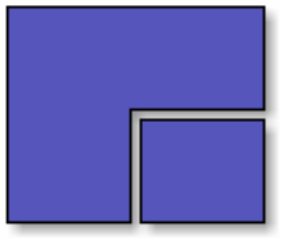
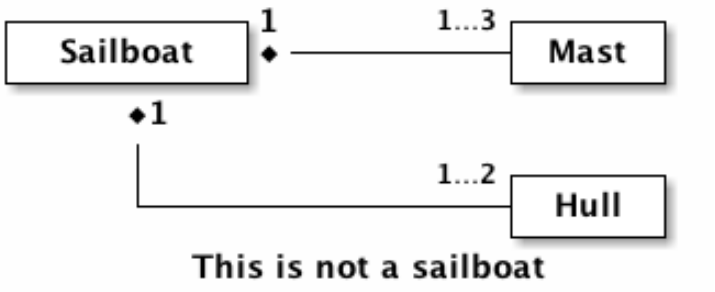
1. Ditaa

   When a drop of water joins the ocean it becomes the ocean.

```
+----------+
| cBLU     |
|          |
|    +----+
|    |cBLU|
|    |    |
+----+----+
```

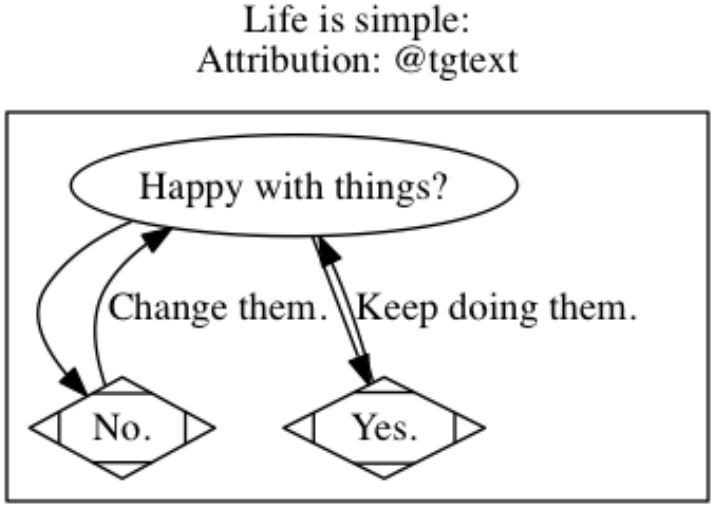


```
+-----------+1       1...3+------+
| Sailboat +-----------+ Mast |
+-----+----+            +------+
      1
      |
      |              1...2+------+
      +-----------------+ Hull |
                        +------+
      This is not a sailboat
```



2. Graphviz

```
digraph graphviz {
  subgraph cluster {
    ayh [label="Happy with things?", shape=ellipse];
    no [label="No.", shape=Mdiamond];
    yes [label="Yes.", shape=Mdiamond];
    ayh -> no;
    ayh -> yes;
    no -> ayh [label="Change them."];
    yes -> ayh [label="Keep doing them."];
  }
  labelloc="t";
  label="Life is simple:\nAttribution: @tgtext";
}
```

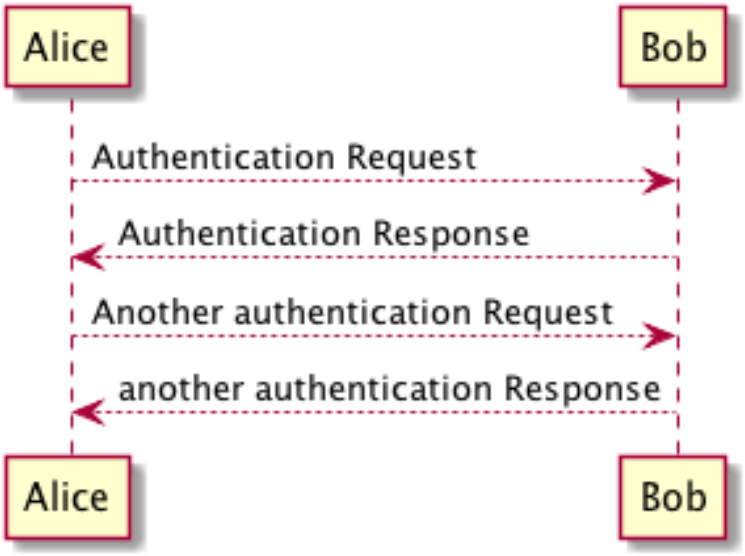


3. PlantUML

   (a) **Sequence Diagram**

   - Feature rich.
   - Information rich.

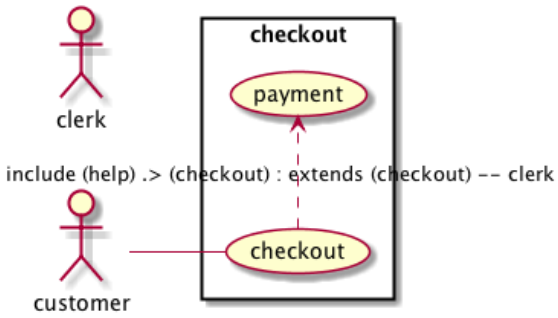
```
Alice --> Bob: Authentication Request
Bob --> Alice: Authentication Response

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



(b) **Use Case Diagram**

- Actor variable aliasing feature.

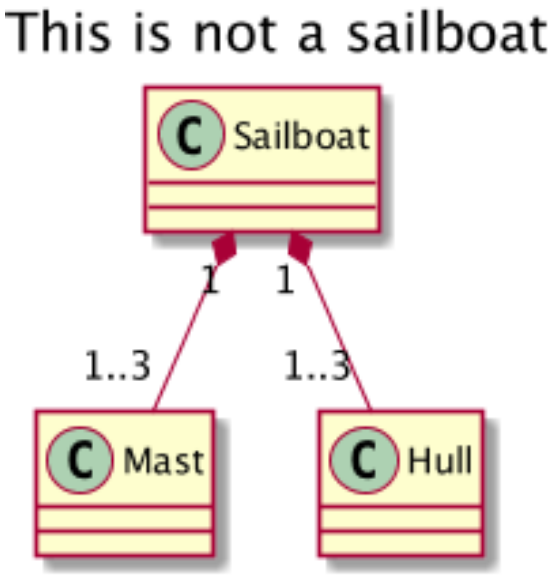
```
left to right direction
skinparam packageStyle rect
actor customer
actor clerk
rectangle checkout {
customer -- (checkout)
(checkout) .> (payment) : include (help) .> (checkout) : extends (checkout) -- clerk
}
```



(c) **Class Diagram**

- Valuable for all sorts of ideas in addition to classes.
- `hide`, `show`, and `include` are mentioned.
  - Could be a great reuse mechanism combined with noweb and tangling.
- Spotted characters might be useful to indicating other things.
  - Example is data which is clearly a first-class citizen.
  - Six package visualization types.
  - Packaging vs. namespaceing.
  - Good support for splitting large images among output pages.

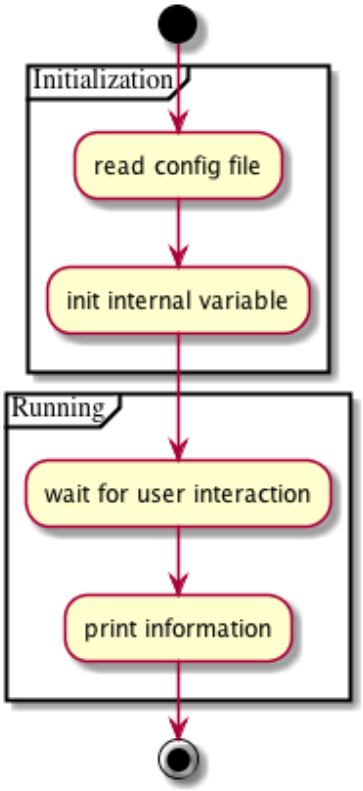
```
title This is not a sailboat
scale 200 width
Sailboat "1" *-- "1..3" Mast
Sailboat "1" *-- "1..3" Hull
```



(d) **Activity Diagram**

- May indicate top to bottom flow using `top`.
- May label arrows.
- May force arrow direction.
- if/else structure for branching.
- Partition construct.
- New syntax with more examples.

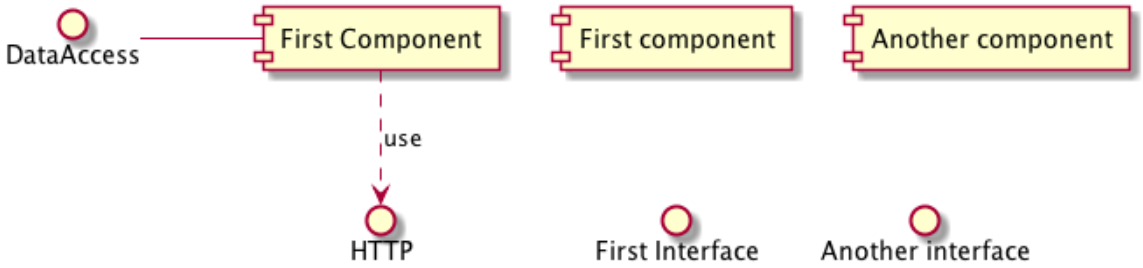
```
start
partition Initialization {
:read config file;
:init internal variable;
}
partition Running {
:wait for user interaction;
:print information;
}

stop
```



(e) **Component Diagram**

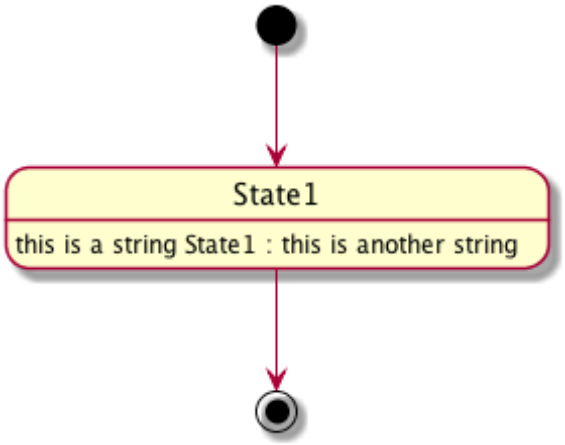
- The names to define all of the diagram entity types.
- Identify “Modern UML”.
- Good for summaries.

```
[First component]
[Another component]
() "First Interface"
() "Another interface" as Interf2
DataAccess - [First Component]
[First Component] ..> HTTP : use
```



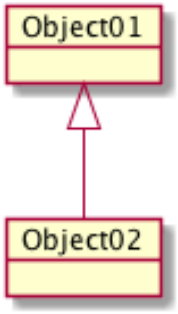
(f) **State Diagram**

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



(g) **Object Diagram**

```
object Object01
object Object02
Object01 <|-- Object02
```



(h) **Options**

- Commands.
  - Header and footer values.
  - Zoom level.
  - Creole markup for most text elements.
  - Lists and sub-lists.
  - Horizontal lines. Will appear in most containers.
  - Headings.
  - Plain old HTML.
  - Tables, LaTeX style.
  - Use OpenIconic icons anywhere.
- Fonts and colors.
  - You can change just about everything.
  - You may nest definitions.
  - `monochrome true` option.
    - If you are printing
    - Or don't want color.
- Internationalization.
  - Full Unicode character support.