



M.Sc. in Computer Science
Department of Computer Science
Faculty of Applied Sciences
University of Sri Jayewardenepura

CSC 540 2.0 Software Engineering

Presented By:

Surani Tissera(PhD)

Department of Computer Science

System Modeling



This is a property of Department of Computer Science, Faculty of Applied Science, University of Sri Jayewardenepura.

Topics to be covered

- System Modeling
 - Interaction models
 - Use case diagram
 - Sequence diagram
 - Structural models
 - Class Diagram
 - Behavioral models
 - Activity Diagram
 - State Diagram



Recommended Reading

- Sommerville I. (2015), Software Engineering 10th Edition, Addison-Wesley
- Roger S. Pressman (2010), Software Engineering: A Practitioner's Approach 7th Edition, McGraw-Hill



Objectives

You will:

- understand **how graphical models can be used to represent software systems** and **why several types of model are needed to fully represent a system**;
- understand the **fundamental system modeling perspectives of context, interaction, structure, and behavior**;
- understand the principle diagram types in the **Unified Modeling Language (UML)** and **how these diagrams may be used in system modeling**;



Structural Model

- The structural perspective model represents a system's organization in terms of the parts that build the system and their relationships.
- Structural models can be static models or dynamic models.
- The static models represent the structure of the system design
- The dynamic models represent the system's organization during execution.
- The structural model presents in
 - Class diagrams



Class diagram

- A UML class diagram is a visual tool that represents the structure of a system
- It shows its classes, attributes, methods, and the relationships between them.
- It helps everyone involved in a project—like developers and designers—understand how the system is organized and how its components interact.



Purpose of Class Diagrams

- Shows static structure of classifiers in a system
- Diagram provides a basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from a business perspective

A UML class diagram is made up of:

- A set of classes and
- A set of relationships between classes



Class Notation

A class notation consists of three parts:

1. Class Name

- The name of the class appears in the first partition.

2. Class Attributes

- Attributes are shown in the second partition.
- The attribute type is shown after the colon.
- The attribute visibility is shown before the name



Visibility of Class attributes and Operations

UML identifies four types of visibility: public, protected, private, and package

- + denotes public attributes or operations
- denotes private attributes or operations
- # denotes protected attributes or operations
- ~ denotes package attributes or operations

Class Visibility Example

MyClass
+attribute1 : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : bool, in p2) : String
-op2(input p3 : int) : float
#op3(out p6) : Class6*



Class Notation

3. Class Operations (Methods)

- Operations are shown in the third partition. They are services the class provides.
- The return type of a method is shown after the colon at the end of the method signature.
- The return type of method parameters is shown after the colon following the parameter name.
- The method visibility is shown before the name



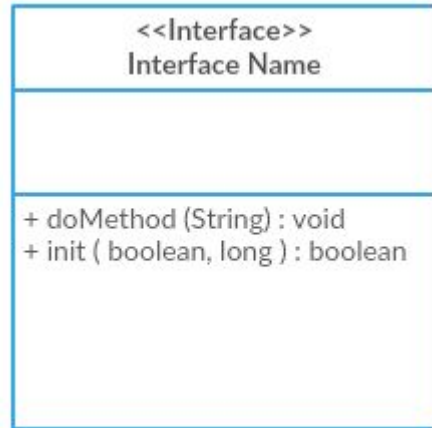
Class Notation

MyClass
+attribute1 : int -attribute2 : float #attribute3 : Circle
+op1(in p1 : bool, in p2) : String -op2(input p3 : int) : float #op3(out p6) : Class6*



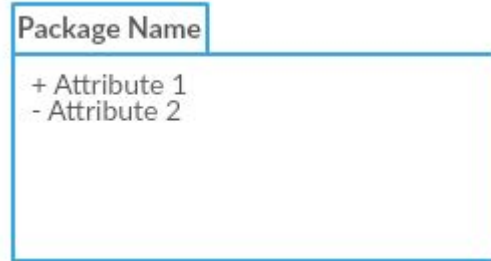
Interface Notation

indicates a set of operations that would detail the responsibility of a class.



Package Notation

The package symbol is used to group classes or interfaces that are either similar in nature or related. Grouping these design elements using the package symbols improves the readability of the diagram



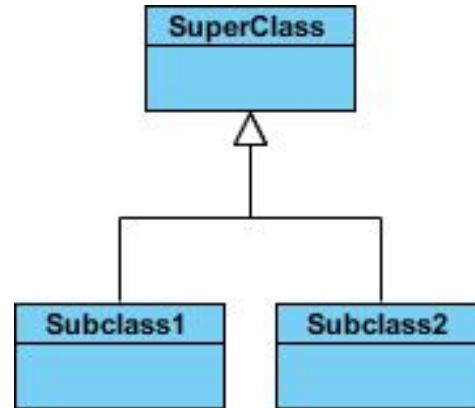
Class Relationships

- A class may be involved in one or more relationships with other classes.
- A relationship can be one of the following types:

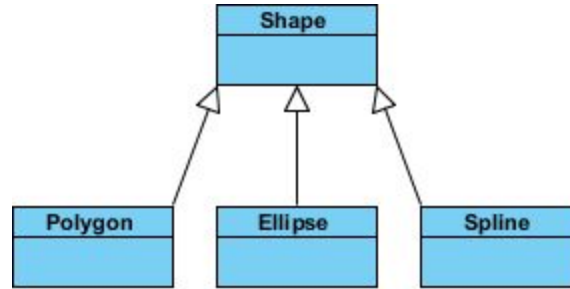


Class Relationships: Inheritance(or Generalization)

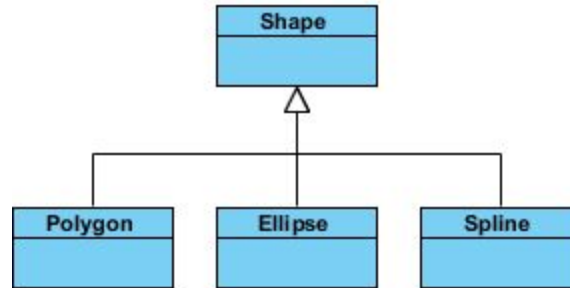
- Represents an "is-a" relationship.
- An abstract class name is shown in italics.
- SubClass1 and SubClass2 are specializations of Super Class.
- A solid line with a hollow arrowhead that point from the child to the parent class



Class Relationships: Inheritance(or Generalization)



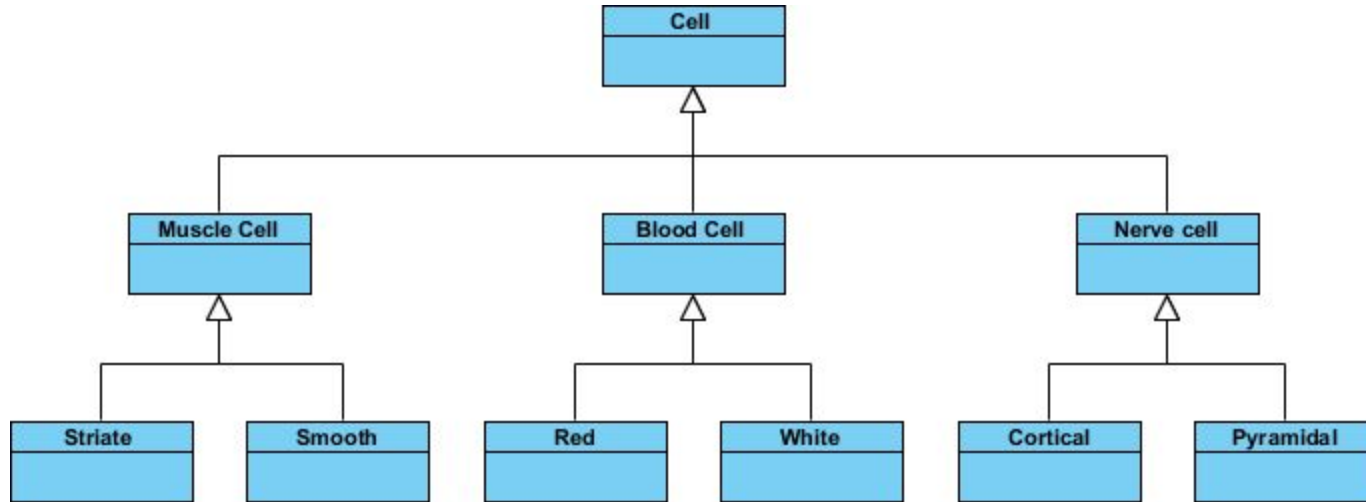
Style 1: Separate target



Style 2: Shared target

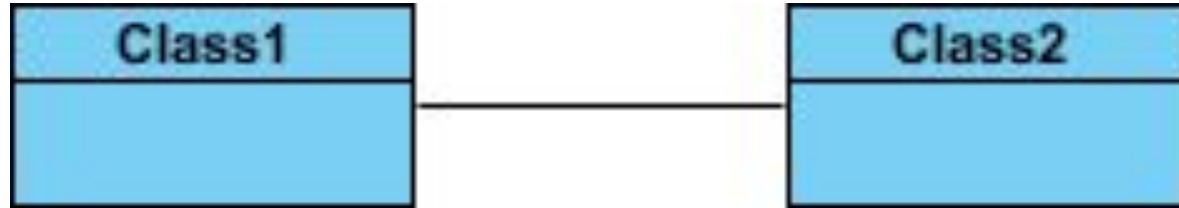


Inheritance Example - Cell Taxonomy

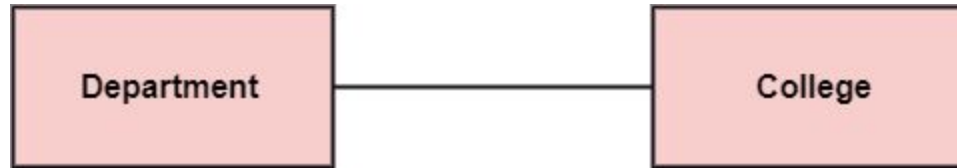


Class Relationships- Simple Association

- A structural link between two peer classes.
- There is an association between Class1 and Class2
- A solid line connecting two classes



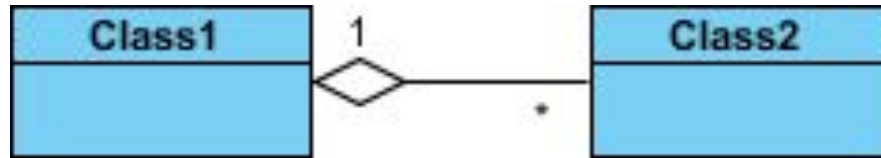
Class Relationships- Simple Association



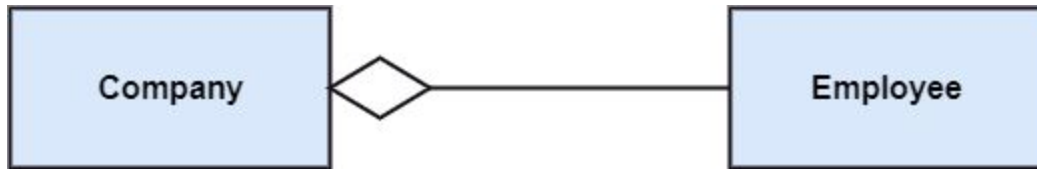
Class Relationships- Aggregation

A special type of association. It represents a "part of" relationship.

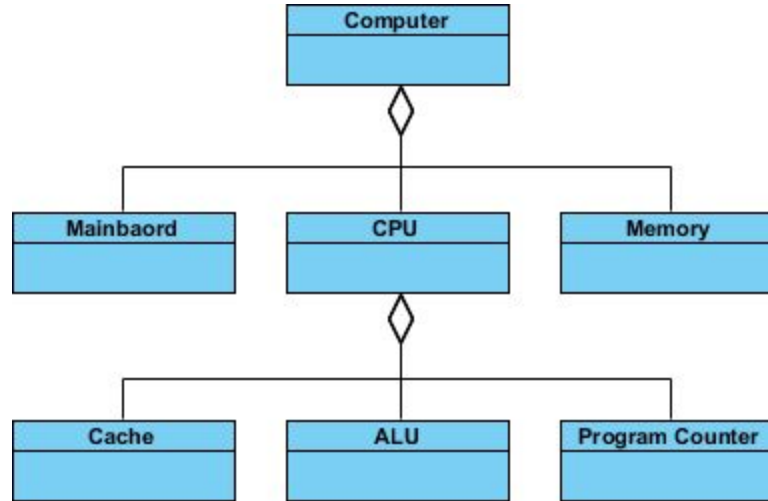
- Class2 is part of Class1.
- Many instances (denoted by the *) of Class2 can be associated with Class1.
- Objects of Class1 and Class2 have separate lifetimes.
- A solid line with an unfilled diamond at the association end connected to the class of composite



Class Relationships- Aggregation



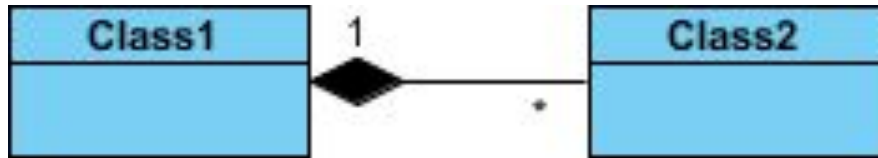
Aggregation Example - Computer and parts



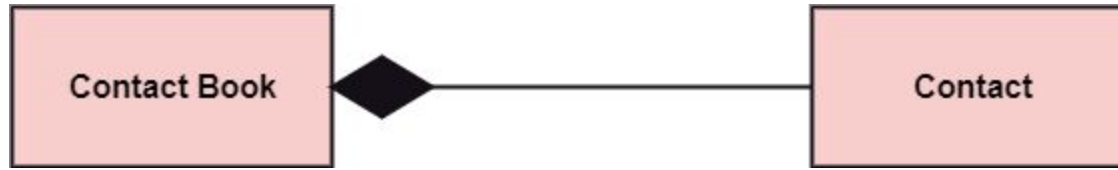
Class Relationships- Composition

A special type of aggregation where parts are destroyed when the whole is destroyed.

- Objects of Class2 live and die with Class1.
- Class2 cannot stand by itself.
- A solid line with a filled diamond at the association connected to the class of composite



Class Relationships- Composition

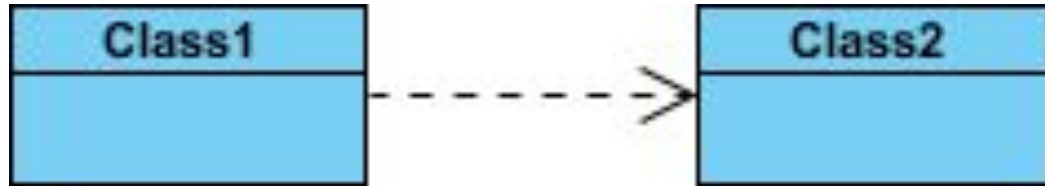


Class Relationships- Dependency:

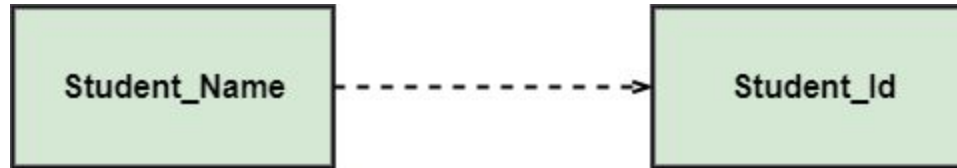
Exists between two classes if the changes to the definition of one may cause changes to the other (but not the other way around).

Class1 depends on Class2

A dashed line with an open arrow



Class Relationships- Dependency:

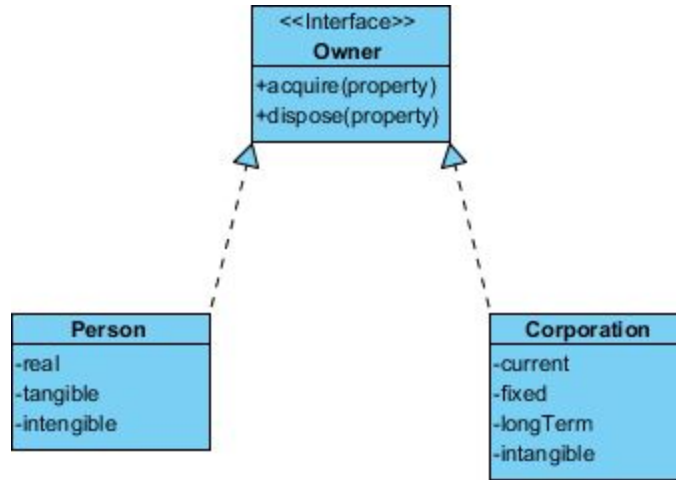


Class Relationships- Realization

In other words, you can understand this as the relationship between the interface and the implementing class.



Class Relationships- Realization



Multiplicity

How many objects of each class take part in the relationships and multiplicity can be expressed as:

Exactly one - 1

Zero or one - 0..1

Many - 0..* or *

One or more - 1..*

Exact Number - e.g. 3..4 or 6

Or a complex relationship - e.g. 0..1, 3..4, 6..* would mean any number of objects other than 2 or 5



How to Draw a Class Diagram

Step 1: Identify the class names

The first step is to identify the primary objects of the system.

Step 2: Distinguish relationships

Next step is to determine how each of the classes or objects are related to one another.



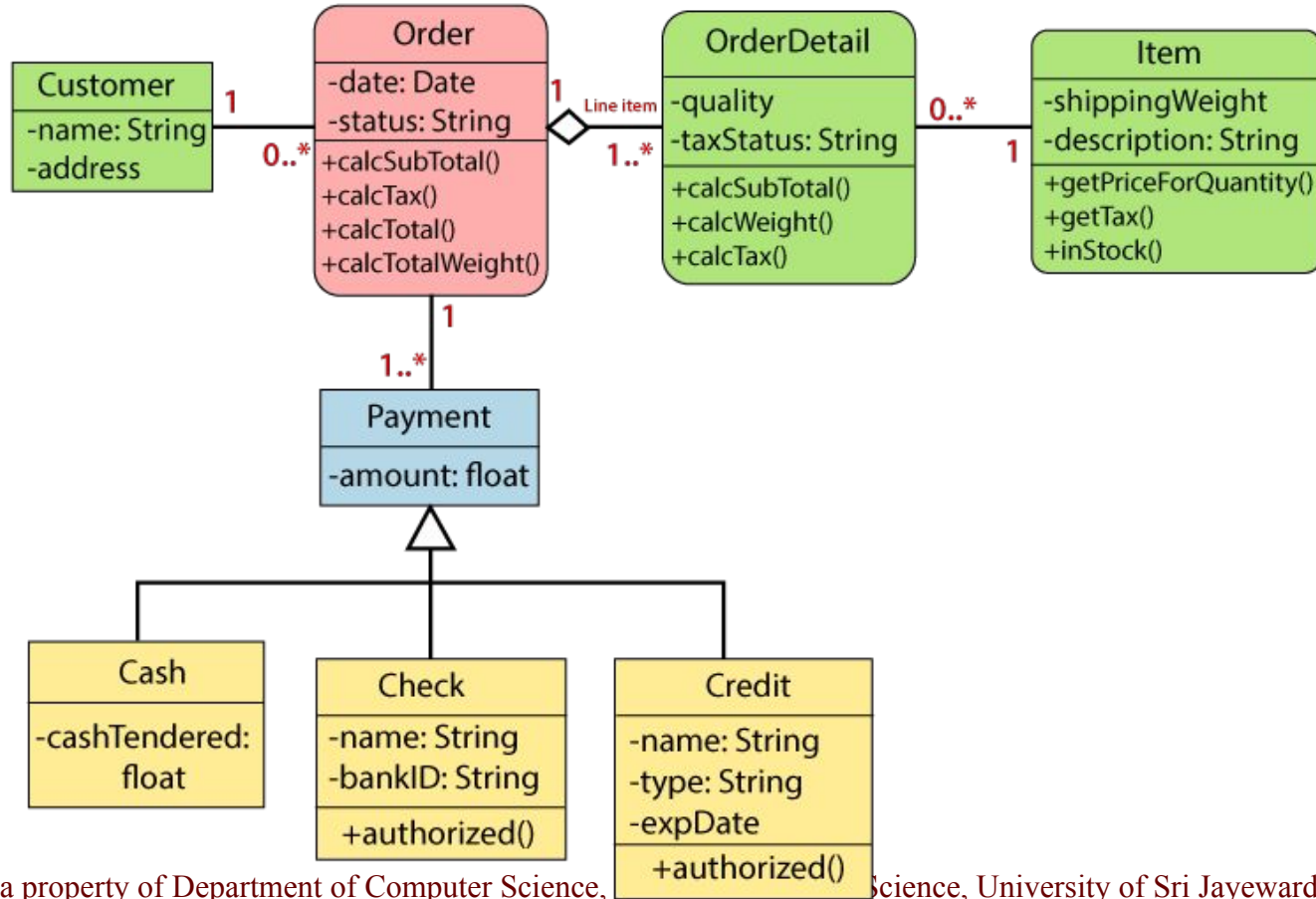
How to Draw a Class Diagram

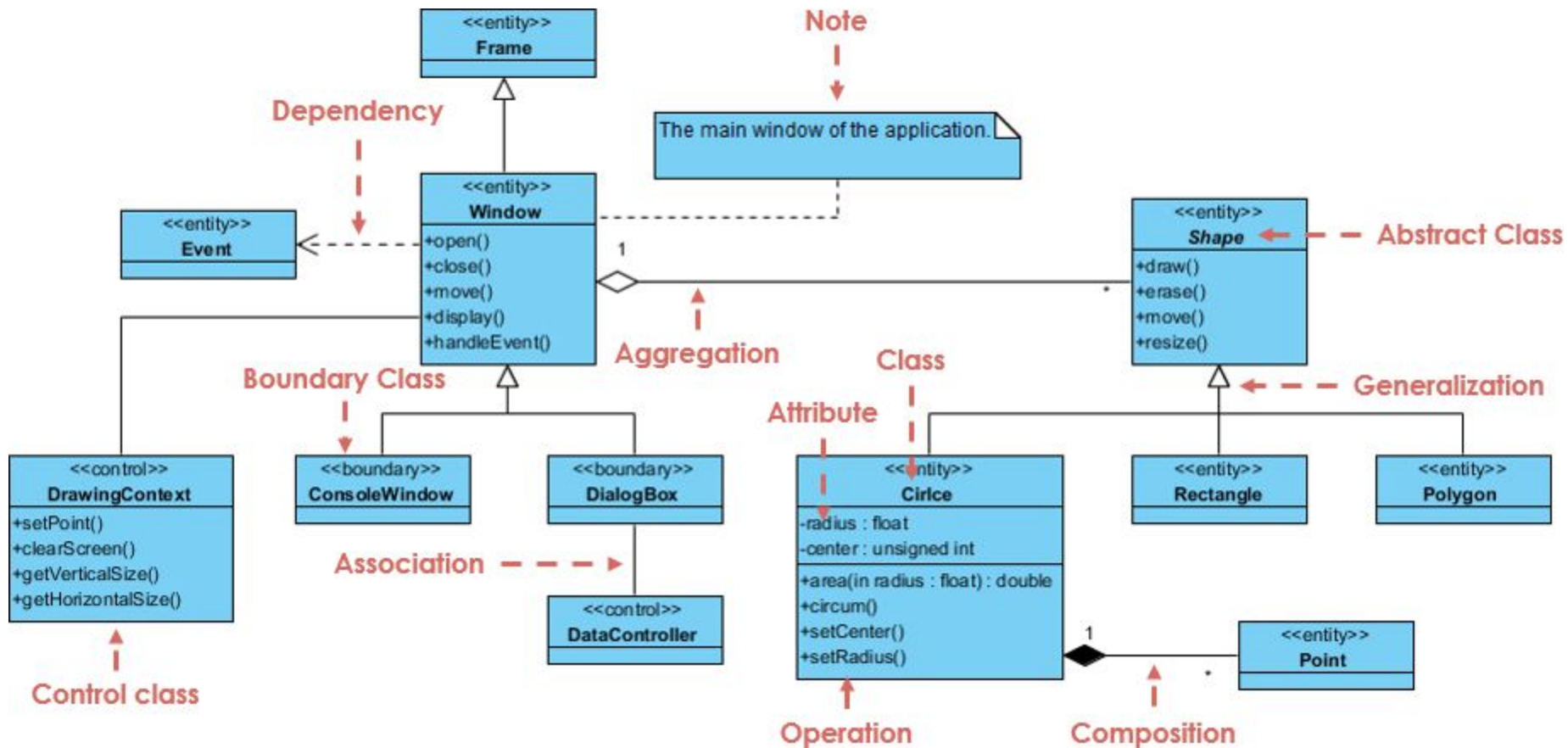
Step 3: Create the Structure

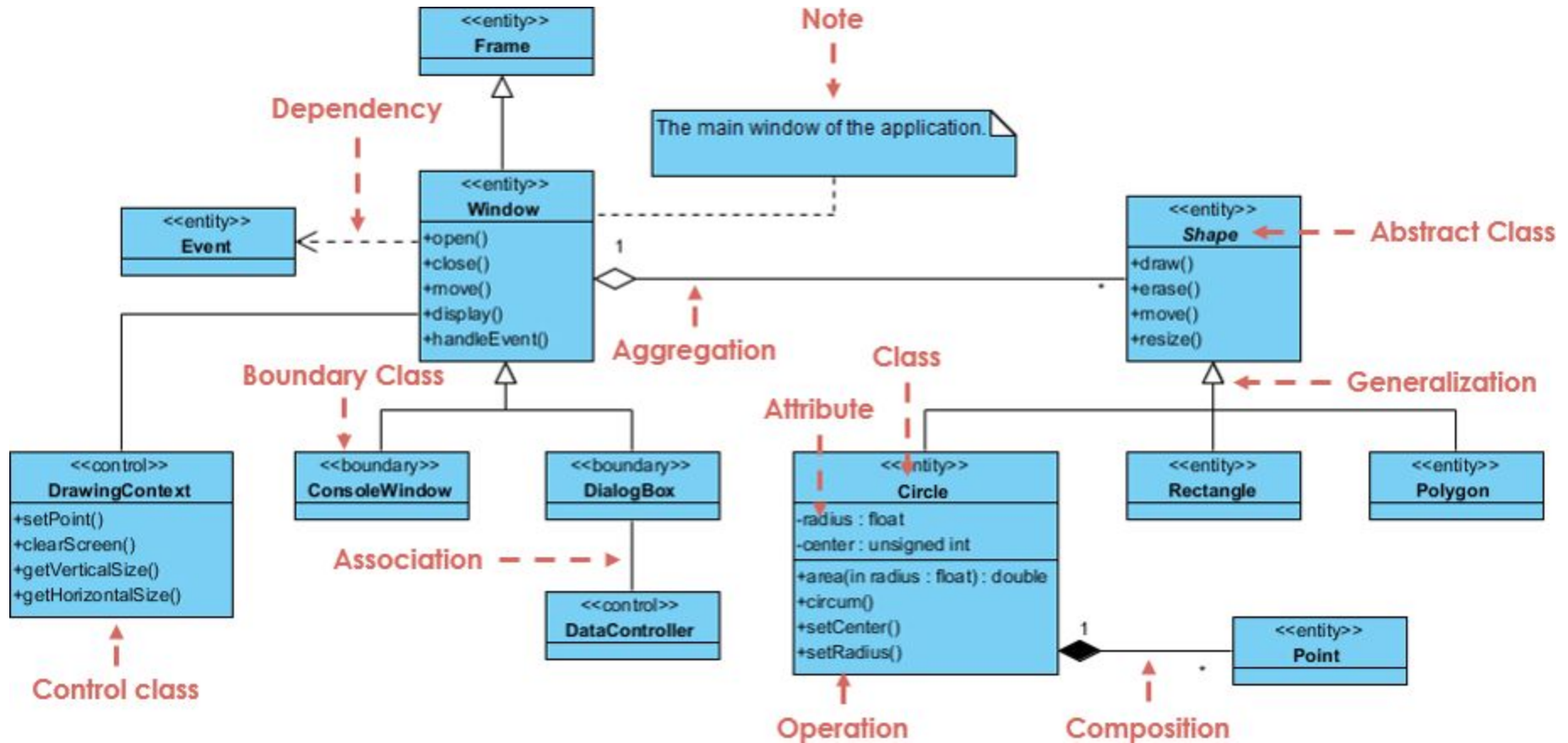
First, add the class names and link them with the appropriate connectors. You can add attributes and functions/ methods/ operations later.



- Sales order class diagram







Behavioral Model

- The behavioural model is used to illustrate the dynamic behaviour of the system at the time when the model is executing. The behavioural model depicts what occurs or what should occur when a system reacts to a stimulus from its environment.

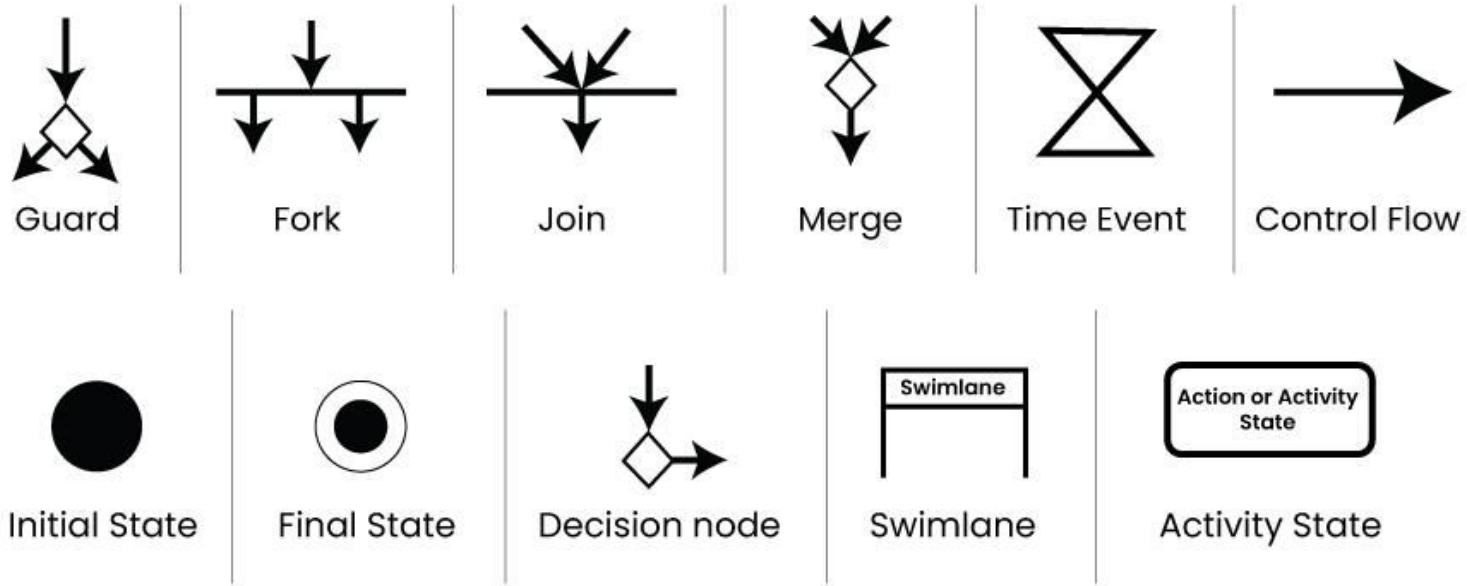


Activity Diagram

- Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration, and concurrency.
- It describes how activities are coordinated to provide a service which can be at different levels of abstraction.



Activity Diagram- Notation



Activity Diagram Notation

Symbol

Name

Use



Start/ Initial Node

Used to represent the starting point or the initial state of an activity



Activity / Action State

Used to represent the activities of the process







Control Flow / Edge

Used to represent the flow of control from one action to the other



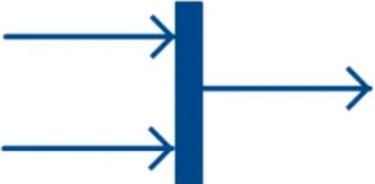


Activity Diagram Notation

Symbol	Name	Use
	Object Flow / Control Edge	Used to represent the path of objects moving through the activity
	Activity Final Node	Used to mark the end of all control flows within the activity
	Flow Final Node	Used to mark the end of a single control flow
	Decision Node	Used to represent a conditional branch point with one input and multiple outputs






Activity Diagram Notation

Symbol	Name	Use
	Merge Node	Used to represent the merging of flows. It has several inputs, but one output.
	Fork	Used to represent a flow that may branch into two or more parallel flows
	Merge	Used to represent two inputs that merge into one output



Activity Diagram Notation

Symbol	Name	Use
	Signal Sending	Used to represent the action of sending a signal to an accepting activity
	Signal Receipt	Used to represent that the signal is received
	Note/ Comment	Used to add relevant comments to elements



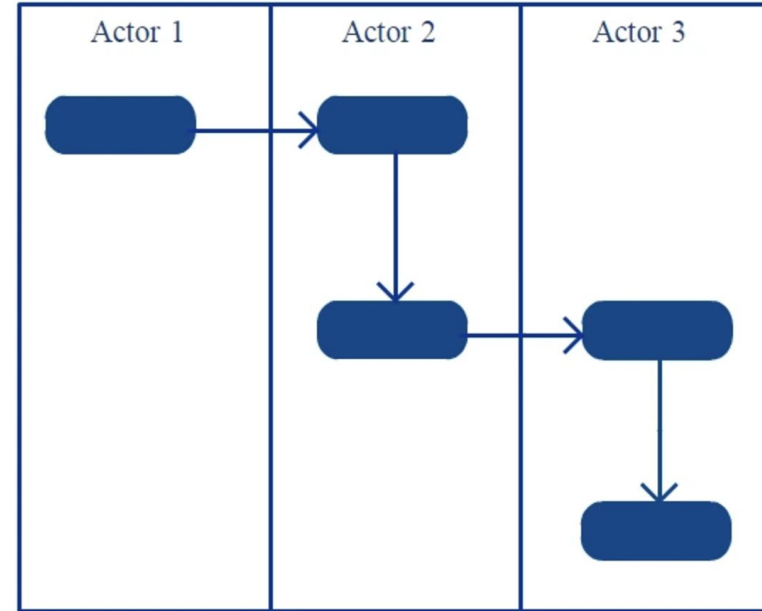
Activity Diagram Notation

- In activity diagrams, swimlanes – also known as partitions – are used to represent or group actions carried out by different actors in a single thread.

Here are a few tips you can follow when using swimlanes.

- Add swimlanes to linear processes. It makes it easy to read.
- Don't add more than 5 swimlanes.
- Arrange swimlanes in a logical

Activity Diagrams with Swimlanes



manner.

How to Draw an Activity Diagram

Step 1: Figure out the action steps from the use case
identify the various activities and actions your business process or system is made up of.

Step 2: Identify the actors who are involved
figured out who the actors are, then it's easier to determine each action they are responsible for.



How to Draw an Activity Diagram

Step 3: Find a flow among the activities

Figure out in which order the actions are processed.

Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram.

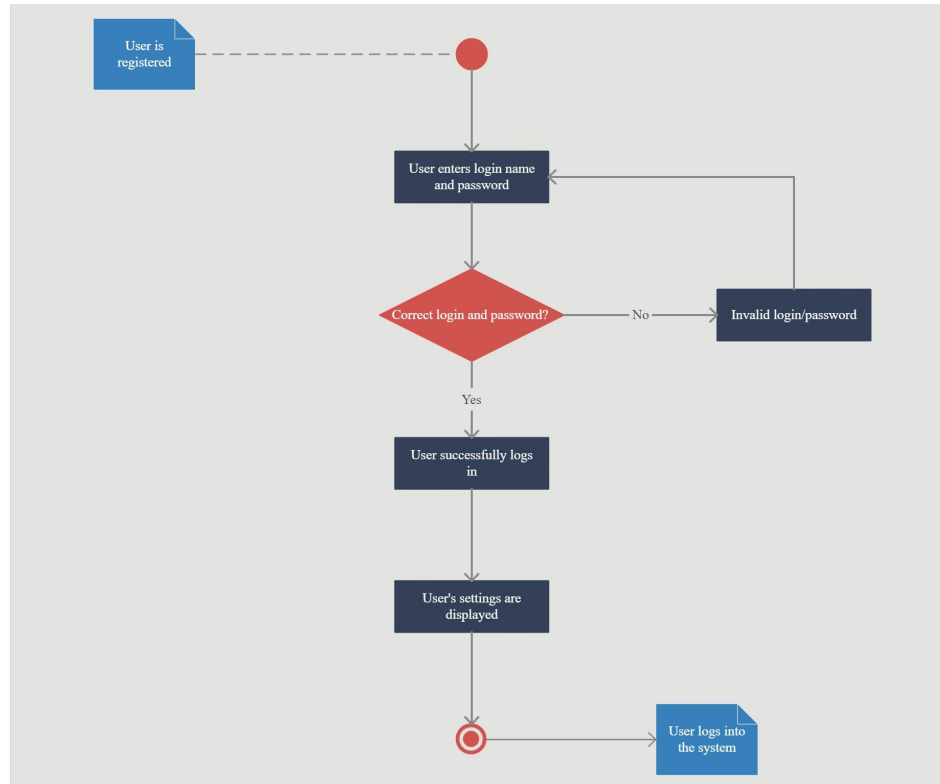
And do you have to complete some actions before you can proceed to others?

Step 4: Add swimlanes

You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.

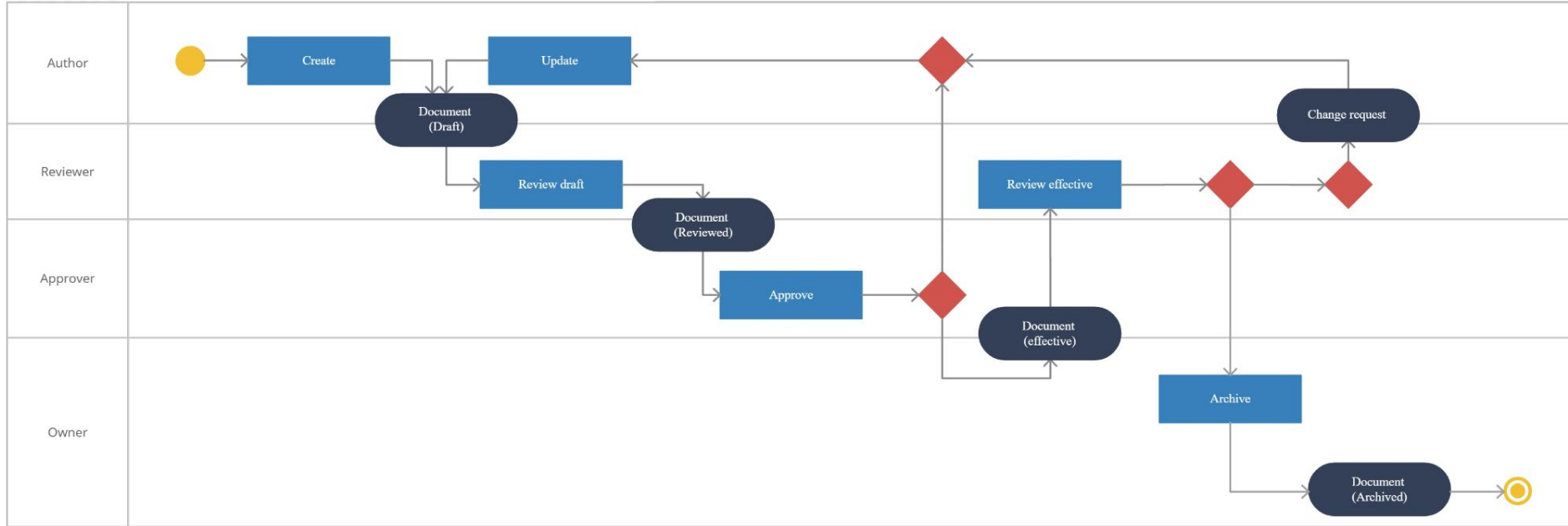


Activity Diagram Example



Activity Diagram- Example

DOCUMENT MANAGEMENT SYSTEM for ABC Co.



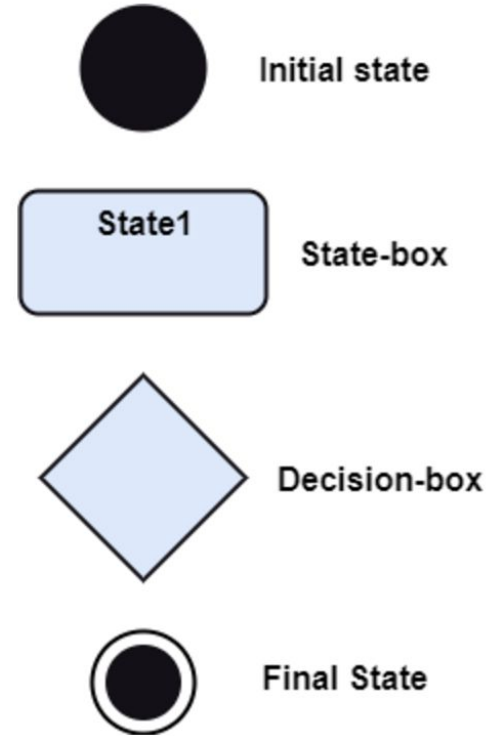
State Diagram

- 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.
- A state machine diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli(events that causes system to changes its state from one to another).



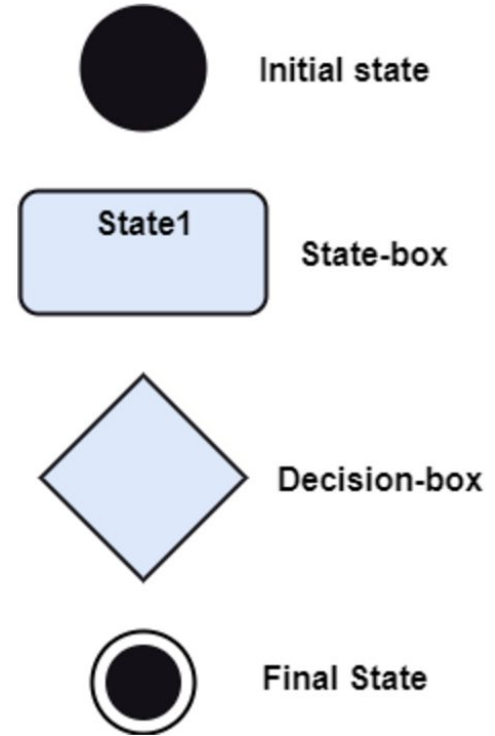
Notation of a State Machine Diagram

- Initial state: It defines the initial state (beginning) of a system, and it is represented by a black filled circle.
- Final state: It represents the final state (end) of a system. It is denoted by a filled circle present within a circle.
- Decision box: It is of diamond shape that represents the decisions to be made on the basis of an evaluated guard.



Notation of a State Machine Diagram

- Transition: A change of control from one state to another due to the occurrence of some event is termed as a transition. It is represented by an arrow labeled with an event due to which the change has ensued.
- State box: It depicts the conditions or circumstances of a particular object of a class at a specific point of time. A rectangle with round corners is used to represent the state box.



Types of State

The UML consist of three states:

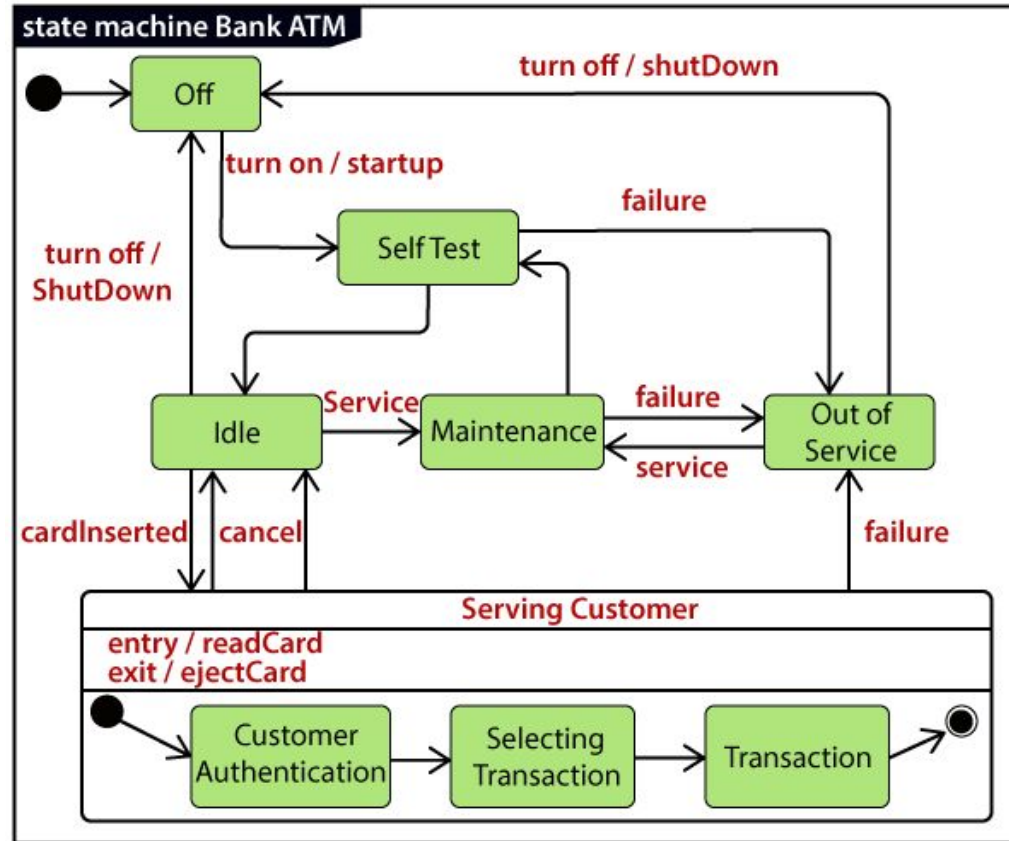
Simple state: It does not constitute any substructure.

Composite state: It consists of nested states (substates), such that it does not contain more than one initial state and one final state. It can be nested to any level.

Submachine state: The submachine state is semantically identical to the composite state, but it can be reused.



Types of State



Q & A



Thank
You!

