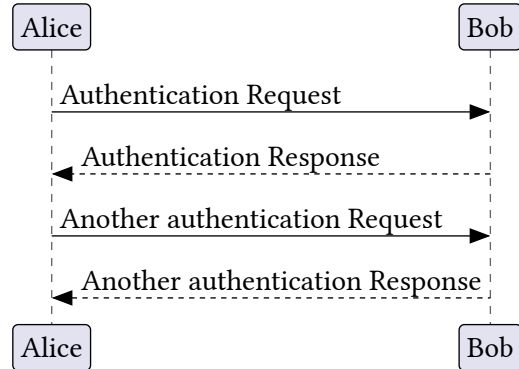


## Basic Examples

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

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

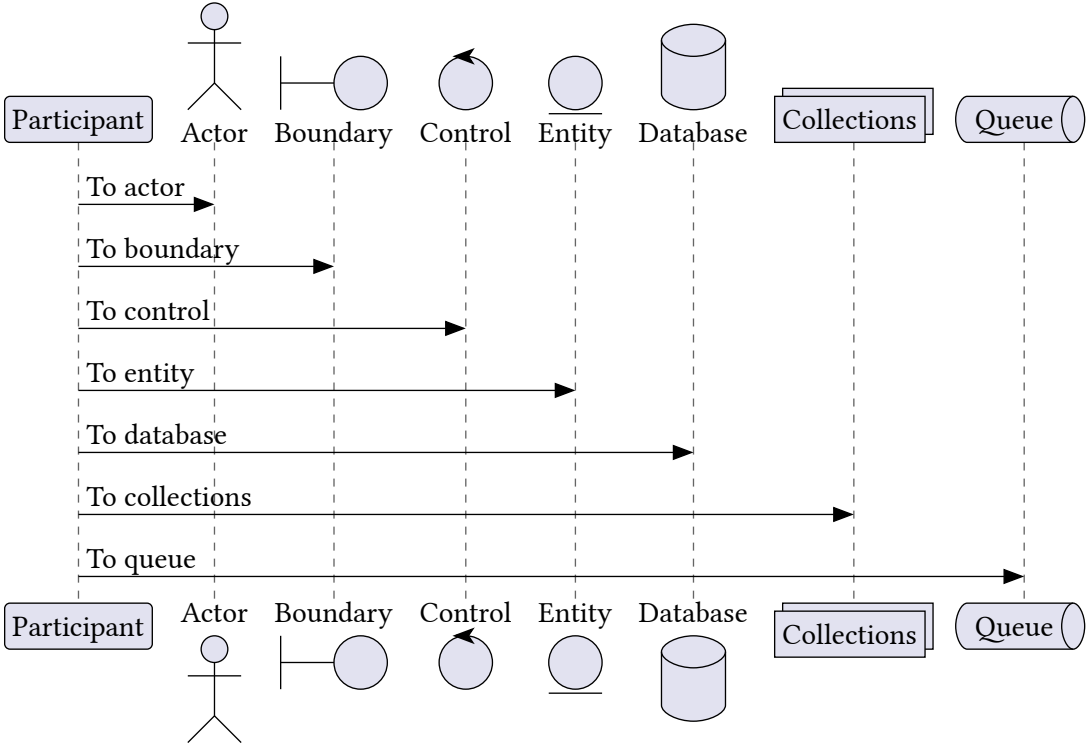


# Declaring participant

```

@startuml
participant Participant as Foo
actor Actor as Foo1
boundary Boundary as Foo2
control Control as Foo3
entity Entity as Foo4
database Database as Foo5
collections Collections as Foo6
queue Queue as Foo7
Foo -> Foo1 : To actor
Foo -> Foo2 : To boundary
Foo -> Foo3 : To control
Foo -> Foo4 : To entity
Foo -> Foo5 : To database
Foo -> Foo6 : To collections
Foo -> Foo7 : To queue
@enduml

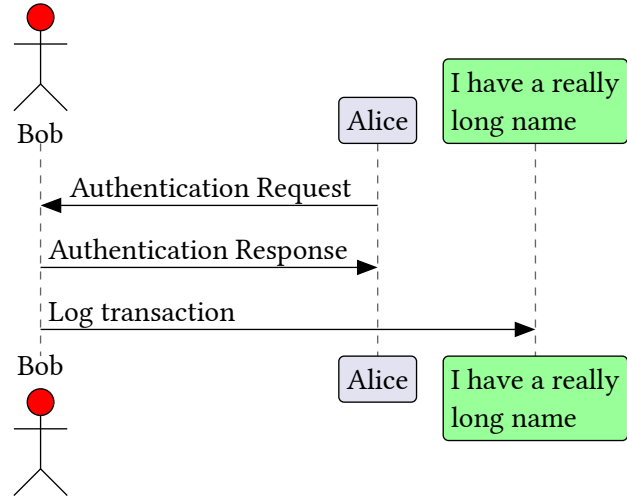
```



## Declaring participant (2)

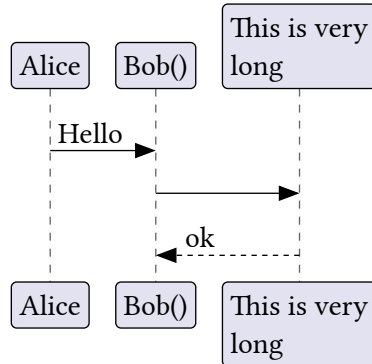
```
@startuml
actor Bob #red
' The only difference between actor
'and participant is the drawing
participant Alice
participant "I have a really\nlong name" as L #99FF99
/' You can also declare:
  participant L as "I have a really\nlong name" #99FF99
  '/

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



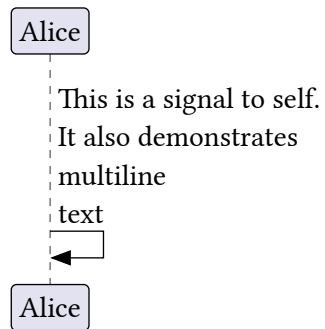
## Use non-letters in participants

```
@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
```



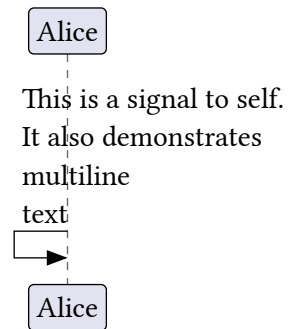
## Message to Self

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

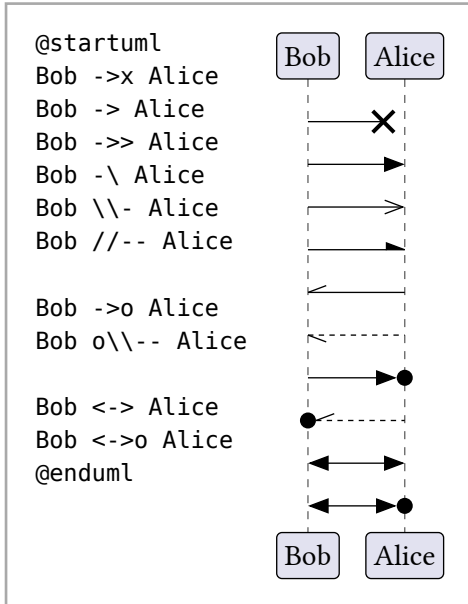


## Message to Self (2)

```
@startuml
Alice <- Alice: This is a signal to self.
It also demonstrates
multiline
text
@enduml
```



## Change arrow style



# Grouping message

```

@startuml
Alice -> Bob: Authentication Request

alt successful case

    Bob -> Alice: Authentication Accepted

else some kind of failure

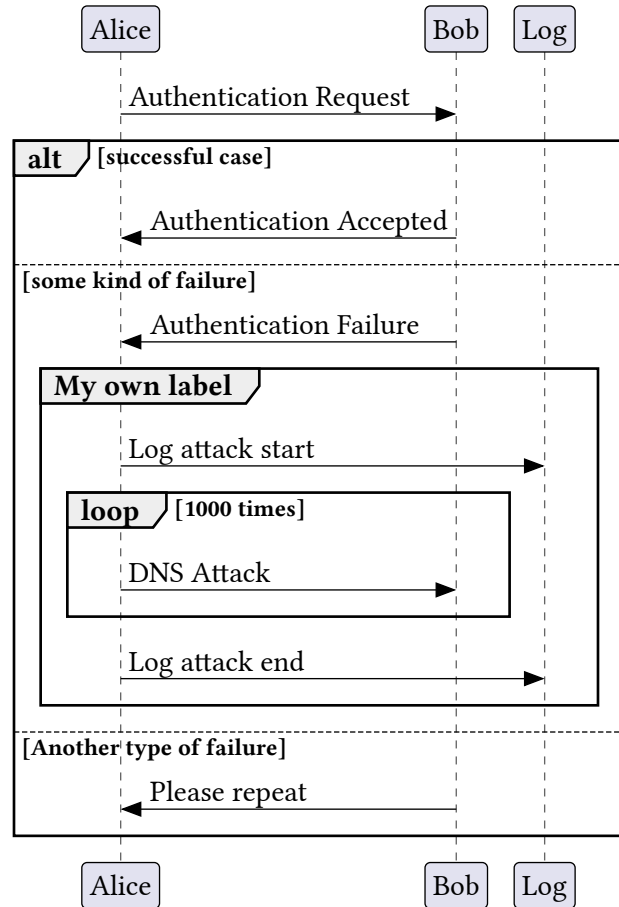
    Bob -> Alice: Authentication Failure
    group My own label
    Alice -> Log : Log attack start
    loop 1000 times
        Alice -> Bob: DNS Attack
    end
    Alice -> Log : Log attack end
end

else Another type of failure

    Bob -> Alice: Please repeat

end
@enduml

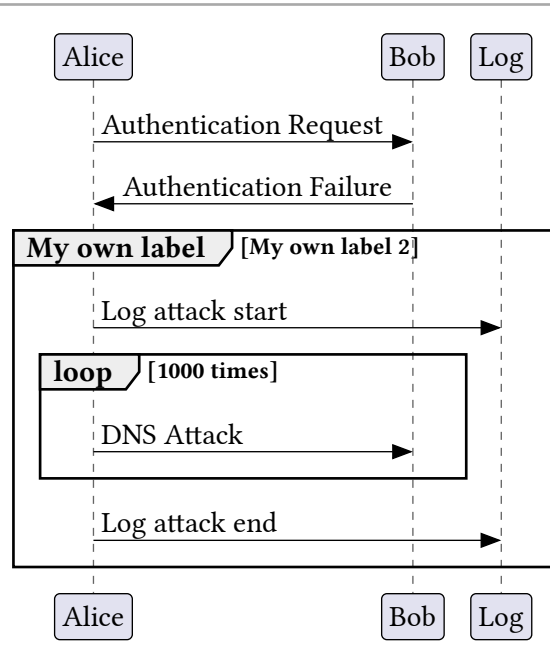
```





## Secondary group label

```
@startuml
Alice -> Bob: Authentication Request
Bob -> Alice: Authentication Failure
group My own label [My own label 2]
  Alice -> Log : Log attack start
  loop 1000 times
    Alice -> Bob: DNS Attack
  end
  Alice -> Log : Log attack end
end
@enduml
```

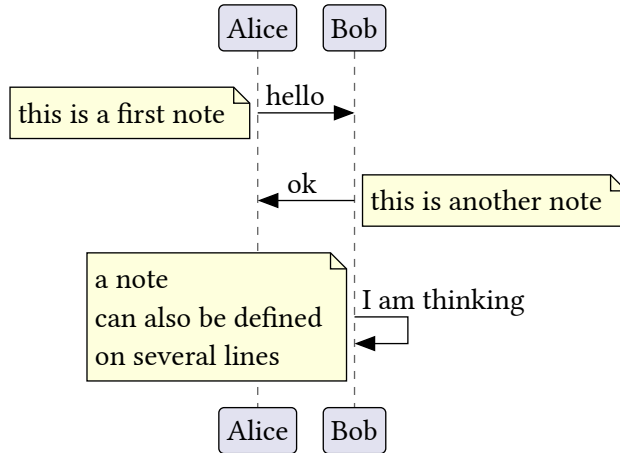


## Notes on messages

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

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

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



## Some other notes

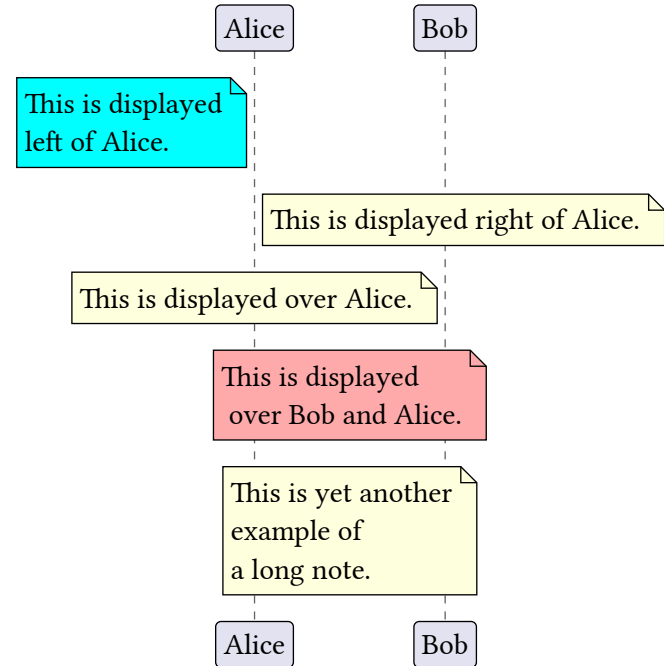
```
@startuml
participant Alice
participant Bob
note left of Alice #aqua
This is displayed
left of Alice.
end note

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

note over Alice: This is displayed over Alice.

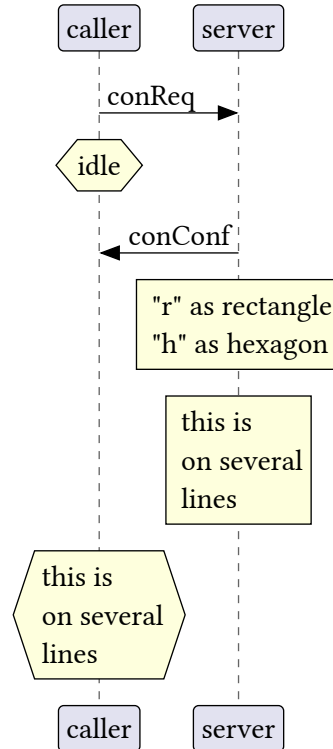
note over Alice, Bob #FFAAAA: This is displayed\n over Bob and Alice.

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



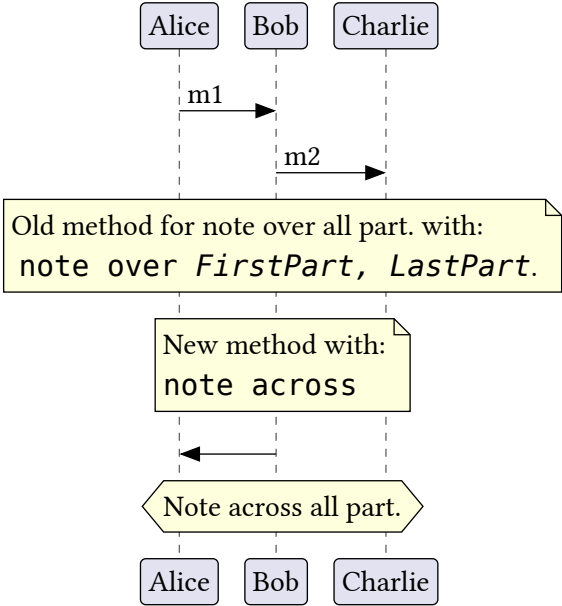
## Changing notes shape [hnote, rnote]

```
@startuml
caller -> server : conReq
hnote over caller : idle
caller <- server : conConf
rnote over server
  "r" as rectangle
  "h" as hexagon
endnote
rnote over server
  this is
  on several
  lines
endnote
hnote over caller
  this is
  on several
  lines
endnote
@enduml
```



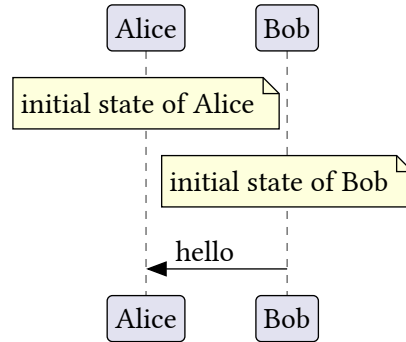
# Note over all participants [across]

```
@startuml
Alice->Bob:m1
Bob->Charlie:m2
note over Alice, Charlie: Old method for note over all part. with:\n ""note over //FirstPart, LastPart//"".
note across: New method with:\n""note across""
Bob->Alice
hnote across:Note across all part.
@enduml
```



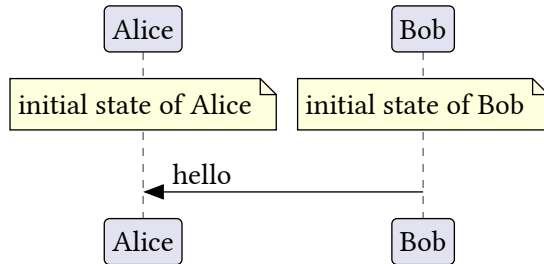
## Several notes aligned at the same level [/]

```
@startuml
note over Alice : initial state of Alice
note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



## Several notes aligned at the same level [/] (2)

```
@startuml
note over Alice : initial state of Alice
/ note over Bob : initial state of Bob
Bob -> Alice : hello
@enduml
```



## Divider or separator

```
@startuml
```

```
== Initialization ==
```

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

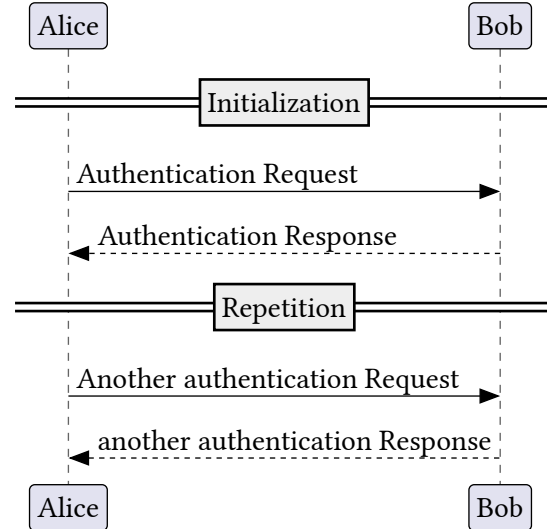
```
Bob --> Alice: Authentication Response
```

```
== Repetition ==
```

```
Alice -> Bob: Another authentication Request
```

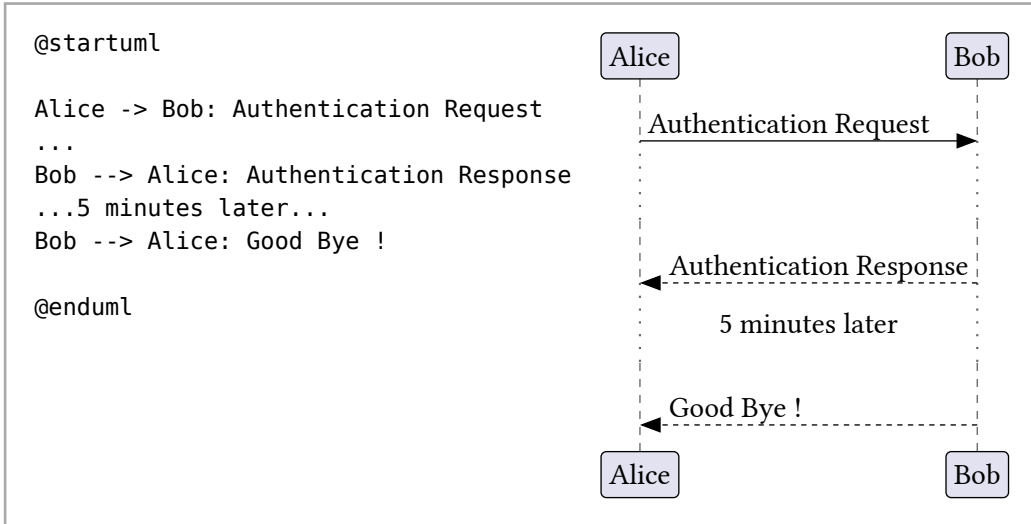
```
Alice <-- Bob: another authentication Response
```

```
@enduml
```

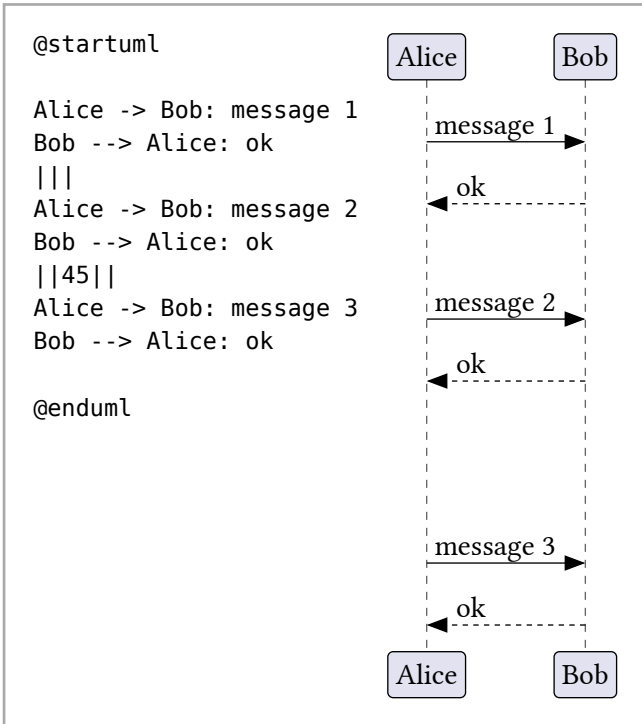




# Delay



# Space



## Lifeline Activation and Destruction

```
@startuml
participant User
```

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

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

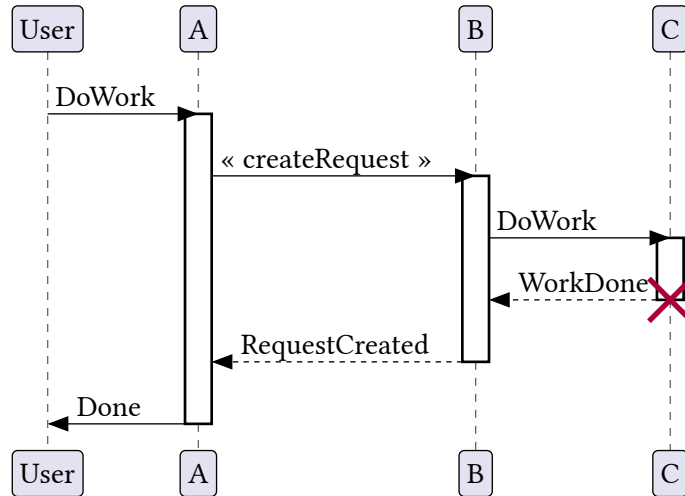
```
B -> C: DoWork
activate C
```

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

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

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

```
@enduml
```



## Lifeline Activation and Destruction (2)

```
@startuml
participant User

User -> A: DoWork
activate A #FFBBBB

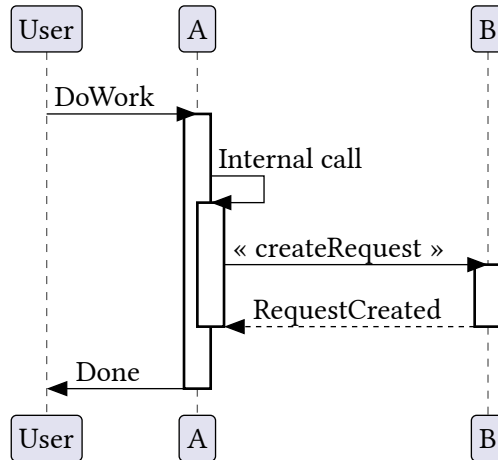
A -> A: Internal call
activate A #DarkSalmon

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

B --> A: RequestCreated
deactivate B
deactivate A

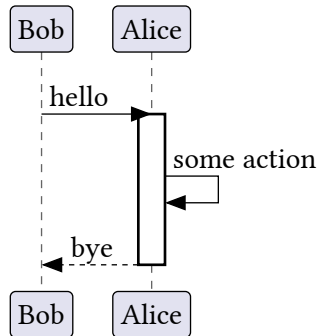
A -> User: Done
deactivate A

@enduml
```

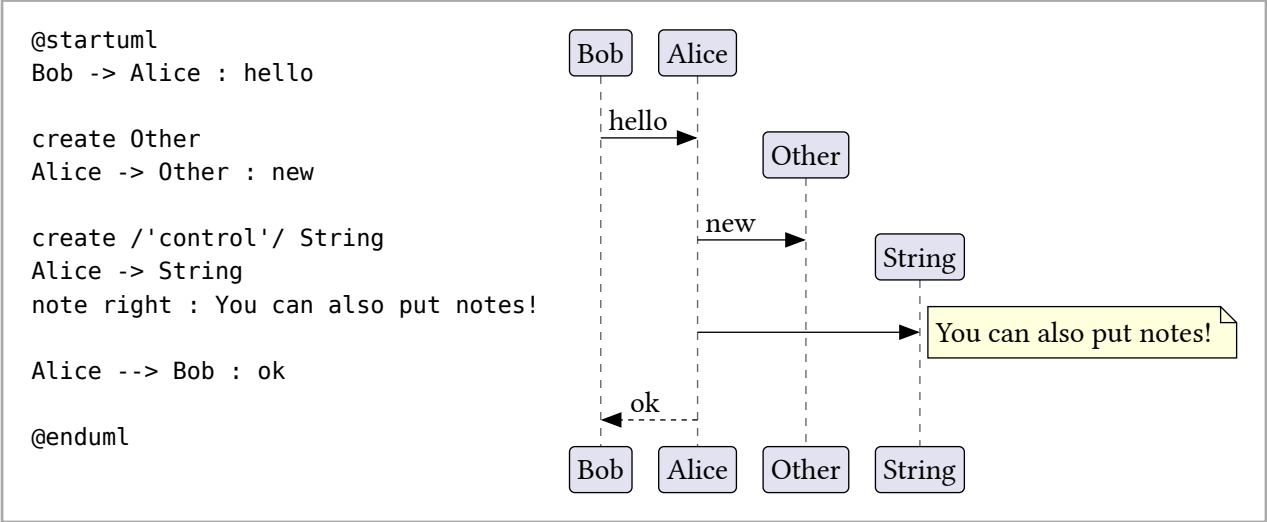


## Return

```
@startuml
Bob -> Alice : hello
activate Alice
Alice -> Alice : some action
return bye
@enduml
```

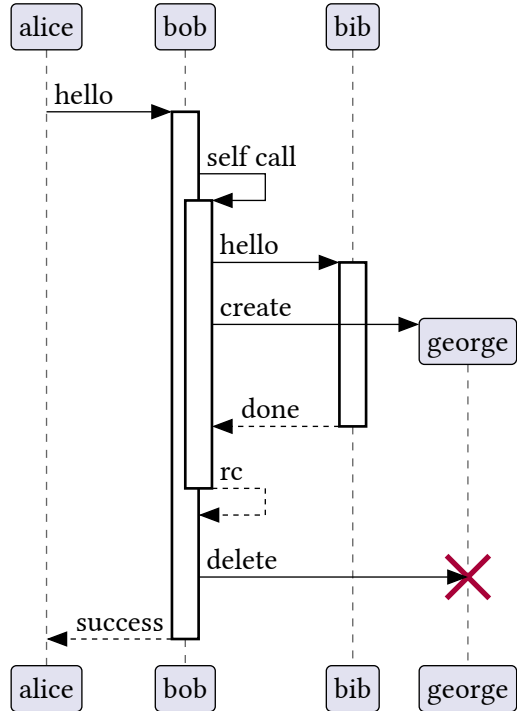


# Participant creation



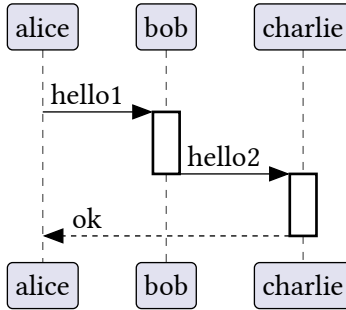
## Shortcut syntax for activation, deactivation, creation

```
@startuml
alice -> bob ++ : hello
bob -> bob ++ : self call
bob -> bib ++ '/' #005500'/' : hello
bob -> george ** : create
return done
return rc
bob -> george !! : delete
return success
@enduml
```



## Shortcut syntax for activation, deactivation, creation (2)

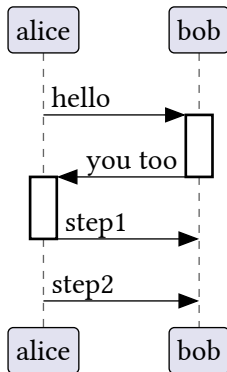
```
@startuml
alice -> bob ++ : hello1
bob -> charlie ---++ : hello2
charlie --> alice -- : ok
@enduml
```





## Shortcut syntax for activation, deactivation, creation (3)

```
@startuml
alice -> bob  ++ /'#gold'/: hello
bob  -> alice --++ /'#gold'/: you too
alice -> bob  --: step1
alice -> bob  : step2
@enduml
@enduml
```



## Incoming and outgoing messages

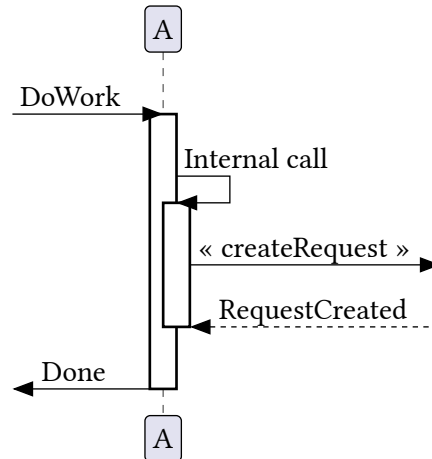
```
@startuml
[-> A: DoWork

activate A

A -> A: Internal call
activate A

A ->] : « createRequest »

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



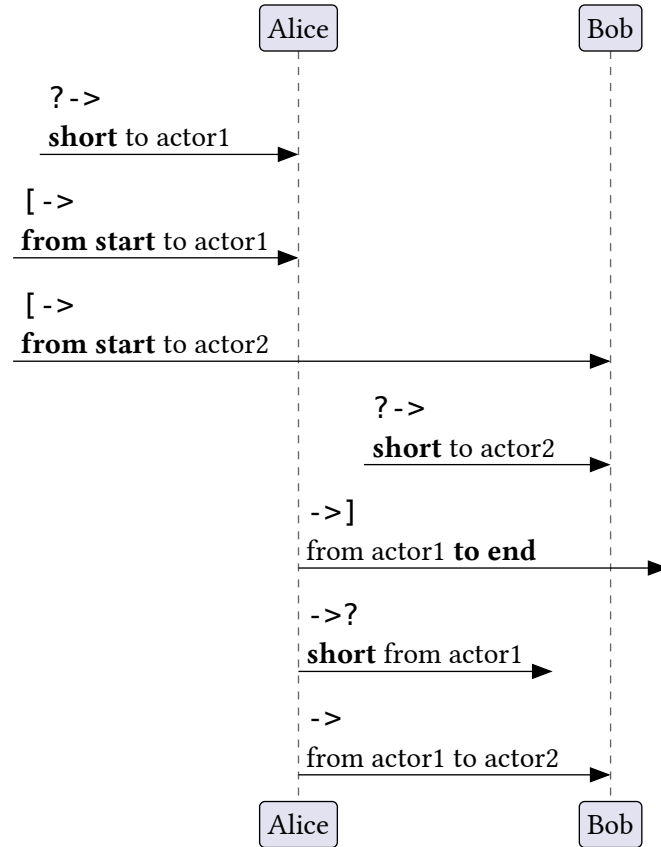


## Short arrows for incoming and outgoing messages

```

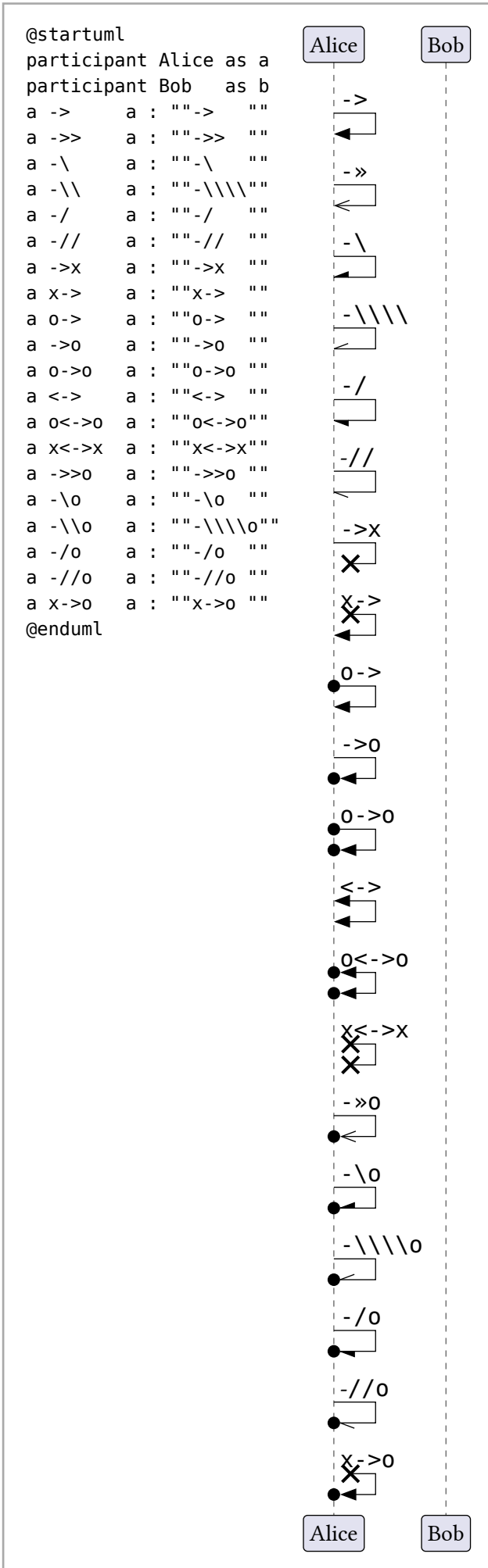
@startuml
?-> Alice : ""?->""\n**short** to actor1
[-> Alice : ""[->""\n**from start** to actor1
[-> Bob : ""[->""\n**from start** to actor2
?-> Bob : ""?->""\n**short** to actor2
Alice ->] : ""->]""\nfrom actor1 **to end**
Alice ->? : ""->?""\n**short** from actor1
Alice -> Bob : ""->"" \nfrom actor1 to actor2
@enduml

```

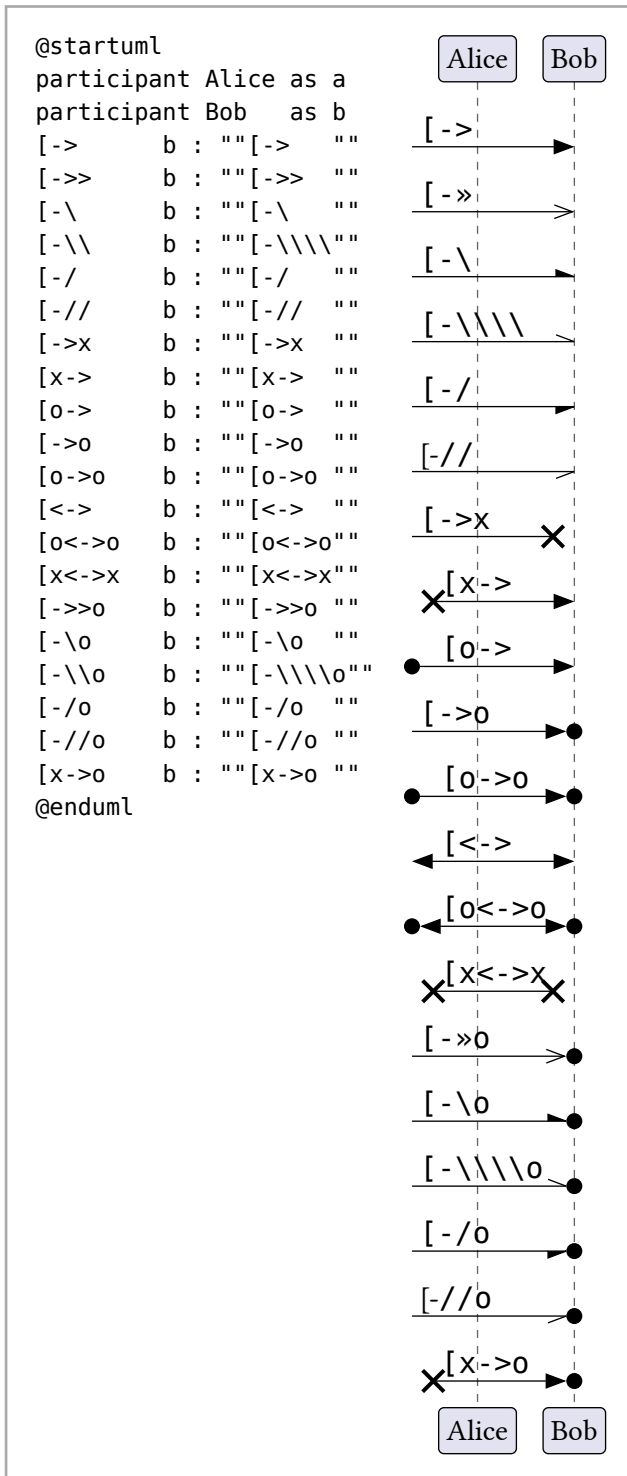




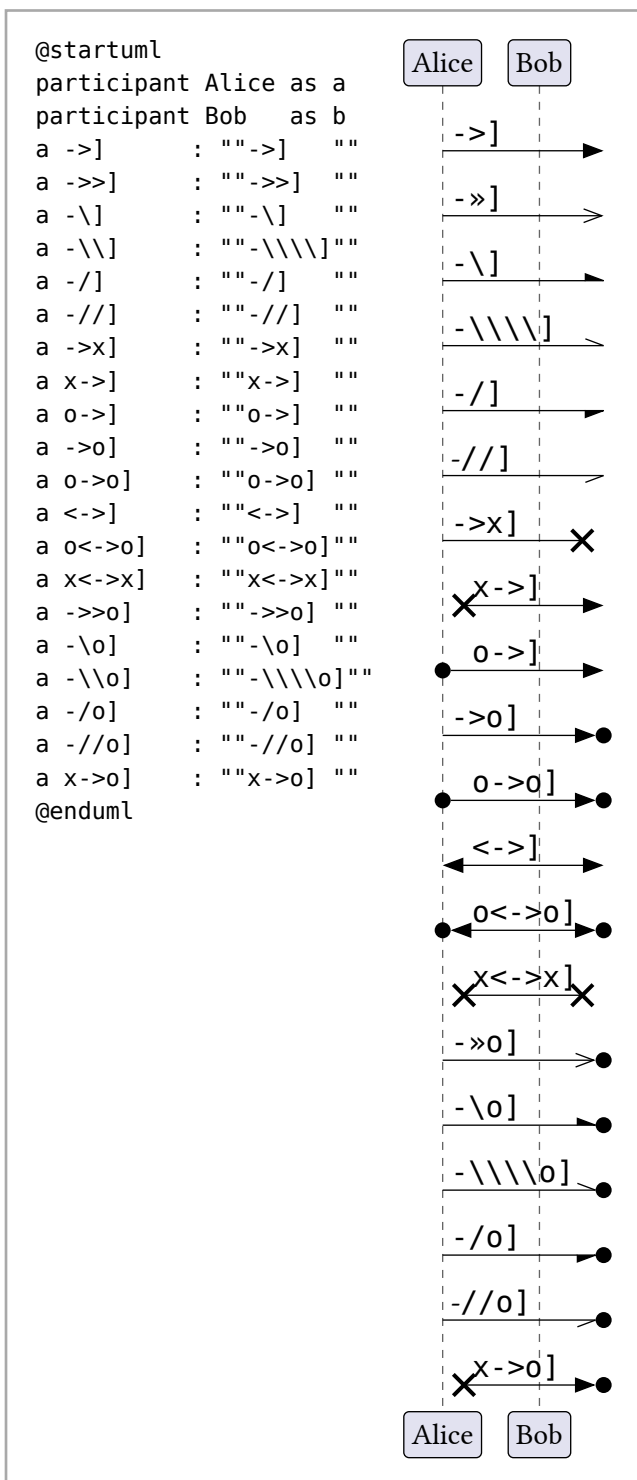
# Itself arrow



# Incoming messages (with ‘\’)

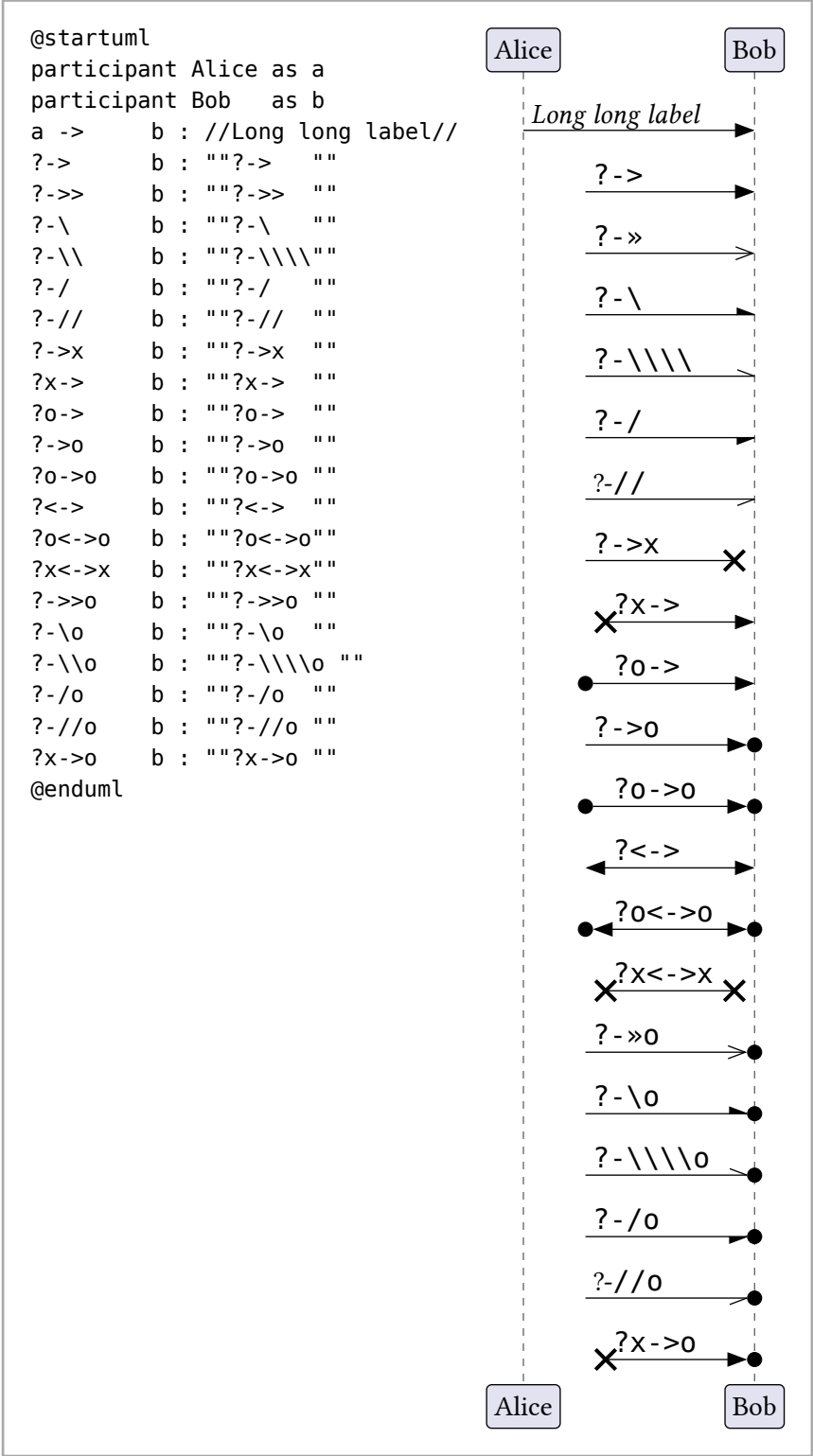


## Outgoing messages (with ‘]’)





# Short incoming (with '?')



# Short outgoing (with '?')

