UML & Markdown Summary

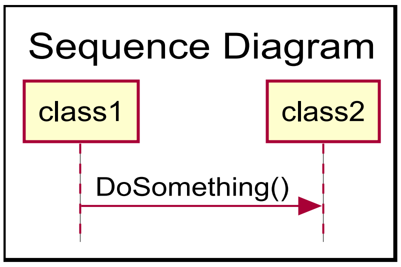
Table of Contents

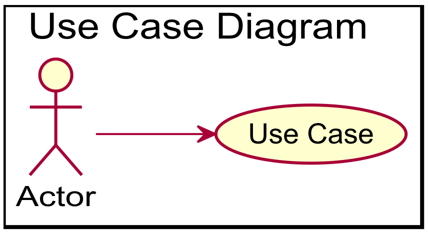
Links & Sources

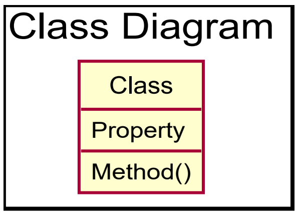
- UML Diagrams: Wikipedia
- UML Diagrams: uml-diagrams.org
- UML Basic Notations: tutorialspoint.com
- UML Quick Reference: nomagic.com
- UML Notation Overview (German): oose.de
- Markdown Syntax: commonmark.org
- plantUML.com
- plantUML Reference Guide: deepu.js.org
- Hyperlinks in plantUML)

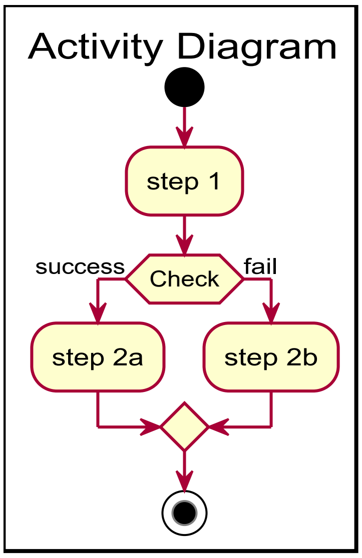
plantUML Introduction

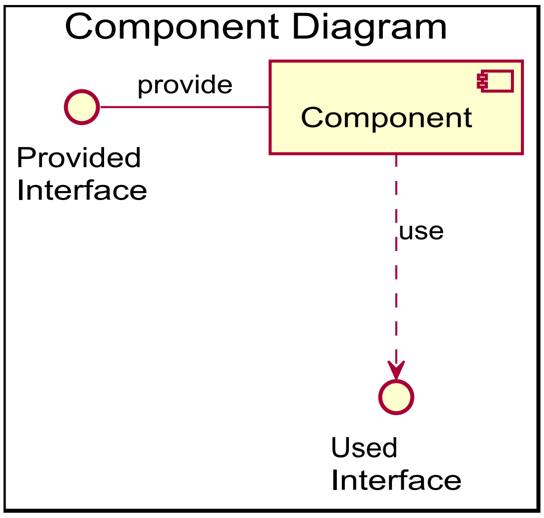
Available Diagrams

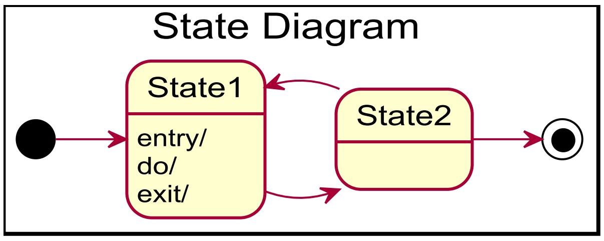


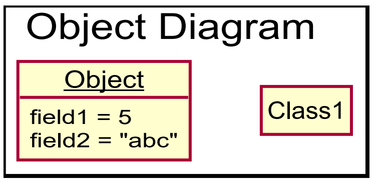


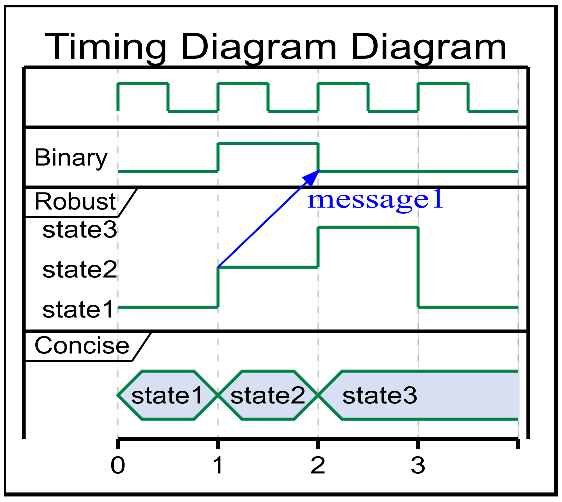












UML Diagrams:

- Sequence diagram
- Usecase diagram
- Class diagram
- Activity diagram
- Component diagram
- State diagram
- Object diagram
- Deployment diagram
- Timing diagram

Non-UML Diagrams:

- Wireframe GUI
- Archimate diagram
- SDL - Specification and Description Language
- Ditaa diagram
- Gantt diagram
- MindMap
- Work Breakdown Structure diagram
- Entity Relationship diagram

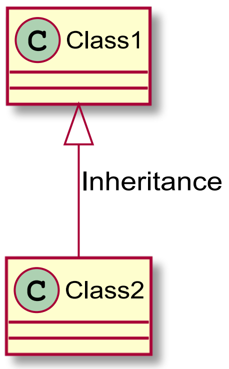
Usage of plantUML within a Markdown document

Within a markdown document a plantUML section is marked with the tags `@startuml` and `@enduml`.

Example:

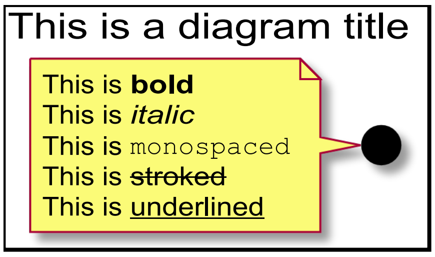
```
@startuml
Class1 <|-- Class2 : Inheritance
@enduml
```

Result:

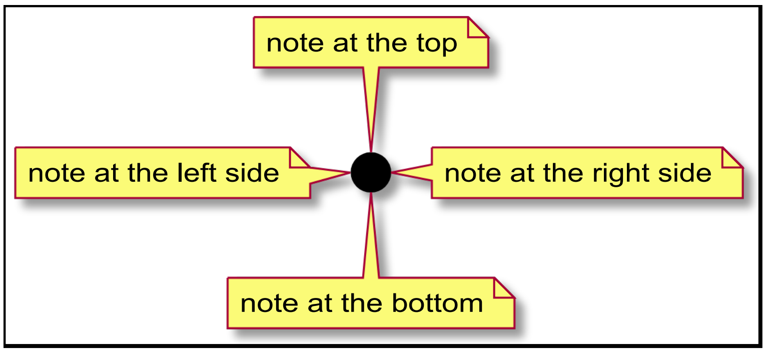


Basic Syntax

- See also https://plantuml.com/de/creole



```
title This is a diagram title

' This is a (one line) comment. This line will be ignored by plantUML.

/' This is ...
   ... a multiline comment.
'/

note left
        This is **bold**
        This is //italic//
        This is ""monospaced""
        This is --stroked--
        This is __underlined__
end note
```



```
note left
    note at the left side
end note

note right
    note at the right side
end note

note bottom
    note at the bottom
end note
```

```
note top
    note at the top
end note
```

UML Diagrams

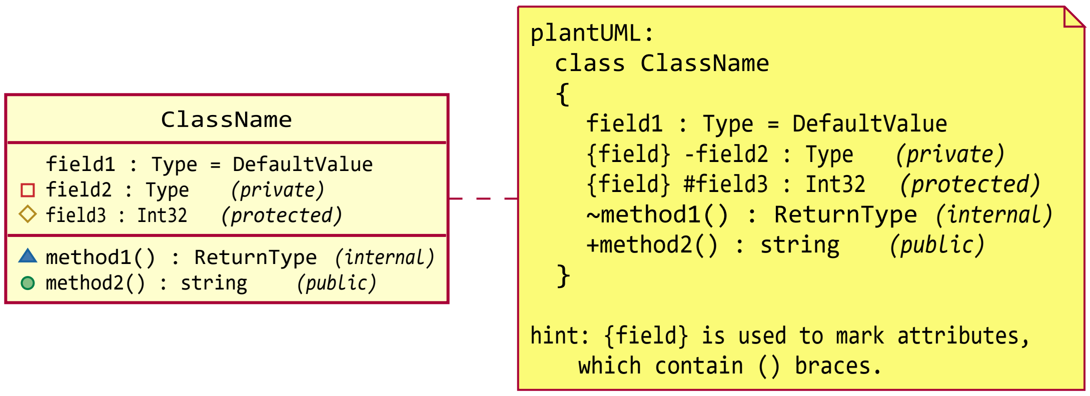
- The standard plantUML style does not perfectly comply with the UML standard. To be more compliant use `skinparam style strictuml`.
- To get a non-colored diagram `skinparam monochrome true` can be used.

Class Diagram

See also:

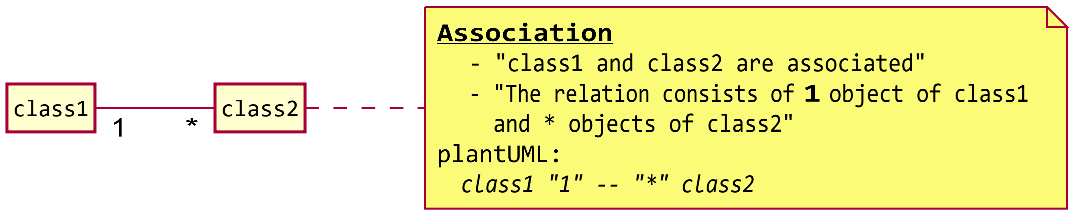
- Class Diagrams Overview: uml-diagrams.org
- Class Diagram: plantuml.com

Class Notation

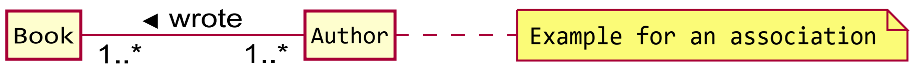

ClassName
field1 : Type = DefaultValue
field2 : Type (private)
field3 : Int32 (protected)
method1() : ReturnType (internal)
method2() : string (public)

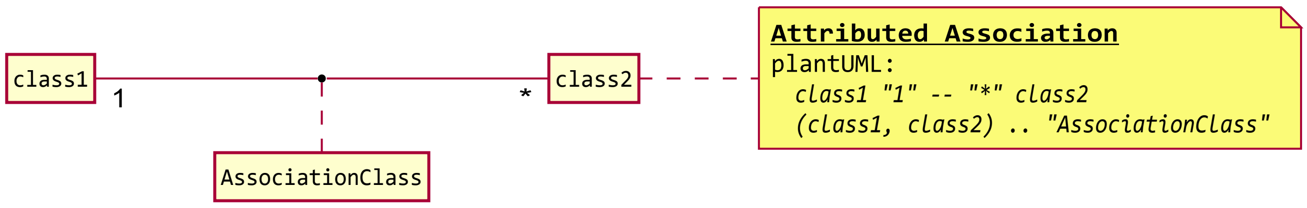
```
plantUML:
  class ClassName
  {
     field1 : Type = DefaultValue
     {field} -field2 : Type    (private)
     {field} #field3 : Int32   (protected)
     ~method1() : ReturnType (internal)
     +method2() : string    (public)
  }

hint: {field} is used to mark attributes,
    which contain () braces.
```

Association



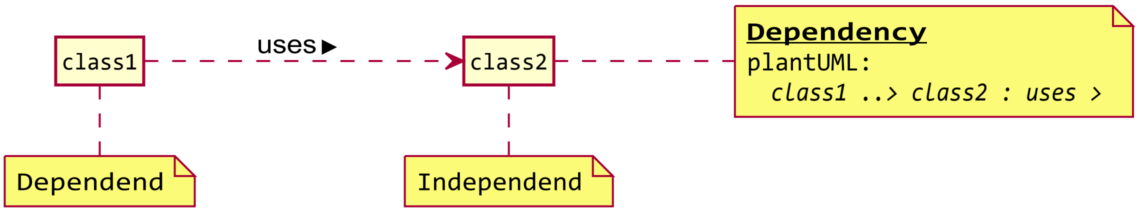
```
Association
   - "class1 and class2 are associated"
   - "The relation consists of 1 object of class1
     and * objects of class2"
plantUML:
  class1 "1" -- "*" class2
```



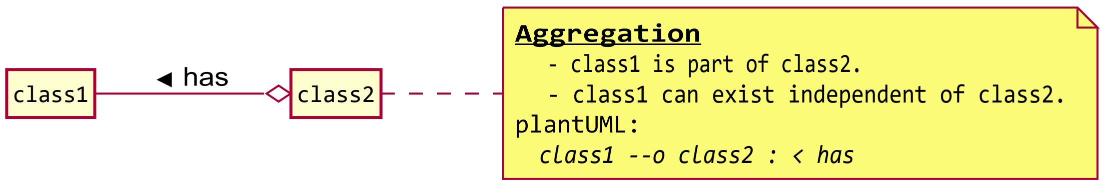


```
Attributed Association
plantUML:
  class1 "1" -- "*" class2
  (class1, class2) .. "AssociationClass"
```

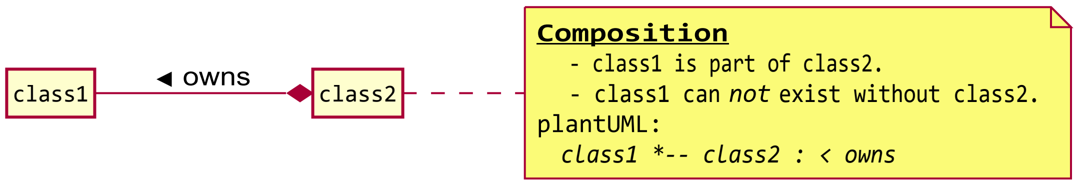
Dependency



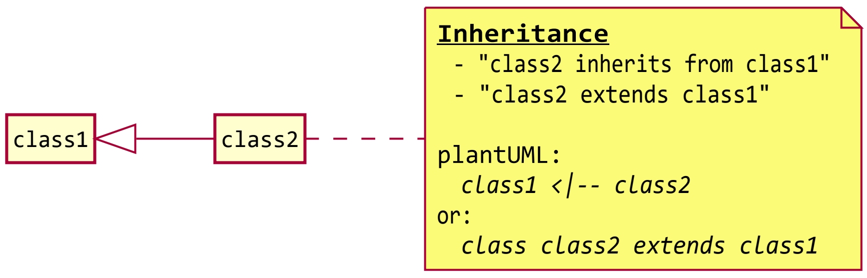
Aggregation


class1
◂ has
class2
Aggregation
- class1 is part of class2.
- class1 can exist independent of class2.
plantUML:
class1 --o class2 : < has

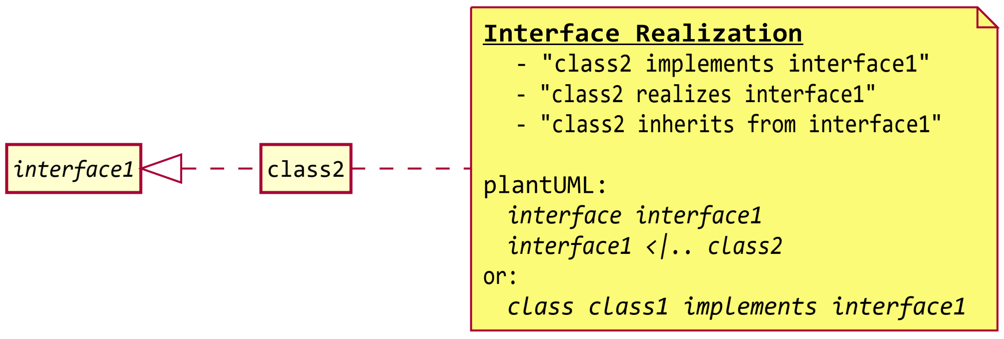
Composition


class1
◂ owns
class2
Composition
- class1 is part of class2.
- class1 can not exist without class2.
plantUML:
class1 *-- class2 : < owns

Inheritance

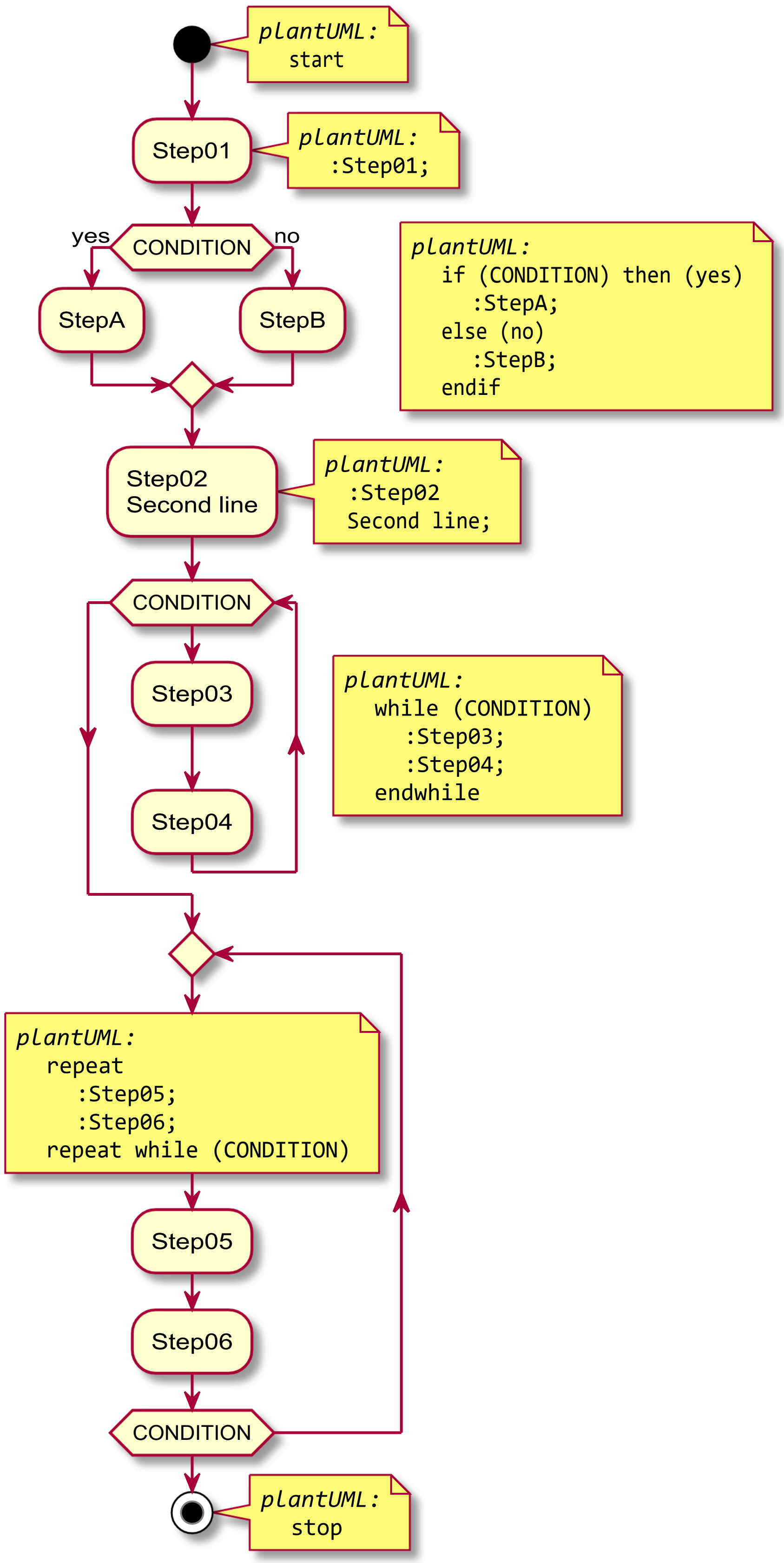

class1
class2
Inheritance
- "class2 inherits from class1"
- "class2 extends class1"
plantUML:
class1 <|-- class2
or:
class class2 extends class1

Interface Realization (Interface Inheritance)


interface1
class2
Interface Realization
- "class2 implements interface1"
- "class2 realizes interface1"
- "class2 inherits from interface1"
plantUML:
interface interface1
interface1 <|.. class2
or:
class class1 implements interface1

Activity Diagram (Flow Chart)

plantUML: NEW Activity Diagram Syntax



```
plantUML:
  start
```

```
plantUML:
  :Step01;
```

```
plantUML:
   if (CONDITION) then (yes)
      :StepA;
   else (no)
      :StepB;
   endif
```

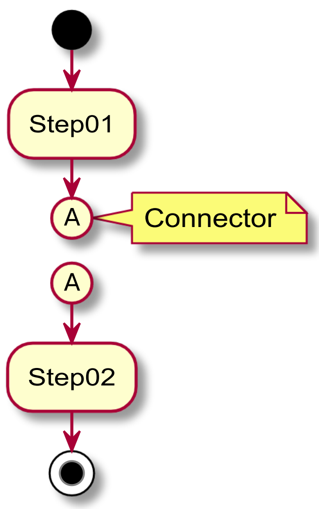
```
plantUML:
  :Step02
  Second line;
```

```
plantUML:
   while (CONDITION)
      :Step03;
      :Step04;
   endwhile
```

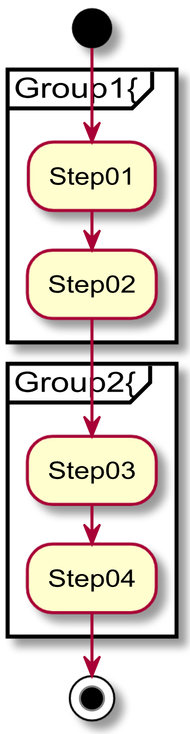
```
plantUML:
   repeat
      :Step05;
      :Step06;
   repeat while (CONDITION)
```

```
plantUML:
  stop
```

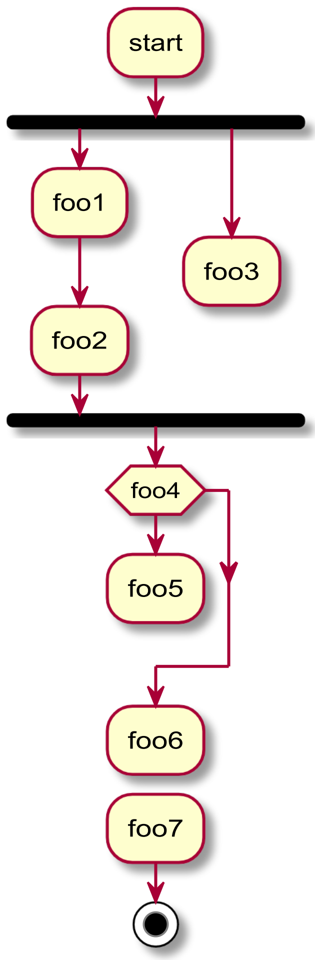
Connector & Detach



Grouping (partitions)



Detach



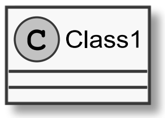
plantUML formatting / styles

Border around the diagrams

```
@plantuml
    skinparam DiagramBorderColor black
    skinparam DiagramBorderThickness 2
    ...
```

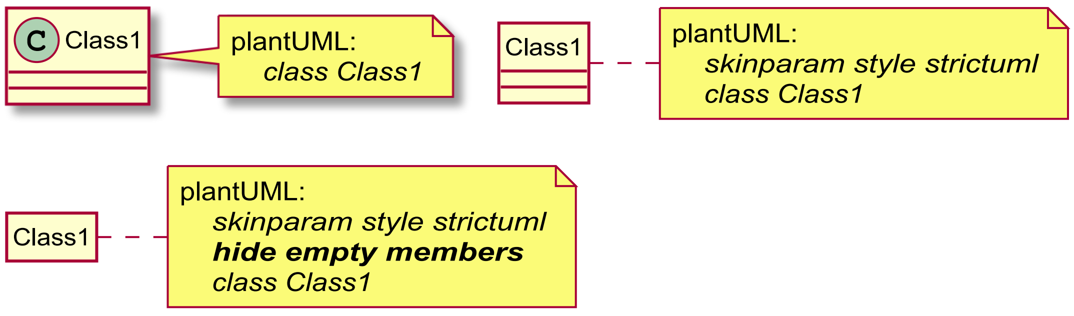
Black White diagrams

```
@plantuml
    skinparam monochrome true
    ...
```



UML compliant appearance

The default style of plantUML does not completely match the official UML definition. To change the style to comply as far as possible with the official UML defintion, set the `strictuml` style.



together

To influence how multiple classes are arranged in large diagrams, the ´Together{}´ keyword can be used:

```
Together{
    class A
    class B
}
```

Non-UML Diagrams

MindMap Diagram

Basic MindMap syntax

@startmindmap

- Root node '* A second root node is not allowed ** Node *** Subsubnode **_ Node without box

left side

** left side node ** second

@endmindmap

@startmindmap

- Root Node ++ Alternative notation with ++ -- '--' Chooses the left side ---_ Subnode ---_ Subnode -- ~~Strike through~~ @endmindmap

Headers etc.

@startmindmap

header My Header title My Title

- one ** two ** three

legend right My Legend end legend

caption My Caption center footer My Footer @endmindmap

Markdown compliant

@startmindmap

- **Markdown syntax**
  - indented by **TAB**
  - intention by space\n is currently not possible
    - one
    - two
  - three @endmindmap

Symbols

@startmindmap

- Symbols
  - <&flag> <&flag >
  - <&globe> <&globe >

  - <&graph> <&graph >
  - <&pulse> <&pulse >
  - <&people> <&people >
  - <&star> <&star >

@endmindmap