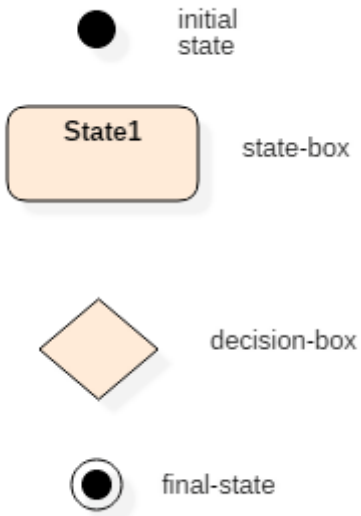
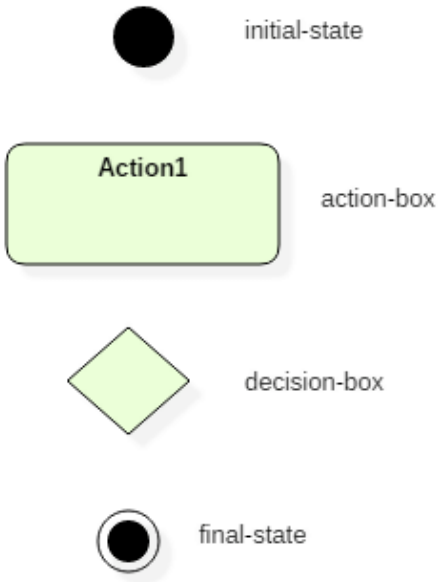
Unified Modeling Language (UML)

Note:

State machine

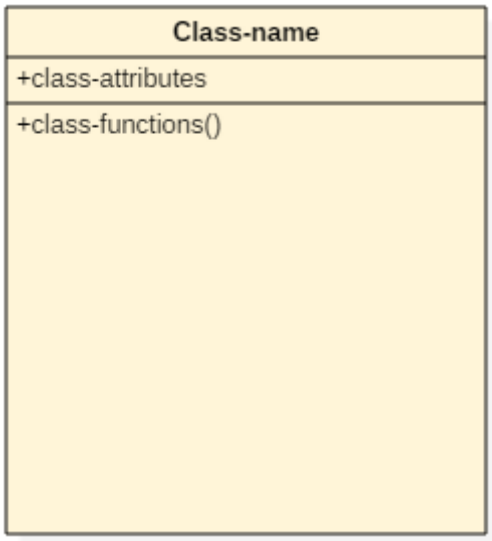


Activity diagram



UML Building Blocks

Class Diagram



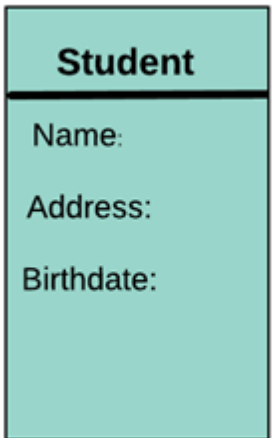


Figure 2 - Class "Student"

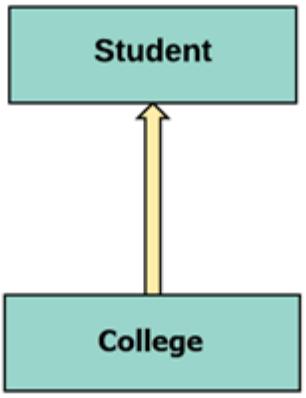
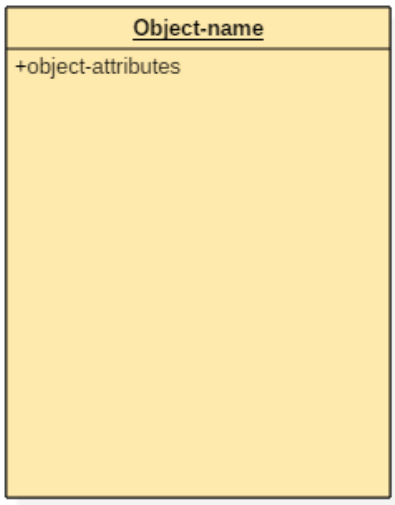
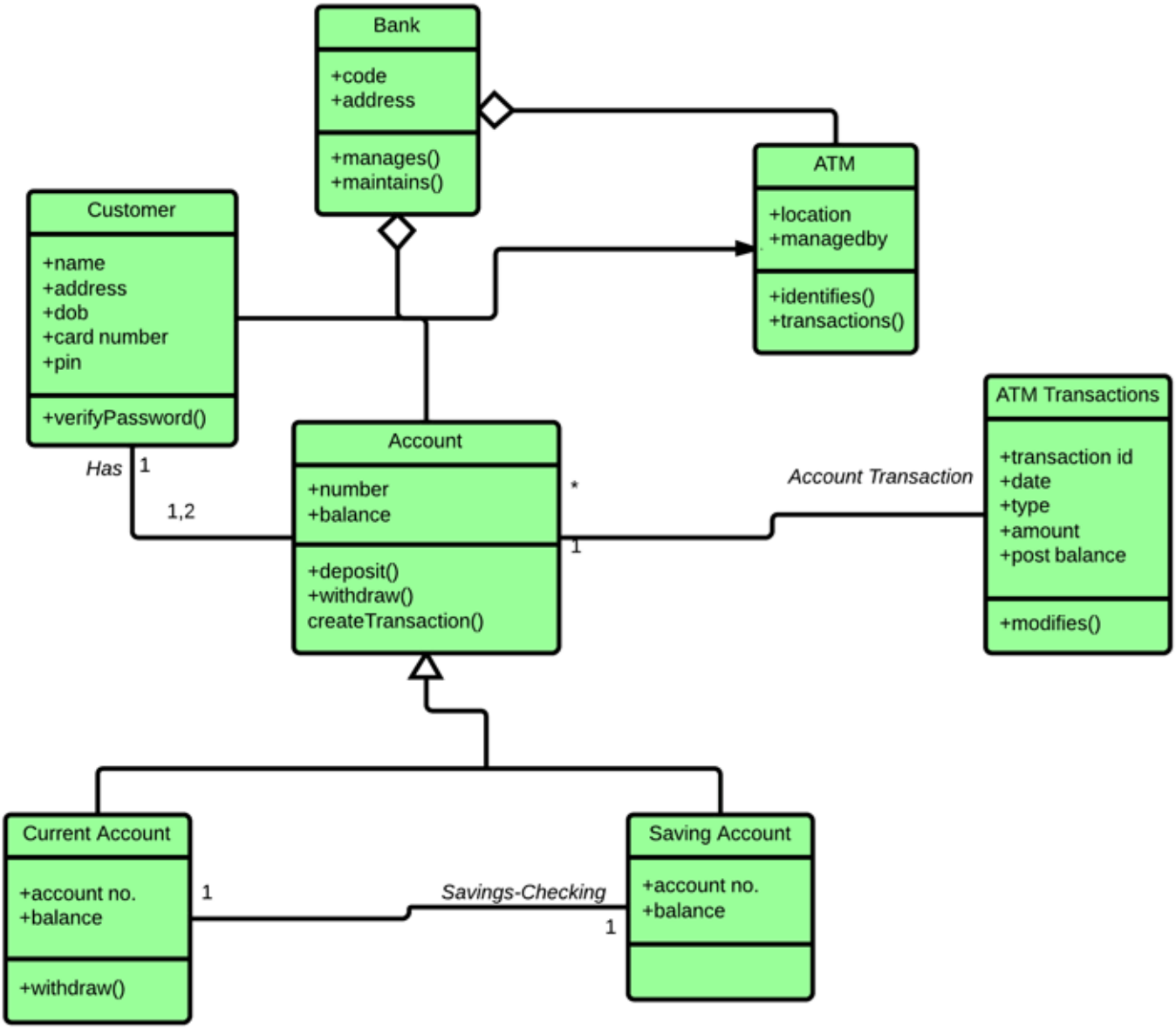


Figure 1 - Subclass

Object

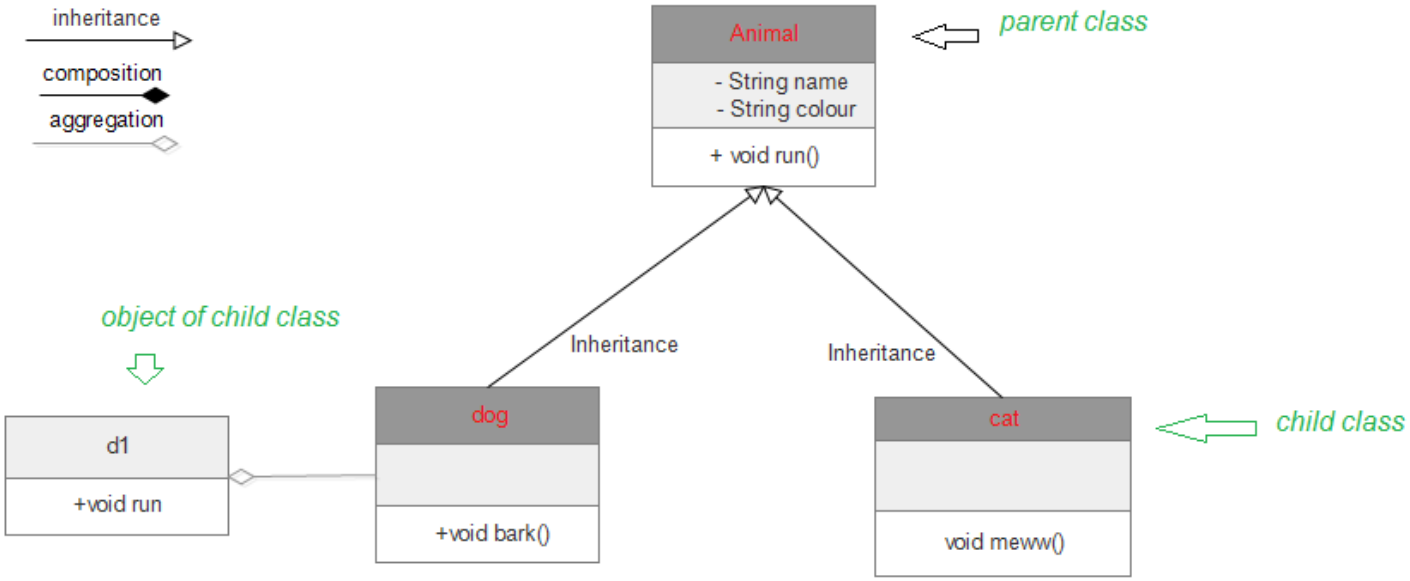


Example of Class Diagram

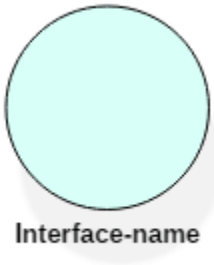


Note: # means 'protected'
+ means public
- means private

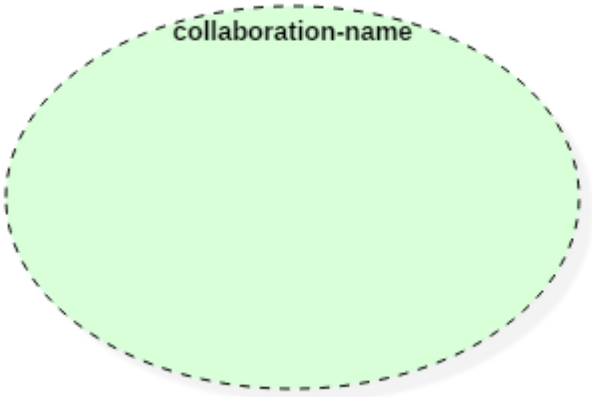
Example of Class Diagram



Interface

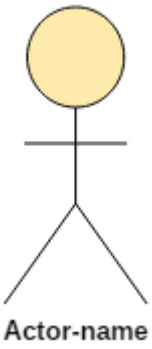


Collaboration:

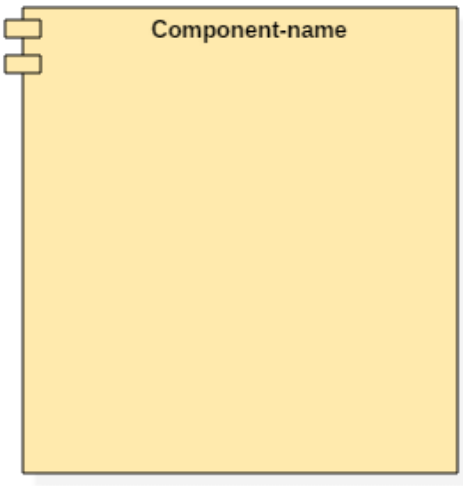


Use-case:

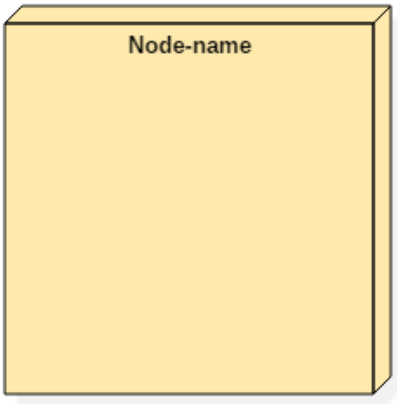
Actor:



Component



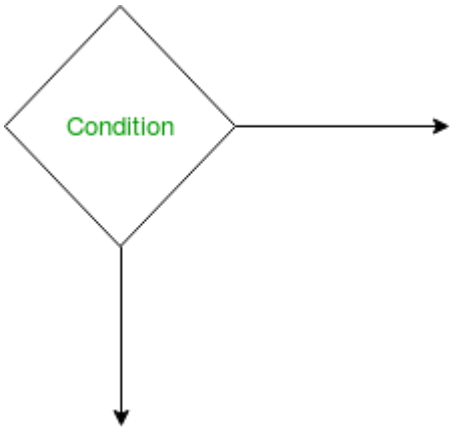
Node:



Activity Diagram Notations –

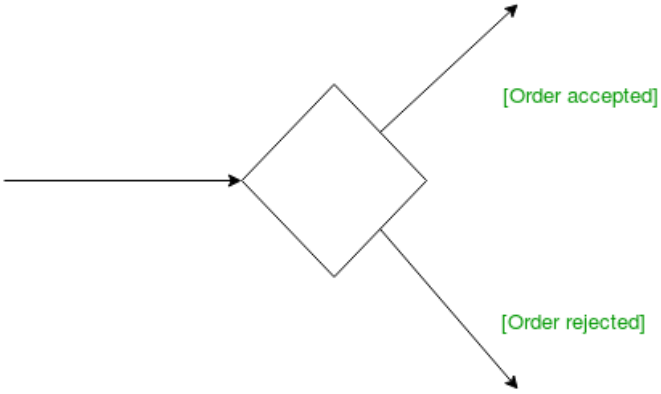
Initial State

Decision Node and Branching

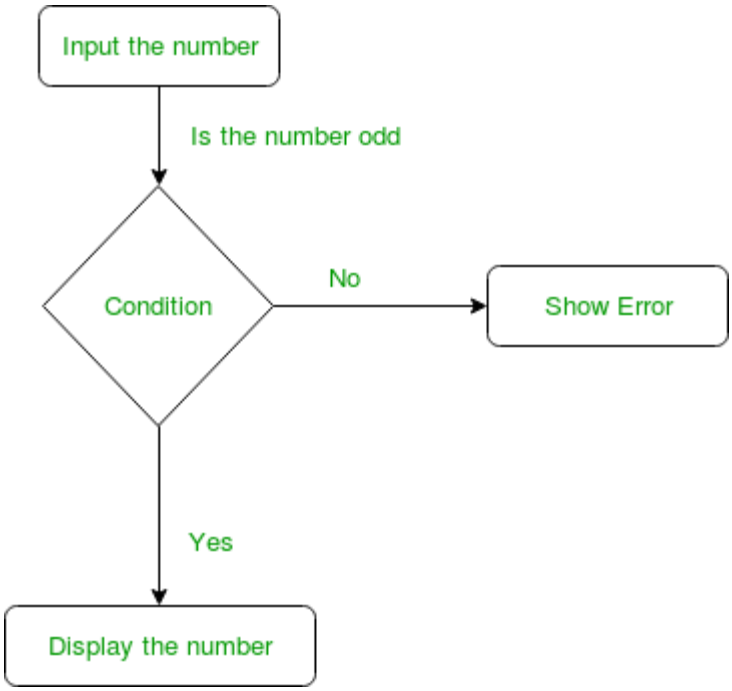


Action or Activity State

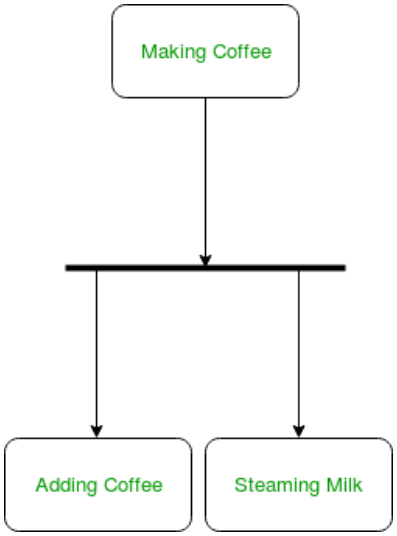
Guards – A Guard refers to a statement written next to a decision node on an arrow sometimes within square brackets.



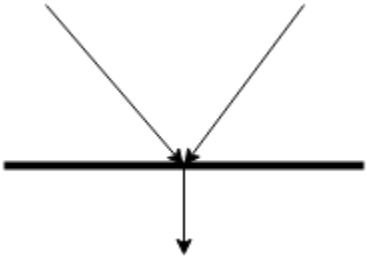
Action Flow or Control flows

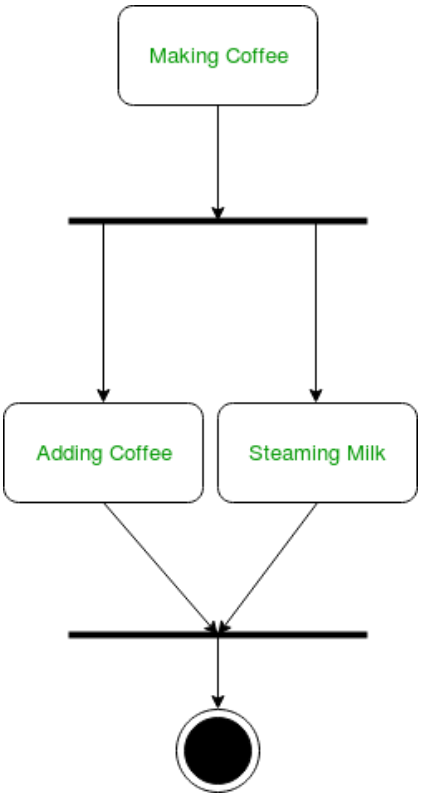


Fork – Fork nodes are used to support concurrent activities.

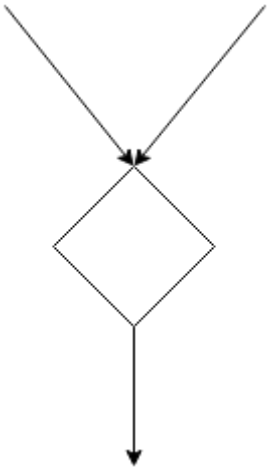


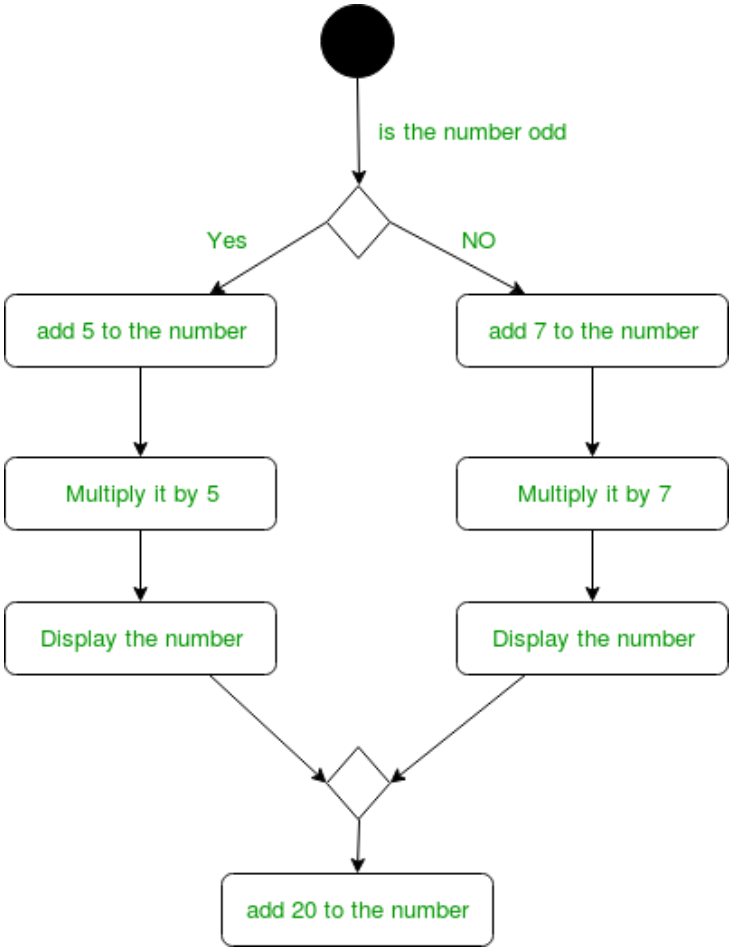
Join – Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge



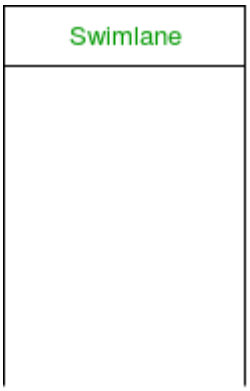


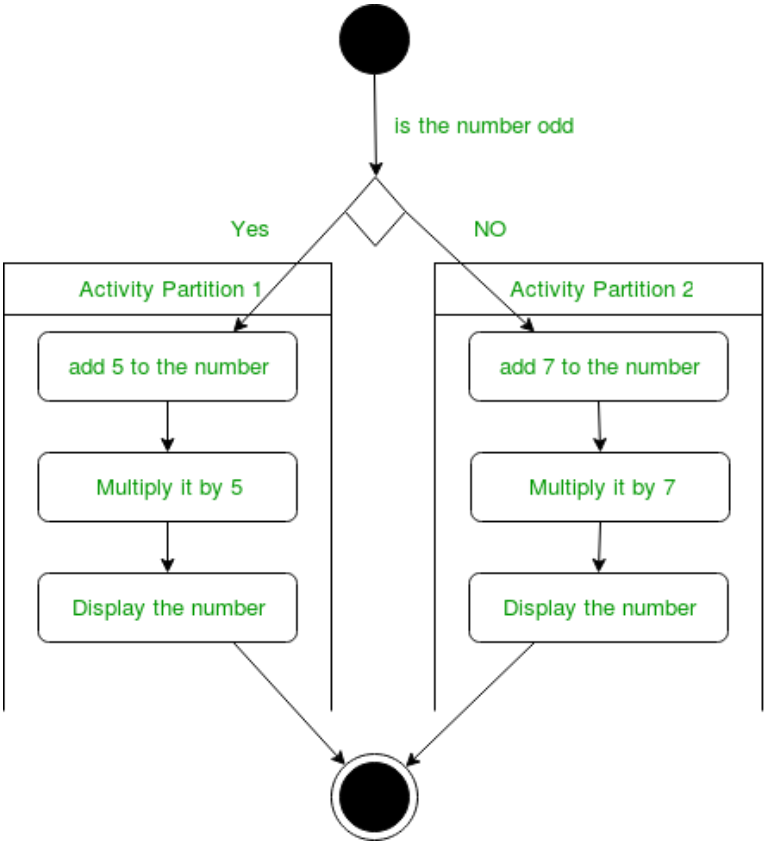
Merge or Merge Event – Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen.



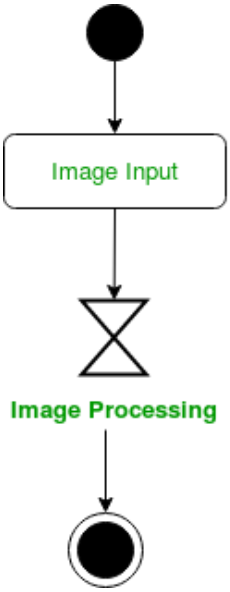


Swimlanes – We use swimlanes for grouping related activities in one column. Swimlanes group related activities into one column or one row. Swimlanes can be vertical and horizontal. Swimlanes are used to add modularity to the activity diagram.

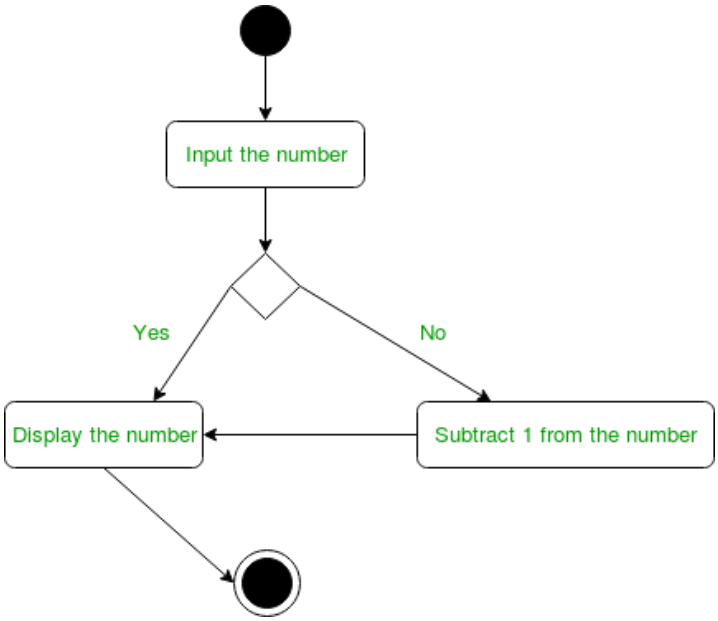




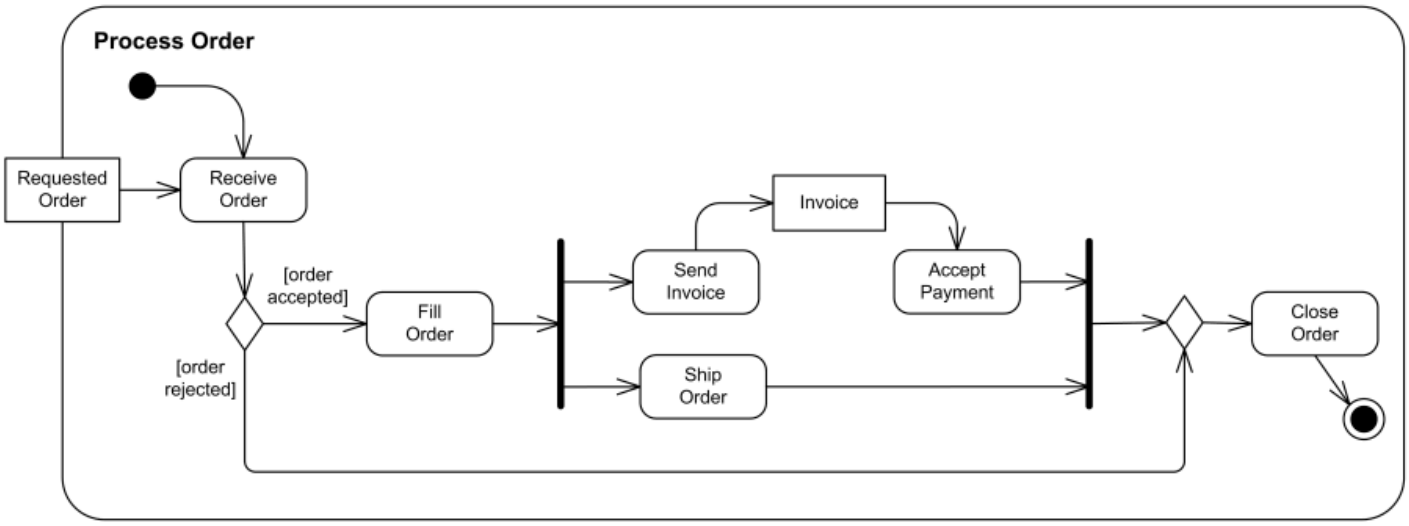
Time Event – **Figure –** time event notation We can have a scenario where an event takes some time to complete. We use an hourglass to represent a time event.



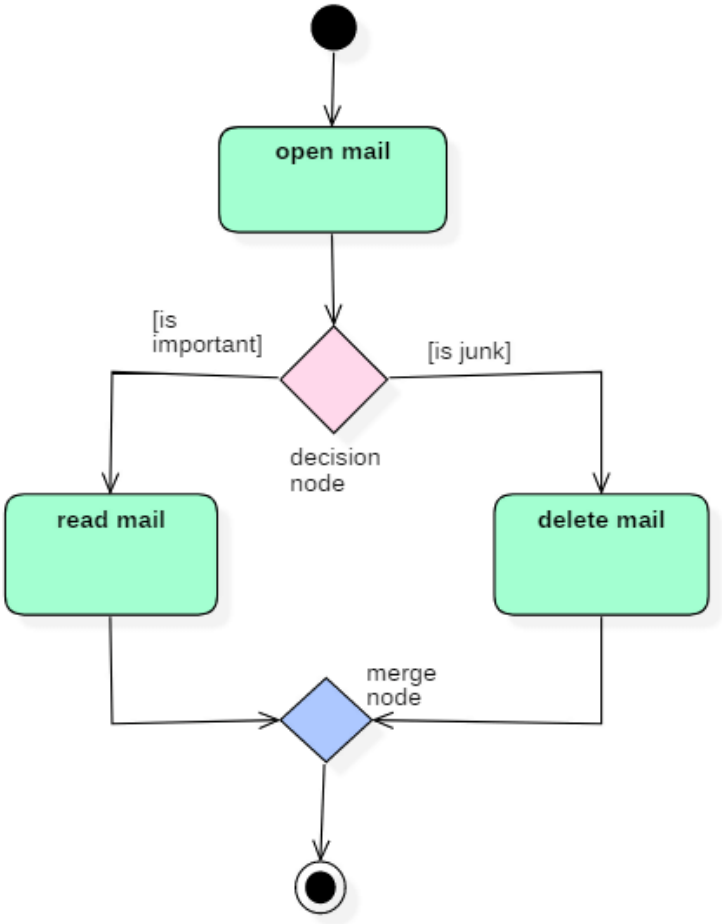
Final State or End State – The state which the system reaches when a particular process or activity ends is known as a Final State or End State. We use a filled circle within a circle notation to represent the final state in a state machine diagram.



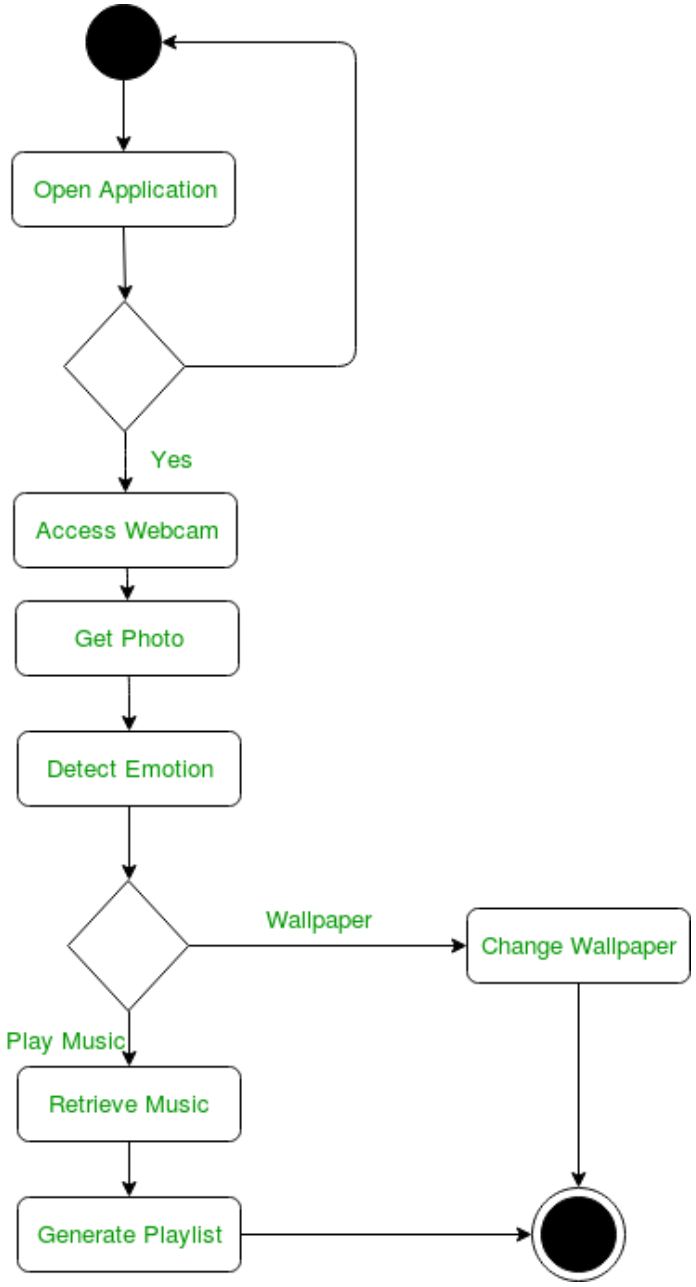
Example of Activity Diagram



Example of Activity Diagram



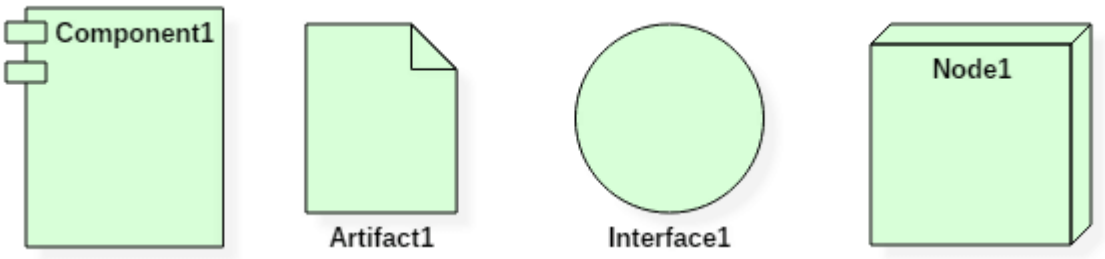
Example of Activity Diagram



Deployment Diagram

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

Deployment Diagram Symbol and notations



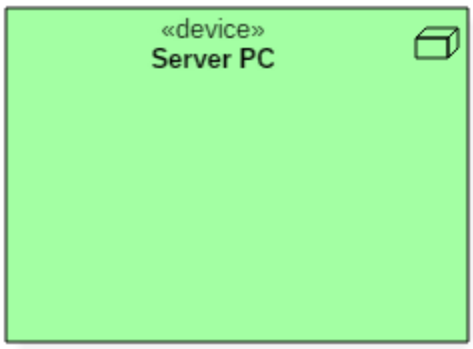
Artifact:
An artifact represents the specification of a concrete real-world entity related to software development. You can use the artifact to describe a framework which is used during the software development process or an executable file. Artifacts are deployed on the nodes. The most common artifacts are as follows,

1. Source files
2. Executable files
3. Database tables
4. Scripts
5. DLL files
6. User manuals or documentation
7. Output files

Artifacts are deployed on the nodes.

What is a node?

Node is a computational resource upon which artifacts are deployed for execution. A node is a physical thing that can execute one or more artifacts. A node may vary in its size depending upon the size of the project.



Node is an essential UML element that describes the execution of code and the communication between various entities of a system. It is denoted by a 3D box with the node-name written inside of it. Nodes help to convey the hardware which is used to deploy the software.

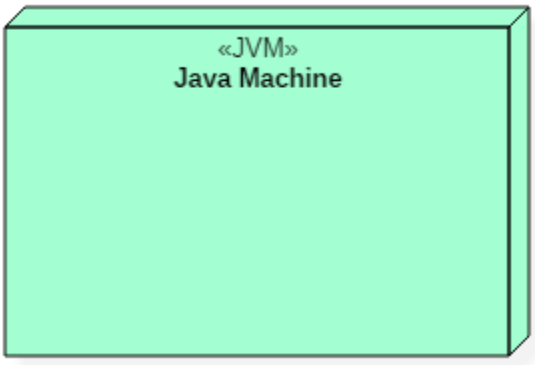
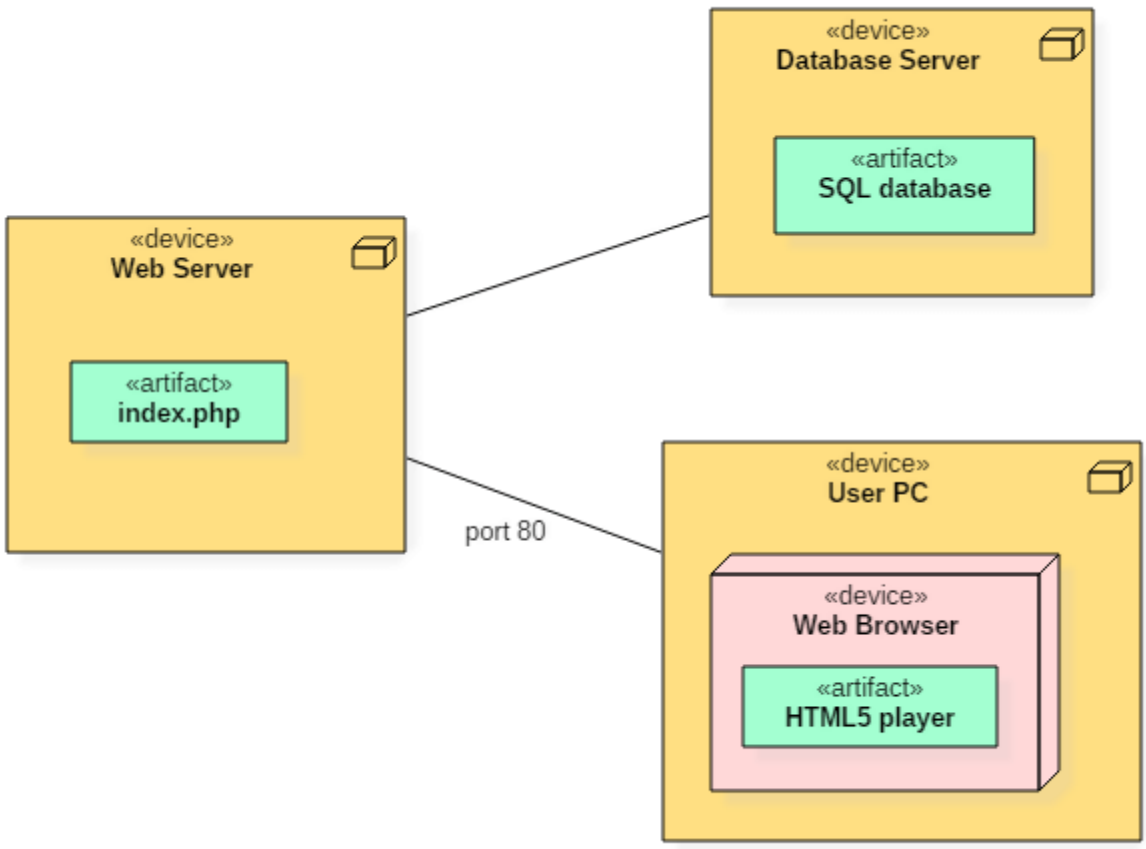


Figure 3 - Execution Environment Node

Example of a Deployment Diagram



Use Case Diagram

Use Case Diagram captures the system's functionality and requirements by using actors and use cases. Use Cases model the services, tasks, function that a system needs to perform. Use cases represent high-level functionalities and how a user will handle the system. Use-cases are the core concepts of Unified Modelling language modeling.

When to use a use-case diagram?

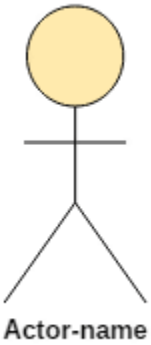
In general use case diagrams are used for:

1. Analyzing the requirements of a system
2. High-level visual software designing
3. Capturing the functionalities of a system
4. Modeling the basic idea behind the system
5. Forward and reverse engineering of a system using various test cases.

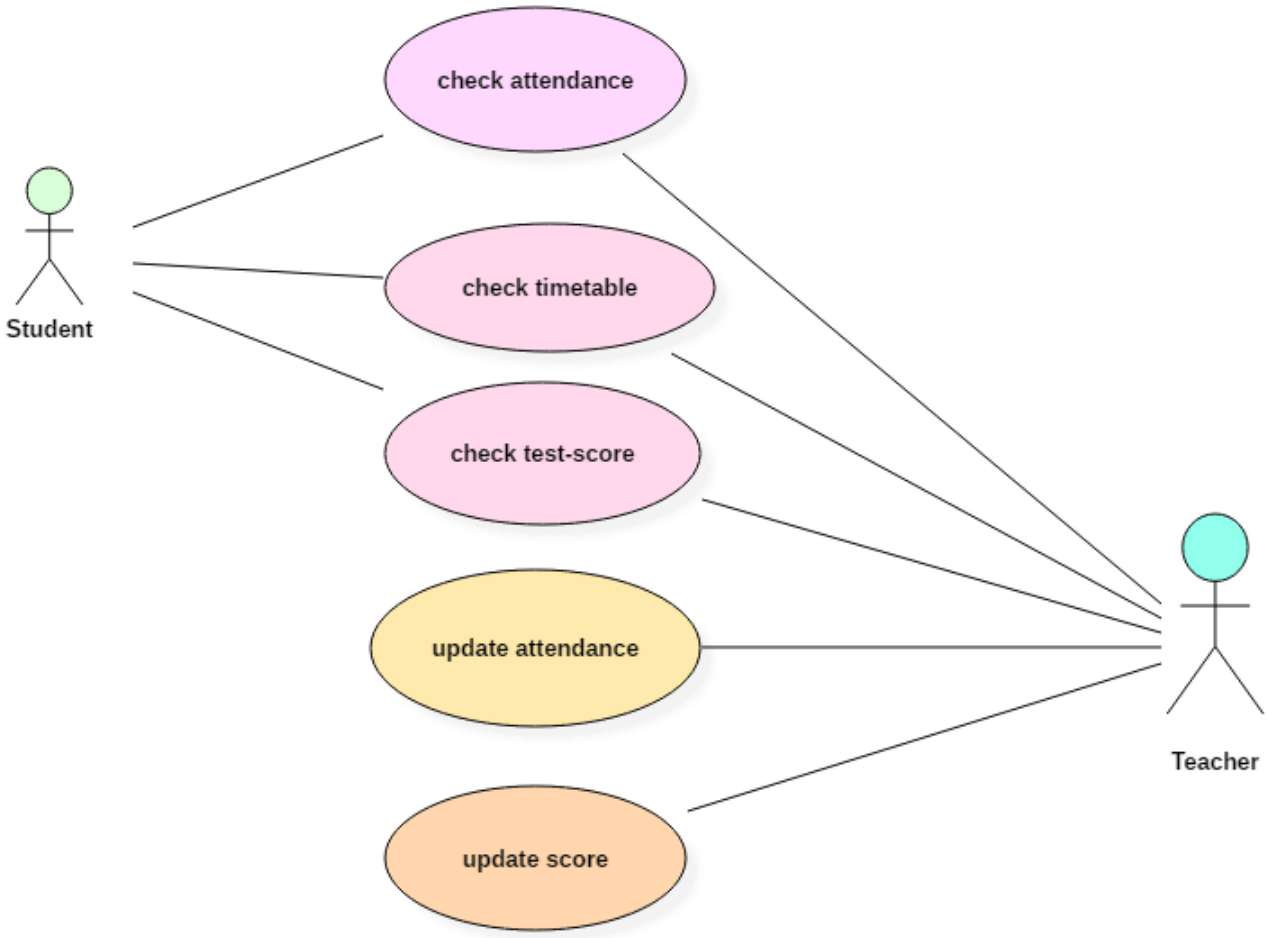
Use cases are intended to convey desired functionality so the exact scope of a use case may vary according to the system and the purpose of creating UML model.

Use Case

Actor



An example of a use-case diagram



State Diagrams

A **state diagram** is used to represent the condition of the system or part of the system at finite instances of time. It's a **behavioral** diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as **State machines** and **State-chart Diagrams**

(1) **Initial state –** We use a black filled circle represent the initial state of a System or a class.
(2) **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.
(3) **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.
(4) **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states.
(5) **Join –** We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events.
(6) **Self transition –** We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event. We use self transitions to represent such cases.
(7) **Composite state –** We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.
(8) **Final state –** We use a filled circle within a circle notation to represent the final state in a state machine diagram.