Plant UML

Plant UML

Objectifs

Représentation graphique des diagrammes

- Open source (écrit en java : plantuml.jar)
- Utilise Graphviz
- Fichier texte

Installation

- Interfaçable avec de nombreux outils
  openoffice, gedit, vim, latex, ...
  Brackets, SublimeText, ...
  netbeans, eclipse, IntelliJ, ...

- Génération automatique de diagramme

Mode console

Variable environnement :

GRAPHVIZ_DOT = /usr/bin/dot

Utilisation :

java -jar plantuml.jar fichier.txt

=> Génération image .png

Plant UML

Eclipse

Install new software

http://plantuml.sourceforge.net/updatesite

Ouvrir perspective PlantUML

Plant UML

libreOffice

Installation

plantuml.jar → Copier dans répertoire home/plantuml

home : outils → options → libreoffice → chemin → mesdocuments

Récupérer fichier plantuml.odt → macro installer plantUML

! Autoriser l'exécution des macros

Utilisation

Créer fichier texte avec code PlantUML

Bouton UML → Génère le schéma dans la feuille

Use Case

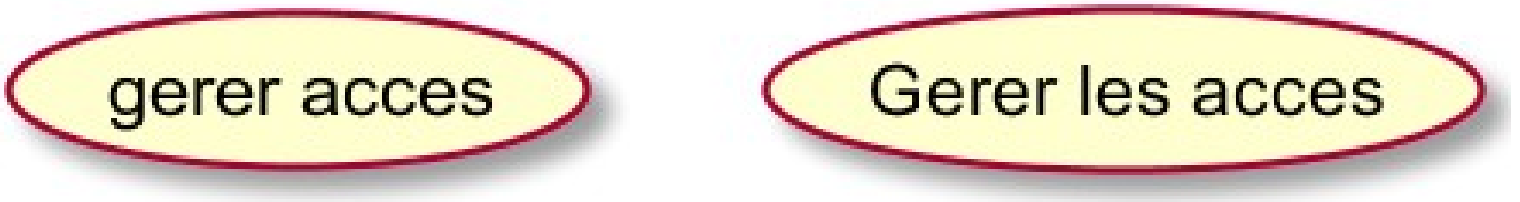
Plant UML

Acteur

```
:admin:
actor :Le petit poucet: as LPP
```

Use Case

```
(Gerer acces)
usecase "Gerer les acces" as UC1
```



Plant UML

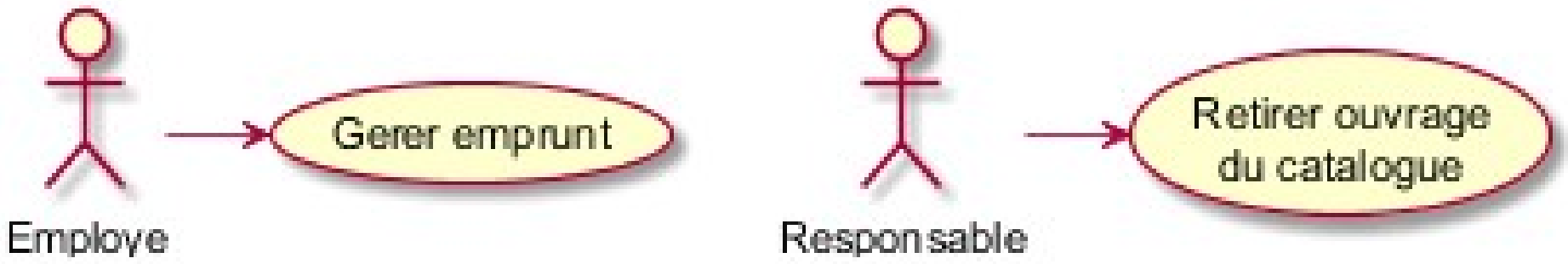
Séparateurs - - . . = = _ _

```
usecase UC1 as "Gerer les acces
- -
description sur
plusieurs lignes
= = Attention = =
Fin"
```

! Contenu délimité par "

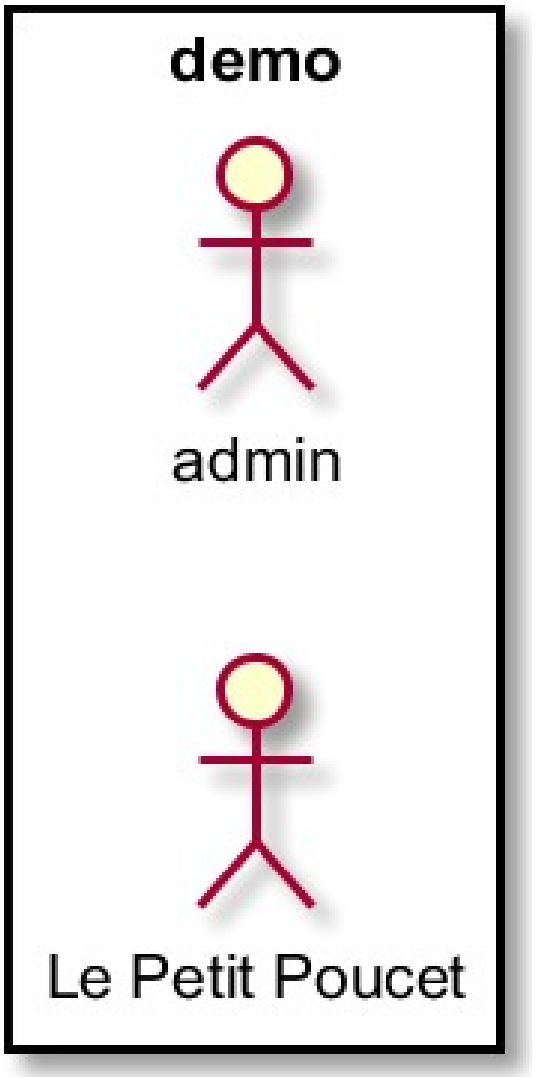
Plant UML

```
@startuml
:Employe: -> (Gerer emprunt)
:Responsable: -> (Retirer ouvrage\ndu catalogue)
@enduml
```



Plant UML

```
@startuml
left to right direction

rectangle demo {
  actor admin
  actor "Le Petit Poucet"
}

@enduml
```



Plant UML

Représentation des liaisons

Liaisons	- >	droite	< -	gauche
	- - >	bas	< - -	haut
	- - - >	Lien plus long		

Liens orientés	- up - >	- down - >
	- right - >	- left - >

! Si ambiguïté, erreur d'interprétation

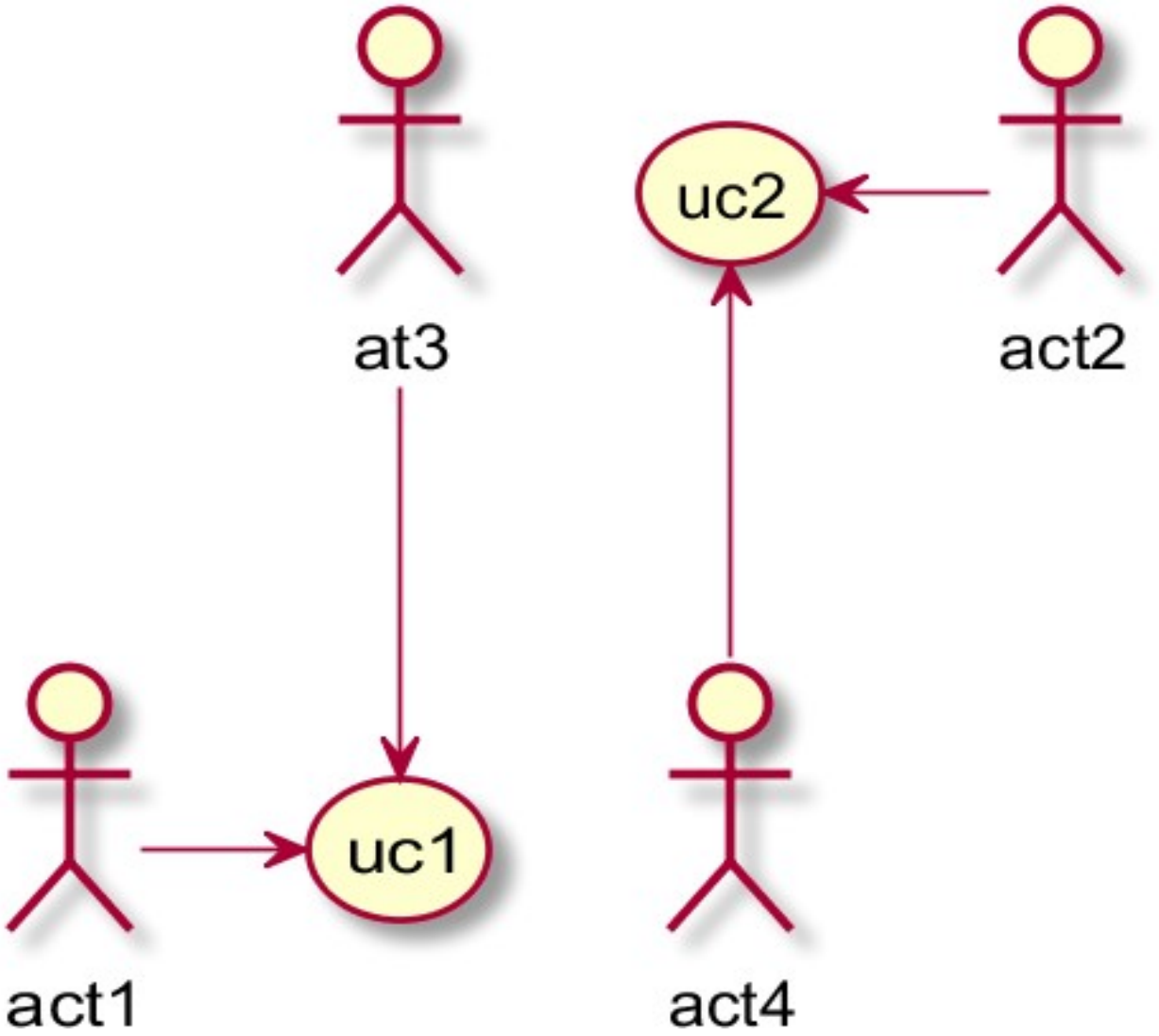
Plant UML

:act1: -> (uc1)

(uc2) <- :act2:

:at3: --> (uc1)

(uc2) <-- :act4:



Plant UML

:act1: -> (uc1)

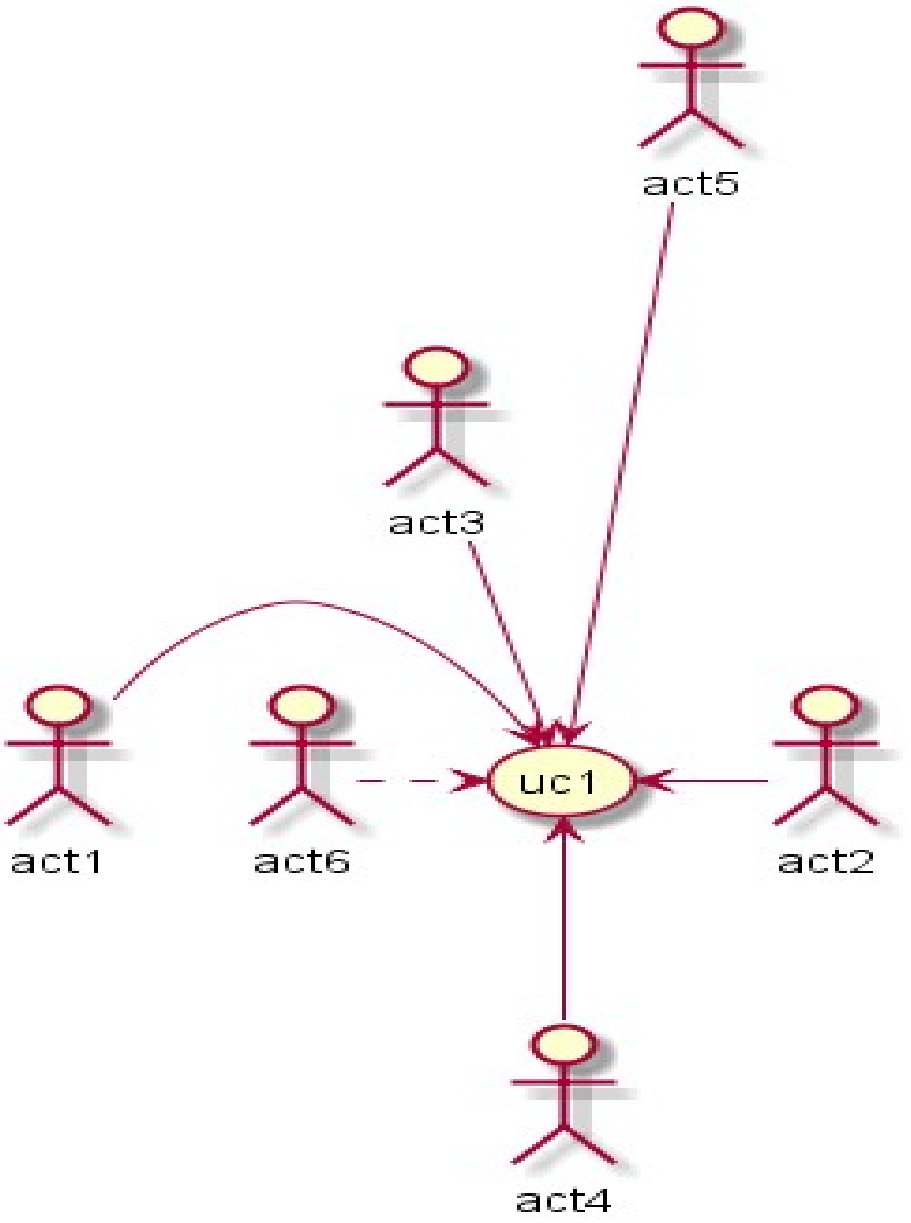
(uc1) <- :act2:

:act3: --> (uc1)

(uc1) <-- :act4:

:act5: ---> (uc1)

:act6: .> (uc1)



Plant UML

left to right direction

:act1: -> (uc1)

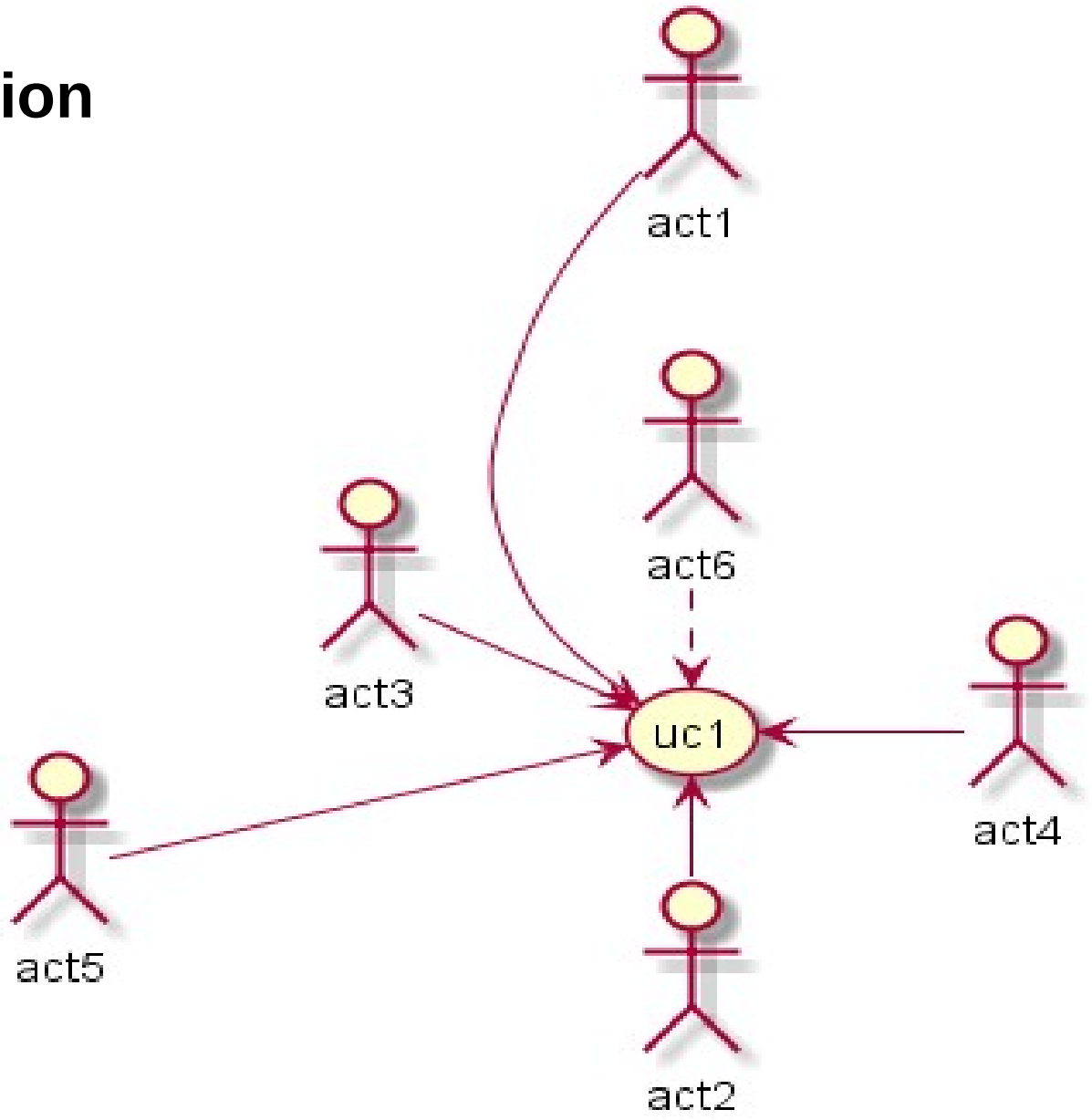
(uc1) <- :act2:

:act3: --> (uc1)

(uc1) <-- :act4:

:act5: ---> (uc1)

:act6: .> (uc1)



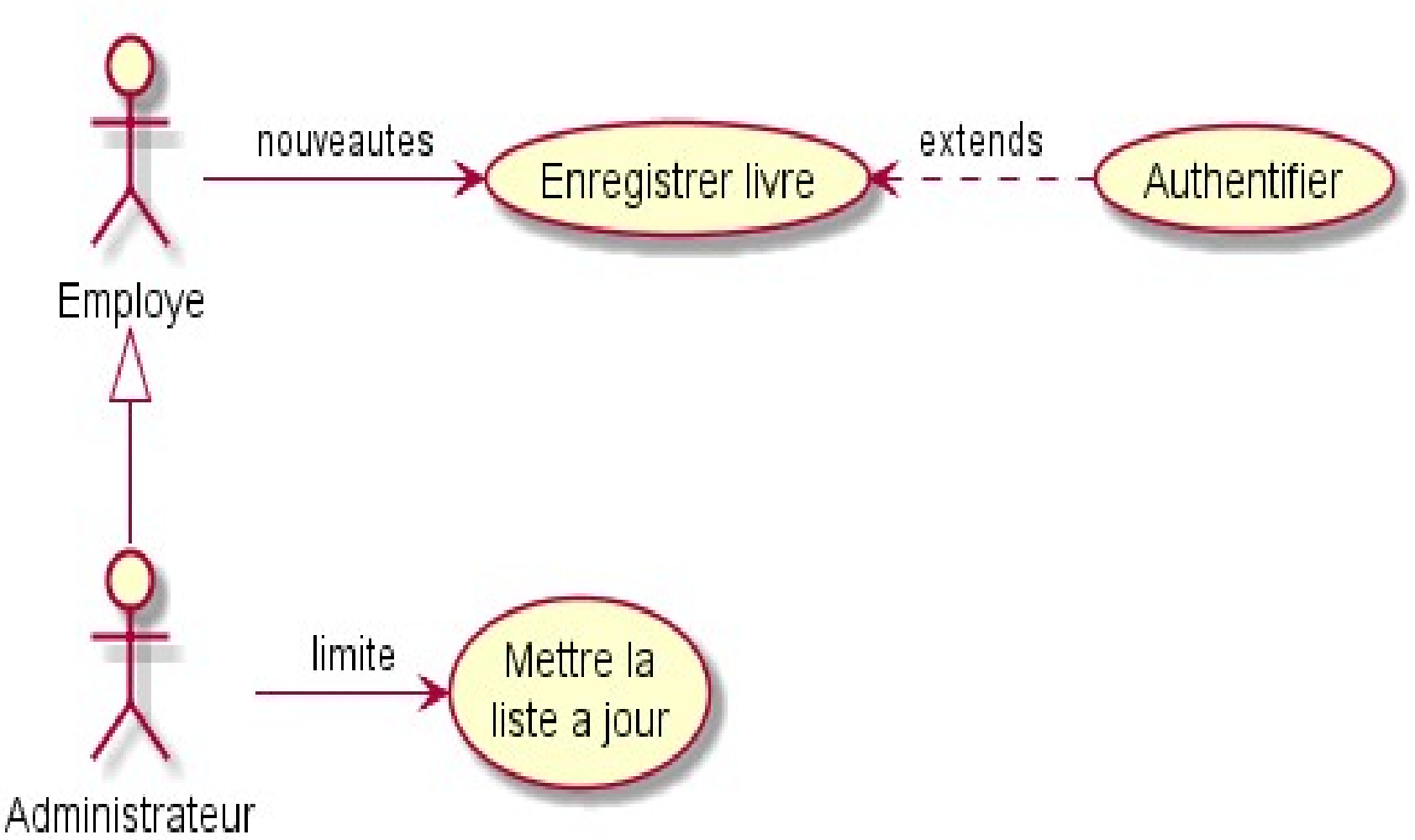
Plant UML

actor Employe **as act1** ← Définition d'alias
actor Administrateur **as act2**

usecase "Enregistrer livre" **as uc1**
usecase "Authentifier" **as uc2**
usecase "Mettre la\nliste a jour" **as uc3**

act1 -> (**uc1**) : nouveautes ← Message sur lien
(**uc1**)<. (**uc2**) : extends
act2 -> (**uc3**) : limite
act1 <|-- **act2** ← Héritage

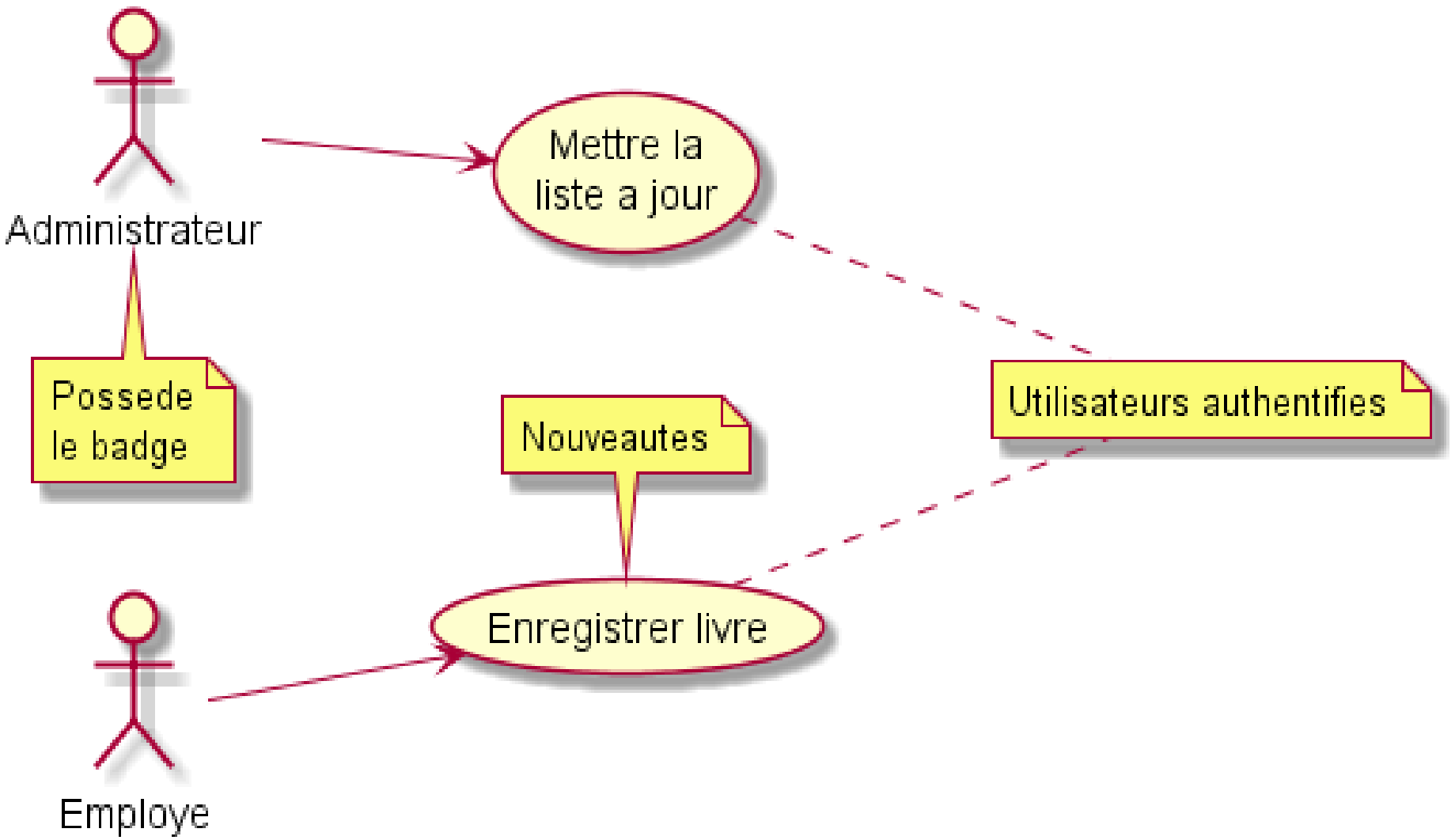
Plant UML


nouveautes
Enregistrer livre
extends
Authentifier
Employe
limite
Mettre la
liste a jour
Administrateur

Plant UML

```
left to right direction
usecase "Enregistrer livre" as UC1
usecase "Mettre la\nliste a jour" as UC2
:Employe: --> (UC1)
:Administrateur: --> (UC2)

note bottom of :Administrateur: : Possede\nle badge
note top of (UC1) : Nouveautes
note "Utilisateurs authentifies" as N1
(UC1) .. N1
(UC2) .. N1
```

Plant UML


Administrateur
Mettre la liste a jour
Possede le badge
Nouveautes
Utilisateurs authentifies
Enregistrer livre
Employe

Plant UML

actor user <<human>>

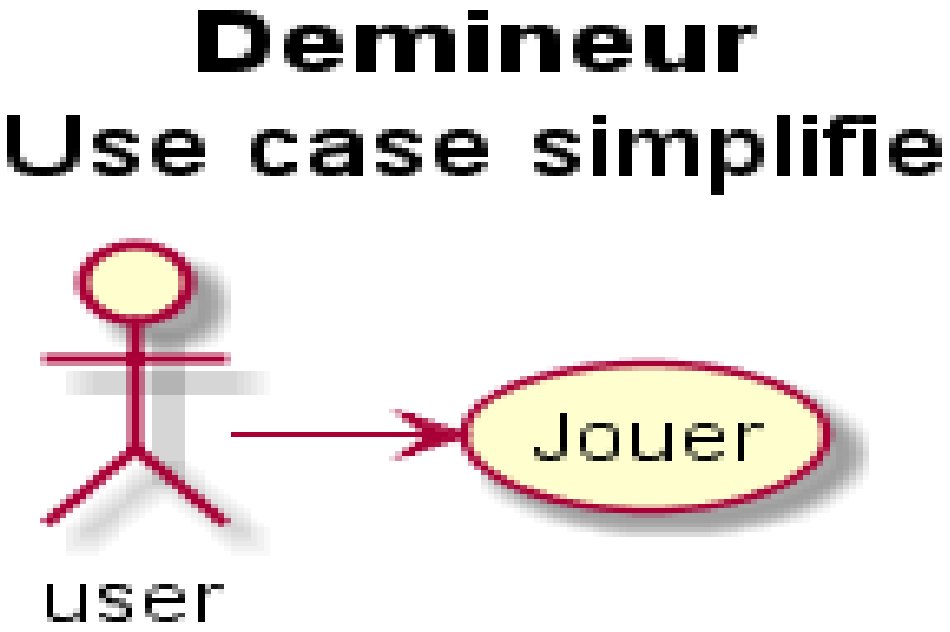
usecase jouer <<main>>

← Stéréotype

Plant UML

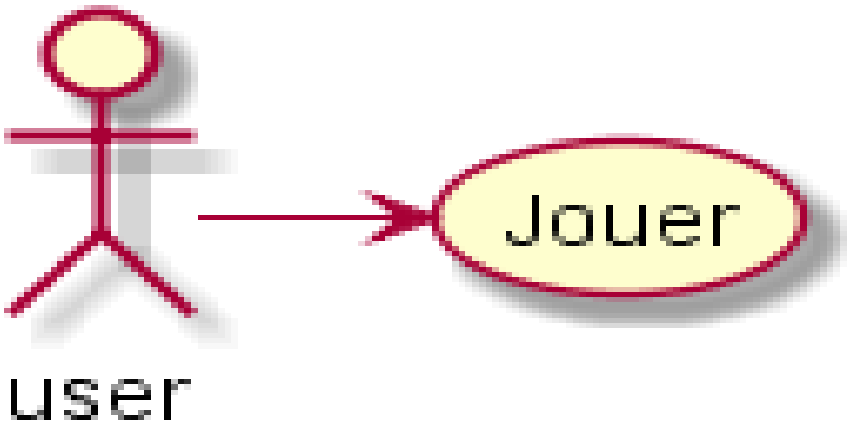
```
title <b>Demineur</b>\nUse case simplifie
:user: -> (Jouer)
```

← Titre



Plant UML

```
:user: -> (Jouer)
legend
Commentaire :
Cas d utilisation
endlegend
```



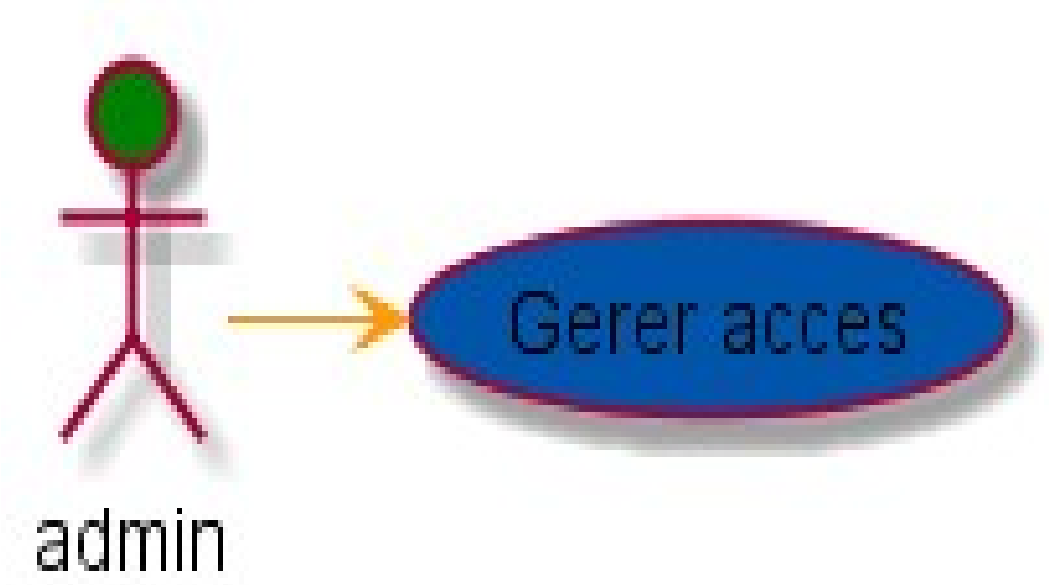
Plant UML

Mise en forme HTML

<b>
<u>
<i>
<s>,<del>,<strike>
<font color="AAAAAA"> or <font color="ColorName">
<color:AAAAAA> or <color:ColorName>
<size:nn> font size
<img src="chemin"> or <img:chemin>

Plant UML

```
:admin: #green
usecase "Gerer acces" as UC1 #0055AA
admin -[#FF9900]> UC1
```



Plant UML

```
skinparam usecase {
  BackgroundColor LightBlue
  BorderColor DarkSlateGray
}
skinparam actor {
  BorderColor Black
  FontColor Red
  FontName TimesNewRoman
}

:user: -> (Jouer)
```

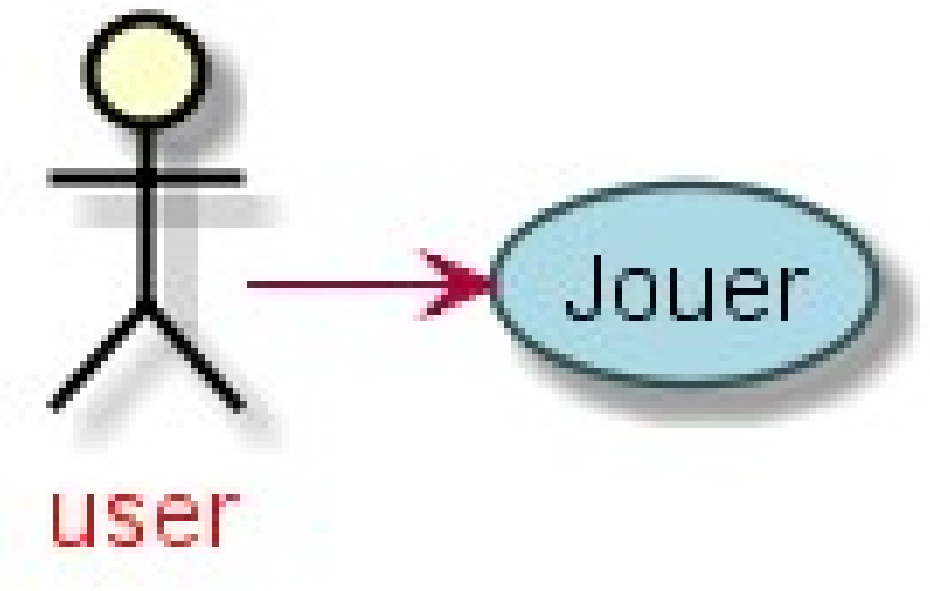
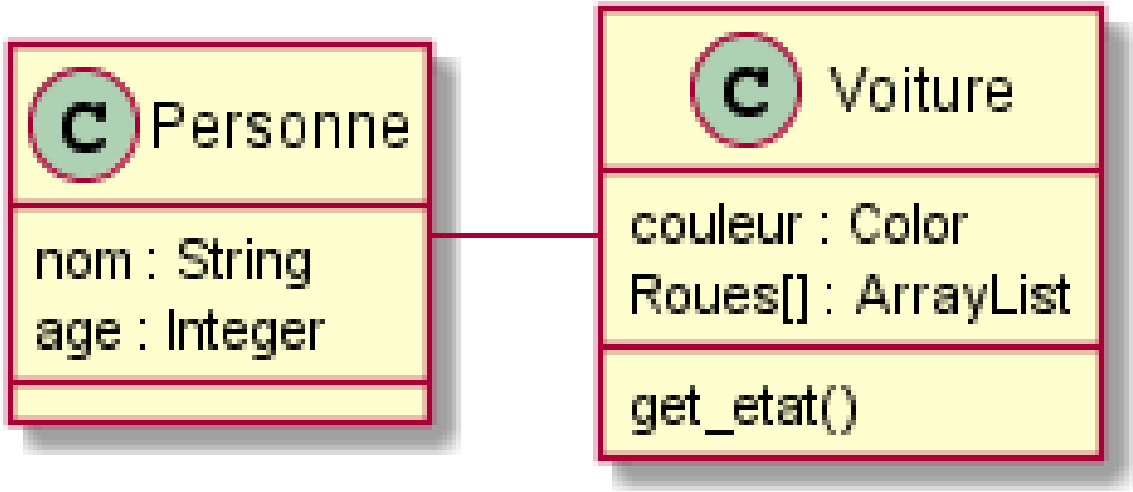


Diagramme de classe

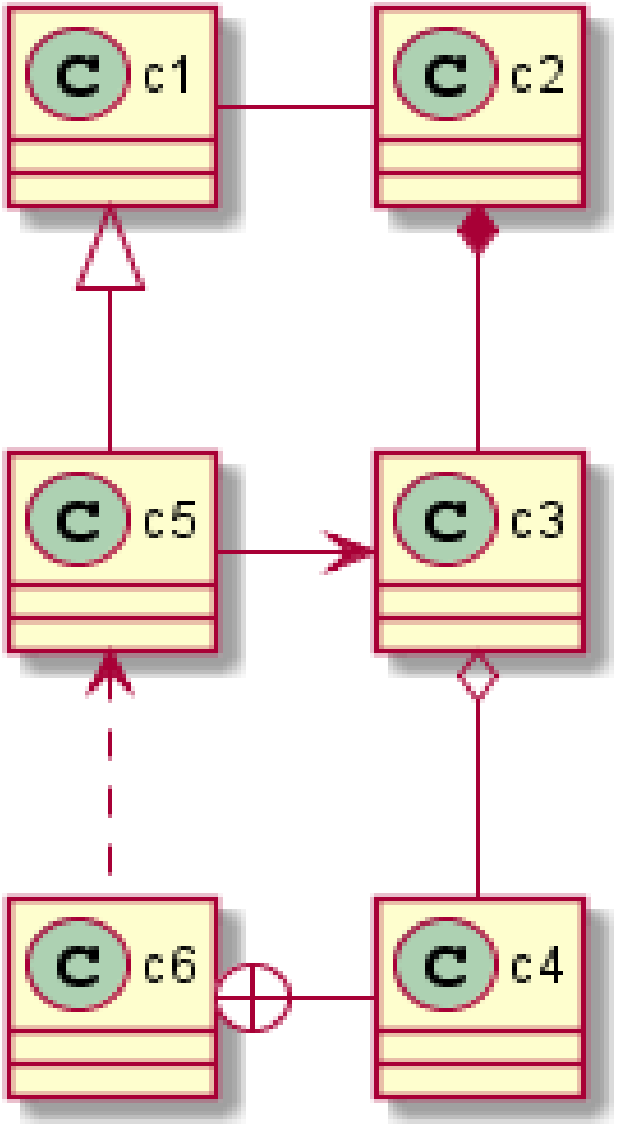
Plant UML

```
class "Personne" as CP {
nom : String
age : Integer
}
class "Voiture" as CV {
couleur : Color
Roues[ ] : ArrayList
get_etat( )
}

CP - CV
```



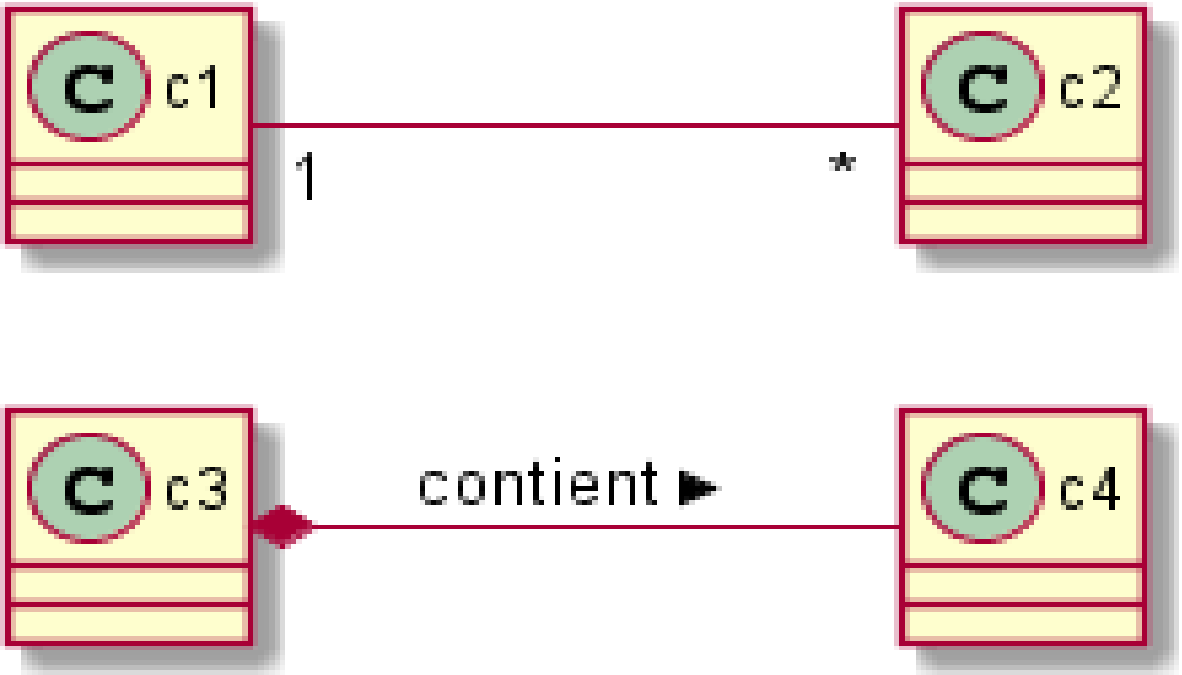
Plant UML

```
class c1
...
c1 - c2
c2 *-- c3
c3 o-- c4
c1 <|-- c5
c5 <.. c6
c5 -> c3
c6 +- c4
```



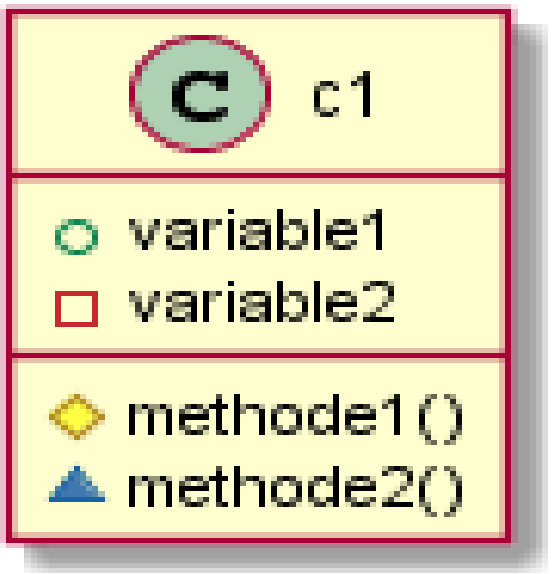
Plant UML

```
left to right direction
class c3
class c4
class c1
class c2

c1 "1" -- "*" c2
c3 *-- c4 : contient >
```



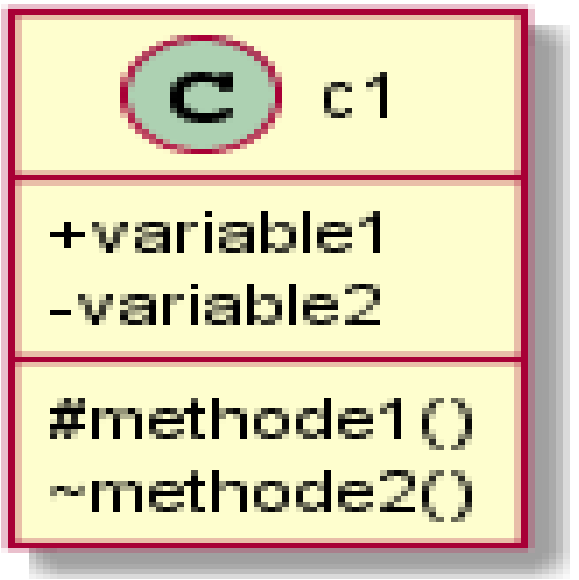
Plant UML

```
class c1 {
  +variable1
  -variable2
  #methode1()
  ~methode2()
}
```



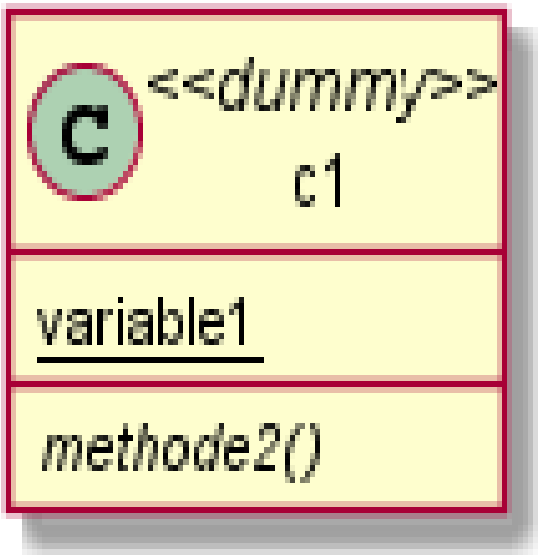
Plant UML

```
skinparam ClassAttributeIconSize 0

class c1 {
  +variable1
  -variable2
  #methode1()
  ~methode2()
}
```



Plant UML

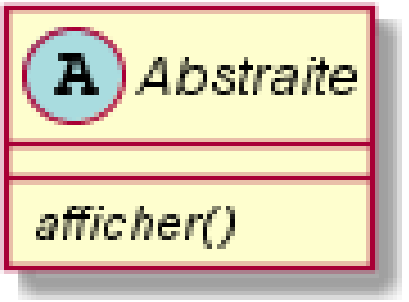
```
class c1 <<dummy>> {
  {static} variable1
  {abstract} methode2()
}
```



Plant UML

```
class Normal
```

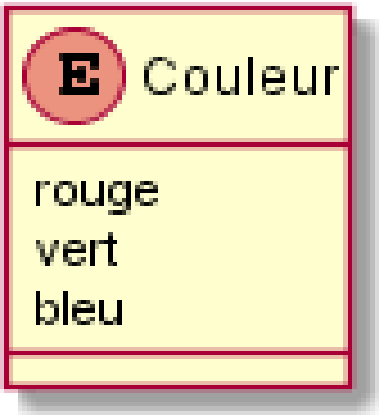

```
abstract Abstraite {
  {abstract} afficher()
}
```




```
interface Interface
```

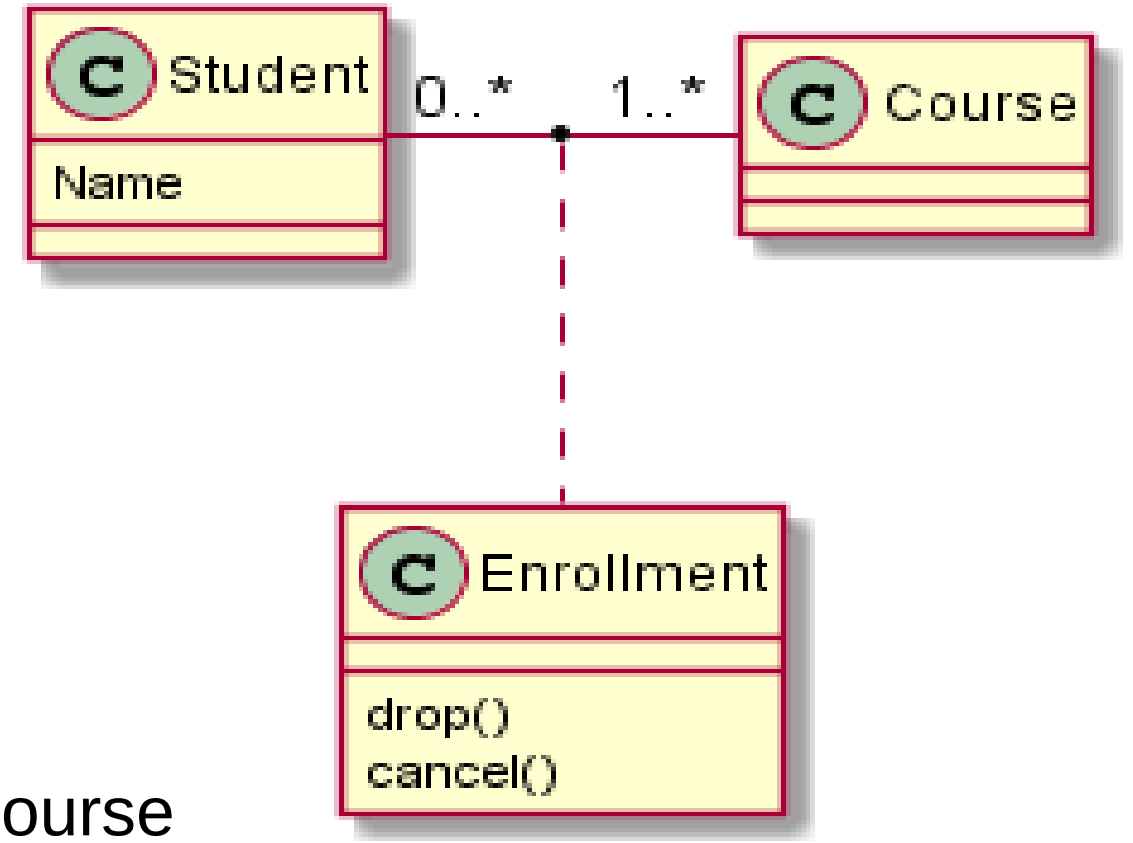
Plant UML

```
enum Couleur {
  rouge
  vert
  bleu
}

annotation Listener
```



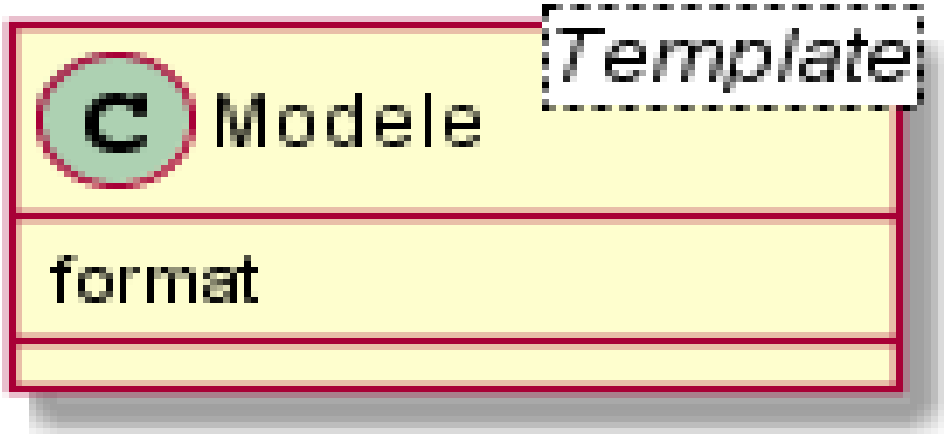
Plant UML

```
class Student {
  Name
}
class Course
class Enrollment {
  drop()
  cancel()
}

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



Plant UML

```
class Modele <Template> {
  format
}
```



Plant UML

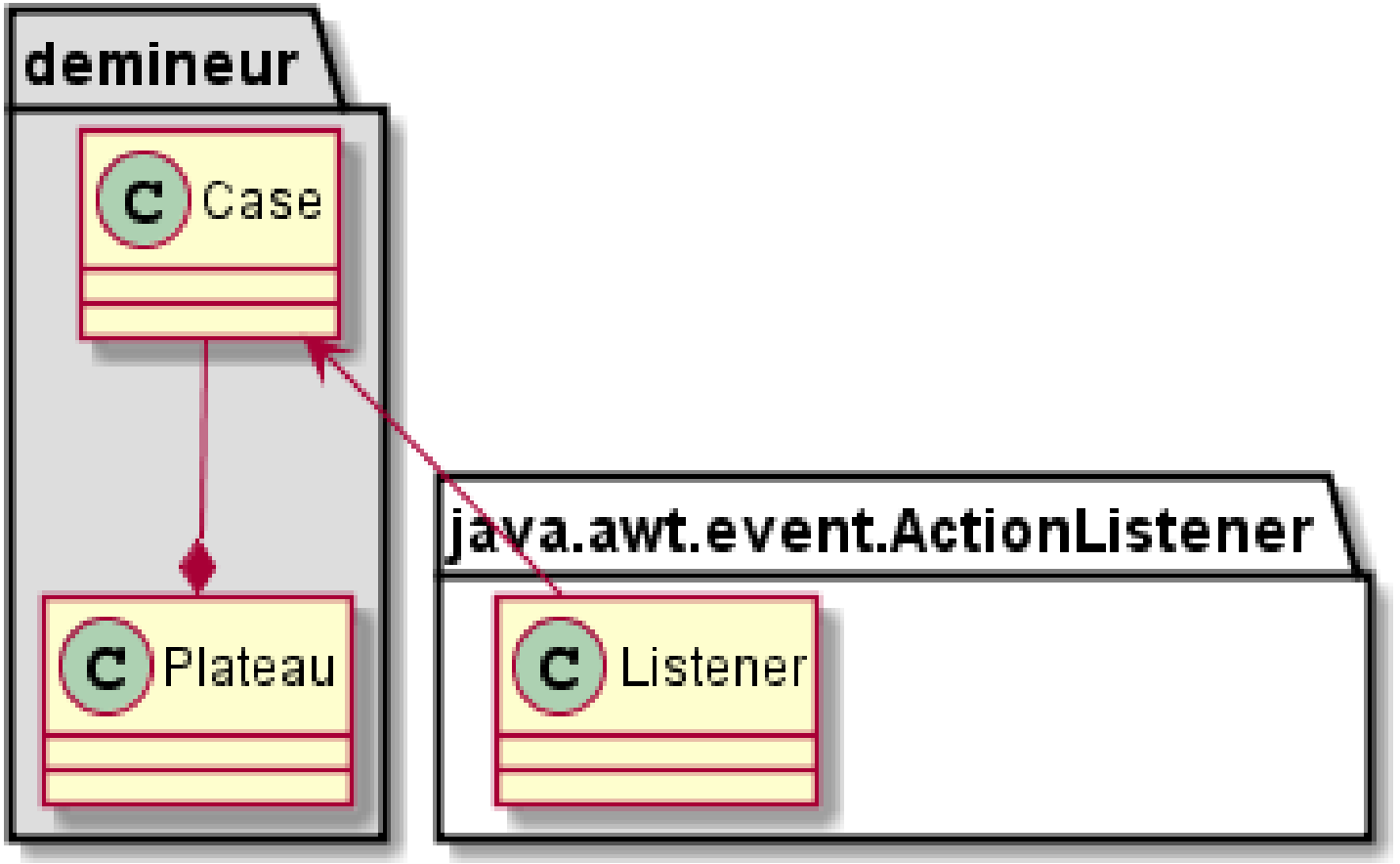
```
skinparam ClassAttributeIconSize 0
class Plateau << (S,FF7700) Singleton>> {
  -instance
  getInstance()
}
```

Diagramme de package

Plant UML

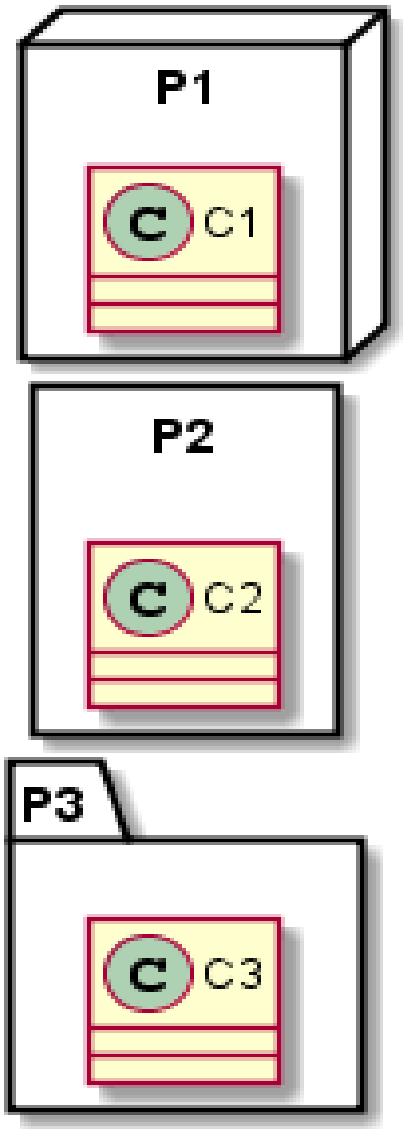
```
package "demineur" #DDDDDD  {
  class Plateau
  class Case
  Case --* Plateau
}

Package "java.awt.event.ActionListener" {
 class Listener
}

Case <- Listener
```

Plant UML


demineur
Case
Plateau
java.awt.event.ActionListener
Listener

Plant UML

```
package P1 <<Node>> {
  class C1
}

package P2 <<Rect>> {
  class C2
}

package P3 <<Folder>> {
  class C3
}
```



Plant UML

```
package P1 <<Database>> {
  class C1
}

package P2 <<Cloud>> {
  class C2
}

package P3 <<Frame>> {
  class C3
}
```

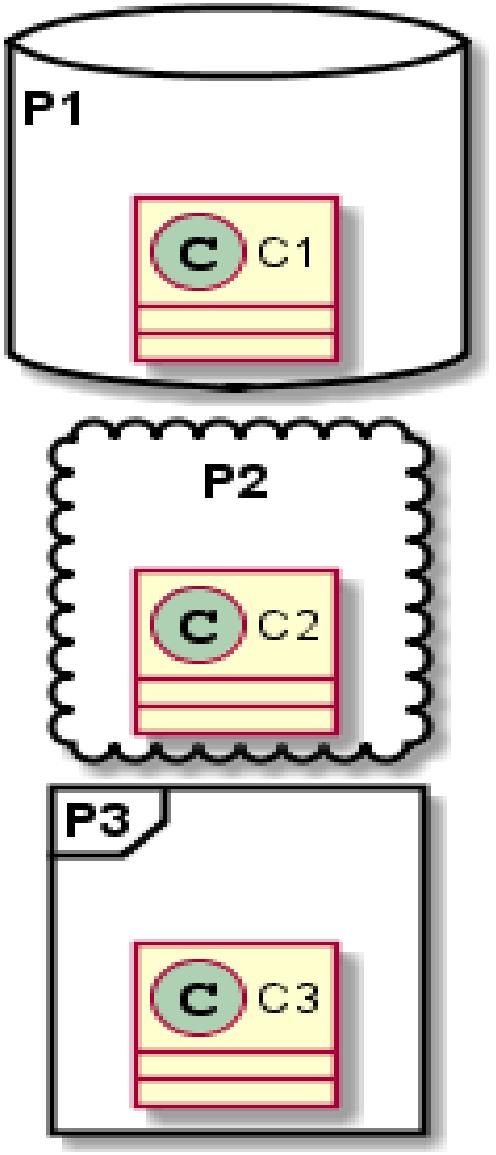
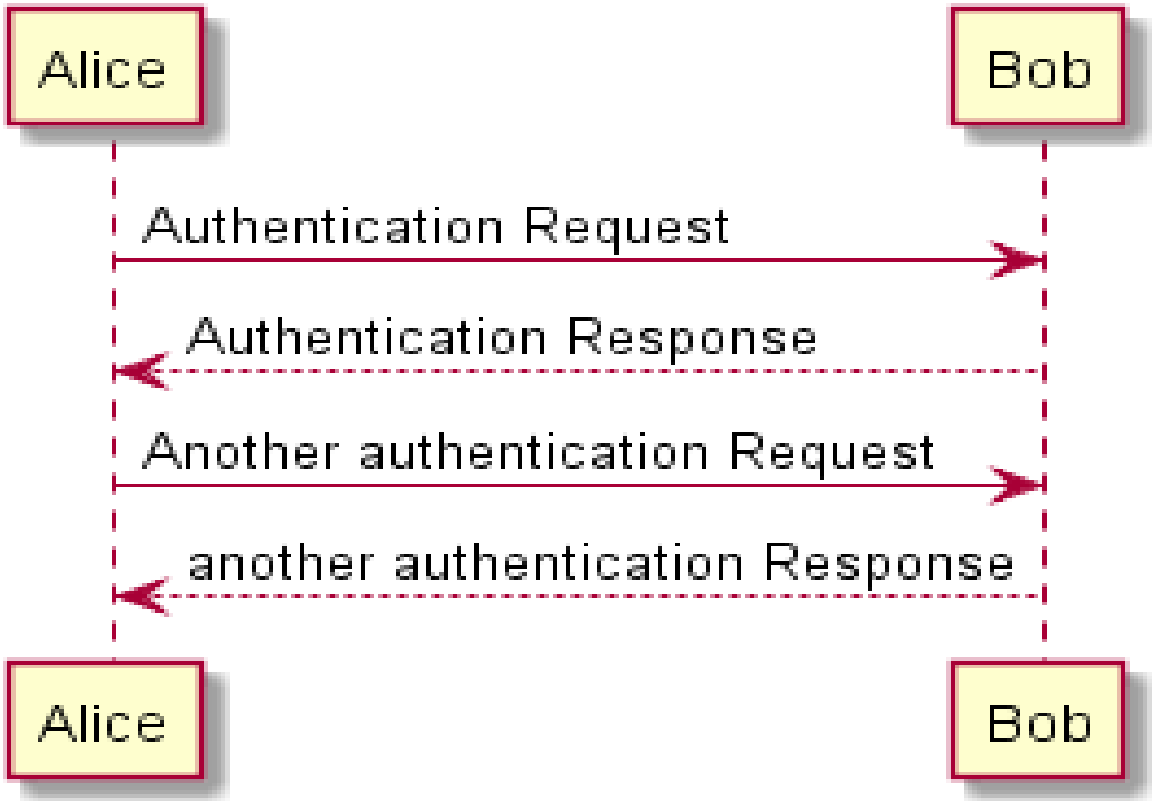


Diagramme de séquence

Plant UML

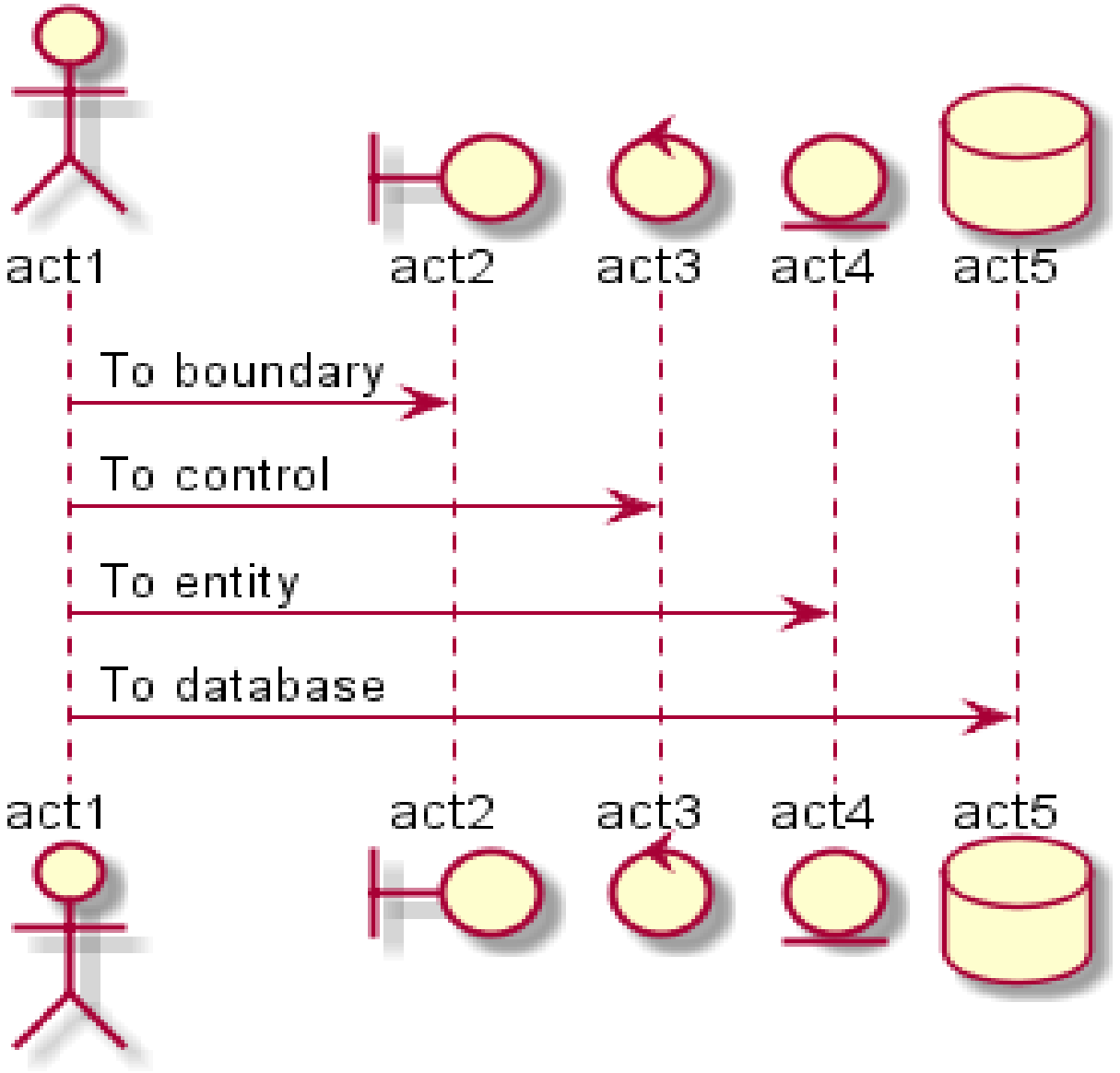
```
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
Alice -> Bob: Another authentication Request
Alice <-- Bob: another authentication Response
```



Plant UML

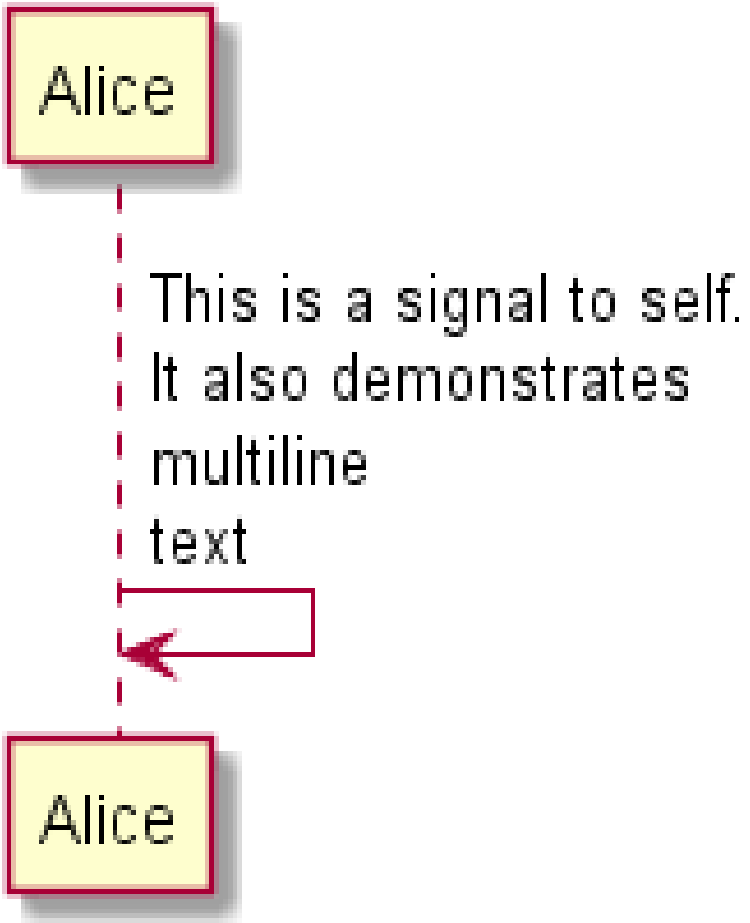
```
actor act1
boundary act2
control act3
entity act4
database act5
act1 -> act2 : To boundary
act1 -> act3 : To control
act1 -> act4 : To entity
act1 -> act5 : To database
```

Plant UML


act1
act2
act3
act4
act5
To boundary
To control
To entity
To database
act1
act2
act3
act4
act5

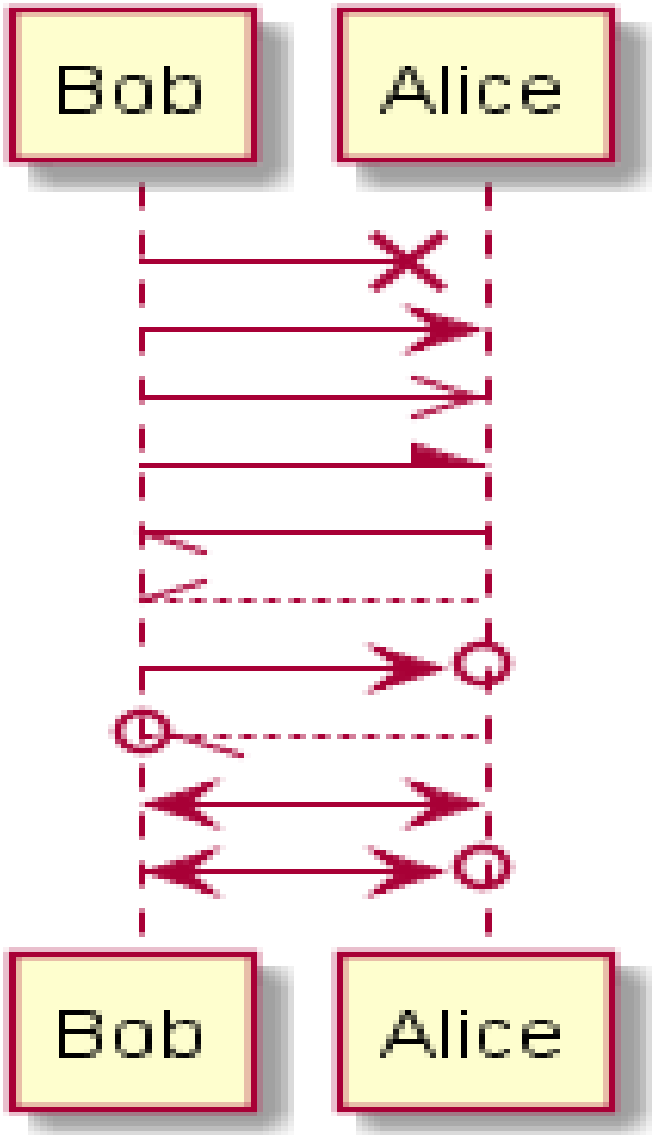
Plant UML

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



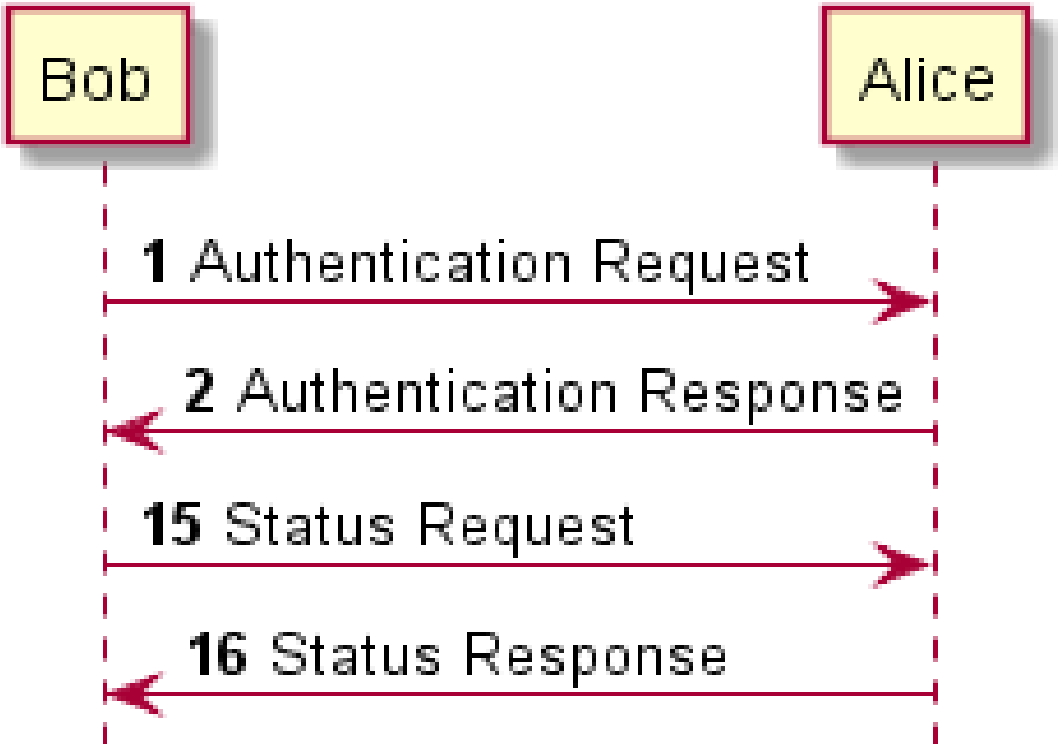
Plant UML

```
Bob ->x Alice
Bob -> Alice
Bob ->> Alice
Bob -\ Alice
Bob \\- Alice
Bob //-- Alice
Bob ->o Alice
Bob o\\-- Alice
Bob <-> Alice
Bob <->o Alice
```



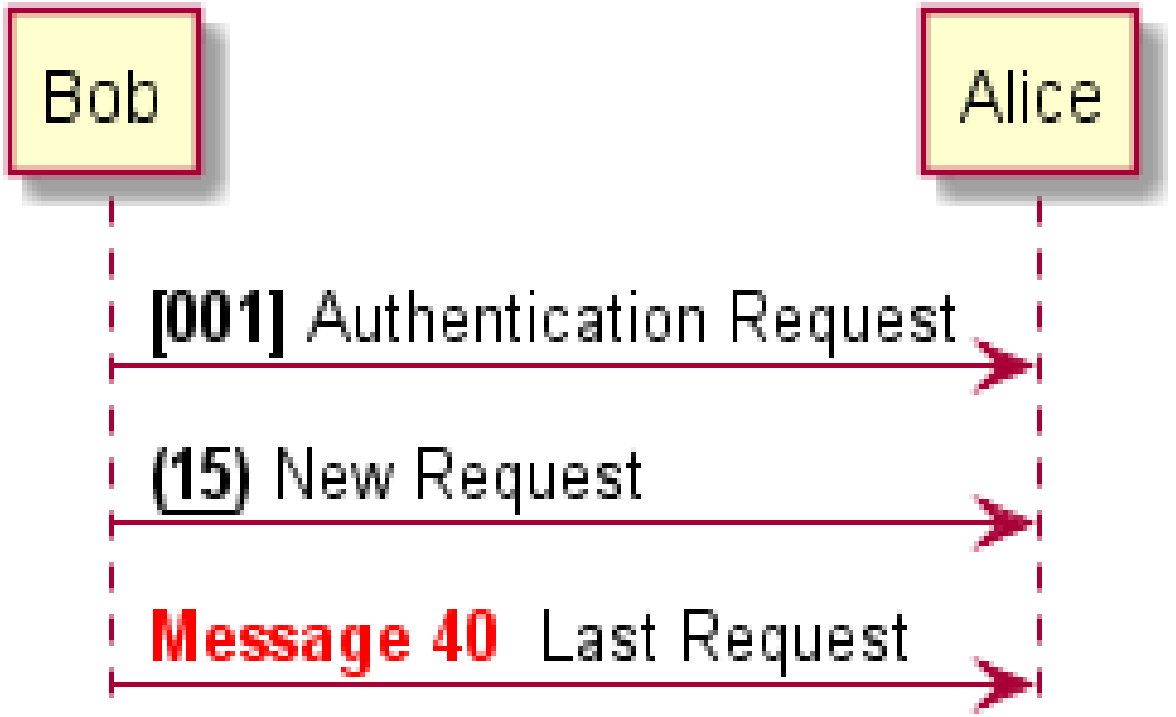
Plant UML

```
autonumber
Bob -> Alice : Authentication Request
Bob <- Alice : Authentication Response
autonumber 15
Bob -> Alice : Status Request
Bob <- Alice : Status Response
```



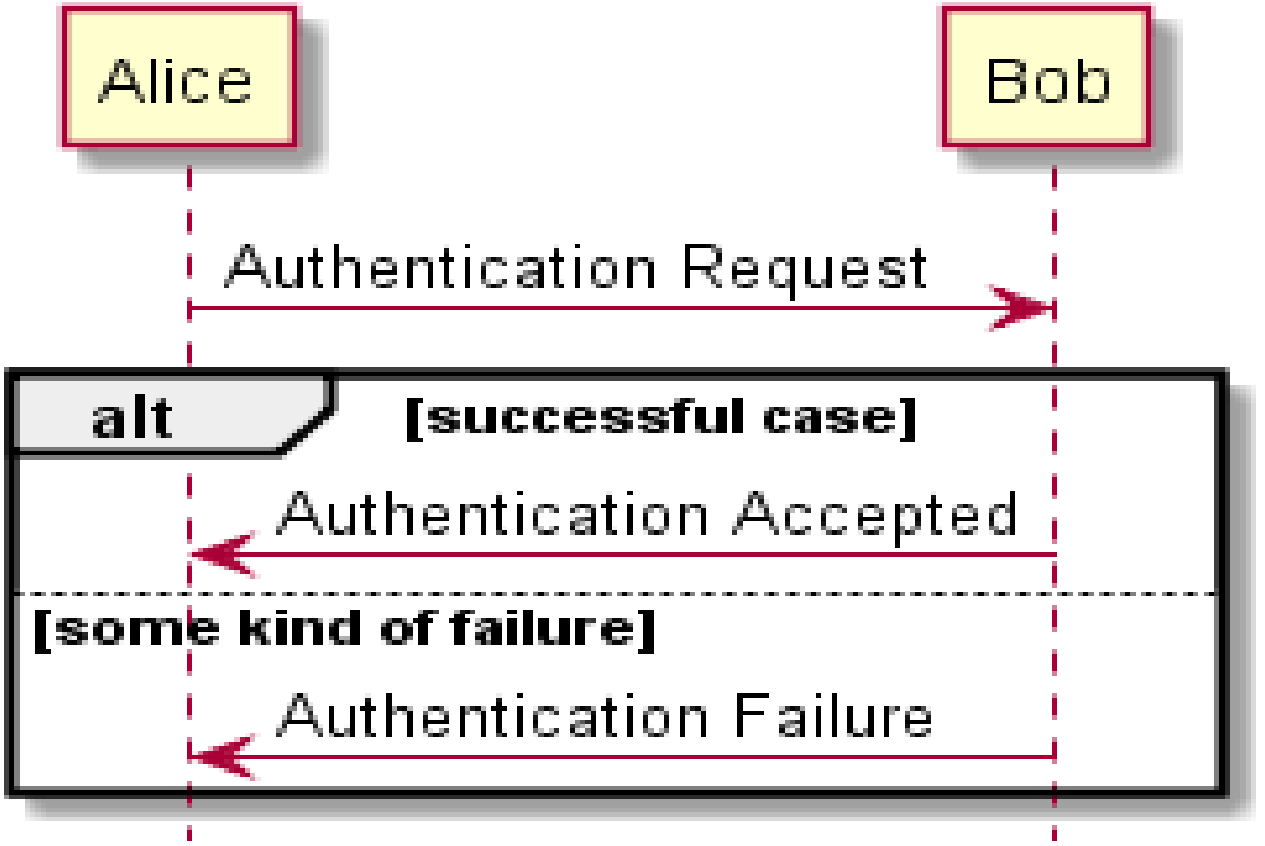
Plant UML

```
autonumber "<b>[000]"
Bob -> Alice : Authentication Request
autonumber 15 "<b>(<u>##</u>)"
Bob -> Alice : New Request
autonumber 40 10 "<font color=red ><b>Message 0 "
Bob -> Alice : Last Request
```



Plant UML

```
Alice -> Bob: Authentication Request
alt successful case
  Bob -> Alice: Authentication Accepted
else some kind of failure
  Bob -> Alice: Authentication Failure
end
```



Plant UML

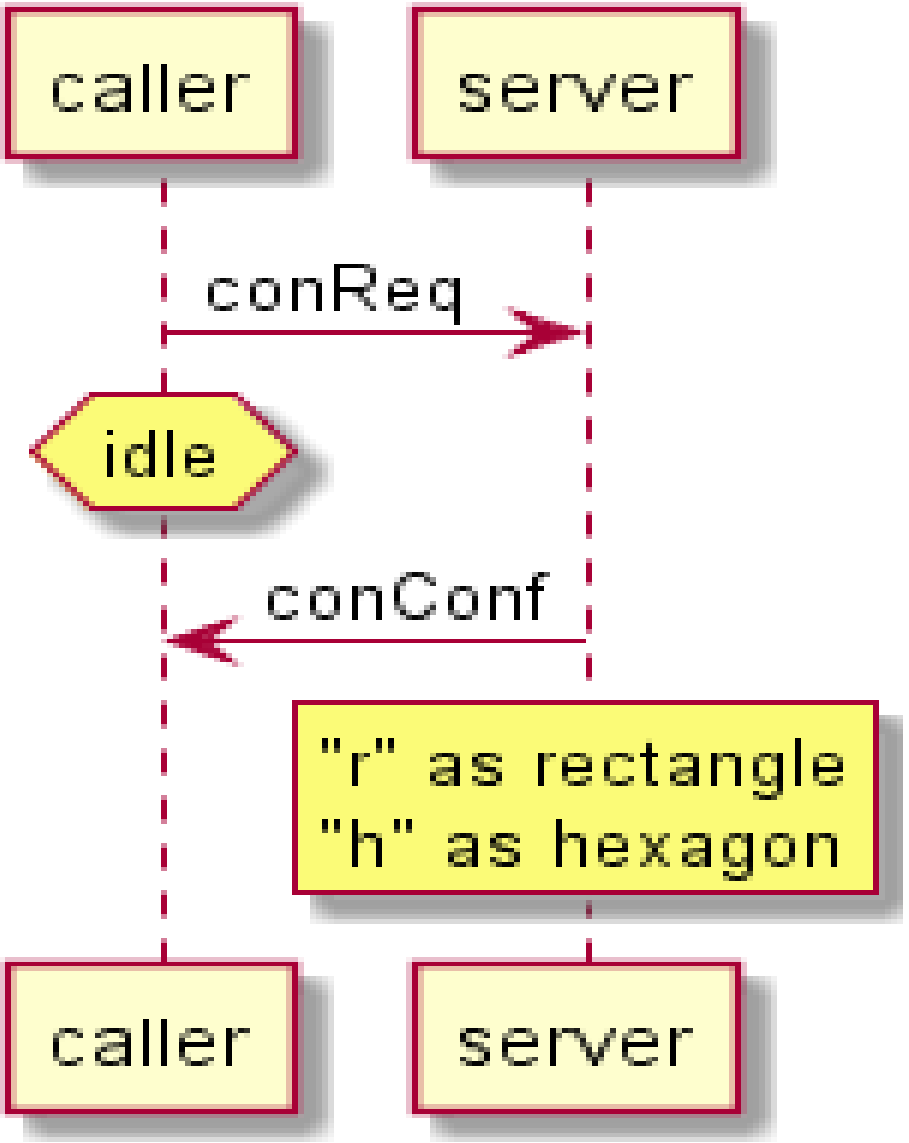
Regroupements :

- alt / else
- opt
- loop
- par
- break
- critical

- group "texte à afficher"

Plant UML

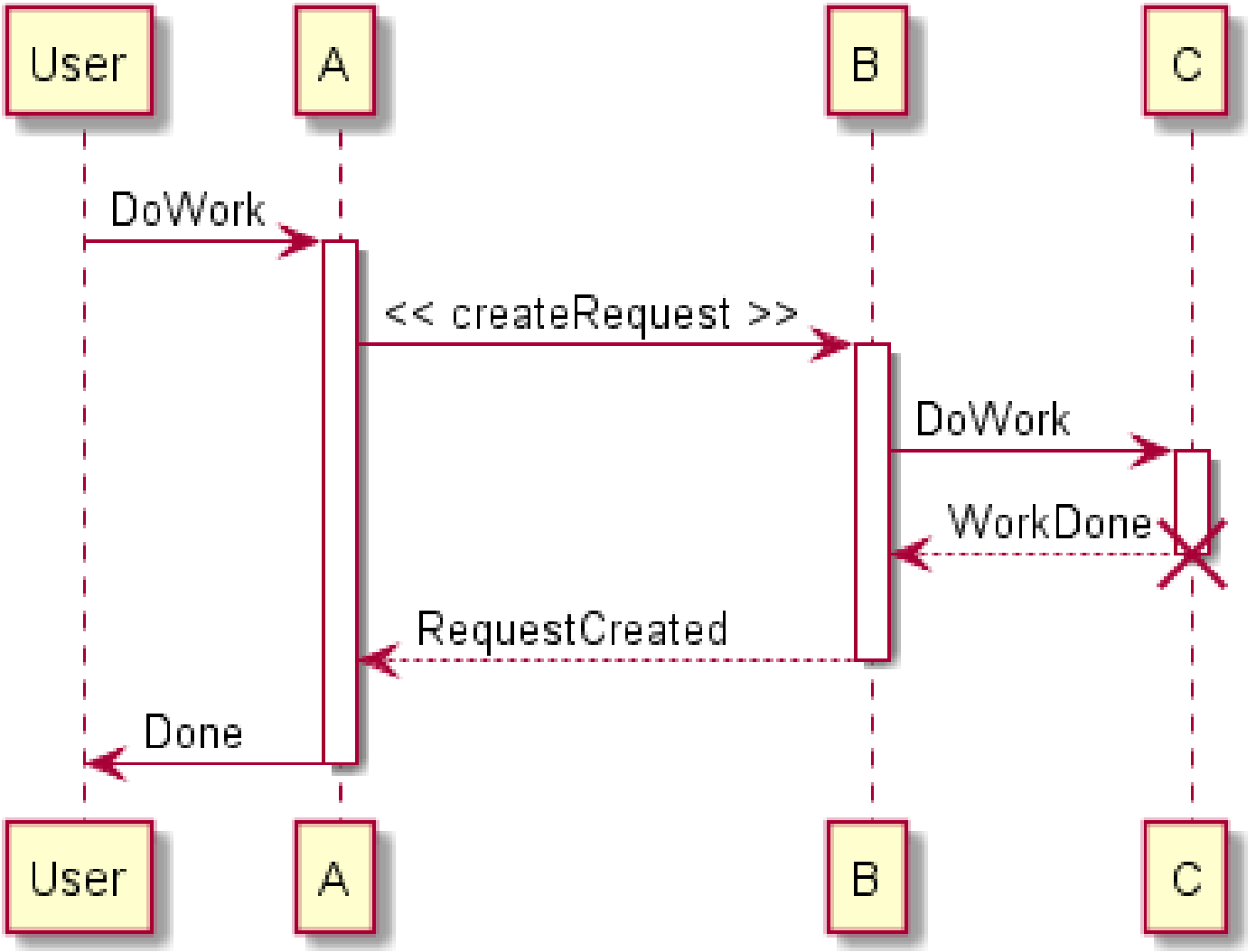
```
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
"r" as rectangle
"h" as hexagon
endrnote
```



Plant UML

```
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
```

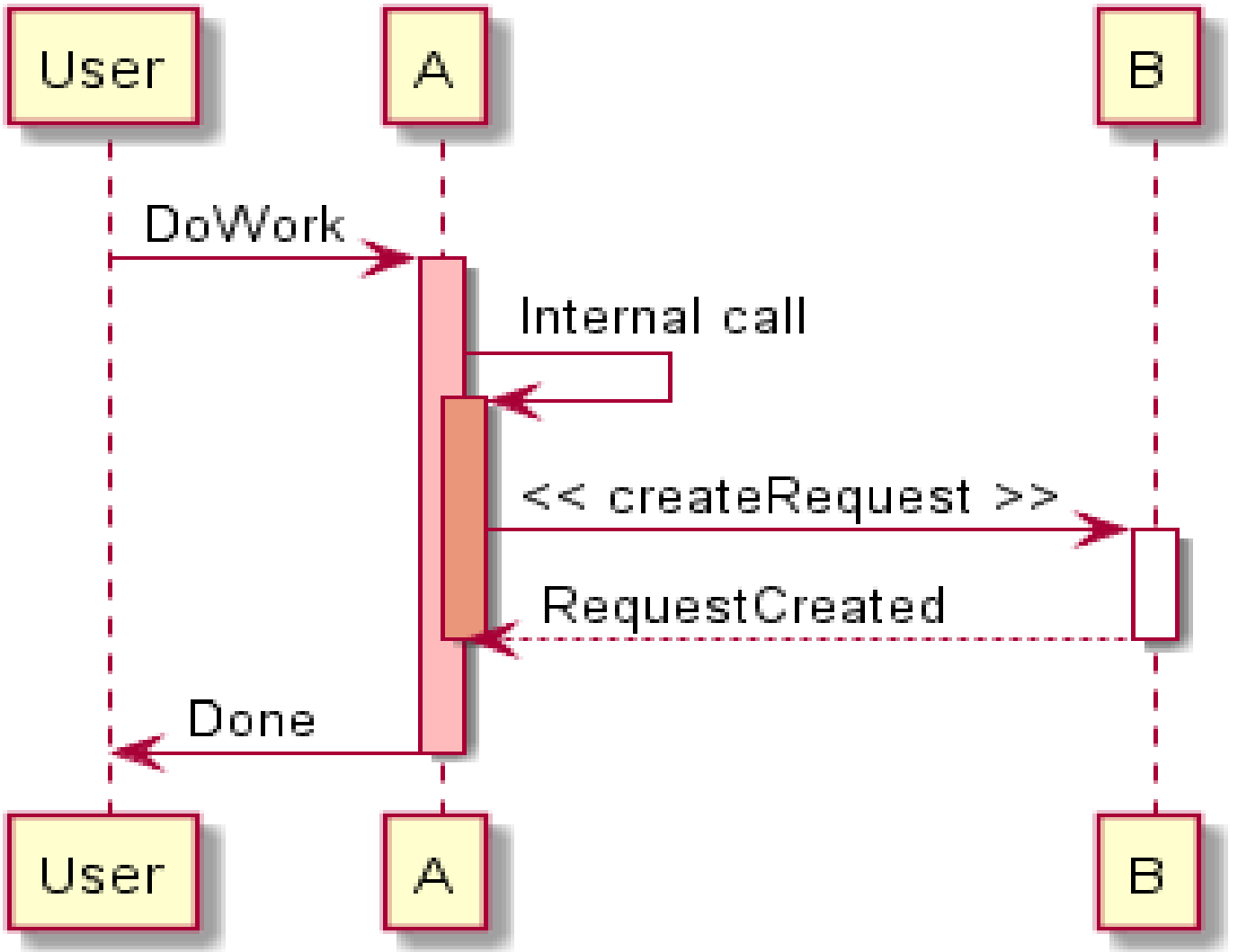
Plant UML


User
A
B
C
DoWork
<< createRequest >>
DoWork
WorkDone
RequestCreated
Done
User
A
B
C

Plant UML

```
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
```

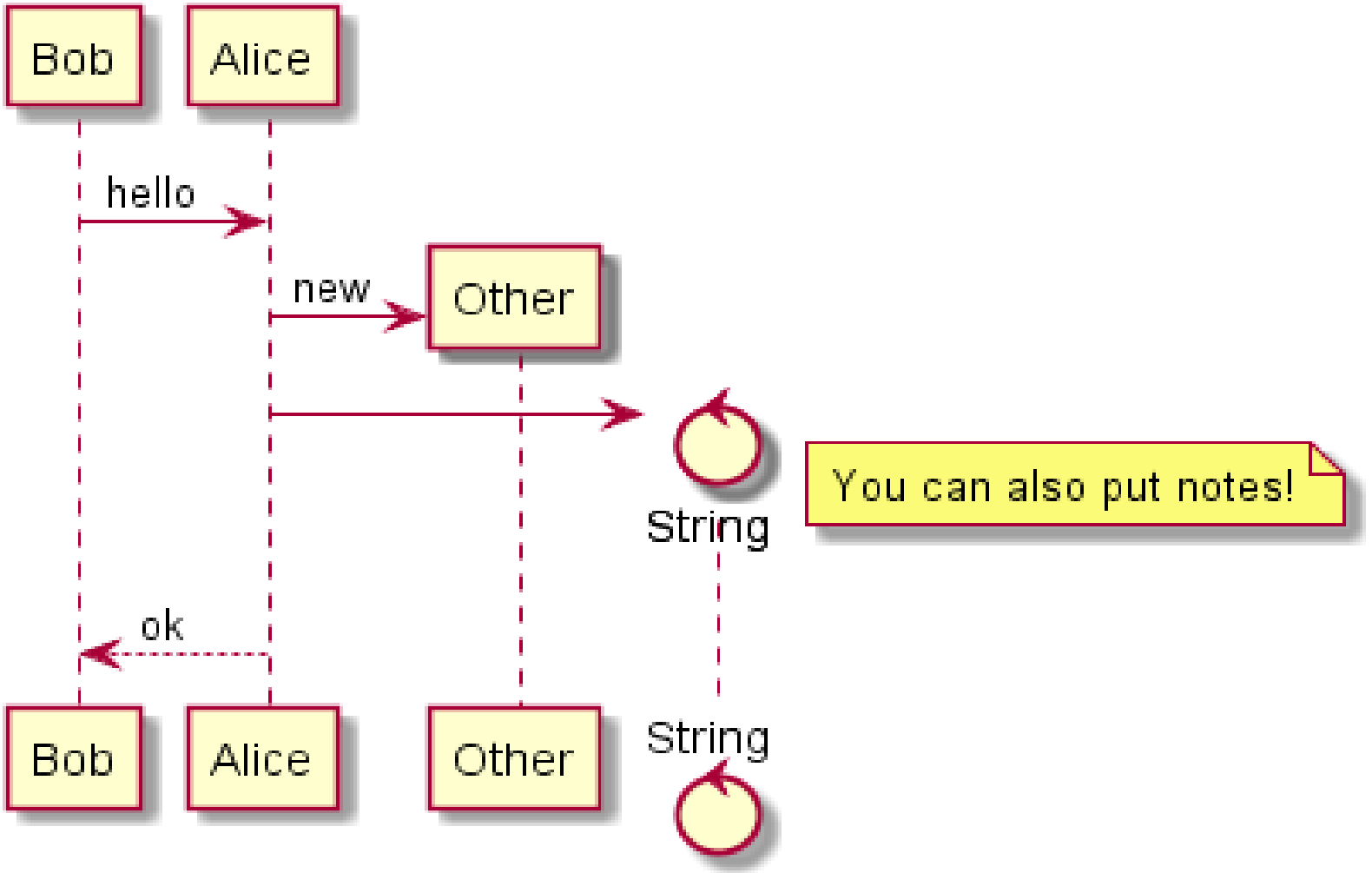
Plant UML


User
A
B
DoWork
Internal call
<< createRequest >>
RequestCreated
Done
User
A
B

Plant UML

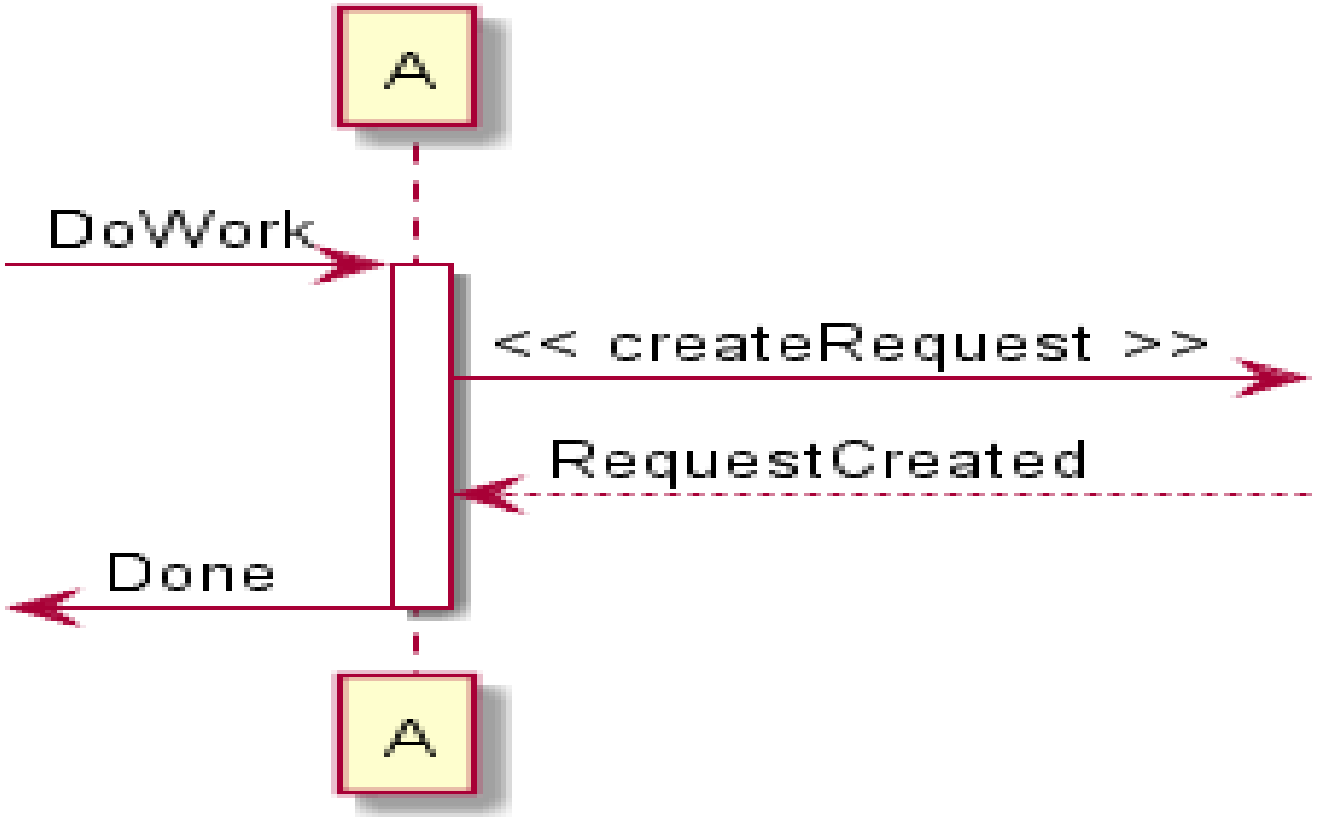
```
Bob -> Alice : hello
create Other
Alice -> Other : new
create control String
Alice -> String
note right : You can also put notes!
Alice --> Bob : ok
```

Plant UML


Bob
Alice
hello
new
Other
String
You can also put notes!
ok
Bob
Alice
Other
String

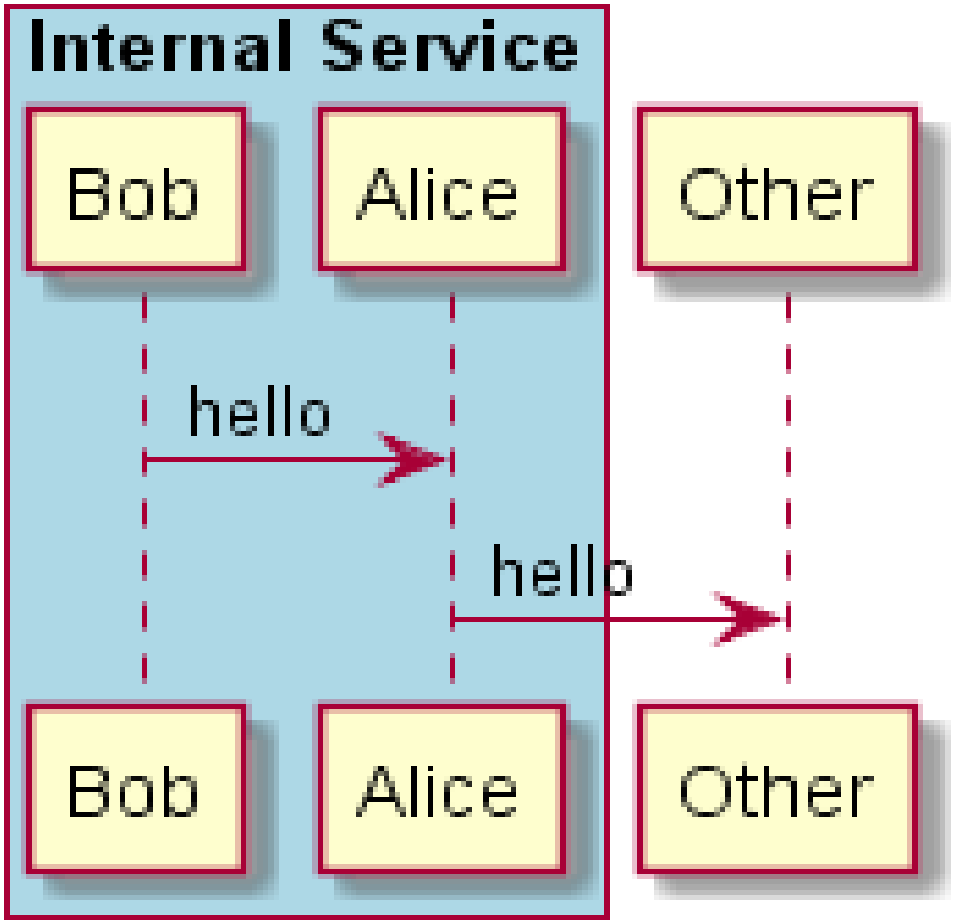
Plant UML

```
[-> A: DoWork
activate A
A ->] : << createRequest >>
A<--] : RequestCreated
[<- A: Done
deactivate A
```



Plant UML

```
box "Internal Service" #LightBlue
participant Bob
participant Alice
end box
participant Other
Bob -> Alice : hello
Alice -> Other : hello
```



Plant UML

```
hide footbox
title Footer removed
Alice -> Bob: Authentication Request
Bob --> Alice: Authentication Response
```

Plant UML

```
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
```

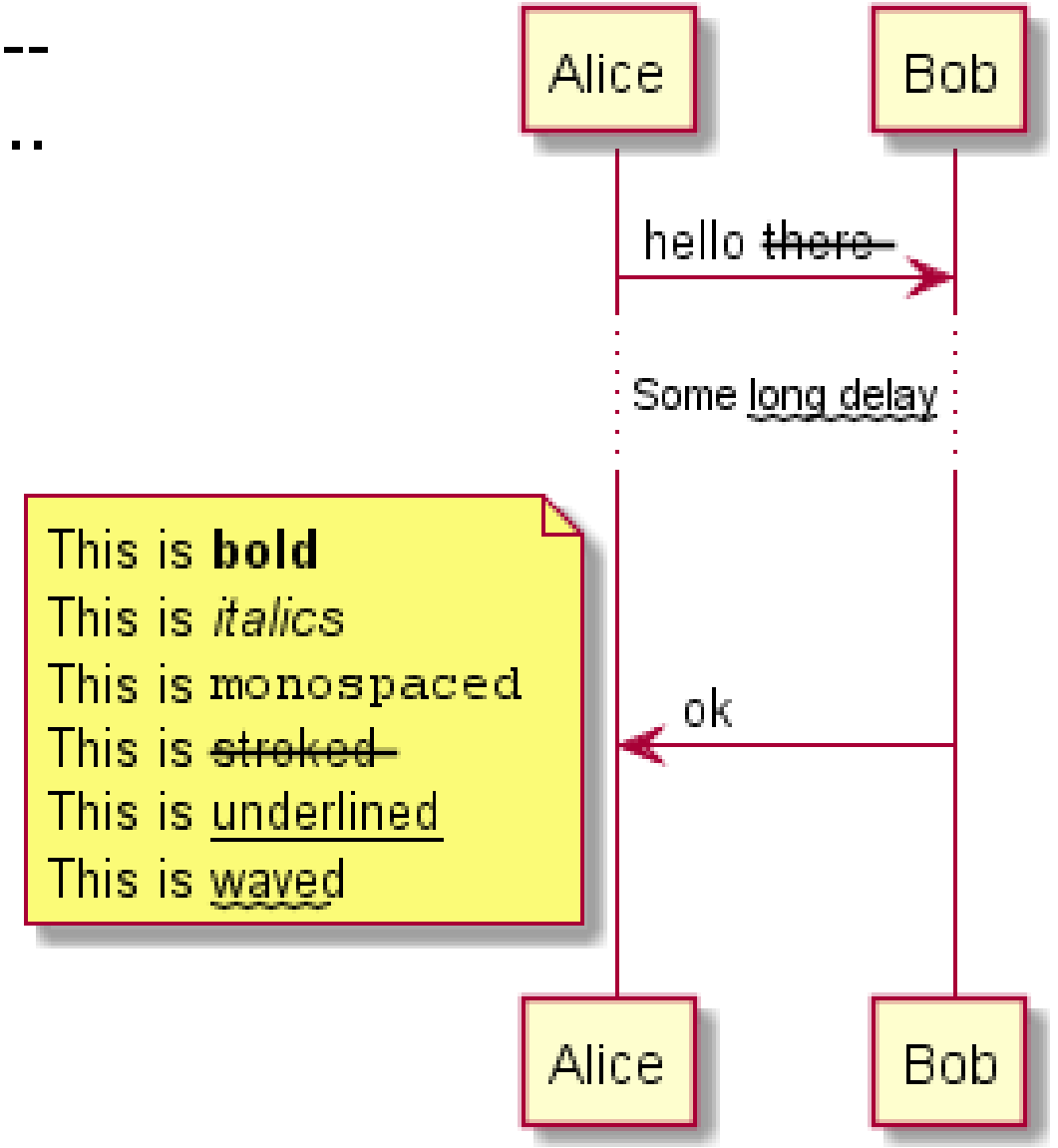
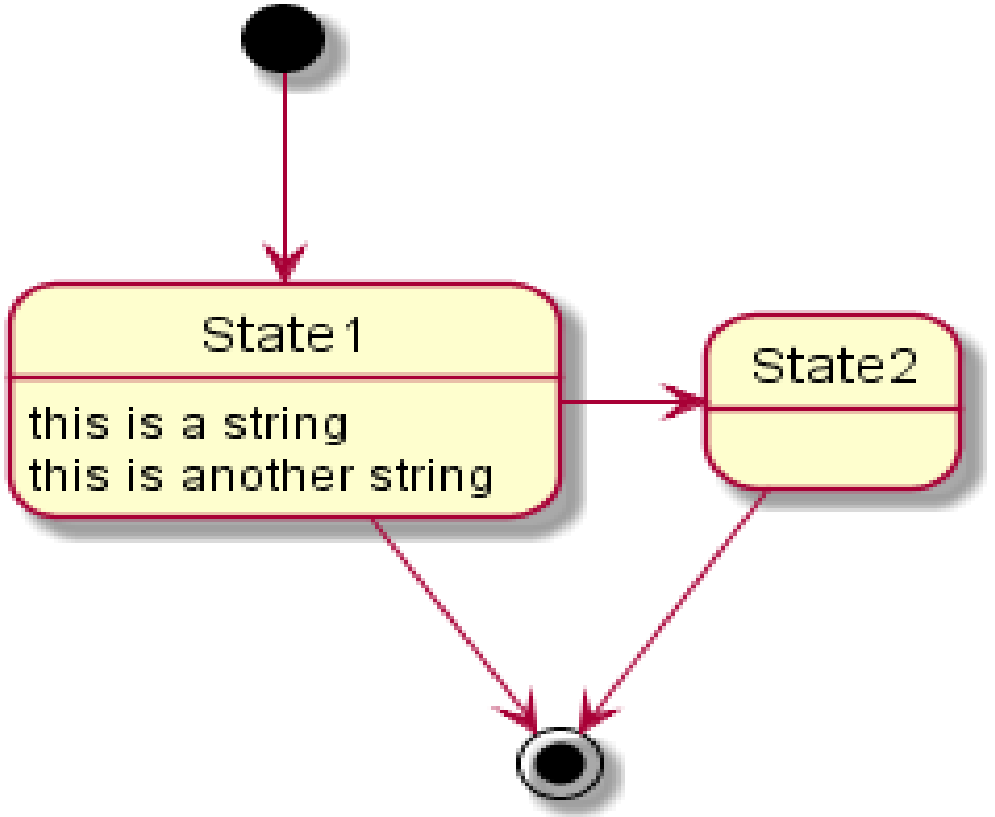


Diagramme d'état

Plant UML

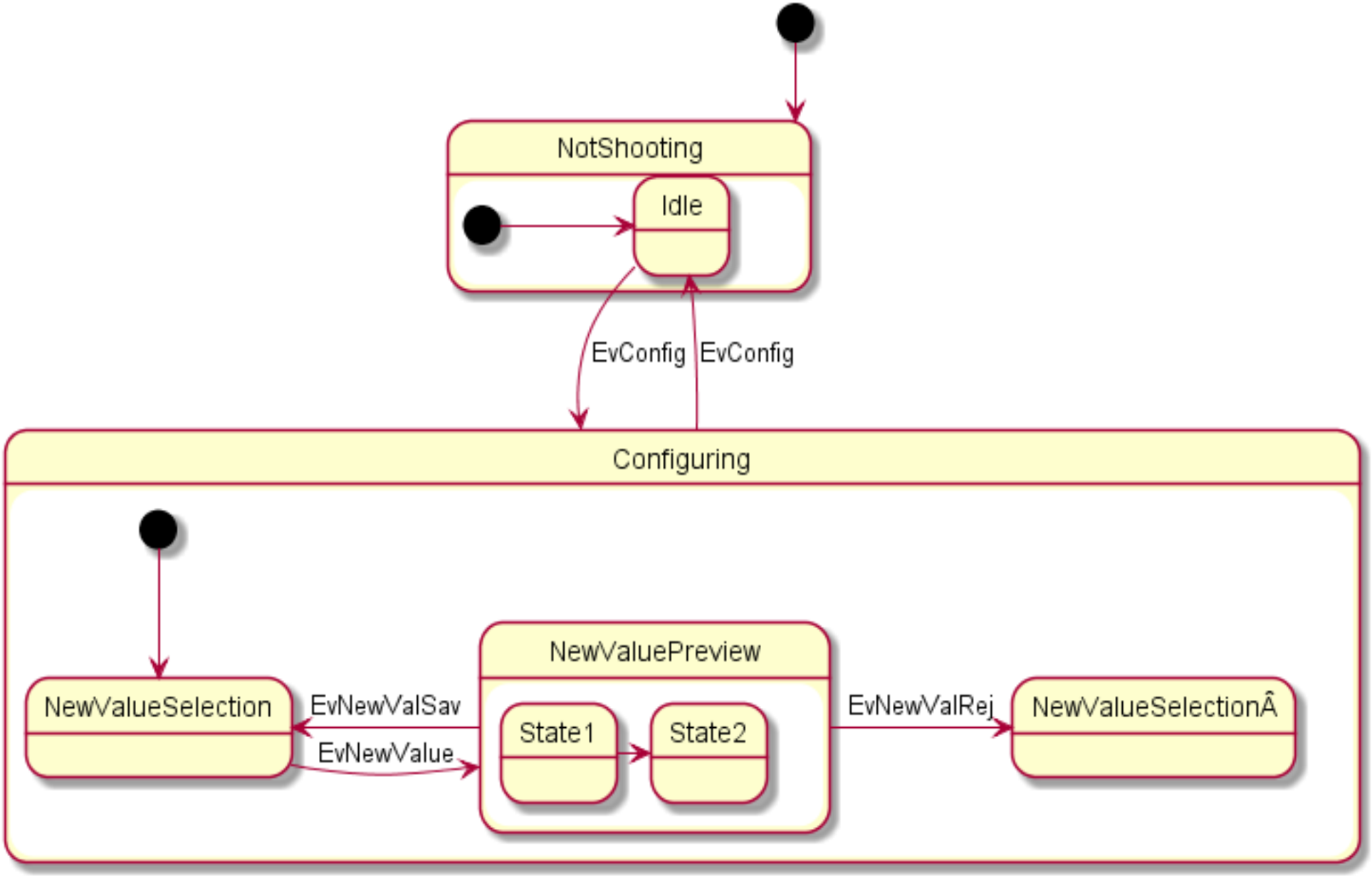
```
[*] --> State1
State1 --> [*]
State1 : this is a string
State1 : this is another string
State1 -> State2
State2 --> [*]
```



Plant UML

```
*] -> NotShooting
state NotShooting {
  [*] -> Idle
  Idle -> Configuring : EvConfig
  Configuring -> Idle : EvConfig
}
state Configuring {
  [*] --> NewValueSelection
  NewValueSelection -> NewValuePreview : EvNewValue
  NewValuePreview -> NewValueSelection : EvNewValRej
  NewValuePreview -> NewValueSelection : EvNewValSav
  state NewValuePreview {
    State1 -> State2
  }
}
```

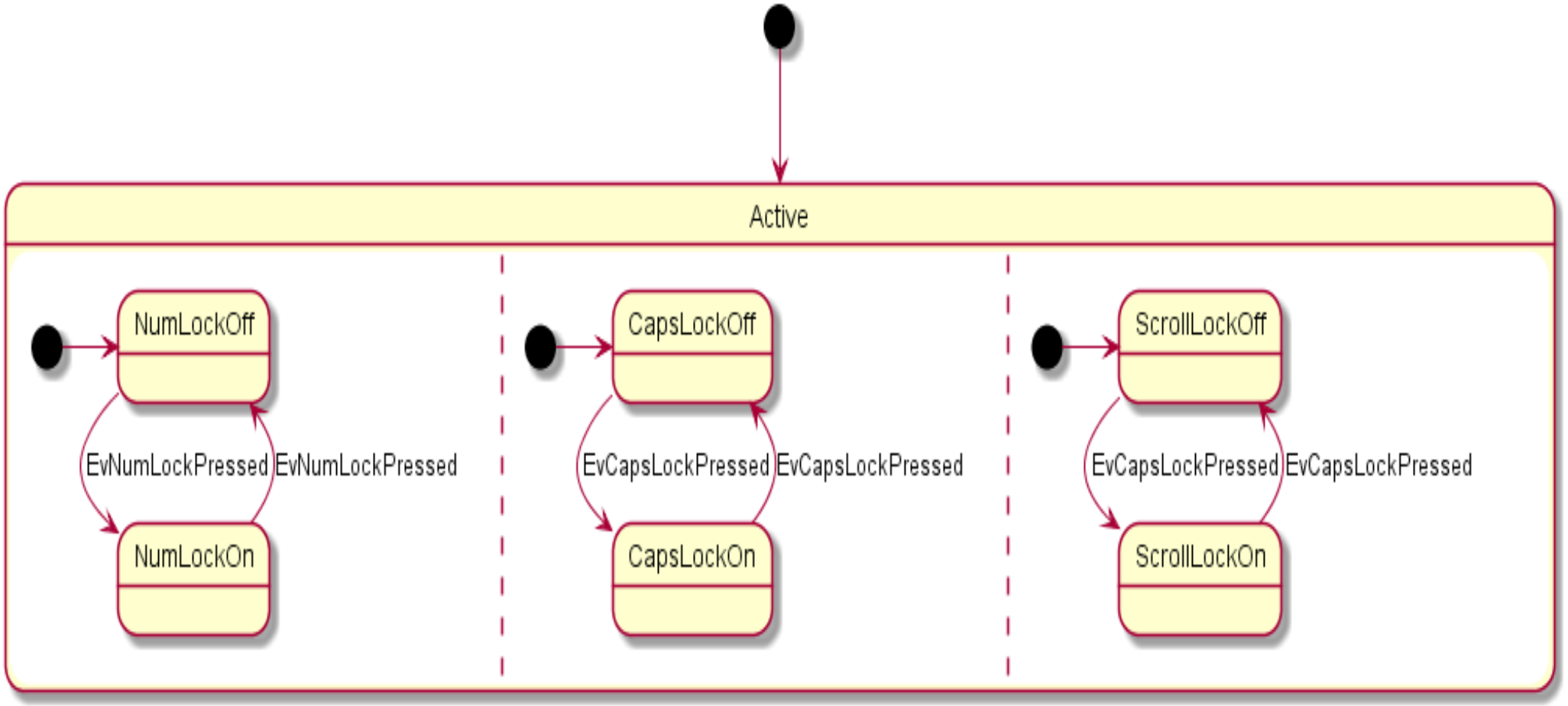
Plant UML


NotShooting
Idle
EvConfig
EvConfig
Configuring
NewValueSelection
EvNewValSav
EvNewValue
NewValuePreview
State1
State2
EvNewValRej
NewValueSelectionÂ

Plant UML

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

Plant UML

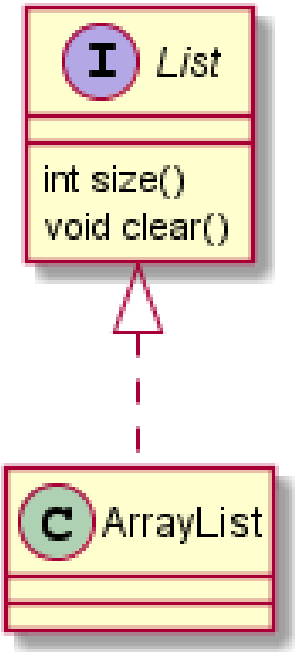

Active
NumLockOff
EvNumLockPressed
EvNumLockPressed
NumLockOn
CapsLockOff
EvCapsLockPressed
EvCapsLockPressed
CapsLockOn
ScrollLockOff
EvCapsLockPressed
EvCapsLockPressed
ScrollLockOn

Préprocesseur

Plant UML

Inclusion de fichier

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



File List.iuml:

```
interface List
List : int size()
List : void clear()
```

Rem : URL → !includeurl http://someurl.com/mypath!0

Plant UML

Utilisation de constantes

```
@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
```

Complément

Plant UML

Facteur d'échelle

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

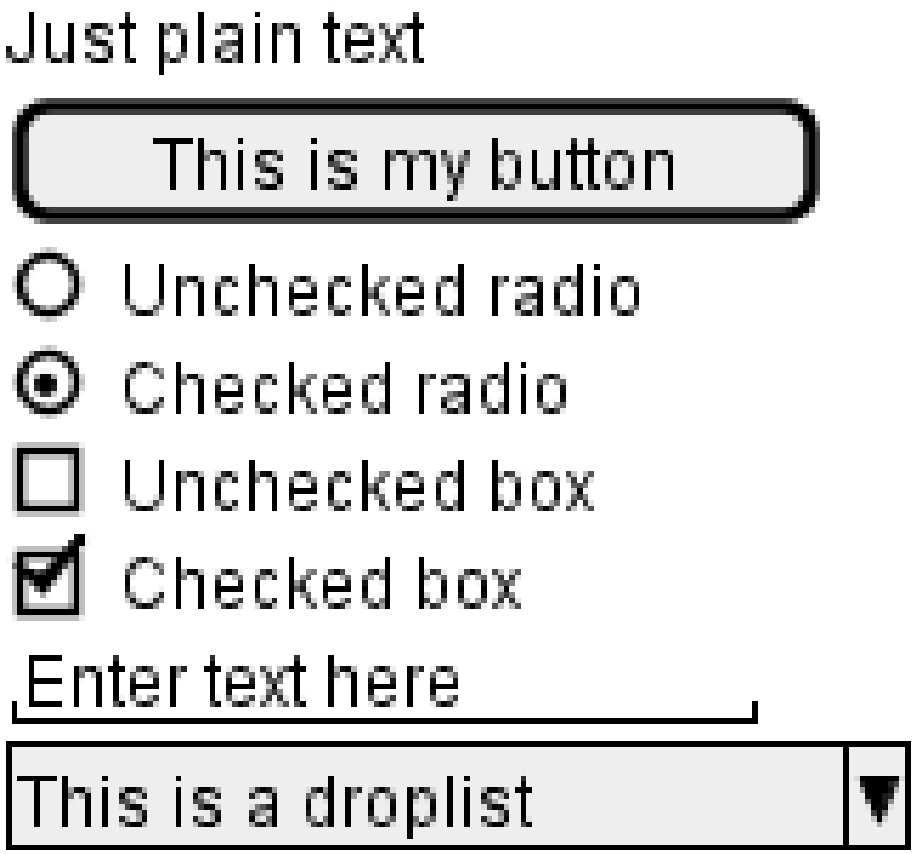
Plant UML

```
object Liste {
* Premier item
* Second item
** Sous item
}

legend
# item a
# item b
## item b.1
## item b.2
# item suivant
end legend
```

1. item a
2. item b
    1. item b.1
    2. item b.2
3. item suivant

Plant UML

```
@startsalt
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^
}
@endsalt
```



Plant UML

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

Plant UML

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

Plant UML

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

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

Plant UML

```
legend
  wifi <&wifi>
  warning <&warning>
end legend
```

Plant UML

Références :

http://plantuml.com	*Site officiel*
http://fr.plantuml.com	*Version française*
www.graphviz.org	*Dot*