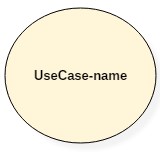
**1.Explain about use case diagram with appropriate example.**

# “Use Case Diagram”

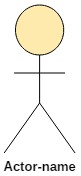
A use case diagram is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. The purpose of the use case diagram is to capture the dynamic aspect of the system. It captures the system’s functionality and the 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. It models how an external entity interacts with the system to make it work. Use case diagrams are responsible for visualizing the external things that interact with the part of the system. Use case diagrams are used to gather the requirements of a system including internal and external influences.

The components of use case diagram are explained below:

**Use-case:**

Use cases are used to represent high-level functionalities and how the user will handle the system. A use case represents a distinct functionality of a system, a component, a package, or a class. It is denoted by an oval shape with the name of a use case written inside the oval shape.

**Actor:**

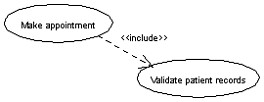
The actor is an entity that interacts with the system. A user is the best example of an actor. An actor is an entity that initiates the use case from outside the scope of a use case. It can be any element that can trigger an interaction with the use case. One actor can be associated with multiple use cases in the system.

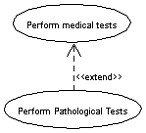
**System boundary:**

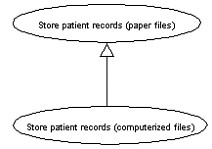
A system boundary defines the scope of what a system will be. A system cannot have infinite functionality. So, it follows that use cases also need to have definitive limits defined. It defines the limits of the system. The system boundary is shown as a rectangle spanning all the use cases in the system.

**Relationships in Use Cases:**

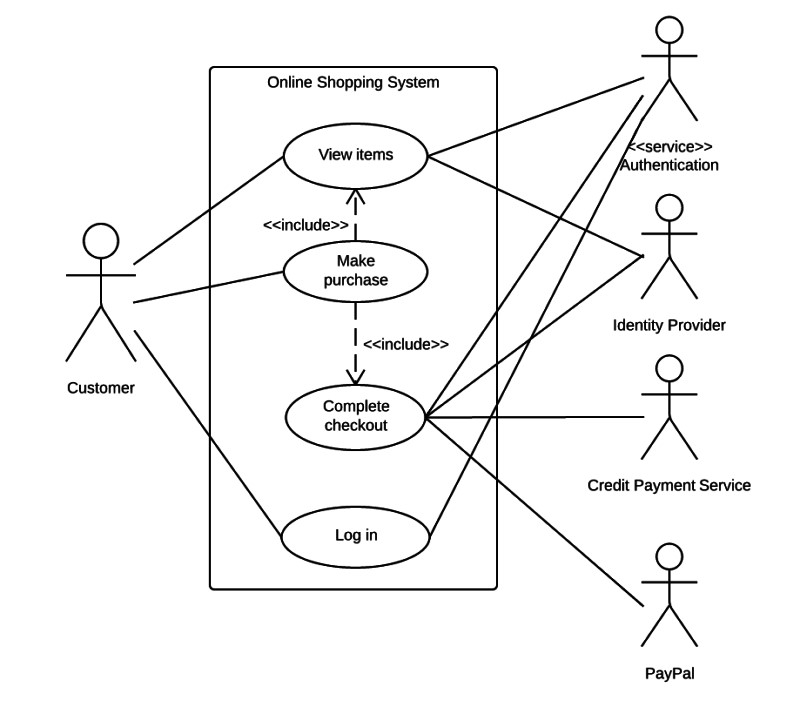
A relationship between two use cases is basically a dependency between the two use cases. Defining a relationship between two use cases is the decision of the modeler of the use case diagram. Use case relationships can be one of the following:

**Include:** When a use case is depicted as using the functionality of another use case in a diagram, this relationship between the use cases is named as an *include* relationship. An include relationship is depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the child use case and the parent use case is connected at the base of the arrow. The stereotype "<<include>>" identifies the relationship as an include relationship.

**Extend:** In an *extend* relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case. An extend relationship is depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<extend>>" identifies the relationship as an extend relationship.

**Generalizations:** A *generalization* relationship is also a parent-child relationship between use cases. The child use case in the generalization relationship has the underlying business process meaning, but is an enhancement of the parent use case. In a use case diagram, generalization is shown as a directed arrow with a triangle arrowhead. The child use case is connected at the base of the arrow. The tip of the arrow is connected to the parent use case.

Example: Online Shopping System



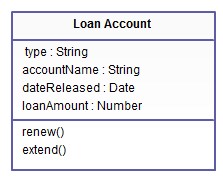
**2.Explain about class diagram with appropriate example.**

# “Class Diagram”

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. It shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram. It represents the static view of an application. The purpose of the class diagram are listed below:

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.

The following figure is an example of a simple class:



In the example, a class called “loan account” is depicted. Classes in class diagrams are represented by boxes that are partitioned into three:

* The top partition contains the name of the class.
* The middle part contains the class’s attributes. The attribute type is shown after the colon. Attributes map onto member variables (data members) in code.
* The bottom partition shows the possible operations that are associated with the class.

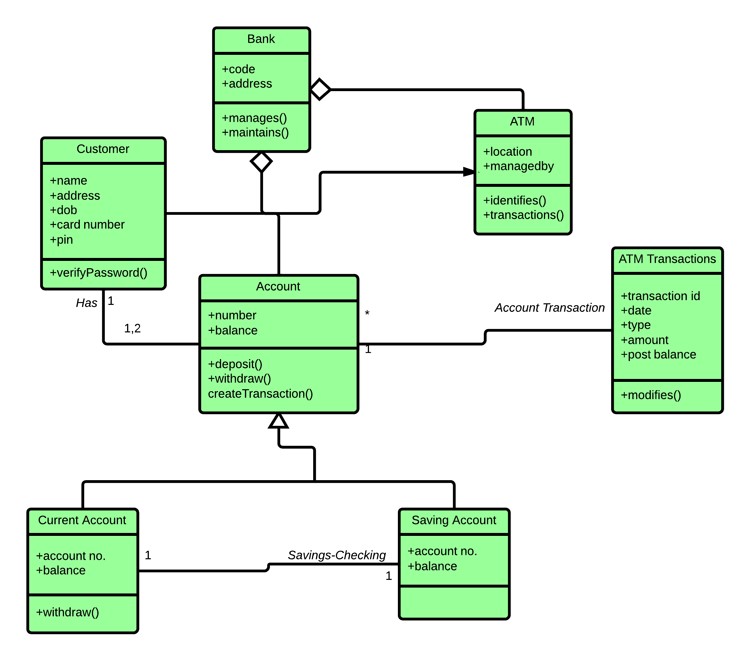
The return type of a method is shown after the colon at the end of the method signature. The return type of method parameters is shown after the colon following the parameter name. Operations map onto class methods in code. A UML class diagram is made up of:

* A set of classes and
* A set of relationships between classes.

For Example: ATM system

ATMs system is very simple as customers need to press some buttons to receive cash. However, there are multiple security layers that any ATM system needs to pass. This helps to prevent fraud and provide cash or need details to banking customers.

Below given is a UML Class Diagram example:



**3.Explain about the relationship in class diagram in detail.Write with example 1:1,1:M,M:1,M:M relationships.**

**“Relationship in the Class Diagram”**

Class diagram is one of the types of UML diagram which is used to represent the static diagram by mapping the structure of the systems using classes, attributes, relations and operations between the various objects. Class diagram has the various classes, each has three part, First partition contains Class name which is the name of the class or entity which is participated in the activity, Second partition contains class attributes shows the various properties of class, third partition contains class operations which shows various operations performed by the class, relationships shows the relation between two classes.

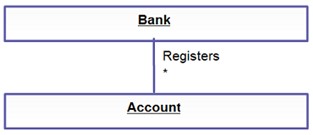
**Relationships**

In a class diagram, it is necessary that there exists a relationship between the classes. The similarity of various relationships often makes it difficult to understand it. Below are the relationships which exist in a class diagram.

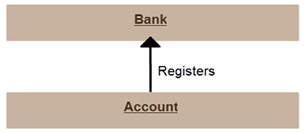
**a. Association**

Between two other classes in an association relationship, an association class forms a part of it. Additional information about the relationship could be obtained by attaching the association relationship with the association class. Various operations, attributes, etc., are present in the association class. Below diagram shows an association of bank and account.

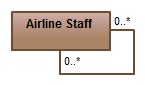
1. **Multiplicity**

The number of elements or cardinality could be defined by multiplicity. It is one of the most misunderstood relationships which describes the number of instances allowed for a particular element by providing an inclusive non-negative integers interval. It has both lower and upper bound. For example, a bank would have many accounts registered to it. Thus near the account class, a star sign is present.

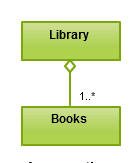
1. **Directed Association**

This is a one-directional relationship in a class diagram which ensures the flow of control from one to another classifier. The navigability is specified by one of the association ends. The relationship between two classifiers could be described by naming any association. The direction of navigation is indicated by an arrow. Below example shows an arrowhead relationship between the container and the contained.

1. **Reflexive Association**

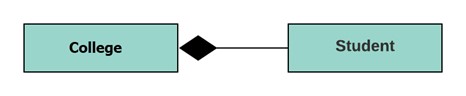
This occurs when a class may have multiple functions or responsibilities. For example, a staff member working in an airport may be a pilot, aviation engineer, a ticket dispatcher, a guard, or a maintenance crew member. If the maintenance crew member is managed by the aviation engineer there could be a managed by relationship in two instances of the same class.

**e**.**Aggregation**

Aggregation is a special type of association that models a whole- part relationship between aggregate and its parts.Aggregation refers to the formation of a particular class as a result of one class being aggregated or built as a collection. For example, the class “library” is made up of one or more books, among other materials. In aggregation, the contained classes are not strongly dependent on the lifecycle of the container. In the same example, books will remain so even when the library is dissolved. To show aggregation in a diagram, draw a line from the parent class to the child class with a diamond shape near the parent class.To show aggregation in a diagram, draw a line from the parent class to the child class with diamond shape near the parent class.

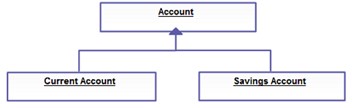
1. **Composition**

It is a form of an aggregation which represents the whole part relationship. Here, the part classifier lifetime is dependent on the whole classifier lifetime. It is a special type of aggregation which denotes strong ownership between two classes when one class is a part of another class. In a class, a strong life-cycle is represented by the composition relationship. There is usually a one direction flow of data here. It is generally indicated by a solid line.



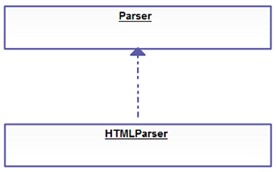
1. **Generalization**

In this kind of relationship, the child model is based on the parent model. The relationship is used to describe various use-case diagrams and ensures that the child class receives the properties present in the parent. The child model could reuse the attributes of the parent model with the help of the generalization relationship. Hence the distinct attributes need to be defined only in the child, rest it would inherit from the parent. There could be single parent, multiple children or multiple parents, single child characteristics in this relationship. There are no names in the generalization relationships. It is also known as the ‘is a’ relationship.



1. **Realization**

The behaviour of one model element is realized by the specified behaviour of another model element. This type of relationships doesn’t have any names.



**i.Dependency**

A dependency means the relation between two or more classes in which a change in one may force changes in the other. However, it will always create a weaker relationship. Dependency indicates that one class depends on another.

In the following example, Student has a dependency on College



**4.Explain about sequence diagram, activity diagram and state machine diagram with examples.**

**“Sequence Diagram”**

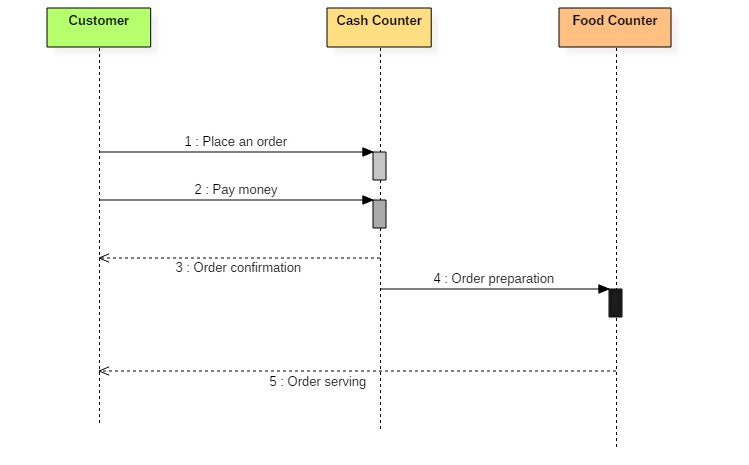
Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.Sequence Diagrams captures:

* + the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
  + high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)

Purpose of Sequence Diagram

* + Model high-level interaction between active objects in a system
  + Model the interaction between object instances within a collaboration that realizes a use case
  + Model the interaction between objects within a collaboration that realizes an operation
  + Either model generic interactions (showing all possible paths through the interaction) or specific instances of a interaction (showing just one path through the interaction)

Example: The following sequence diagram example represents Food ordering system:



Sequence diagram of Food ordering system

The ordered sequence of events in a given sequence diagram is as follows:

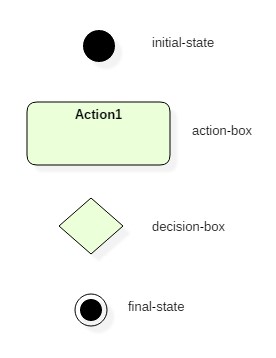
* 1. Place an order.
  2. Pay money to the cash counter.
  3. Order Confirmation.
  4. Order preparation.
  5. Order serving.

If one changes the order of the operations, then it may result in crashing the program. It can also lead to generating incorrect or buggy results. Each sequence in the above-given sequence diagram is denoted using a different type of message. One cannot use the same type of message to denote all the interactions in the diagram because it creates complications in the system.

**“Activity diagram”**

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. They can also describe the steps in a use case diagram. It shows the flow from activity to activity within the system. The activity shows a set of activities, the sequential or branching flow from activity to activity, and objects that act and are acted upon. It shows what activities can be done in parallel, and any alternate paths through the flow. It contains activities, transactions between the activities, decision points and synchronization bars. Activity diagrams emphasize the flow of control among objects.

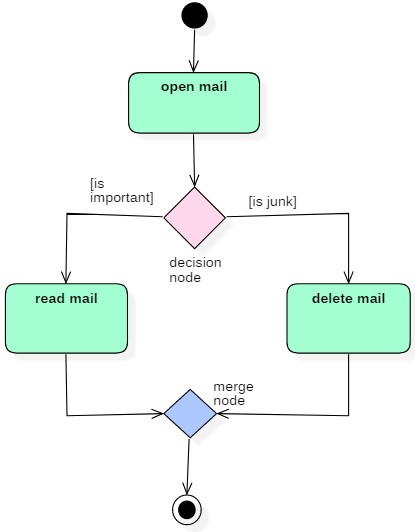
Activity Transition Decision Synchronization Bar

Activity diagrams symbols can be generated by using the following notations:

* Initial states: The starting stage before an activity takes place is depicted as the initial state
* Final states: The state which the system reaches when a specific process ends is known as a Final State
* State or an activity box: An action state represents the non-interruptible action of objects. You can draw an action state in SmartDraw using a rectangle with rounded corners.
* Decision box: It is a diamond shape box which represents a decision with alternate paths. It represents the flow of control.

To draw an activity diagram, one must understand and explore the entire system. All the elements and entities that are going to be used inside the diagram must be known by the user. The central concept which is nothing but an activity must be clear to the user. After analyzing all activities, these activities should be explored to find various constraints that are applied to activities. If there is such a constraint, then it should be noted before developing an activity diagram.

**Example of Activity Diagram**

Let us consider mail processing activity as a sample for Activity Diagram. Following diagram

represents activity for processing emails.

In this activity diagram, three activities are specified. When the mail checking process begins user checks if mail is important or junk. Two guard conditions [is essential] and [is junk] decides the flow of execution of a process. After performing the activity, finally, the process is terminated at termination node.

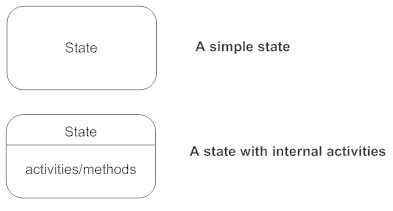
# “State Machine Diagram”

State machine diagram is a behaviour diagram which shows discrete behaviour of a part of designed system through finite state transitions. State machine diagrams can also be used to express the usage protocol of part of a system. It shows the behaviour of classes in response to external stimuli. It describes the behaviour of a single object in response to a series of events in a system. Sometimes it's also known as a Harel state chart or a state diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system. A state machine diagram models the behaviour of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events.

Basic State Machine Diagram Symbols and Notations:

**States:**

States represent situations during the life of an object. You can easily illustrate a state in SmartDraw by using a rectangle with rounded corners.



**Transition:**

A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it. A state can have a transition that points back to itself.



**Initial state:**

A filled circle followed by an arrow represents the object's initial state.



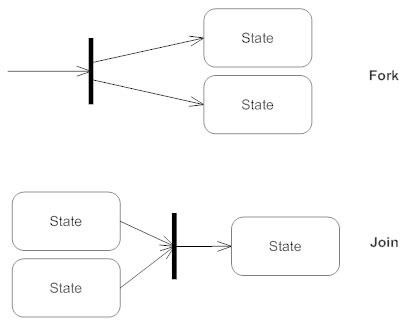
**Final state:**

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



**Synchronization and Splitting of Control:**

A short heavy bar with two transitions entering it represents a synchronization of control. The first bar is often called a fork where a single transition splits into concurrent multiple transitions. The second bar is called a join, where the concurrent transitions reduce back to one.



**For Example:** The following state machine diagram shows the states that a door goes through during its lifetime.

