comando `skinparam classAttributeIconSize 0`:

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

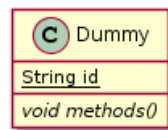


### 3.5 Abstracto y Estático

Puedes definir métodos o propiedades abstractas y estáticas usando los modificadores `static` o `abstract`.

Esos modificadores pueden ser usado al comienzo o al final de un línea. También puedes usar `classifier` en lugar de `static`.

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



### 3.6 Cuerpo avanzado de las clases

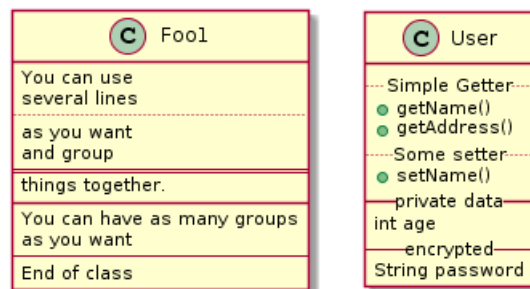
Por defecto, las propiedades y los métodos son agrupados automáticamente por PlantUML. Puedes usar separadores para definir tu propia manera de ordenar las propiedades y los métodos. Son posibles los siguientes separadores: -- .. == \_\_.

También puedes definir títulos dentro de los separadores:

```
@startuml
class Foo1 {
You can use
several lines
..
as you want
and group
==
things together.
--
You can have as many groups
as you want
--
End of class
}

class User {
.. Simple Getter ..
+ getName()
+ getAddress()
.. Some setter ..
+ setName()
__ private data __
int age
-- encrypted --
String password
}

@enduml
```



### 3.7 Notas y estereotipos

Los estereotipos son definidos con la palabra clave **class** " << " and " >> " .

También puedes definir notas usando las palabras claves **note left of** , **note right of** , **note top of** , **note bottom of** .

Además puedes definir una nota en la última clase definida usando **note left**, **note right**, **note top**, **note bottom** .

Una nota también puede definirse solitariamente con la palabra clave **note**, y a continuación relacionarla con otro objeto usando el símbolo . . .

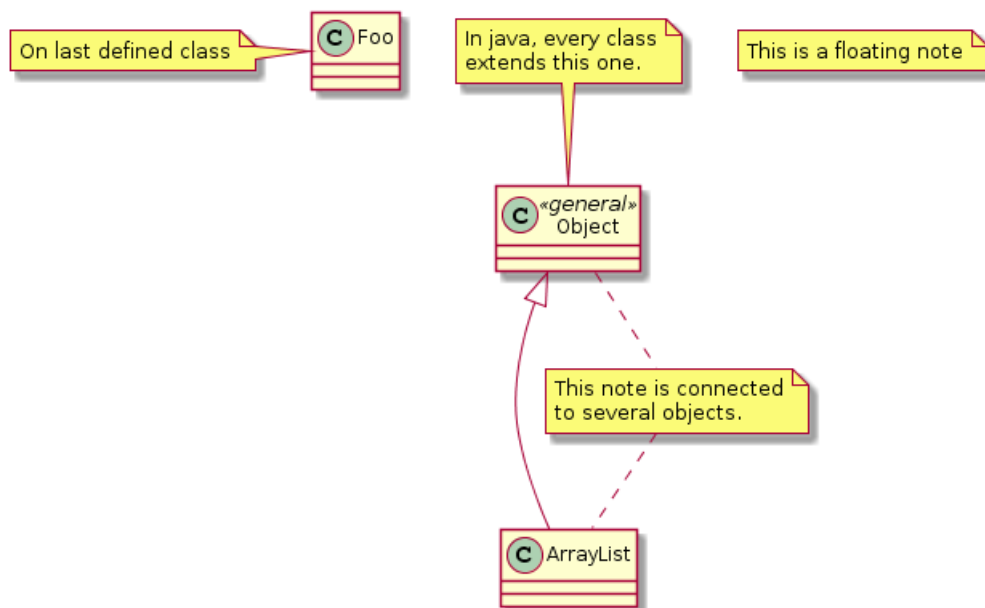
```
@startuml
class Object << general >>
Object <|--- ArrayList

note top of Object : In java, every class\nextends this one.

note "This is a floating note" as N1
note "This note is connected\nto several objects." as N2
Object .. N2
N2 .. ArrayList

class Foo
note left: On last defined class

@enduml
```



### 3.8 Más acerca de notas

También es posible usar algunas etiquetas HTML como:

- `<b>`
- `<u>`
- `<i>`
- `<s>`, `<del>`, `<strike>`
- `<font color="#AAAAAA">` or `<font color="colorName">`
- `<color:#AAAAAA>` or `<color:colorName>`
- `<size:nn>` to change font size
- `` or `<img:file>` : the file must be accessible by the filesystem

También puedes tener una nota en varias líneas.

Es posible definir una nota en la última clase definida usando `note left`, `note right`, `note top`, `note bottom`.

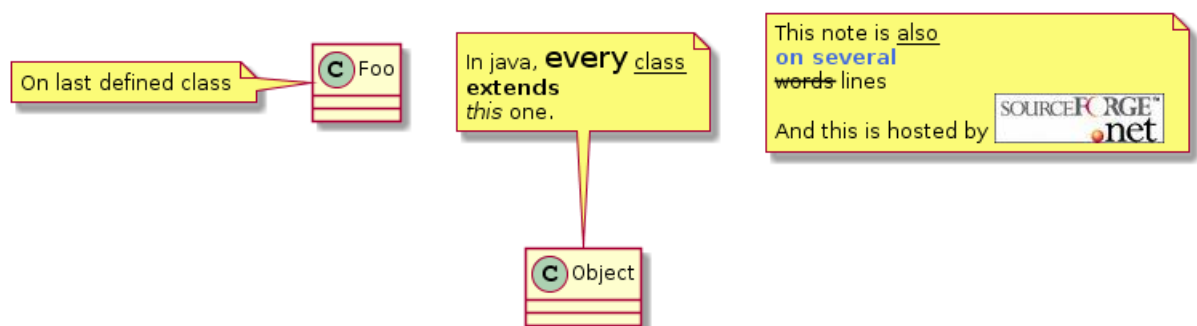
```
@startuml

class Foo
note left: On last defined class

note top of Object
In java, <size:18>every</size> <u>class</u>
<b>extends</b>
<i>this</i> one.
end note

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

@enduml
```



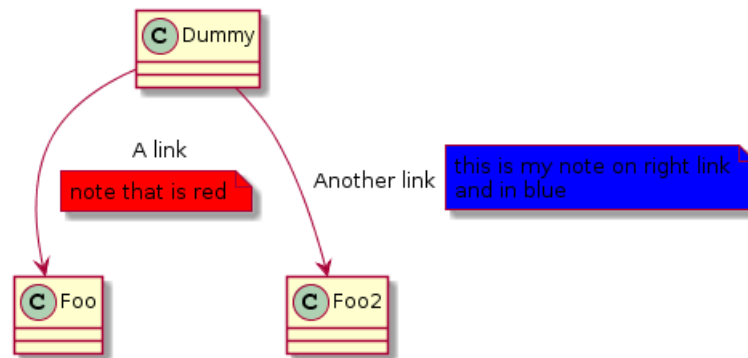
### 3.9 Notas en enlaces

Es posible añadir una nota en un enlace, justo después de la definición de dicho enlace, usando `note on link`.

También puedes usar `note left on link`, `note right on link`, `note top on link`, `note bottom on link` si quieres cambiar la posición de la nota, en relación a una etiqueta.

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

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



### 3.10 Clases abstractas e interfaces

Puedes declarar una clase como abstracta usando las palabras claves "abstract" or "abstract class".

La clase será impresa en *italic*.

Puedes usar también las palabras claves `interface`, `annotation` and `enum`.

```
@startuml

abstract class AbstractList
abstract AbstractCollection
interface List
interface Collection

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

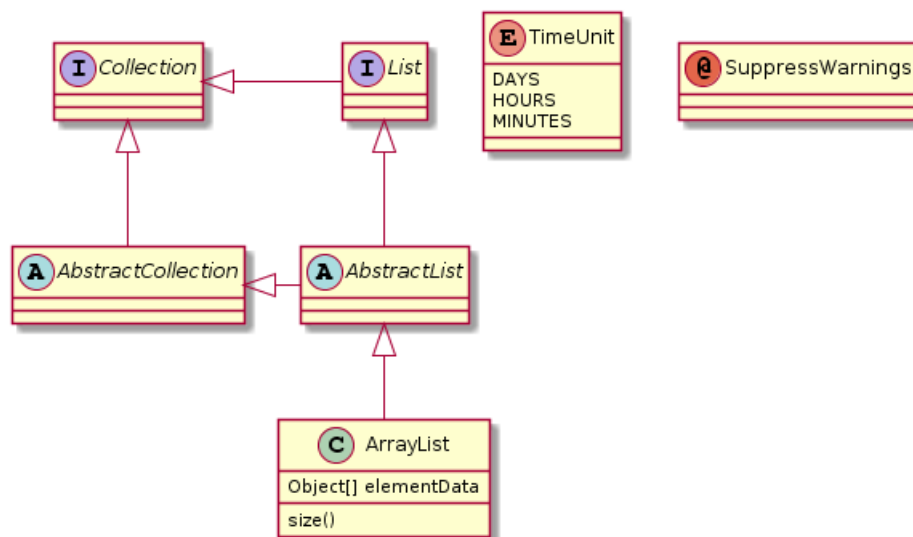
Collection <|-- List
AbstractCollection <|-- AbstractList
AbstractList <|-- ArrayList

class ArrayList {
Object[] elementData
size()
}

enum TimeUnit {
DAYS
HOURS
MINUTES
}

annotation SuppressWarnings

@enduml
```





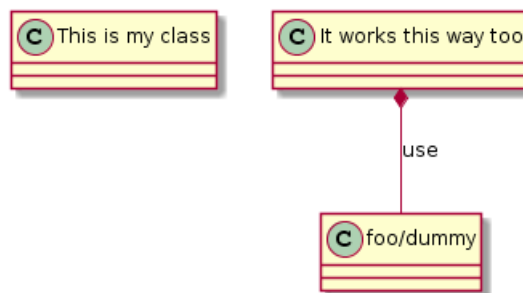
### 3.11 Sin usar letras

Si no desea usar letras en la visualización de la clase (o enum...), puede:

- Utilizar la palabra reservada **as** en la definición de la clase
- Colocar comillas "" alrededor del nombre de la clase

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

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



### 3.12 Atributos, métodos... ocultos

Puede parametrizar la visualización de las clases usando el comando `hide/show` .

El comando básico es: `hide empty members`. Este comando ocultará atributos y métodos si están vacíos.

En lugar de `empty members` , puedes usar:

- `empty fields` o `empty attributes` para atributos vacíos.
- `empty methods` para métodos vacíos,
- `fields` o `attributes` que ocultará atributos, incluso si son descritos,
- `methods` que ocultará métodos, incluso si son descritos,
- `members` que ocultará atributos y métodos, incluso si son descritos.
- `circle` para el carácter encerrado en un círculo, en frente del nombre de clase.
- `stereotype` para el estereotipo.

También puede proporcionar, justo después las palabras clave `hide` o `show`:

- `class` para todas las clases,
- `interface` para todas las interfaces,
- `enum` para todos los enums,
- `<<foo1>>` para clases que son estereotipadas con *foo1*,
- un nombre de clase existente.

Puedes usar varios comandos `show/hide` para definir reglas y excepciones.

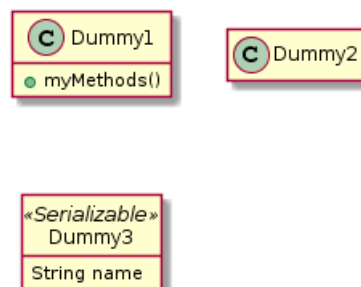
```
@startuml
class Dummy1 {
+myMethods()
}

class Dummy2 {
+hiddenMethod()
}

class Dummy3 <<Serializable>> {
String name
}

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

@enduml
```



### 3.13 Clases ocultas

También puedes usar el comando `show/hide` para ocultar clases.

Esto puede llegar a ser útil si defines una archivo `!included` muy grande y si deseas ocultar algunas clases después de la inclusión de dicho archivo.

```
@startuml
class Foo1
class Foo2

Foo2 --- Foo1

hide Foo2

@enduml
```

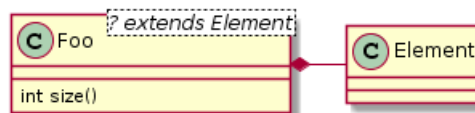


### 3.14 Uso de clases genéricas

También puedes usar los signos menor `<` y mayor `>` para definir el uso de clases genéricas.

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

@enduml
```



Es posible desactivar este dibujo con el comando `skinparam genericDisplay old`.

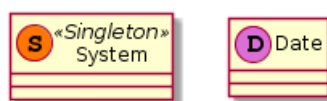
### 3.15 Círculo enmarcador específico

Usualmente, un carácter enmarcado en un círculo (C,I,E o A) es usado por clases, interfaces, enum y clases abstractas.

Pero puedes definir tu propio enmarcado para una clase cuando defines un estereotipo, añadiendo un carácter y un color, así como en el ejemplo:

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

@enduml
```



### 3.16 Paquetes

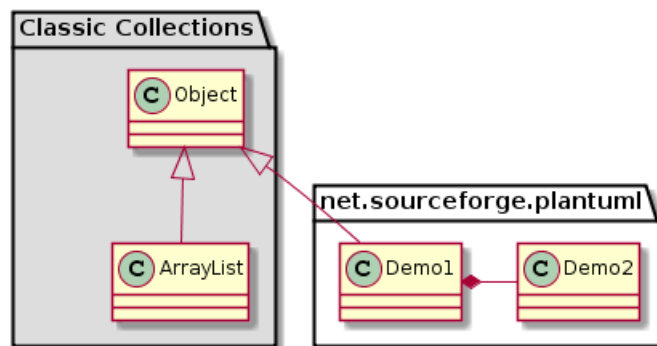
Puedes definir un paquete usando la palabra reservada **package**, y opcionalmente declarar un color de fondo para tu paquete (Usando el nombre o el código HTML del color).

Tenga en cuenta que las definiciones de paquetes pueden ser anidadas.

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

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

@enduml
```



### 3.17 Estilos de paquetes

Hay diferentes estilos disponibles para paquetes.

Puedes especificarlos, ya sea configurando un estilo por defecto con el comando : **skinparam packageStyle** , o usando un estereotipo en el paquete.

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

package foo2 <<Rectangle>> {
class Class2
}

package foo3 <<Folder>> {
class Class3
}

package foo4 <<Frame>> {
class Class4
}

package foo5 <<Cloud>> {
class Class5
}

package foo6 <<Database>> {
class Class6
}

@enduml
```





Puedes también definir enlaces entre paquetes, como en el siguiente ejemplo:

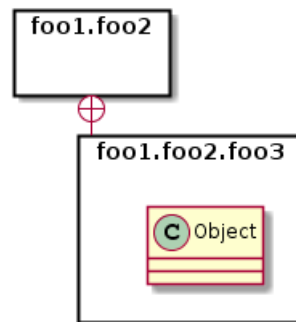
```
@startuml
skinparam packageStyle rectangle

package foo1.foo2 {
}

package foo1.foo2.foo3 {
class Object
}

foo1.foo2 +-- foo1.foo2.foo3

@enduml
```



### 3.18 Espacios de nombre

En los paquetes, el nombre de una clase es el único identificador de esta clase. Quiere decir que no puedes tener dos clases con el mismo nombre en diferentes paquetes.

En este caso, deberías usar espacios de nombres en lugar de paquetes.

Puedes referir a clases de otros espacios de nombre describiendo su ruta completamente. A clases del espacio de nombre por defecto son descritas colocando un punto al inicio.

Tenga en cuenta que no tiene que especificar explícitamente un espacio de nombre : una clase altamente clasificada es automáticamente colocada en el espacio de nombre correcto.

```
@startuml
class BaseClass

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

.BaseClass <|-- Meeting
}

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



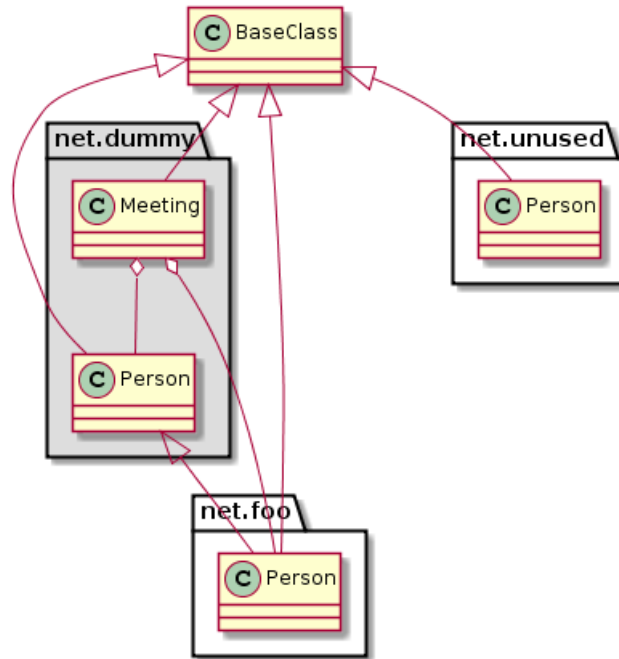
```

net.dummy.Meeting o-- Person
}

BaseClass <|-- net.unused.Person

@enduml

```



### 3.19 Creación automática del espacio de nombre

Puedes definir otro separador (otro además del punto) usando el comando : `set namespaceSeparator ???`.

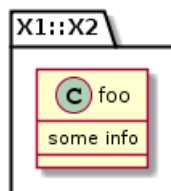
```

@startuml

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

@enduml

```



Puedes deshabilitar la creación automática de paquetes usando el comando `set namespaceSeparator none`.

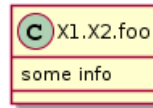
```

@startuml

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

@enduml

```

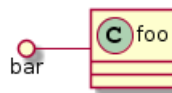


### 3.20 Interface Lollipop

También puedes definir interfaces lollipops en clases, usando la siguiente sintaxis:

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

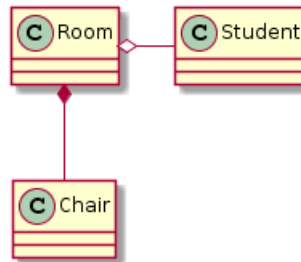
```
@startuml
class foo
bar ()- foo
@enduml
```



### 3.21 Cambiando la dirección de las flechas

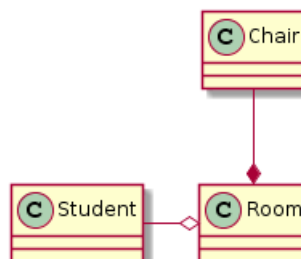
Por defecto, enlaces entre clases tienen dos guiones -- y son verticalmente orientados. Es posible usar un enlace horizontal colocando un solo guión (o punto), así:

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



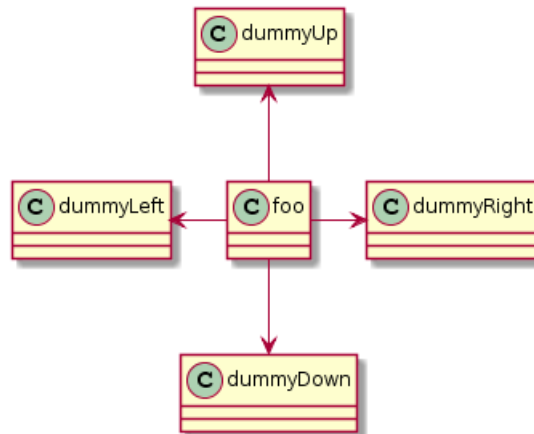
También puedes cambiar las direcciones revirtiendo el enlace:

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



También es posible cambiar la dirección de la flecha añadiendo las palabras claves `left`, `right`, `up` or `down` dentro de la flecha:

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



Puedes acortar la flecha usando el primer carácter de la dirección (por ejemplo, `-d-` en lugar de `-down-`) o los primeros dos caracteres.

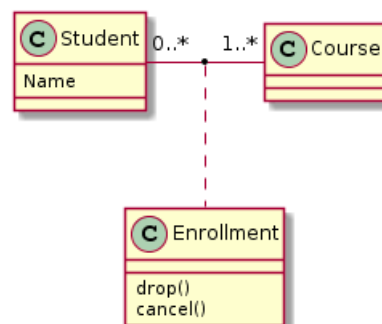
Por favor tenga en cuenta que no debería abusar de esta funcionalidad : *Graphviz* usualmente otorga buenos resultados sin necesidad de ajustar.

### 3.22 Asociación de clases

Puedes definir *association class* después de que una relación haya sido establecida entre dos clases, como en este ejemplo:

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

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



Puedes definirla en otra dirección:

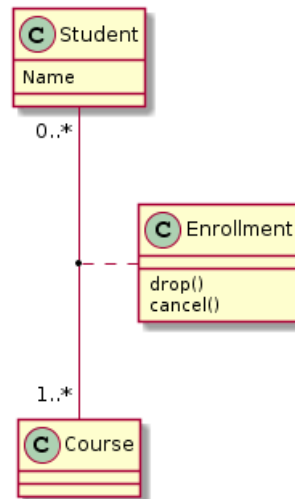




```

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

```



### 3.23 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y fuentes en el diagrama.

Puedes usar este comando:

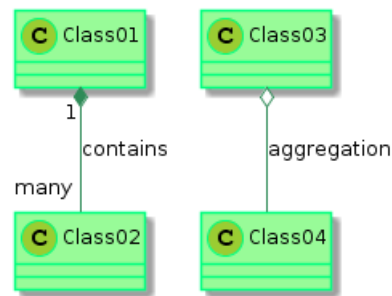
- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

```

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

Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
@enduml

```



### 3.24 Estereotipos personalizados

Puedes definir colores y fuentes específicas para clases esterotipadas.

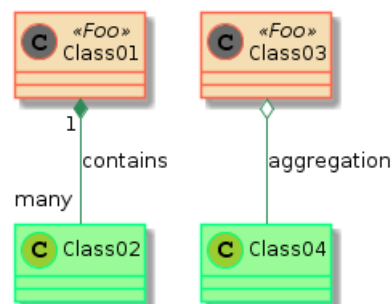
```

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

Class01 <<Foo>>
Class03 <<Foo>>
Class01 "1" *-- "many" Class02 : contains

Class03 o-- Class04 : aggregation

@enduml
  
```



### 3.25 Degrado de colores

Es posible declarar colores individuales para clases o notas usando la notación .

Puedes usar el nombre estándar del color o el código RGB.

También puedes usar degradación de color en el fondo, con la siguiente sintaxis: dos nombres de colores separados por cualquier de los siguientes:

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

dependiendo de la dirección del degradado.

Por ejemplo, podrías tener:



```

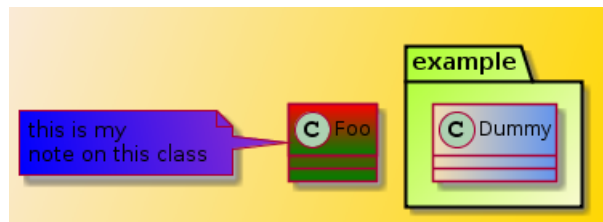
@startuml
skinparam backgroundColor AntiqueWhite/Gold
skinparam classBackgroundColor Wheat|CornflowerBlue

class Foo #red-green
note left of Foo #blue\9932CC
this is my
note on this class
end note

package example #GreenYellow/LightGoldenRodYellow {
class Dummy
}

@enduml

```



### 3.26 Ayudar en el diseño

Sometimes, the default layout is not perfect...

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

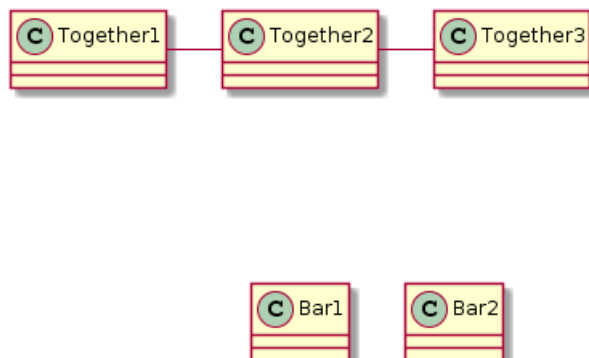
You can also use **hidden** links to force the layout.

```

@startuml
class Bar1
class Bar2
together {
class Together1
class Together2
class Together3
}
Together1 - Together2
Together2 - Together3
Together2 -[hidden]--> Bar1
Bar1 -[hidden]> Bar2

@enduml

```



### 3.27 Dividiendo archivos grandes

A veces, puedes obtener imágenes bastante grandes.

Puedes usar el comando "page (hpages)x(vpages)" para dividir la imagen generada, en varias imágenes:

**hpages** es el número que indica la cantidad de páginas horizontales, y **vpages** es el número que indica la cantidad de páginas verticales.

También puede utilizar algunos ajustes específicos skinparam poner fronteras en las páginas splitted (ver ejemplo).

```
@startuml
' Split into 4 pages
page 2x2
skinparam pageMargin 10
skinparam pageExternalColor gray
skinparam pageBorderColor black

class BaseClass

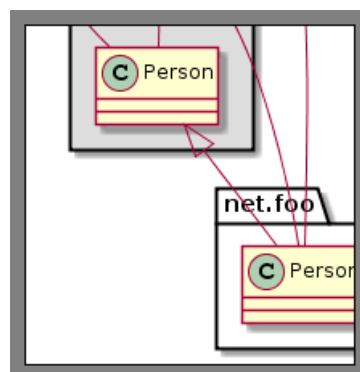
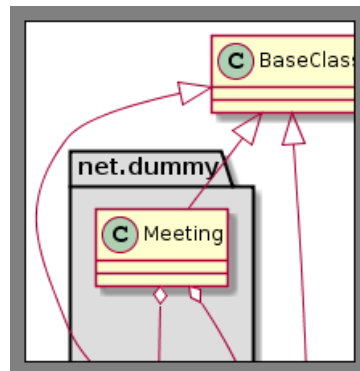
namespace net.dummy #DDDDDD {
.BaseClass <|-- Person
Meeting o-- Person

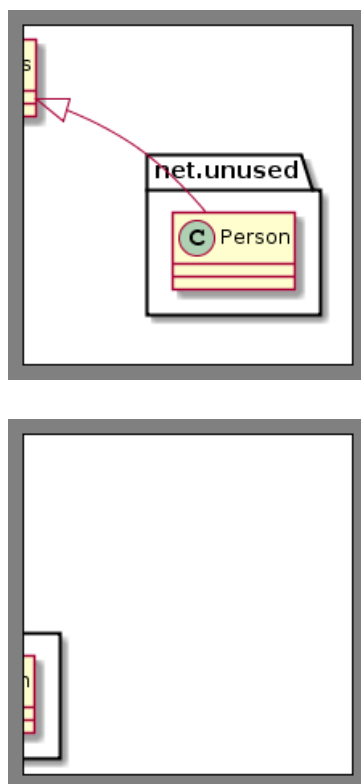
.BaseClass <|-- Meeting
}

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

net.dummy.Meeting o-- Person
}

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





## 4 Diagrama de Actividades

### 4.1 Actividades simples

Puedes usar (\*) para el punto de inicio y de final en un diagrama de actividades.

En algunos casos, quizás quieras usar (\*top) para forzar a que el punto de inicio se ubique en la parte superior del diagrama.

Utilice --> para las flechas.

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

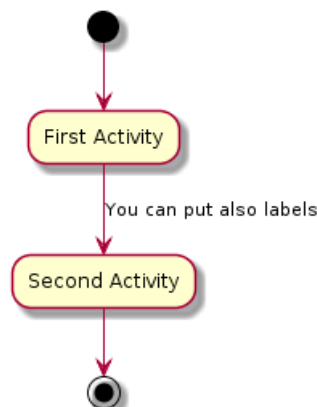


### 4.2 Etiquetas en las flechas

Por defecto, una flecha comienza en la última actividad usada.

Puedes colocar una etiqueta sobre una flecha usando corchetes, [ ] , justo después de la definición de la flecha.

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



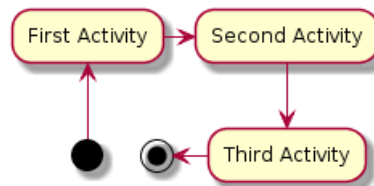
### 4.3 Cambiando la dirección de la flecha

Puedes usar -> para flechas horizontales. Es posible forzar la dirección de una flecha usando la siguiente sintaxis:



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

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

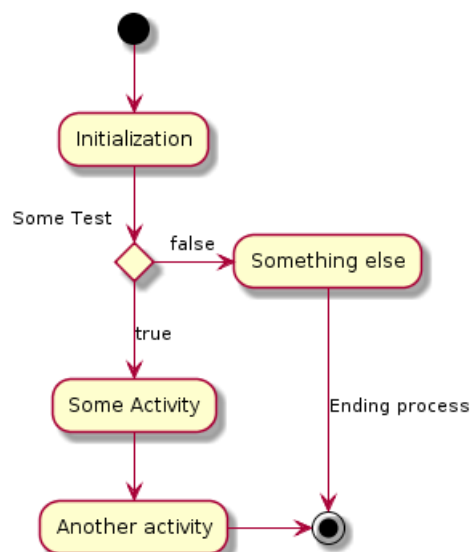


#### 4.4 Ramas (bifurcaciones)

Puedes usar las palabras reservadas `if/then/else` para definir ramas o caminos alternos.

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

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

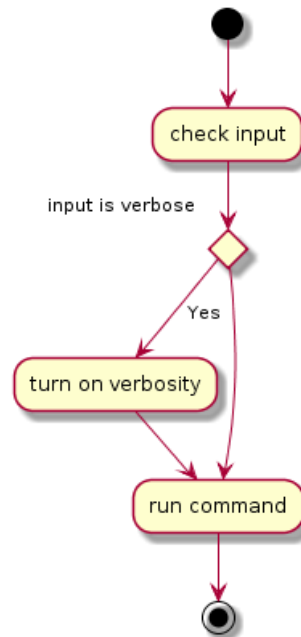


Desafortunadamente, a veces tendrás que repetir la misma actividad en el texto del diagrama:

```

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

```



## 4.5 Más acerca de las Ramas

Por defecto, una rama es conectada con la última actividad definida, pero es posible sobrescribir esto y definir un enlace con la palabra reservada `if`.

También es posible anidar ramas.

```

@startuml
(*) --> if "Some Test" then

-->[true] "activity 1"

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

else

->[false] "activity 2"

endif

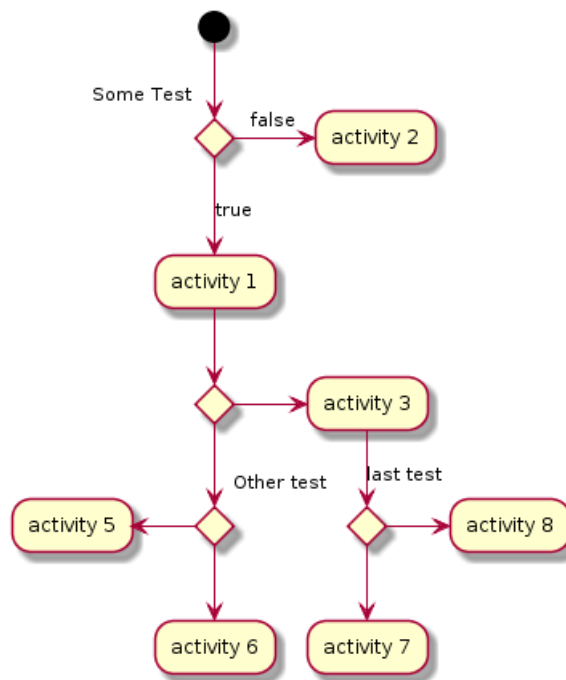
a3 --> if "last test" then
--> "activity 7"
else
-> "activity 8"
endif

```





```
endif
@enduml
```



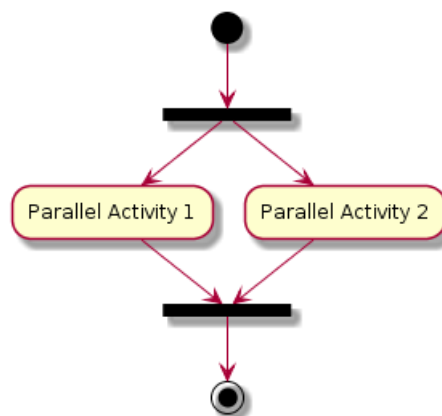
## 4.6 Sincronización

Puedes usar `=== code ===` para mostrar barras de sincronización.

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

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

--> (*)
@enduml
```



## 4.7 Descripción de actividades de gran contenido

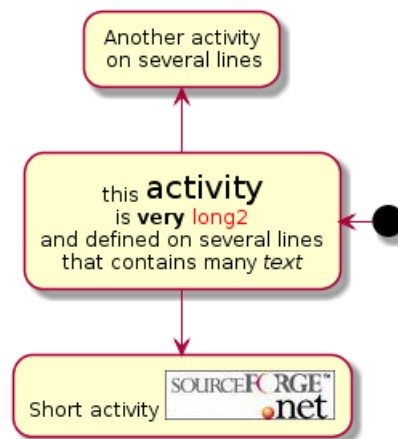
Cuando declaras actividades, puedes abarcar la descripción del texto, en varias líneas. También puedes añadir `\n` en la descripción.

También puedes colocar una pequeña cantidad de código en la actividad, con la palabra reservada `as`. Este código puede usarse más adelante en la descripción del diagrama.

```
@startuml
(*) -left-> "this <size:20>activity</size>
is <b>very</b> <color:red>long2</color>
and defined on several lines
that contains many <i>text</i>" as A1

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

A1 --> "Short activity <img:sourceforge.jpg>"
@enduml
```



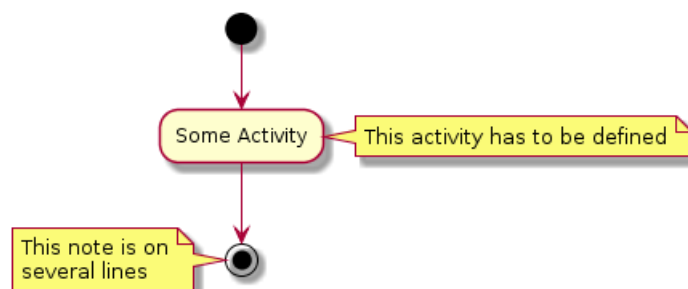
## 4.8 Notas

Puedes añadir notas sobre la actividad usando los comandos `note left`, `note right`, `note top` or `note bottom`, justo después de la descripción de la actividad a la quieres añadirle la nota.

Si quieres colocar una nota sobre el punto de inicio, define la nota al comienzo de la descripción del diagrama.

También puedes tener una nota de varias líneas, usando la palabra reservada `endnote`.

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



## 4.9 Partición

Puedes definir una partición usando la palabra reservada `partition` y opcionalmente declarar un color de fondo para tu partición (Usando el nombre o código HTML del color)

Cuando declaras actividades, éstas son automáticamente colocadas en la última partición usada.

Puedes cerrar la partición usando una llave de cierre `}`.

```
@startuml

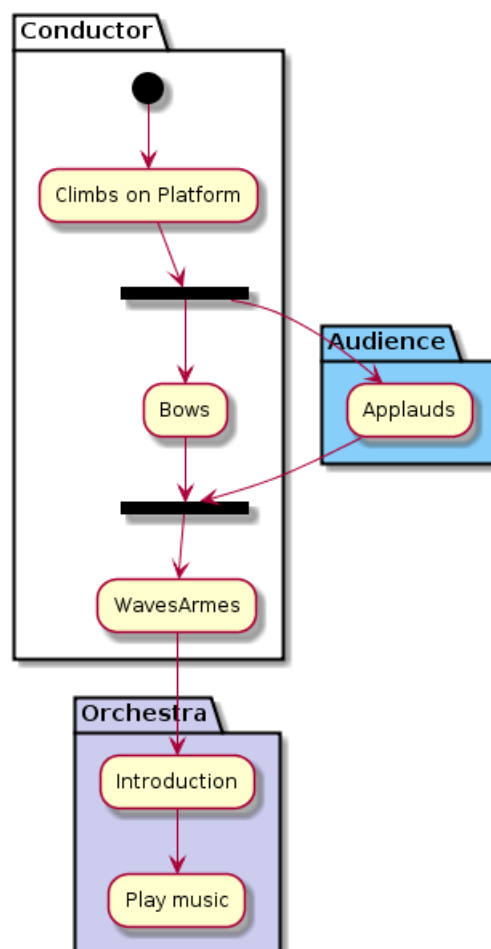
partition Conductor {
(*) --> "Climbs on Platform"
--> === S1 ===
--> Bows
}

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

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

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

@enduml
```



## 4.10 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar las fuentes y colores en el diagrama.

Puedes usar este comando :

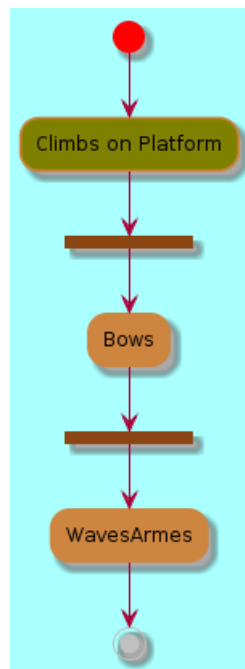
- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para actividades estereotipadas.

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

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

@enduml
```



## 4.11 Octágono

Puedes cambiar la forma de las actividades a un octágono usando el comando `skinparam activityShape octagon`.

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



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

@enduml
```



## 4.12 Un ejemplo completo

```
@startuml
title Servlet Container

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

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

if "isForward?" then
->[no] "Process controls"

if "continue processing?" then
-->[yes] ===RENDERING===
else
->[no] ===REDIRECT_CHECK===
endif

else
-->[yes] ===RENDERING===
endif

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

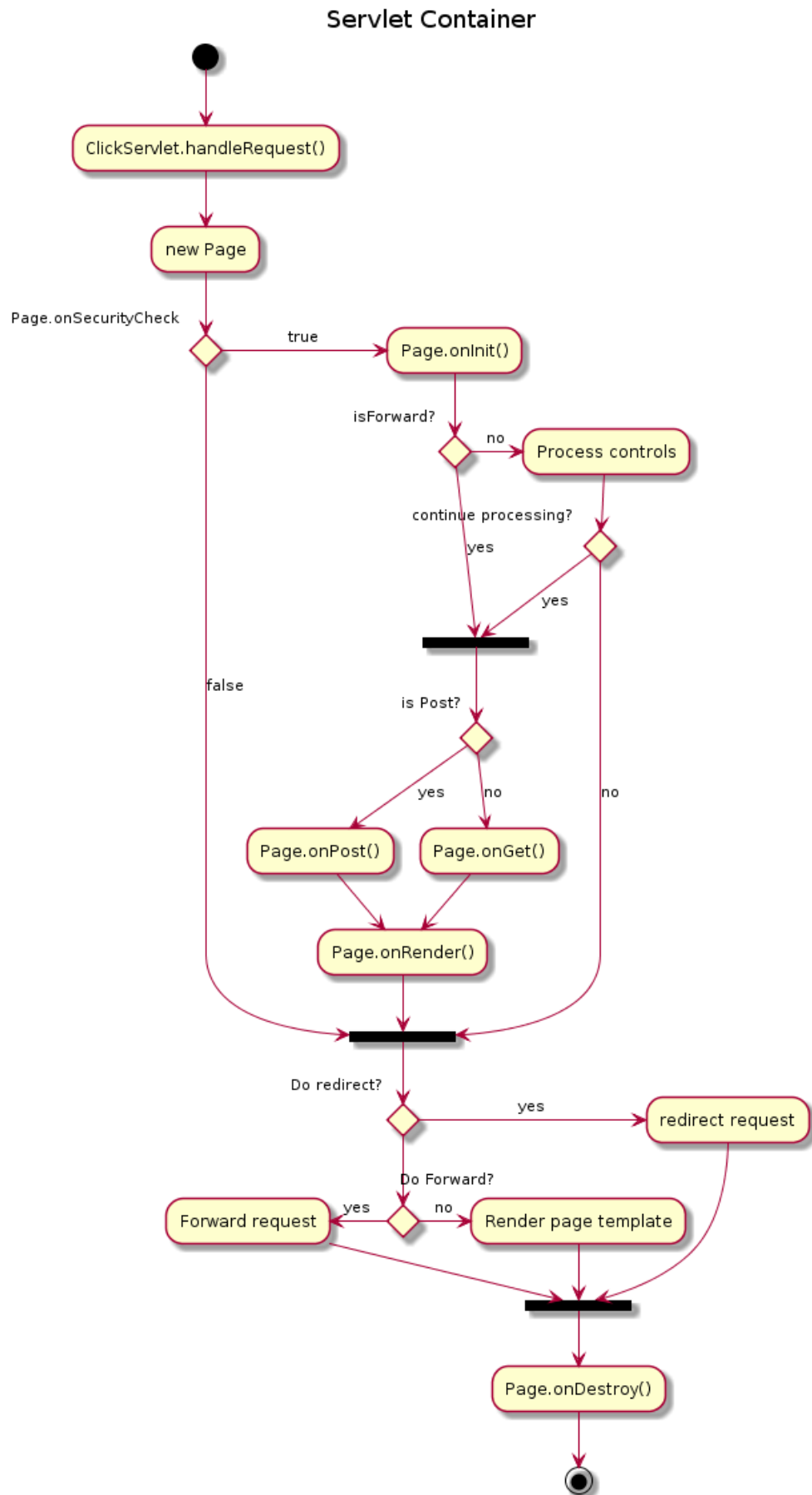
else
-->[false] ===REDIRECT_CHECK===
endif

if "Do redirect?" then
->[yes] "redirect request"
--> ==BEFORE_DESTROY==
else
if "Do Forward?" then
-left->[yes] "Forward request"
--> ==BEFORE_DESTROY==
else
-right->[no] "Render page template"
--> ==BEFORE_DESTROY==
endif
endif

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

@enduml





## 5 Diagrama de Actividades (beta)

La actual sintaxis para los diagramas de actividades tiene varias limitaciones e inconvenientes (por ejemplo, es difícil de mantener).

Entonces, una implementación y sintaxis nueva propuesta como **versión beta**, es ofrecida a los usuarios (empezando con V7947), sólo así podremos definir un mejor formato y sintaxis.

Otra ventaja de esta nueva implementación es que acaba con la necesidad de tener Graphviz instalado (como en los diagramas de secuencia).

La nueva sintaxis reemplazará la anterior. Sin embargo, por razones de compatibilidad, la sintaxis vieja seguirá siendo reconocida, para asegurarse la *retrocompatibilidad*.

Los usuarios están siendo motivados para migrarse a la nueva sintaxis.

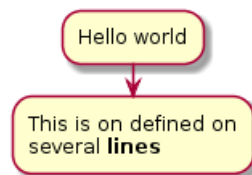
### 5.1 Una Actividad simple

Las etiquetas de las actividades inician con un dos puntos (:) y terminan con un punto y coma (;).

Se puede aplicar formato a un texto usando sintaxis de WikiCreole.

Están implícitamente enlazados en el orden de su definición.

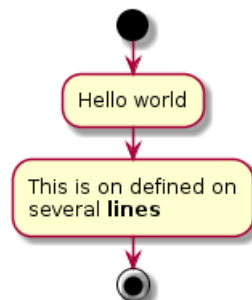
```
@startuml
:Hello world;
:This is on defined on
several lines;
@enduml
```



### 5.2 Start/Stop

Puedes usar las palabras reservadas **start** y **stop** para denotar el comienzo y el final del diagrama.

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



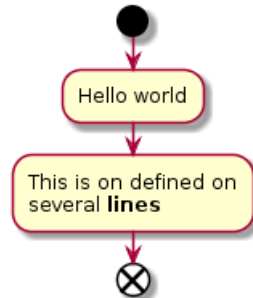
También puedes usar la palabra reservada **end**.



```

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

```



### 5.3 Condicionales

Puedes usar las palabras reservadas `if`, `then` y `else` para colocar condiciones en tus diagramas. Las etiquetas pueden ser proporcionadas usando paréntesis.

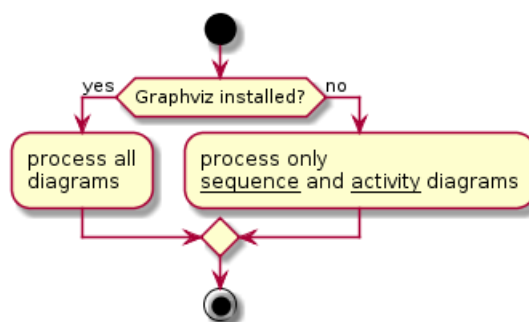
```

@startuml
start

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

stop
@enduml

```



Puedes usar la palabra reservada `elseif` para tener varias condiciones :

```

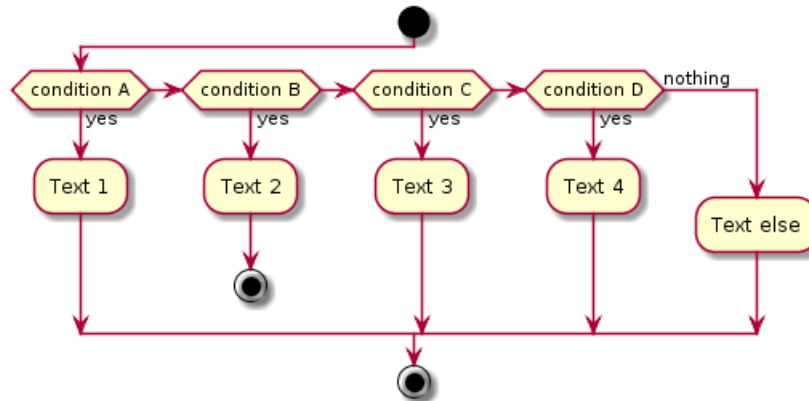
@startuml
start
if (condition A) then (yes)
:Text 1;
elseif (condition B) then (yes)
:Text 2;
stop
elseif (condition C) then (yes)
:Text 3;
elseif (condition D) then (yes)
:Text 4;
endif

```

```

else (nothing)
:Text else;
endif
stop
@enduml

```



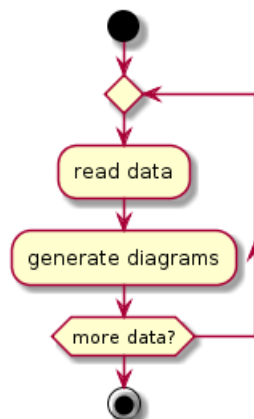
## 5.4 El ciclo Repeat

Puedes usar las palabras reservadas `repeat` y `repeatwhile` para colocar bucles.

```

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

```



## 5.5 El ciclo While

Puedes usar las palabras reservadas `while` y `end while` para un ciclo repetitivo.

```

@startuml
start
while (data available?)

```

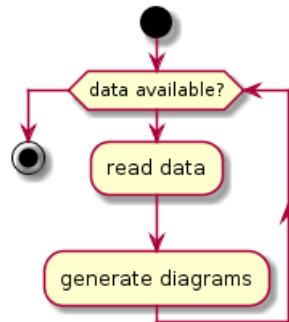
```

:read data;
:generate diagrams;
endwhile

stop

@enduml

```

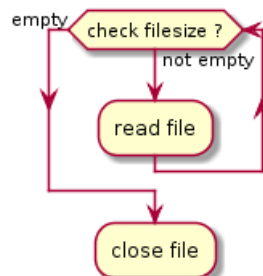


Es posible proporcionar una etiqueta después de la palabra reservada `endwhile`, o usar la palabra reservada `is`.

```

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

```



## 5.6 Procesamiento paralelo

Puedes usar las palabras reservadas `fork`, `fork again` y `end fork` para denotar procesamientos paralelos.

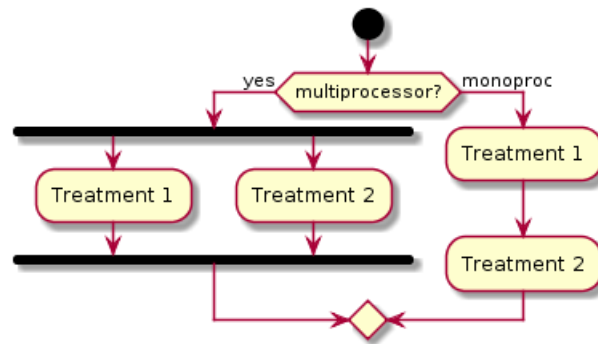
```

@startuml
start

if (multiprocessor?) then (yes)
fork
:Treatment 1;
fork again
:Treatment 2;
end fork
else (monoproc)
:Treatment 1;
:Treatment 2;
endif

@enduml

```



## 5.7 Notas

Se puede aplicar formato a un texto usando sintaxis de WikiCreole.

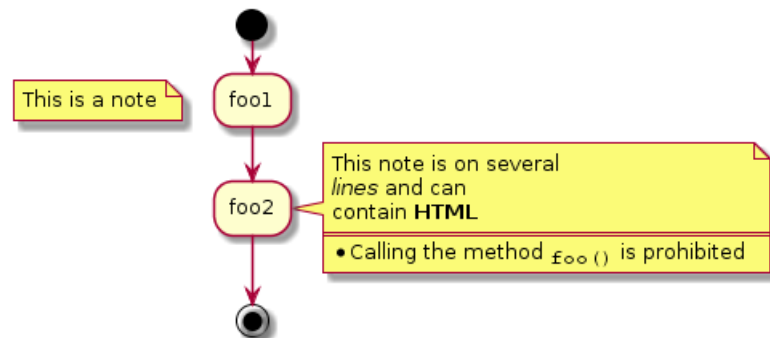
Una nota puede ser flotante, usando la palabra clave **floating**.

```

@startuml

start
:foo1;
floating note left: This is a note
:foo2;
note right
This note is on several
//lines// and can
contain <b>HTML</b>
====
* Calling the method "'foo()'" is prohibited
end note
stop

@enduml
  
```



## 5.8 Colores

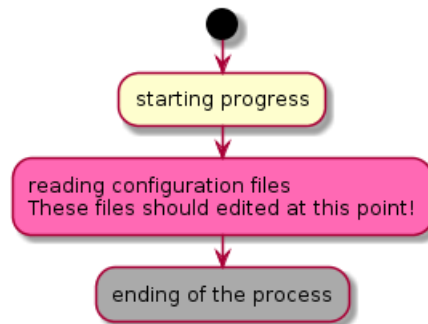
Puedes especificar colores en algunas actividades.

```

@startuml

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

@enduml
  
```



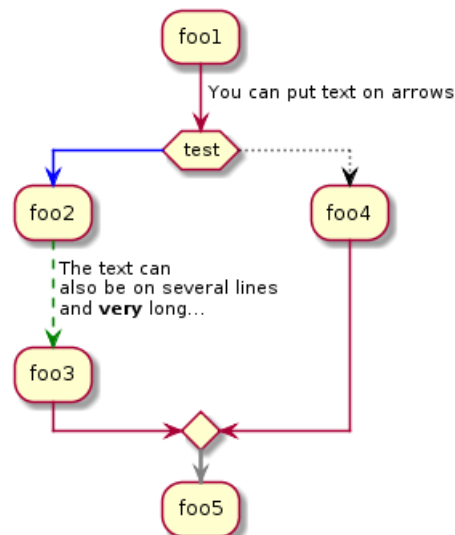
## 5.9 Flechas

Usando la notación `->`, puedes añadir texto a una flecha y cambiar su color.

También es posible tener flechas punteadas, en línea discontinua, en negrita u ocultas.

```

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



## 5.10 Agrupación

Puedes agrupar actividades definiendo particiones:

```

@startuml
start
partition Initialization {
:read config file;
}
  
```

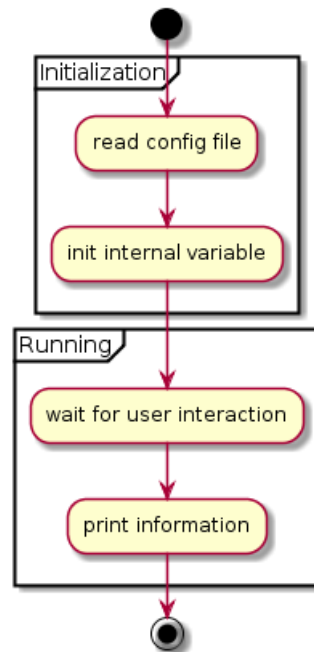


```

: init internal variable;
}
partition Running {
: wait for user interaction;
: print information;
}

stop
@enduml

```



## 5.11 Carriles

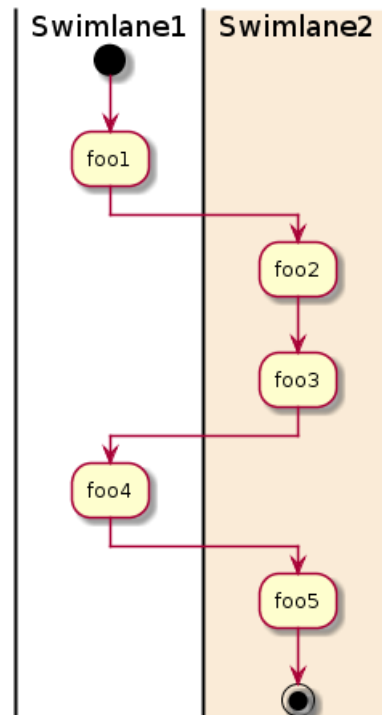
Usando la tecla pipe |, puedes definir carriles.

También es posible cambiar el color de los carriles.

```

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

```

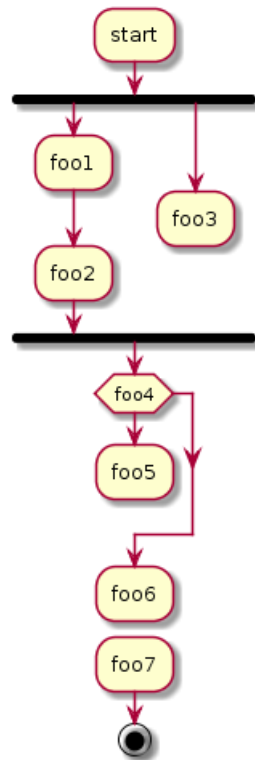


## 5.12 Desacoplar y remover

Es posible remover una flecha usando la palabra reservada `detach`.

```

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



### 5.13 Otras formas de representación de actividades

Cambiando el separador final, ; , puedes configurar diferentes representaciones para una actividad:

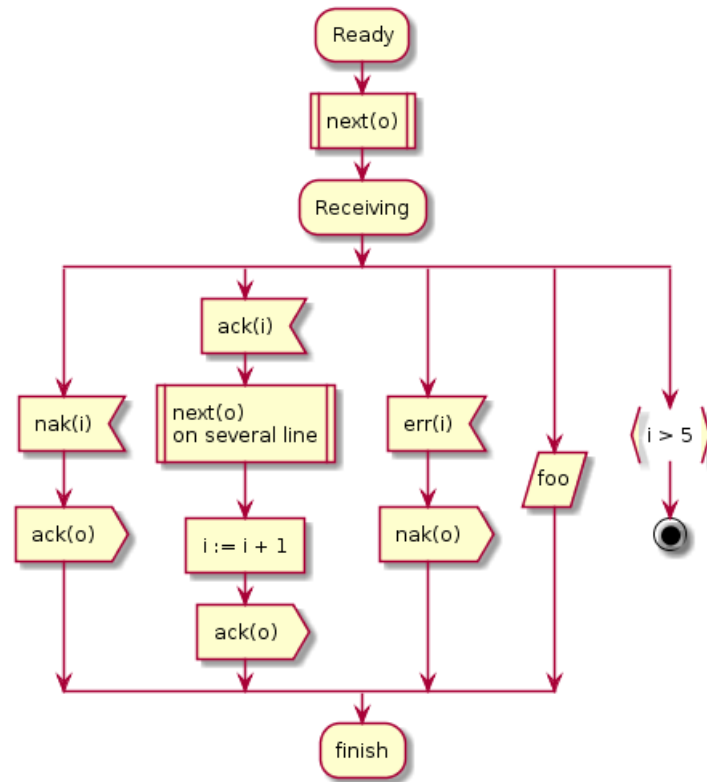
- |
- <
- >
- /
- ]
- }

```

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

```





## 5.14 Un ejemplo completo

```

@startuml

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

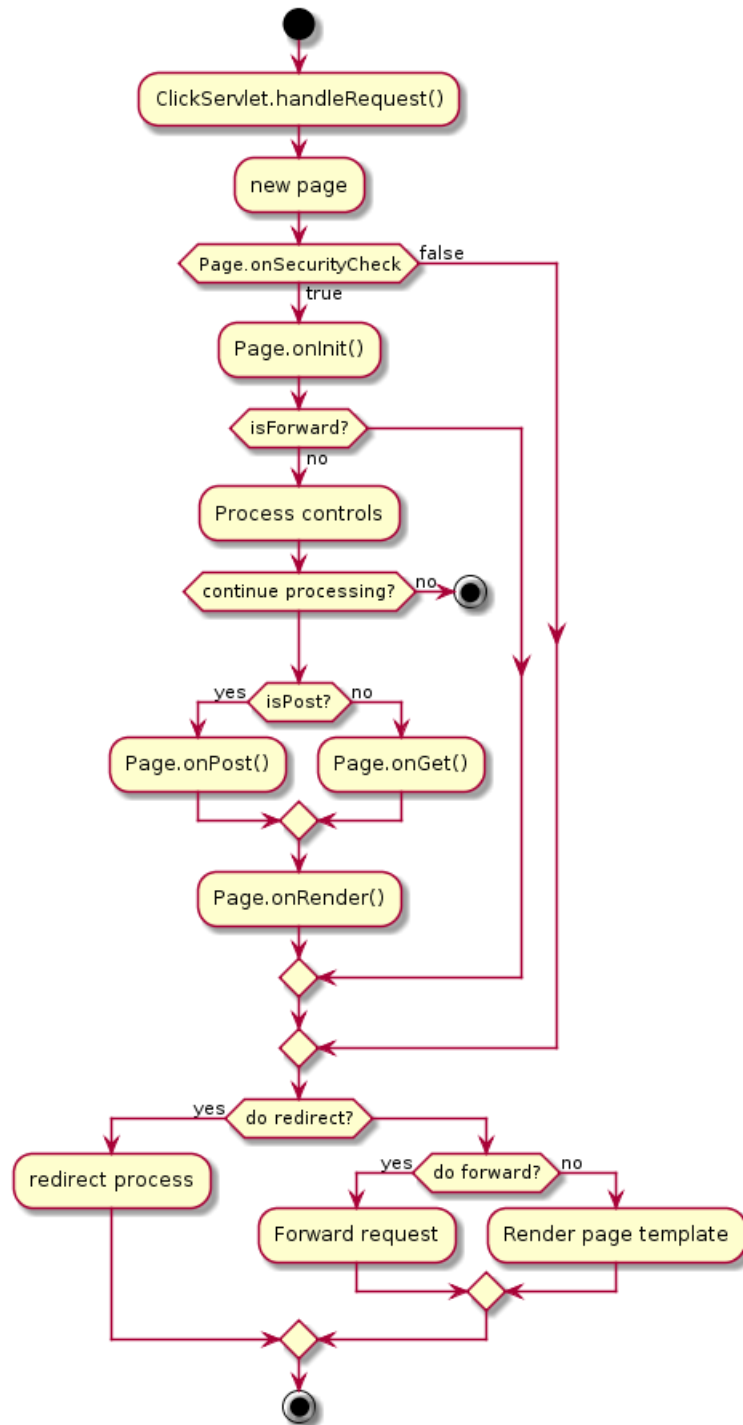
if (isPost?) then (yes)
:Page.onPost();
else (no)
:Page.onGet();
endif
:Page.onRender();
endif
else (false)
endif

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

stop

@enduml

```



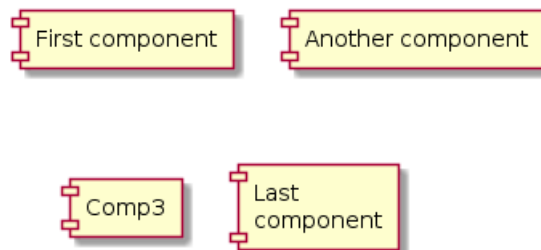
## 6 Diagrama de Componentes

### 6.1 Componentes

Los componentes deberían ser encerrados entre llaves "".

También puedes usar la palabra reservada **component** para definir un componente. Y puedes definir un alias, usando la palabra reservada **as**. Este alias será usado más adelante, cuando definamos relaciones.

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



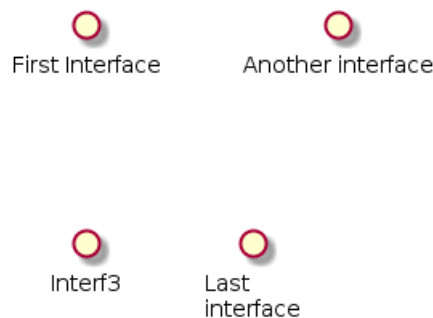
### 6.2 Interfaces

Puedes definir una interfaz usando el símbolo () (porque esto luce como un círculo).

Puedes usar también la palabra reservada **interface** para definir una interfaz. Y puedes definir un alias, usando la palabra reservada **as**. Este alias sera usado luego, definiendo las relaciones.

Nosotros veremos mas adelante que la definición de interfaz es opcional.

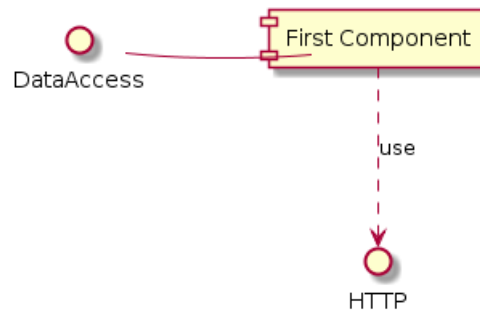
```
@startuml
() "First Interface"
() "Another interface" as Interf2
interface Interf3
interface "Last\ninterface" as Interf4
@enduml
```



### 6.3 Ejemplos basicos

Los links entre los elementos son hechos usando la combinación de símbolos de linea de puntos (.), linea recta (--), y flechas (-->)

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



### 6.4 Usando notas

Puedes usar el `note left of`, `note right of`, `note top of`, `note bottom of`. Las palabras reservadas para definir notas relacionadas a un objeto simple.

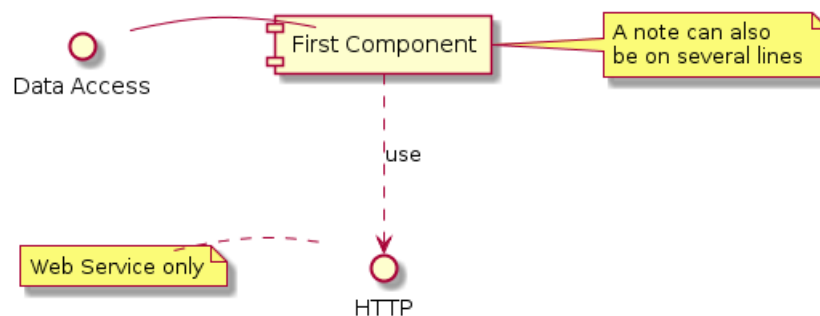
Una nota puede ser definida sola usando la palabra reservada `note`, luego linkea a otro objeto usando el símbolo `...`

```
@startuml
interface "Data Access" as DA

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

note left of HTTP : Web Service only

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



## 6.5 Agrupando componentes

Puedes usar varias palabras reservadas para agrupar componentes e interfaces juntos:

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

```
@startuml

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

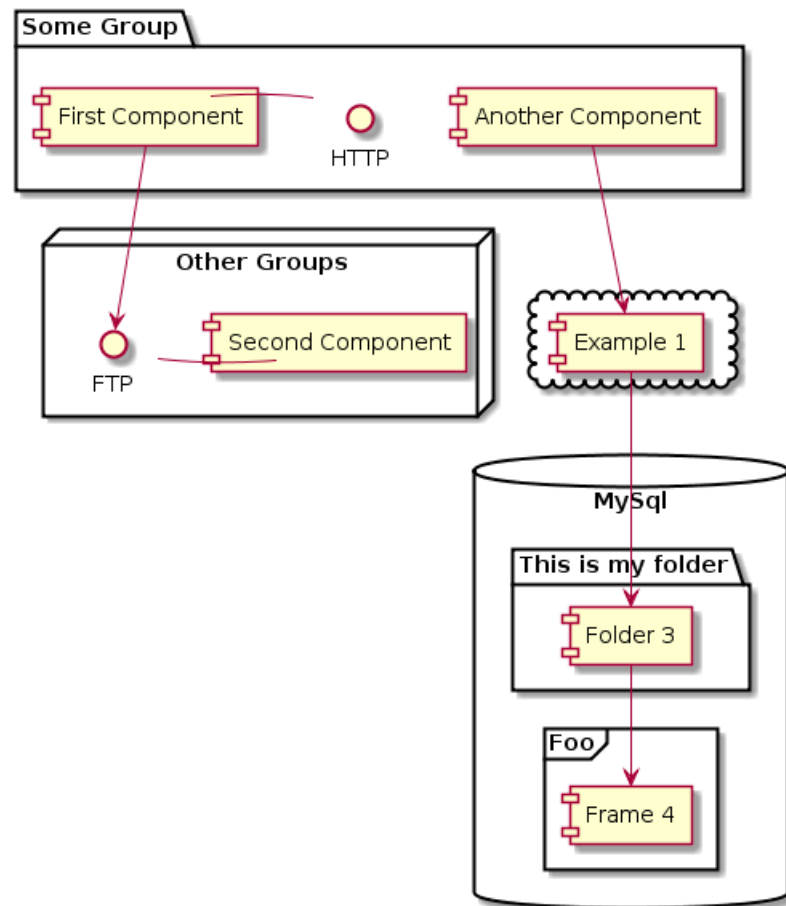
node "Other Groups" {
  FTP - [Second Component]
  [First Component] --> FTP
}

cloud {
  [Example 1]
}

database "MySQL" {
  folder "This is my folder" {
    [Folder 3]
  }
  frame "Foo" {
    [Frame 4]
  }
}

[Another Component] --> [Example 1]
[Example 1] --> [Folder 3]
[Folder 3] --> [Frame 4]

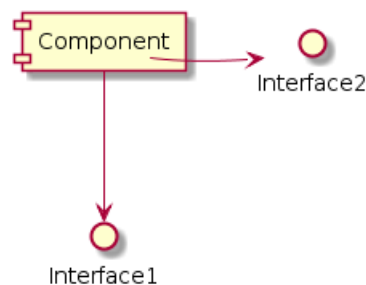
@enduml
```



## 6.6 Cambiando la dirección de las flechas

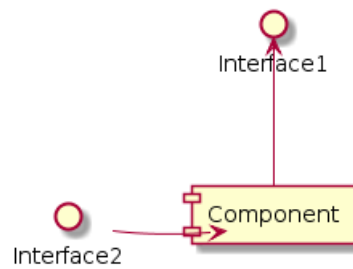
Por defecto los links entre clases tienen dos guiones --y son orientados verticalmente. Puedes usar la orientación horizontal para un link poniendo un guion (o punto) como en el siguiente ejemplo:

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



Puedes también cambiar direcciones invirtiendo el link:

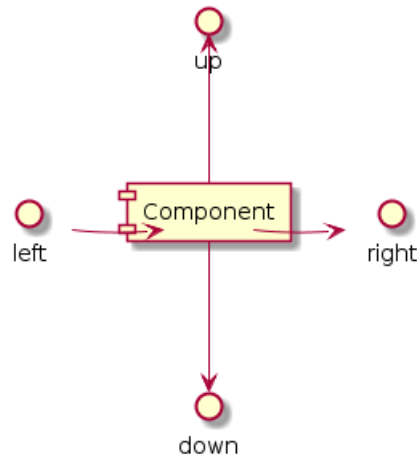
```
@startuml
Interface1 <-- [Component]
Interface2 <- [Component]
@enduml
```



También es posible cambiar la dirección de las flechas agregando la palabra reservada `left`, `right`, `up` o `down` dentro de la flecha:

```

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



Puedes acortar la flecha usando el primer caracter (por ejemplo, `-d-` en lugar de `-down-`) o los dos primeros caracteres (`-do-`).

Por favor nota que no puedes abusar de esta funcionalidad *Graphviz* que usualmente otorga buenos resultados sin ajustes.

## 6.7 Utiliza la notación UML2

El comando `skinparam componentStyle uml2` es usado para cambiar hacia la notación UML2.

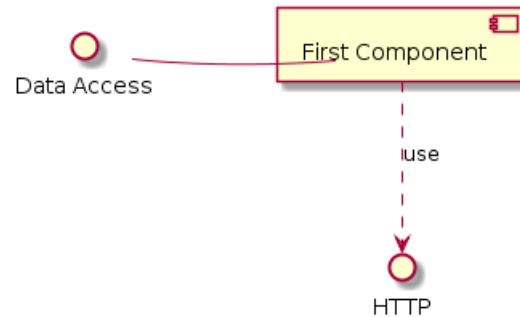
```

@startuml
skinparam componentStyle uml2

interface "Data Access" as DA

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

@enduml
  
```

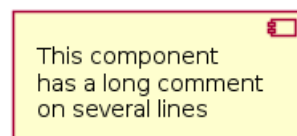


## 6.8 Long description

It is possible to put description on several lines using square brackets.

```

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



## 6.9 Colores individuales

Puedes especificar un color despues de la definición del componente.

```

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



## 6.10 Using Sprite in Stereotype

You can use sprites within stereotype components.

```

@startuml
sprite $businessProcess [16x16/16] {
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FF0000000000FFF
FF0000000000FFF
FF0000000000FFF
FFFFFFFFF0FFFF
FFFFFFFFF0FFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}
@enduml
  
```

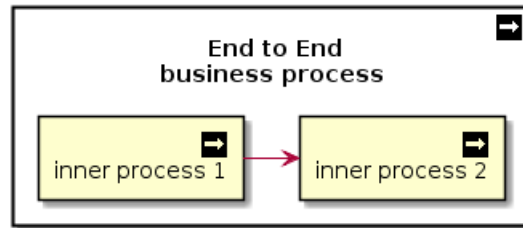


```

FFFFFFFFFFFFFFFF
FFFFFFFFFFFFFFFF
}

rectangle " End to End\nbusiness process" <<$businessProcess>> {
rectangle "inner process 1" <<$businessProcess>> as src
rectangle "inner process 2" <<$businessProcess>> as tgt
src -> tgt
}
@enduml

```



## 6.11 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar colores y fuentes en el diagrama.

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en la ANT Task

Puedes definir colores y fuentes específicas para interfaces y componentes estereotipados.

```

@startuml

skinparam interface {
backgroundColor RosyBrown
borderColor orange
}

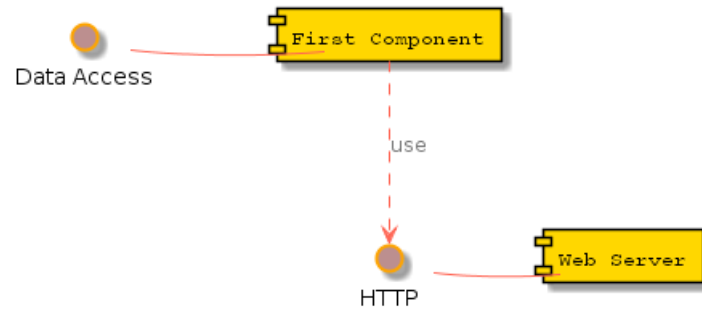
skinparam component {
FontSize 13
BackgroundColor<<Apache>> Red
BorderColor<<Apache>> #FF6655
FontName Courier
BorderColor black
BackgroundColor gold
ArrowFontName Impact
ArrowColor #FF6655
ArrowFontColor #777777
}

() "Data Access" as DA

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

@enduml

```



```

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

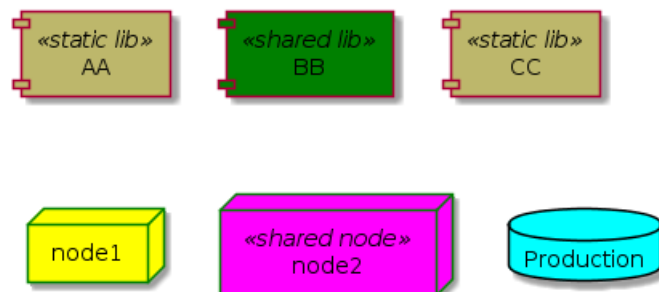
node node1
node node2 <<shared node>>
database Production

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

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

@enduml

```



## 7 Diagrama de Estados

### 7.1 Un Estado Simple

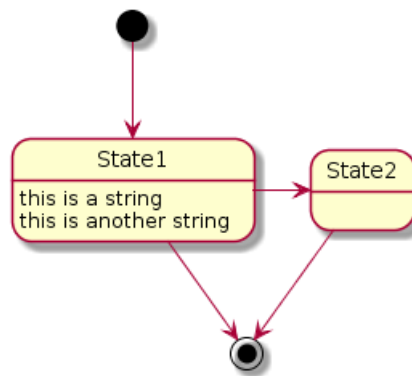
Puedes usar [\*] para el punto de inicio y finalización del diagrama de estados.

Utilice --> para las flechas.

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

State1 -> State2
State2 --> [*]

@enduml
```



### 7.2 Estados compuestos

Un estado también puede ser compuesto. Puedes definirlo usando la palabra reservada **state** y llaves.

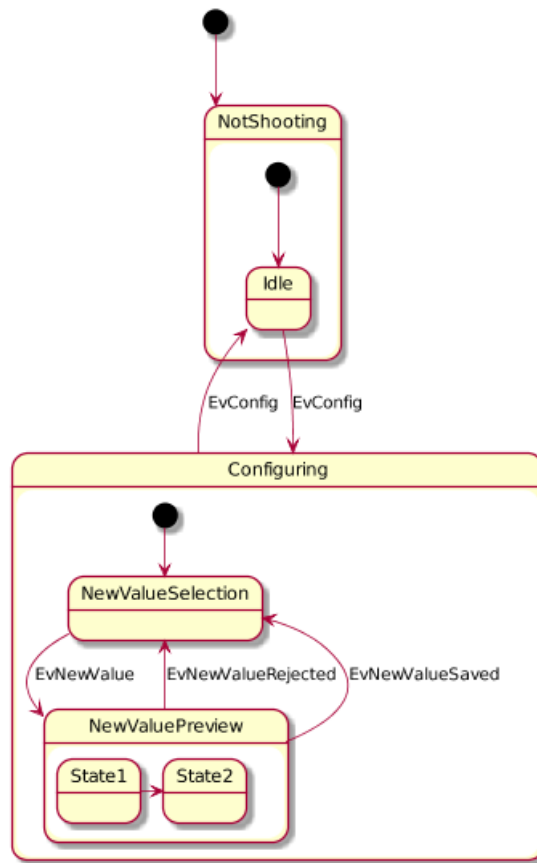
```
@startuml
scale 350 width
[*] --> NotShooting

state NotShooting {
  [*] --> Idle
  Idle --> Configuring : EvConfig
  Configuring --> Idle : EvConfig
}

state Configuring {
  [*] --> NewValueSelection
  NewValueSelection --> NewValuePreview : EvNewValue
  NewValuePreview --> NewValueSelection : EvNewValueRejected
  NewValuePreview --> NewValueSelection : EvNewValueSaved
}

state NewValuePreview {
  State1 -> State2
}

@enduml
```



### 7.3 Nombres largos

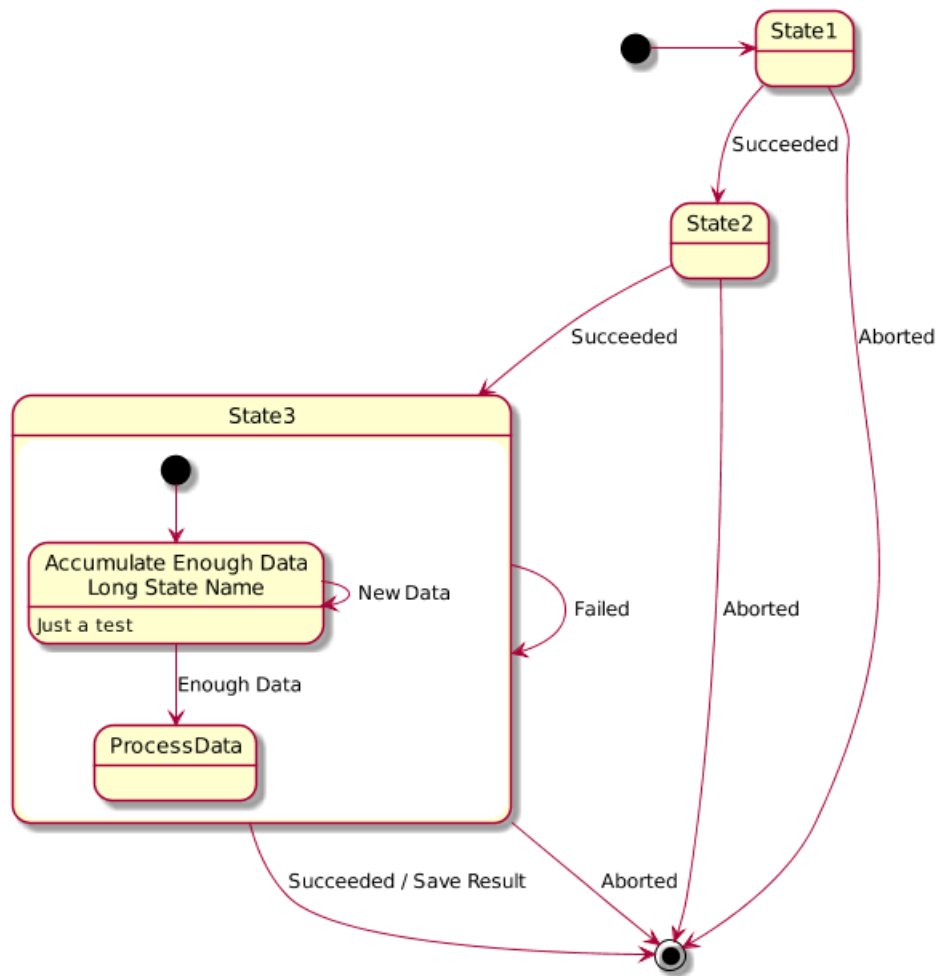
También puedes usar la palabra reservada **state** para definir largas descripciones en un estado.

```

@startuml
scale 600 width

[*] -> State1
State1 --> State2 : Succeeded
State1 --> [*] : Aborted
State2 --> State3 : Succeeded
State2 --> [*] : Aborted
state State3 {
state "Accumulate Enough Data\nLong State Name" as long1
long1 : Just a test
[*] --> long1
long1 --> long1 : New Data
long1 --> processData : Enough Data
}
State3 --> State3 : Failed
State3 --> [*] : Succeeded / Save Result
State3 --> [*] : Aborted

@enduml
  
```



## 7.4 Estados simultáneos

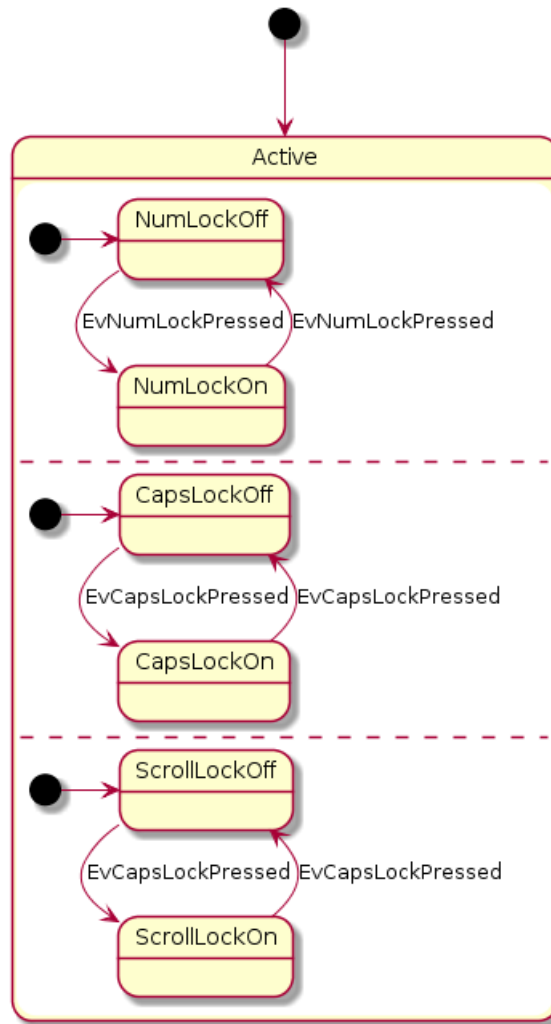
Puedes definir estados simultáneos dentro de un estado compuesto usando los símbolos `--` or `||` como separadores.

```

@startuml
[*] --> Active

state Active {
    [*] --> NumLockOff
    NumLockOff --> NumLockOn : EvNumLockPressed
    NumLockOn --> NumLockOff : EvNumLockPressed
    --
    [*] --> CapsLockOff
    CapsLockOff --> CapsLockOn : EvCapsLockPressed
    CapsLockOn --> CapsLockOff : EvCapsLockPressed
    --
    [*] --> ScrollLockOff
    ScrollLockOff --> ScrollLockOn : EvCapsLockPressed
    ScrollLockOn --> ScrollLockOff : EvCapsLockPressed
}

@enduml
  
```

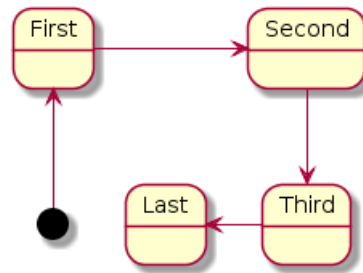


## 7.5 Dirección de la flecha

Puedes usar `->` para flechas horizontales. Es posible forzar la dirección de las flechas usando la siguiente sintaxis:

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

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



Puedes acortar un flecha usando sólo el primer carácter del nombre de la dirección (por ejemplo, `-d-` en lugar de `-down-`) o los dos primeros caracteres (`-do-`).

Por favor tenga en cuenta que no debería de esta funcionalidad : *Graphviz* usualmente devuelve buenos resultados sin necesidad de configuración.

## 7.6 Notas

También puedes definir notas usando las palabras reservadas `note left of`, `note right of`, `note top of`, `note bottom of`.

También puedes definir notas de varias líneas.

```

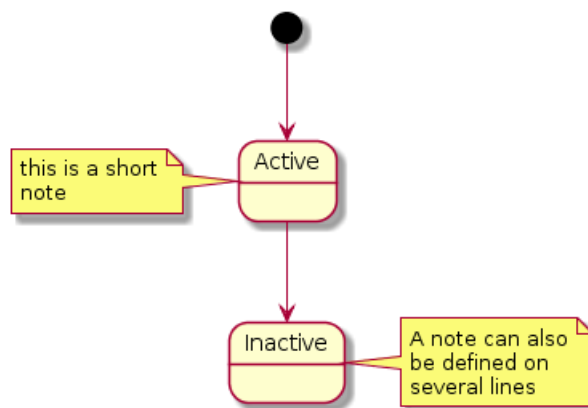
@startuml

[*] --> 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
  
```



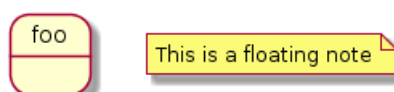
También puedes tener notas flotantes.

```

@startuml

state foo
note "This is a floating note" as N1

@enduml
  
```



## 7.7 Más sobre notas

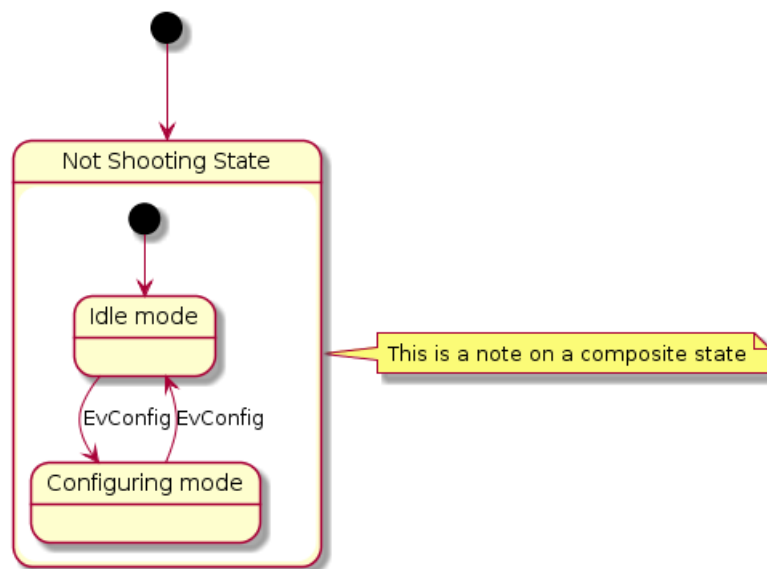
Puedes colocar notas a estados compuestos.

```
@startuml
[*] --> NotShooting

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

note right of NotShooting : This is a note on a composite state

@enduml
```



## 7.8 Personalización (Skinparam)

Puedes usar el comando `skinparam` para cambiar los colores y fuentes de un diagrama.

Puedes usar este comando :

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido,
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

Puedes definir colores y fuentes específicas para estados estereotipados.

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

[*] --> NotShooting

state "Not Shooting State" as NotShooting {
```



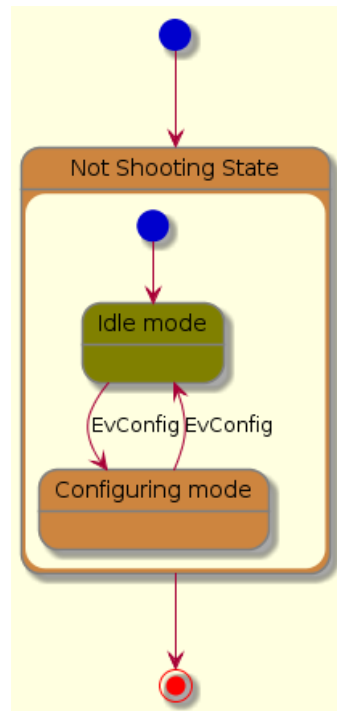


```

state "Idle mode" as Idle <<Warning>>
state "Configuring mode" as Configuring
[*] --> Idle
Idle --> Configuring : EvConfig
Configuring --> Idle : EvConfig
}

NotShooting --> [*]
@enduml

```

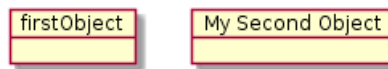


## 8 Diagrama de Objetos

### 8.1 Definición de objetos

Puedes definir instancias de objetos usando la palabra reservada `object`.

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



### 8.2 Relaciones entre objetos

Las relaciones entre objetos son definidas usando los siguientes símbolos:

Extensión	< --	
Composición	*--	
Agregación	o--	

Es posible reemplazar `--` con `..` para obtener una línea de puntos.

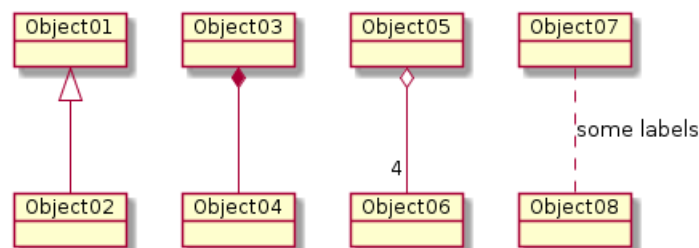
Sabiendo estas reglas, es posible dibujar los siguientes diagramas.

Es posible agregar una etiqueta sobre una relación usando `:"`, seguido del texto de la etiqueta.

Para la cardinalidad puedes usar doble comillas `"` en cada lado de la relación.

```
@startuml
object Object01
object Object02
object Object03
object Object04
object Object05
object Object06
object Object07
object Object08

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



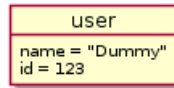
### 8.3 Agregando campos

Para declarar campos, puedes usar el símbolo `:"` seguido del nombre del campo.

```
@startuml
object user

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

@enduml
```

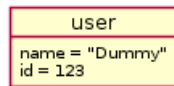


También es posible declarar entre llaves {} todos los campos.

```
@startuml
```

```
object user {
name = "Dummy"
id = 123
}
```

```
@enduml
```



## 8.4 Características comunes en diagramas de clases

- Visibilidad
- Definición de notas
- Uso de paquetes
- Personalización de la salida (skinparam)

## 9 Comandos comunes

### 9.1 Comentarios

Todo lo que comienza con comillas simples ' es un comentario.

También puedes colocar comentarios de varias líneas, usando '/' para empezar y '/' para finalizar.

### 9.2 Encabezado y pie de página

Puedes usar los comandos `header` o `footer` para añadir un encabezado o pie de página en cualquier diagrama generado.

Puedes opcionalmente especificar si quieres un encabezado o pie de página centrado, a la izquierda o a la derecha especificando las palabras clave `center`, `left` o `right`, respectivamente.

Para el título, es posible definir un encabezado o pie de página en varias líneas.

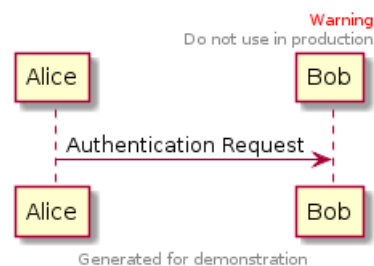
También es posible introducir algo de HTML en un encabezado o pie de página.

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

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

center footer Generated for demonstration

@enduml
```



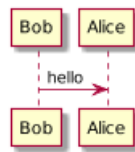
### 9.3 Zoom (acercamiento)

Puedes usar el comando `scale` para hacer zoom a la imagen generada.

Puedes usar tanto un número como una fracción para definir el factor escalar. Puedes también especificar tanto el ancho como el alto (en píxeles). Y también puedes especificar el ancho y altura en un mismo comando : la imagen es adaptada para cumplir con las dimensiones especificadas.

- `scale 1.5`
- `scale 2/3`
- `scale 200 width`
- `scale 200 height`
- `scale 200*100`
- `scale max 300*200`
- `scale max 1024 width`
- `scale max 800 height`

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



## 9.4 Título

La palabra reservada **title** se usa para colocar un título. Puedes añadir una nueva línea usando **\n** en la descripción del título.

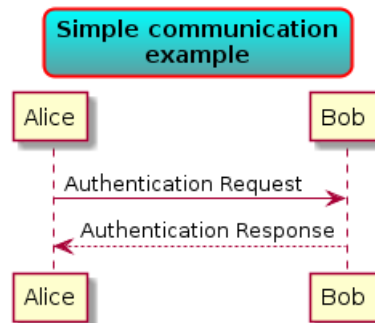
Algunas ronfiguraciones usando **skinparam** están disponibles para colocar bordes en el título.

```
@startuml
skinparam titleBorderRoundCorner 15
skinparam titleBorderThickness 2
skinparam titleBorderColor red
skinparam titleBackgroundColor Aqua-CadetBlue

title Simple communication\nexample

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

@enduml
```



Puedes usar sintaxis de Creole en el título.

Además puedes definir un título en varias líneas usando las palabras reservadas **title** y **end title**.

```
@startuml

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

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

@enduml
```

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



## 9.5 Subtítulo

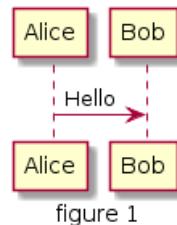
Existe también la palabra clave `caption` para colocar un subtítulo bajo el diagrama.

```

@startuml

caption figure 1
Alice -> Bob: Hello

@enduml
  
```



## 9.6 Leyenda del diagrama

Las palabras reservadas `legend` y `end legend` son usadas para colocar una leyenda.

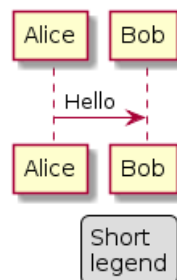
Puedes opcionalmente especificar `left`, `right` o `center` para establecer la alineación de la leyenda.

```

@startuml

Alice -> Bob : Hello
legend right
Short
legend
endlegend

@enduml
  
```



## 10 Salt

**Salt** is a subproject included in PlantUML that may help you to design graphical interface.

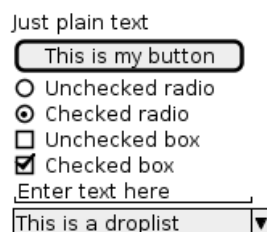
You can use either `@startsalt` keyword, or `@startuml` followed by a line with `salt` keyword.

### 10.1 Basic widgets

A window must start and end with brackets. You can then define:

- Button using `[` and `]`.
- Radio button using `(` and `)`.
- Checkbox using `[` and `]`.
- User text area using `"`.

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



The goal of this tool is to discuss about simple and sample windows.

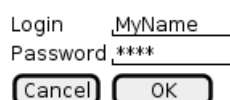
### 10.2 Using grid

A table is automatically created when you use an opening bracket `{`.

And you have to use `|` to separate columns.

For example:

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



Just after the opening bracket, you can use a character to define if you want to draw lines or columns of the grid :

# To display all vertical and horizontal lines

! To display all vertical lines

- To display all horizontal lines

+ To display external lines

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

Login	MyName
Password	****
Cancel	OK

### 10.3 Using separator

You can use several horizontal lines as separator.

```
@startsalt
{
Text1
..
"Some field"
==
Note on usage
~~
Another text
--
[Ok]
}
@endsalt
```

Text1
<u>Some field</u>
<u>Note on usage</u>
Another text
<b>Ok</b>

### 10.4 Tree widget

To have a Tree, you have to start with {T and to use + to denote hierarchy.

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





```

++ Africa
}
}
@endsalt

```



## 10.5 Enclosing brackets

You can define subelements by opening a new opening bracket.

```

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

```

Name: \_\_\_\_\_

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

Superclass: java.lang.Object Browse...

## 10.6 Adding tabs

You can add tabs using {/ notation. Note that you can use HTML code to have bold text.

```

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

```

**General**
Fullscreen
Behavior
Saving

Open image in: Smart Mode ▼

☒ Smooth images when zoomed

☒ Confirm image deletion

☐ Show hidden images

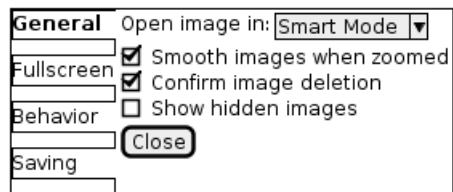
Close

Tab could also be vertically oriented:

```

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

```



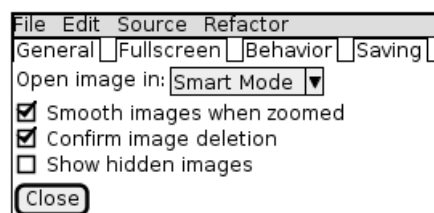
## 10.7 Using menu

You can add a menu by using {\* notation.

```

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

```



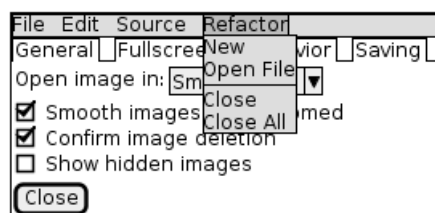
It is also possible to open a menu:

```

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

```





## 10.8 Advanced table

You can use two special notations for table :

- \* to indicate that a cell with span with left
- . to denotate an empty cell

```
@startsalt
{#
. | Column 2 | Column 3
Row header 1 | value 1 | value 2
Row header 2 | A long cell | *
}
@endsalt
```

	Column 2	Column 3
Row header 1	value 1	value 2
Row header 2	A long cell	

## 11 Creole

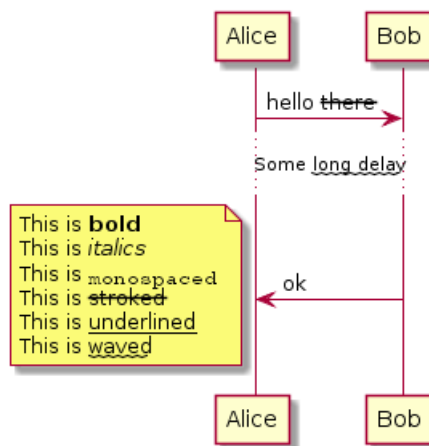
A light Creole engine have been integrated into PlantUML to have a standardized way of defining text style.

All diagrams are now supporting this syntax.

Note that ascending compatibility with HTML syntax is preserved.

### 11.1 Emphasized text

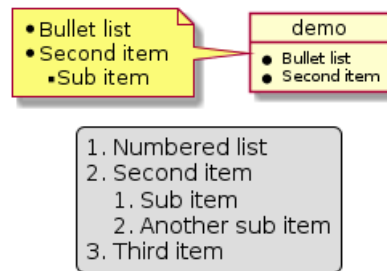
```
@startuml
Alice -> Bob : hello --there--
... Some ~~long delay~~ ...
Bob -> Alice : ok
note left
This is bold
This is italics
This is "monospaced"
This is --stroked--
This is __underlined__
This is ~~waved~~
end note
@enduml
```



### 11.2 List

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

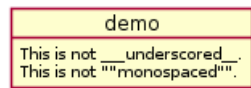
legend
# Numbered list
# Second item
## Sub item
## Another sub item
# Third item
end legend
@enduml
```



### 11.3 Escape character

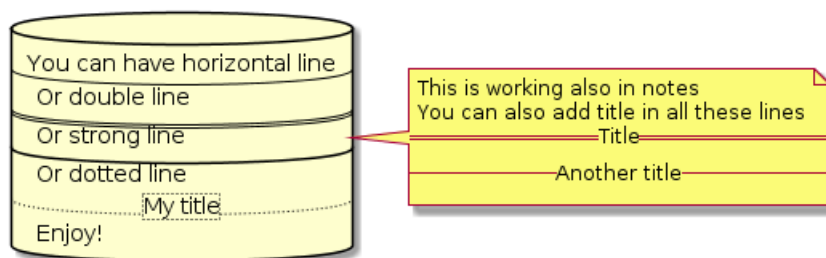
You can use the tilde ~ to escape special creole characters.

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



### 11.4 Horizontal lines

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

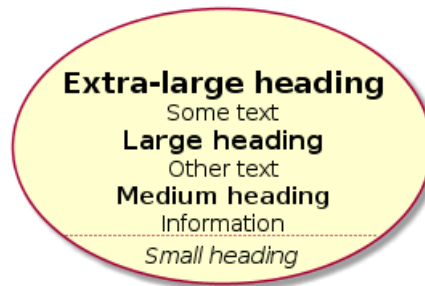


### 11.5 Headings

```

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

```



## 11.6 Legacy HTML

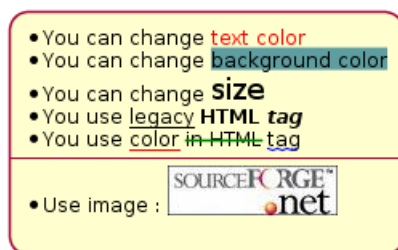
Some HTML tags are also working:

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

```

@startuml
:* You can change <color:red>text color</color>
* You can change <back:cadetblue>background color</back>
* You can change <size:18>size</size>
* You use <u>legacy</u> <b>HTML <i>tag</i></b>
* You use <u:red>color</u> <s:green>in HTML</s> <w:#0000FF>tag</w>
----
* Use image : <img:sourceforge.jpg>
;
@enduml

```



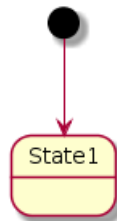
## 11.7 Table

It is possible to build table.

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

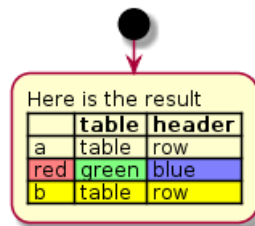
Example of simple table

	table	header
a	table	row
b	table	row



You can specify background colors for cells and lines.

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



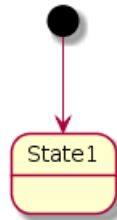
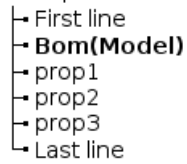
## 11.8 Tree

You can use |\_ characters to build a tree.

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



Example of Tree



## 11.9 Special characters

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

```

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



## 11.10 OpenIconic

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

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

```

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

### ♥Use of OpenIconic♥



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

```

@startuml
listopeniconic
@enduml
  
```



## List Open Iconic

Credit to

<https://useiconic.com/open>

<b>List Open Iconic</b>						
<b>Credit to</b>						
<a href="https://useiconic.com/open">https://useiconic.com/open</a>						



## 11.11 Definiendo y usando sprites

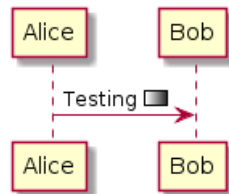
Un *Sprite* es un pequeño elemento gráfico que puede ser usado en diagramas.

En PlantUML, los sprites son monocromos y pueden tener 4, 8 o 16 niveles de grises.

Para definir un sprite, tienes que usar notación hexadecimal entre el 0 y la letra F, por cada píxel.

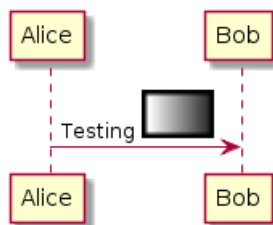
Entonces puedes usar los sprites con la etiqueta <\$XXX> donde XXX es el nombre del sprite.

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



You can scale the sprite.

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



## 11.12 Encoding Sprite

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

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



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

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

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

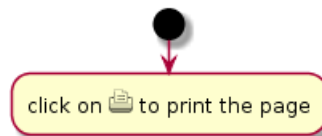
### 11.13 Importing Sprite

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

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

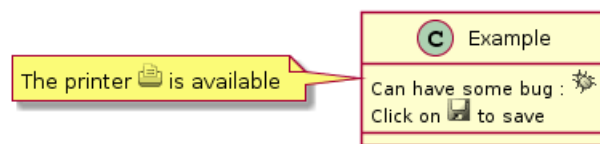
After copying an image into you clipboard, several possible definitions of the corresponding sprite will be displayed : you will just have to pickup the one you want.

### 11.14 Ejemplos



```
@startuml
sprite $printer [15x15/8z] N0tH3WOW208HxFz_kMAhj7lHWpa1XC716szOPq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvFfPEo
start
:click on <$printer> to print the page;
@enduml
```

#### Use of sprites (🖨️, 🐛...)



```
@startuml
sprite $bug [15x15/16z] PKzR2i0m2BFMi15p_FEjQEqB1z27aeqCqixa8S40T7C53cKpsHpaYPDJY_12MHM-BLRyywPhrrlw3qu
sprite $printer [15x15/8z] N0tH3WOW208HxFz_kMAhj7lHWpa1XC716szOPq4MVPEWfBHIuxP3L6kbTcizR8tAhzaqFvXwvFfPEo
sprite $disk {
444445566677881
436000000009991
43600000000ACA1
53700000001A7A1
537000000012B8A1
53800000123B8A1
63800001233C9A1
634999AABBC99B1
744566778899AB1
7456AAAAA99AAB1
8566AFC228AABB1
8567AC8118BBBB1
867BD4433BBBBB1
39AAAAABBBBBBC1
}

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

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

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

## 12 Cambiar fuentes y colores

### 12.1 Uso

Puedes cambiar los colores y fuentes de un diagrama usando el comando `skinparam`. Ejemplo:

```
skinparam backgroundColor yellow
```

Puedes usar este comando:

- En la definición del diagrama, como cualquier otro comando,
- En un archivo incluido (vea *Preprocesamiento*),
- En un archivo de configuración, proporcionado en la consola de comandos o en el ANT task.

### 12.2 Anidado

Para evitar repeticiones, es posible anidar definiciones. Entonces, la siguiente definición:

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

es estrictamente equivalente a:

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

## 12.3 Color

Puedes usar, tanto el nombre del color, como el código RGB.

Nombre del parámetro	Defecto Valor	Color	Comentario
backgroundColor	white		Fondo de la página
activityArrowColor	#A80036		Color de las flechas en diagramas de actividades
activityBackgroundColor	#FEFECE		Fondo de las actividades
activityBorderColor	#A80036		Color de los bordes de la actividad
activityStartColor	black		Círculo de comienzo en diagramas de actividades
activityEndColor	black		Círculo de finalización en diagramas de actividades
activityBarColor	black		Barra de sincronización en diagramas de actividades
usecaseArrowColor	#A80036		Color de las flechas en diagramas de casos de uso
usecaseActorBackgroundColor	#FEFECE		Color de la cabeza del actor en diagramas de casos de uso
usecaseActorBorderColor	#A80036		Color de los bordes del actor en diagramas de casos de uso
usecaseBackgroundColor	#FEFECE		Fondo de los casos de uso
usecaseBorderColor	#A80036		Color de los bordes de un caso de uso en diagramas de casos de uso
classArrowColor	#A80036		Color de las flechas en diagramas de clases
classBackgroundColor	#FEFECE		Fondo de classes/interface/enum en diagramas de clases
classBorderColor	#A80036		Bordes de classes/interface/enum en diagramas de clases
packageBackgroundColor	#FEFECE		Fondo de paquetes en diagramas de clases
packageBorderColor	#A80036		Bordes de paquetes en diagramas de clases
stereotypeCBackgroundColor	#ADD1B2		Fondo de puntos de una clase en diagramas de clases
stereotypeABackgroundColor	#A9DCDF		Fondo de puntos de clases abstractas en diagramas de clases
stereotypeIBackgroundColor	#B4A7E5		Fondo de puntos de interface en diagramas de clases
stereotypeEBackgroundColor	#EB937F		Fondo de puntos de enum en diagramas de clases
componentArrowColor	#A80036		Color de flechas en diagramas de componentes
componentBackgroundColor	#FEFECE		Fondo de componentes
componentBorderColor	#A80036		Bordes de componentes
componentInterfaceBackgroundColor	#FEFECE		Fondo de interface en diagramas de componentes
componentInterfaceBorderColor	#A80036		Bordes de interface en diagramas de componentes
noteBackgroundColor	#FBFB77		Fondo de notas
noteBorderColor	#A80036		Bordes de notas
stateBackgroundColor	#FEFECE		Fondo de estados en diagramas de estados
stateBorderColor	#A80036		Bordes de estados en diagramas de estados
stateArrowColor	#A80036		Color de flechas en diagramas de estados
stateStartColor	black		Círculo de comienzo en diagramas de estados
stateEndColor	black		Círculo de finalización en diagramas de estados
sequenceArrowColor	#A80036		Color de flechas en diagramas de secuencia
sequenceActorBackgroundColor	#FEFECE		Color de la cabeza del actor en diagramas de secuencia
sequenceActorBorderColor	#A80036		Borde del actor en diagramas de secuencia
sequenceGroupBackgroundColor	#EEEEEE		Color de la cabeza de alt/opt/loop en diagramas de secuencia
sequenceLifeLineBackgroundColor	white		Fondo de línea de vida en diagramas de secuencia
sequenceLifeLineBorderColor	#A80036		Borde de línea de vida en diagramas de secuencia
sequenceParticipantBackgroundColor	#FEFECE		Fondo de participante en diagramas de secuencia
sequenceParticipantBorderColor	#A80036		Borde de participante en diagramas de secuencia



## 12.4 Color de fuente, nombre y tamaño

Puedes cambiar la fuente para el dibujo usando los parámetros `xxxFontColor`, `xxxFontSize` y `xxxFontName`.

Ejemplo:

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

Puedes también cambiar la fuente por defecto para todas las fuentes, usando `skinparam defaultFontName`.

Ejemplo:

```
skinparam defaultFontName Aapex
```

Por favor tenga en cuenta que fontname es altamente dependiente del sistema, por lo tanto no lo sobre use, si está buscando portabilidad.

Parámetro Nombre	Defecto Valor	Comentario
activityFontColor activityFontSize activityFontStyle activityFontName	black 14 plain	Usado para caja de actividad
activityArrowFontColor activityArrowFontSize activityArrowFontStyle activityArrowFontName	black 13 plain	Usado para texto en las flechas de los diagramas de actividades
circledCharacterFontColor circledCharacterFontSize circledCharacterFontStyle circledCharacterFontName circledCharacterRadius	black 17 bold Courier 11	Usado para texto en círculo para class, enum u otros
classArrowFontColor classArrowFontSize classArrowFontStyle classArrowFontName	black 10 plain	Usado para texto en las flechas de los diagramas de clases
classAttributeFontColor classAttributeFontSize classAttributeIconSize classAttributeFontStyle classAttributeFontName	black 10 10 plain	Atributos y métodos de clase
classFontColor classFontSize classFontStyle classFontName	black 12 plain	Usado para nombres de clases
classStereotypeFontColor classStereotypeFontSize classStereotypeFontStyle classStereotypeFontName	black 12 italic	Usado para estereotipos de clases
componentFontColor componentFontSize componentFontStyle componentFontName	black 14 plain	Usado para nombres de componentes
componentStereotypeFontColor componentStereotypeFontSize componentStereotypeFontStyle componentStereotypeFontName	black 14 italic	Usado para estereotipos de componentes



componentArrowFontColor componentArrowFontSize componentArrowFontStyle componentArrowFontName	black 13 plain	Usado para texto en las flechas de los diagramas de componentes
noteFontColor noteFontSize noteFontStyle noteFontName	black 13 plain	Usado para notas en todos los diagramas excepto diagramas de secuencia
packageFontColor packageFontSize packageFontStyle packageFontName	black 14 plain	Usado para nombres de paquetes y particiones
sequenceActorFontColor sequenceActorFontSize sequenceActorFontStyle sequenceActorFontName	black 13 plain	Usado para actores en diagramas de secuencia
sequenceDividerFontColor sequenceDividerFontSize sequenceDividerFontStyle sequenceDividerFontName	black 13 bold	Usado para texto en divisores en diagramas de secuencia
sequenceArrowFontColor sequenceArrowFontSize sequenceArrowFontStyle sequenceArrowFontName	black 13 plain	Usado para texto en las flechas de los diagramas de secuencia
sequenceGroupingFontColor sequenceGroupingFontSize sequenceGroupingFontStyle sequenceGroupingFontName	black 11 plain	Usado para texto del "else" en diagramas de secuencia
sequenceGroupingHeaderFontColor sequenceGroupingHeaderFontSize sequenceGroupingHeaderFontStyle sequenceGroupingHeaderFontName	black 13 plain	Usado para texto de encabezados "alt/opt/loop" en diagramas de secuencia
sequenceParticipantFontColor sequenceParticipantFontSize sequenceParticipantFontStyle sequenceParticipantFontName	black 13 plain	Usado para texto del participante en diagramas de secuencia
sequenceTitleFontColor sequenceTitleFontSize sequenceTitleFontStyle sequenceTitleFontName	black 13 plain	Usado para títulos en diagramas de secuencia
titleFontColor titleFontSize titleFontStyle titleFontName	black 18 plain	Usado para títulos en todos los diagramas excepto diagramas de secuencia
stateFontColor stateFontSize stateFontStyle stateFontName	black 14 plain	Usado para estados en diagramas de estados
stateArrowFontColor stateArrowFontSize stateArrowFontStyle stateArrowFontName	black 13 plain	Usado para texto en las flechas de los diagramas de estados
stateAttributeFontColor stateAttributeFontSize stateAttributeFontStyle stateAttributeFontName	black 12 plain	Usado para descripciones de estados en diagramas de estados



usecaseFontColor usecaseFontSize usecaseFontStyle usecaseFontName	black 14 plain	Usado para etiquetas de casos de uso en diagramas de casos de uso
usecaseStereotypeFontColor usecaseStereotypeFontSize usecaseStereotypeFontStyle usecaseStereotypeFontName	black 14 italic	Usado para estereotipo en un caso de uso
usecaseActorFontColor usecaseActorFontSize usecaseActorFontStyle usecaseActorFontName	black 14 plain	Usado para etiquetas de actores en diagramas de casos de uso
usecaseActorStereotypeFontColor usecaseActorStereotypeFontSize usecaseActorStereotypeFontStyle usecaseActorStereotypeFontName	black 14 italic	Usado para estereotipo de actor
usecaseArrowFontColor usecaseArrowFontSize usecaseArrowFontStyle usecaseArrowFontName	black 13 plain	Usado para texto en las flechas de los diagramas de casos de uso
footerFontColor footerFontSize footerFontStyle footerFontName	black 10 plain	Usado para pie de página
headerFontColor headerFontSize headerFontStyle headerFontName	black 10 plain	Usado para encabezado

## 12.5 Blanco y negro

Puedes forzar el uso de salida en blanco y negro usando el comando `skinparam monochrome true`.

```
@startuml
skinparam monochrome true

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

User -> A: DoWork
activate A

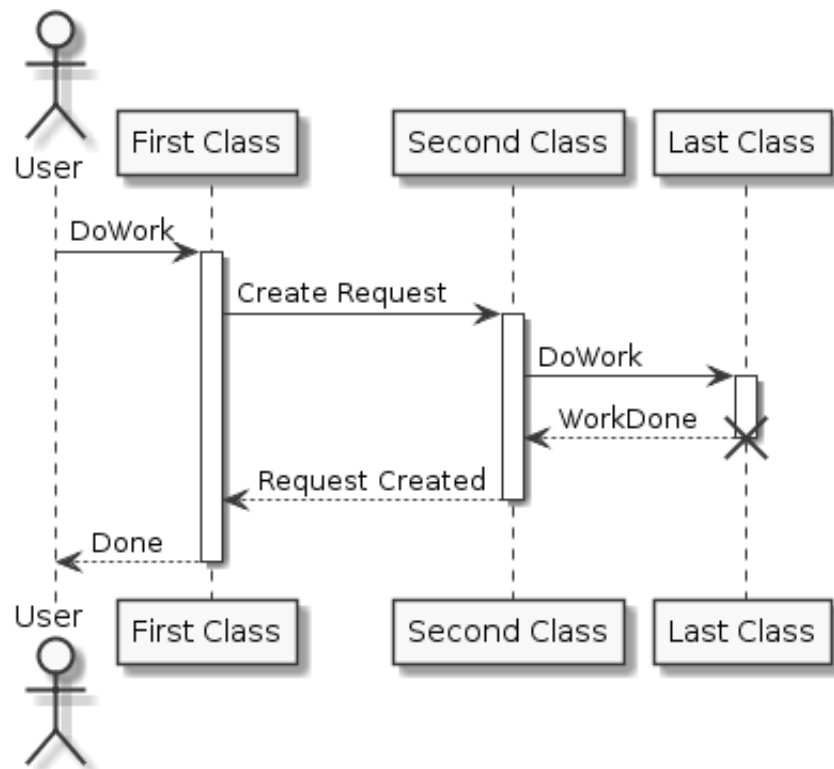
A -> B: Create Request
activate B

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

B --> A: Request Created
deactivate B

A --> User: Done
deactivate A

@enduml
```



## 13 Preprocesamiento

Algunas capacidades de preprocesamiento menores son incluidas **PlantUML**, y disponibles para *todos* los diagramas.

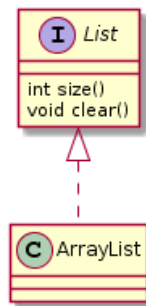
Aquellas funcionalidades son bastante similares al preprocesador del lenguaje C, excepto que el carácter especial (#) ha sido cambiado por el signo de exclamación (!).

### 13.1 Incluyendo archivos

Use la directiva `!include` para incluir archivos en su diagrama.

Imagine que tiene la misma clase que aparece en varios diagramas. En lugar de duplicar la descripción de esa clase, puede definir un archivo que contenga esa descripción.

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



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

El archivo `List.iuml` puede ser incluido en muchos diagramas, y cualquier modificación en ese archivo repercutirá en todos los diagramas que lo incluyan.

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

Puedes también colocar varios bloques de texto `@startuml/@enduml` en un archivo incluido y luego especificar que bloque quieres incluir, añadiendo `!0` donde 0 es el número del bloque.

Por ejemplo, si usas `!include foo.txt!1`, el segundo bloque `@startuml/@enduml` dentro de `foo.txt`, será incluido.

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

### 13.2 Incluyendo URL

Use la directiva `!includeurl` para incluir archivos de Internet/Intranet en su diagrama.

También puede usar `!includeurl http://someurl.com/mypath!0` para especificar que bloque `@startuml/@enduml` de `http://someurl.com/mypath` quiere incluir. La notación `!0` denota el primer diagrama.

### 13.3 Definición de constante

Puede definir una constante usando la directiva `!define`. Así como en el lenguaje C, un nombre de constante puede únicamente usar caracteres alfanuméricos y de subrayado, y no puede comenzar con un dígito.



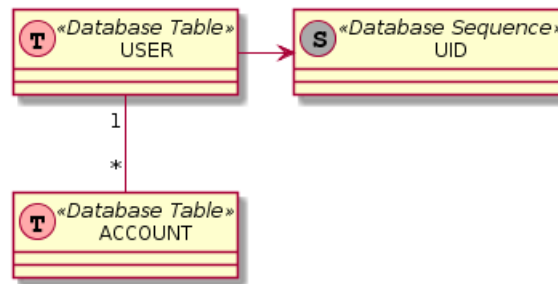
```

@startuml

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

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

```



Por supuesto, puedes usar la directiva `!include` para definir todas las constantes en un solo archivo, e incluirlo en tu diagrama.

Una constante puede darse de baja con la directiva `!undef XXX`.

También puedes especificar constantes en la línea de comandos con las banderas `-D`.

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

Tenga en cuenta que la bandera `-D` debe colocarse después de la sección `"-jar plantuml.jar"`.

## 13.4 Definición de Macro

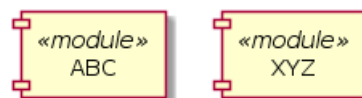
También puedes definir macros con argumentos.

```

@startuml

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

```

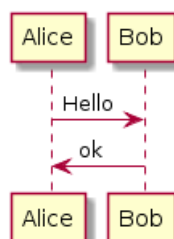


Una macro puede tener varios argumentos.

```

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

```



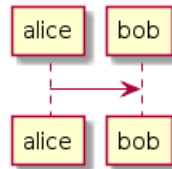
## 13.5 Adding date and time

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

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

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

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



## 13.6 Other special variables

You can also use the following special variables:

`%dirpath%` Path of the current file

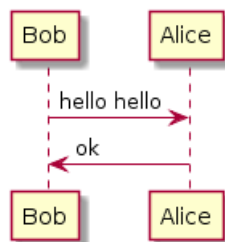
`%filename%` Name of the current file

## 13.7 Macro en varias líneas

También puedes definir una macro en varias líneas usando `!definelong` y `!enddefinelong`.

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

AUTHEN(Bob,Alice)
@enduml
```

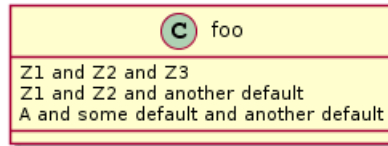


## 13.8 Default values for macro parameters

It is possible to assign default values to macro parameters.

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





## 13.9 Condiciones

Puedes usar las directivas `!ifdef XXX` y `!endif` para tener dibujos condicionales.

Las líneas entre aquellas dos directivas, serán incluidas únicamente si la constante después la directiva `!ifdef`, ha sido previamente definida.

También puedes establecer la parte `!else`, que será incluida si la constante **no** fue definida.

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

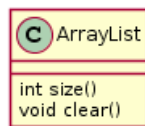


File `ArrayList.iuml`:

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

Puedes también usar la directiva `!define` para activar la parte condicional del diagrama.

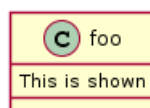
```
@startuml
!define SHOW_METHODS
!include ArrayList.iuml
@enduml
```



También puedes usar la directiva `!ifndef` que incluye líneas si la constante provista, **NO** fue definida.

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

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



## 13.10 Ruta de búsqueda

Puedes especificar la propiedad java "plantuml.include.path" en la línea de comandos.

Por ejemplo:

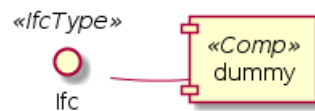
```
java -Dplantuml.include.path="c:/mydir" -jar plantuml.jar atest1.txt
```

Tenga en cuenta que la opción -D debe ser puesta antes de la opción -jar. Las opciones -D después de la opción -jar será usada para definir constantes dentro del preprocesador de plantuml.

## 13.11 Características avanzadas

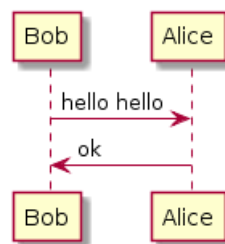
Es posible anexar texto al argumento de una macro usando la sintaxis ##.

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



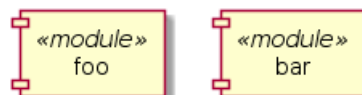
Una macro puede ser definida por otra macro.

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



Una macro puede ser polimórfica con el contador de argumentos.

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



Puedes usar una variable de entorno de sistema o definición de constante cuando usas "include":

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

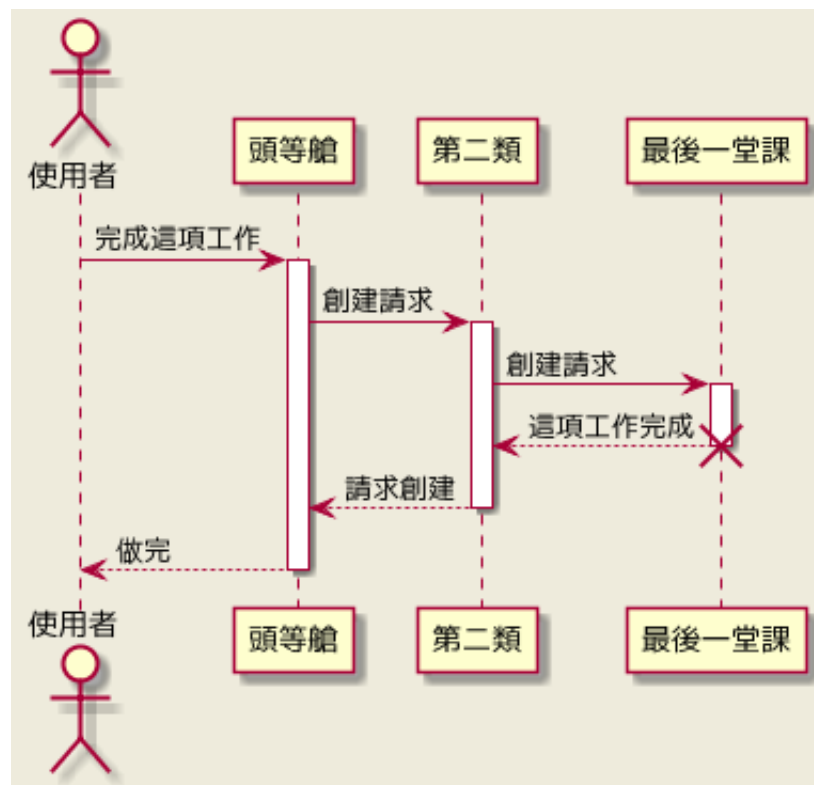




## 14 Internacionalización

El lenguaje de PlantUML usa *letras* para definir actores, casos de uso y demás. Pero *letras* no son únicamente caracteres latinos A-Z, podría *ser cualquier tipo de letra de cualquier lenguaje*.

```
@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 别的東西
使用者 -> A: 完成這項工作
activate A
A -> B: 創建請求
activate B
B -> 别的東西: 創建請求
activate 别的東西
别的東西 --> B: 這項工作完成
destroy 别的東西
B --> A: 請求創建
deactivate B
A --> 使用者: 做完
deactivate A
@enduml
```



### 14.1 Codificación de caracteres (Charset)

La codificación de caracteres usada cuando se lee los archivos de texto que contienen el texto descriptivo UML, depende del sistema. Normalmente, debería estar bien, pero en algunos casos, quizás quieras usar alguna otra codificación de caracteres. Por ejemplo, con la línea de comandos:

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



O, con el ANT task:

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

Dependiendo de tu instalación de Java, la siguiente codificación de caracteres debería estar disponible: ISO-8859-1, UTF-8, UTF-16BE, UTF-16LE, UTF-16.

## 15 Nombres de colores

Esta es la lista de los colores reconocidos por PlantUML. Los nombres son sensibles a mayúsculas y minúsculas.

	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		



## Contents

<b>1</b>	<b>Diagrama de Secuencia</b>	<b>1</b>
1.1	Ejemplo básico . . . . .	1
1.2	Declarando participantes . . . . .	1
1.3	Sin usar letras en participantes . . . . .	2
1.4	Auto-Mensaje . . . . .	3
1.5	Cambiar estilo de la flecha . . . . .	3
1.6	Cambiar el color de la flecha . . . . .	4
1.7	Numeración de la secuencia de mensajes . . . . .	4
1.8	Dividiendo diagramas . . . . .	6
1.9	Agrupando mensajes . . . . .	7
1.10	Notas en mensajes . . . . .	8
1.11	Algunas otras notas . . . . .	8
1.12	Cambiando el aspecto de las notas . . . . .	9
1.13	Creole y HTML . . . . .	10
1.14	Divisor . . . . .	11
1.15	Referencia . . . . .	11
1.16	Retardo . . . . .	12
1.17	Espaciado . . . . .	12
1.18	Activación y Destrucción de la Línea de vida . . . . .	13
1.19	Creación de participante . . . . .	14
1.20	Mensajes entrantes y salientes . . . . .	15
1.21	Estereotipos y marcas . . . . .	16
1.22	Mayor información en los títulos . . . . .	17
1.23	Entorno de participante . . . . .	18
1.24	Removiendo pie de página . . . . .	19
1.25	Personalización (Skinparam) . . . . .	19
1.26	Cambiando el relleno . . . . .	21
<b>2</b>	<b>Diagrama de Casos de Uso</b>	<b>22</b>
2.1	Casos de uso . . . . .	22
2.2	Actores . . . . .	22
2.3	Descripción de Casos de uso . . . . .	22
2.4	Ejemplo básico . . . . .	23
2.5	Extensión . . . . .	23
2.6	Usando notas . . . . .	24
2.7	Estereotipos . . . . .	25
2.8	Cambiar dirección a las flechas . . . . .	25
2.9	Dividiendo los diagramas . . . . .	26
2.10	Dirección: de izquierda a derecha . . . . .	27
2.11	Personalización (Skinparam) . . . . .	28
2.12	Un ejemplo completo . . . . .	29

<b>3</b>	<b>Diagrama de Clases</b>	<b>30</b>
3.1	Relación entre clases . . . . .	30
3.2	Etiquetas en las relaciones . . . . .	31
3.3	Añadiendo métodos . . . . .	32
3.4	Definiendo la visibilidad . . . . .	33
3.5	Abstracto y Estático . . . . .	34
3.6	Cuerpo avanzado de las clases . . . . .	35
3.7	Notas y estereotipos . . . . .	36
3.8	Más acerca de notas . . . . .	37
3.9	Notas en enlaces . . . . .	38
3.10	Clases abstractas e interfaces . . . . .	39
3.11	Sin usar letras . . . . .	40
3.12	Atributos, métodos... ocultos . . . . .	41
3.13	Clases ocultas . . . . .	42
3.14	Uso de clases genéricas . . . . .	42
3.15	Círculo enmarcador específico . . . . .	42
3.16	Paquetes . . . . .	43
3.17	Estilos de paquetes . . . . .	43
3.18	Espacios de nombre . . . . .	44
3.19	Creación automática del espacio de nombre . . . . .	45
3.20	Interface Lollipop . . . . .	46
3.21	Cambiando la dirección de las flechas . . . . .	46
3.22	Asociación de clases . . . . .	47
3.23	Personalización (Skinparam) . . . . .	48
3.24	Estereotipos personalizados . . . . .	49
3.25	Degrado de colores . . . . .	49
3.26	Ayudar en el diseño . . . . .	50
3.27	Dividiendo archivos grandes . . . . .	51
<b>4</b>	<b>Diagrama de Actividades</b>	<b>53</b>
4.1	Actividades simples . . . . .	53
4.2	Etiquetas en las flechas . . . . .	53
4.3	Cambiando la dirección de la flecha . . . . .	53
4.4	Ramas (bifurcaciones) . . . . .	54
4.5	Más acerca de las Ramas . . . . .	55
4.6	Sincronización . . . . .	56
4.7	Descripción de actividades de gran contenido . . . . .	57
4.8	Notas . . . . .	57
4.9	Partición . . . . .	58
4.10	Personalización (Skinparam) . . . . .	59
4.11	Octágono . . . . .	59
4.12	Un ejemplo completo . . . . .	60

<b>5</b>	<b>Diagrama de Actividades (beta)</b>	<b>63</b>
5.1	Una Actividad simple . . . . .	63
5.2	Start/Stop . . . . .	63
5.3	Condicionales . . . . .	64
5.4	El ciclo Repeat . . . . .	65
5.5	El ciclo While . . . . .	65
5.6	Procesamiento paralelo . . . . .	66
5.7	Notas . . . . .	67
5.8	Colores . . . . .	67
5.9	Flechas . . . . .	68
5.10	Agrupación . . . . .	68
5.11	Carriles . . . . .	69
5.12	Desacoplar y remover . . . . .	70
5.13	Otras formas de representación de actividades . . . . .	71
5.14	Un ejemplo completo . . . . .	72
<b>6</b>	<b>Diagrama de Componentes</b>	<b>74</b>
6.1	Componentes . . . . .	74
6.2	Interfaces . . . . .	74
6.3	Ejemplos basicos . . . . .	75
6.4	Usando notas . . . . .	75
6.5	Agrupando componentes . . . . .	76
6.6	Cambiando la dirección de las flechas . . . . .	77
6.7	Utiliza la notación UML2 . . . . .	78
6.8	Long description . . . . .	79
6.9	Colores individuales . . . . .	79
6.10	Using Sprite in Stereotype . . . . .	79
6.11	Personalización (Skinparam) . . . . .	80
<b>7</b>	<b>Diagrama de Estados</b>	<b>82</b>
7.1	Un Estado Simple . . . . .	82
7.2	Estados compuestos . . . . .	82
7.3	Nombres largos . . . . .	83
7.4	Estados simultáneos . . . . .	84
7.5	Dirección de la flecha . . . . .	85
7.6	Notas . . . . .	86
7.7	Más sobre notas . . . . .	87
7.8	Personalización (Skinparam) . . . . .	87
<b>8</b>	<b>Diagrama de Objetos</b>	<b>89</b>
8.1	Definición de objetos . . . . .	89
8.2	Relaciones entre objetos . . . . .	89
8.3	Agregando campos . . . . .	89
8.4	Características comunes en diagramas de clases . . . . .	90

<b>9 Comandos comunes</b>	<b>91</b>
9.1 Comentarios . . . . .	91
9.2 Encabezado y pie de página . . . . .	91
9.3 Zoom (acercamiento) . . . . .	91
9.4 Título . . . . .	92
9.5 Subtítulo . . . . .	93
9.6 Leyenda del diagrama . . . . .	93
<b>10 Salt</b>	<b>94</b>
10.1 Basic widgets . . . . .	94
10.2 Using grid . . . . .	94
10.3 Using separator . . . . .	95
10.4 Tree widget . . . . .	95
10.5 Enclosing brackets . . . . .	96
10.6 Adding tabs . . . . .	96
10.7 Using menu . . . . .	97
10.8 Advanced table . . . . .	98
<b>11 Creole</b>	<b>99</b>
11.1 Emphasized text . . . . .	99
11.2 List . . . . .	99
11.3 Escape character . . . . .	100
11.4 Horizontal lines . . . . .	100
11.5 Headings . . . . .	100
11.6 Legacy HTML . . . . .	101
11.7 Table . . . . .	102
11.8 Tree . . . . .	102
11.9 Special characters . . . . .	103
11.10 OpenIconic . . . . .	103
11.11 Definiendo y usando sprites . . . . .	105
11.12 Encoding Sprite . . . . .	105
11.13 Importing Sprite . . . . .	106
11.14 Ejemplos . . . . .	106
<b>12 Cambiar fuentes y colores</b>	<b>108</b>
12.1 Uso . . . . .	108
12.2 Anidado . . . . .	108
12.3 Color . . . . .	109
12.4 Color de fuente, nombre y tamaño . . . . .	110
12.5 Blanco y negro . . . . .	113

<b>13 Preprocesamiento</b>	<b>114</b>
13.1 Incluyendo archivos . . . . .	114
13.2 Incluyendo URL . . . . .	114
13.3 Definición de constante . . . . .	114
13.4 Definición de Macro . . . . .	115
13.5 Adding date and time . . . . .	116
13.6 Other special variables . . . . .	116
13.7 Macro en varias líneas . . . . .	116
13.8 Default values for macro parameters . . . . .	116
13.9 Condiciones . . . . .	117
13.10 Ruta de búsqueda . . . . .	118
13.11 Características avanzadas . . . . .	118
<b>14 Internacionalización</b>	<b>120</b>
14.1 Codificación de caracteres (Charset) . . . . .	120
<b>15 Nombres de colores</b>	<b>122</b>