

**Figure 1.5** Pyramid of UML diagrams.

**Table 1.1** List of UML Diagrams

S. No	UML Diagram	Purpose of the UML Diagram
1	UML Usecase Diagram	It focuses on the identification of functional requirements of the system under consideration.
2	UML Activity Diagram	It focuses on sequential and parallel activities involved in each functional requirement of the system.
3	UML Class Diagram	It describes the structure of the system in terms of classes and objects.
4	UML Sequence Diagram	It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality.
5	UML Collaboration Diagram or UML Communication Diagram	It shows interactions between objects using sequenced messages in a free-form arrangement.
6	UML State Machine Diagram	It describes the life of an object using three main elements: states of an object, transitions between states and events that trigger the transitions.

(Continued)

**Table 1.1 (Continued) List of UML Diagrams**

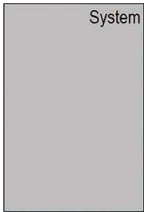
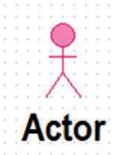
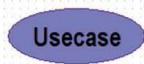
S. No	UML Diagram	Purpose of the UML Diagram
7	UML Component Diagram	The purpose of a component diagram is to show the relationship between different components in a system.
8	UML Deployment Diagram	Deployment diagrams are used for describing the hardware components, where software components are deployed. The purpose of deployment diagrams can be described as: <ul style="list-style-type: none"> <li>• Visualize the hardware topology of a system.</li> <li>• Describe the hardware components used to deploy software components.</li> <li>• Describe the runtime processing nodes.</li> </ul>

### 1.1.2 UML Usecase Diagram

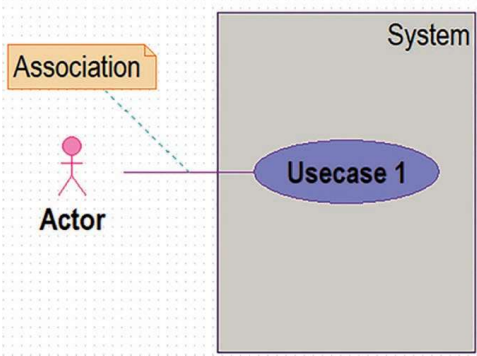
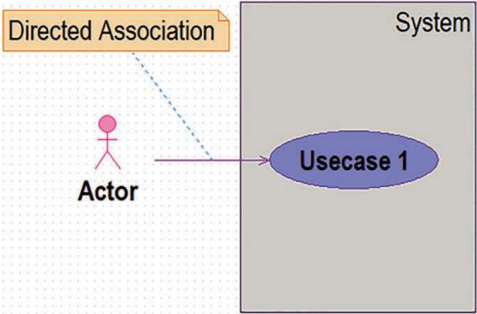
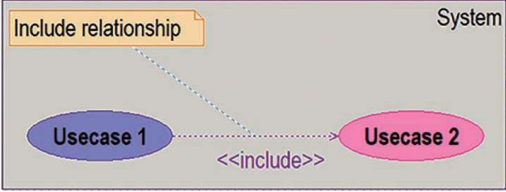
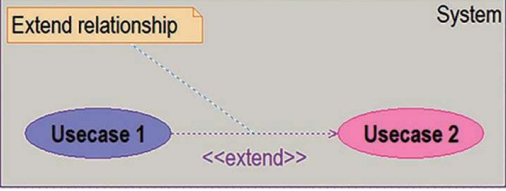
Table 1.2 contains the details of components of the UML Usecase Diagram.

Table 1.3 contains the various usecase relationships.

**Table 1.2 Components of UML Usecase Diagram**

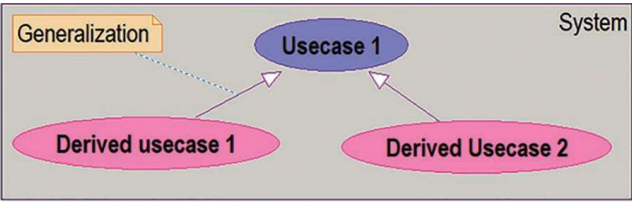
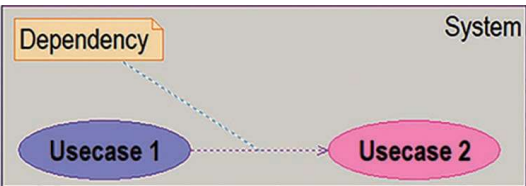
S. No	Name of the Component	UML Notation	Purpose
1	System Boundary		<ul style="list-style-type: none"> <li>• It represents the scope of the system.</li> <li>• It encapsulates the complete set of functionalities of the system.</li> </ul>
2	Actors		<ul style="list-style-type: none"> <li>• The users that interact with a system.</li> <li>• An actor can be a person, an organization or an outside system that interacts with your application or system.</li> <li>• An actor in a usecase diagram interacts with a usecase.</li> <li>• It is up to the developer to consider what actors make an impact on the functionality that they want to model.</li> </ul>
3	Usecases		<ul style="list-style-type: none"> <li>• A usecase is a visual representation of a distinct business functionality in a system.</li> <li>• It ensures that the business process is discrete in nature.</li> <li>• List the discrete business functions in given problem statement.</li> <li>• Identifying usecases is a discovery.</li> </ul>

**Table 1.3 Usecase Relationships**

S. No	Usecase Relationship	UML Notation and Its Functionality
1	Association	 <ul style="list-style-type: none"> <li>• Association represents the relationship between an actor and a usecase.</li> </ul>
2	Directed Association	 <ul style="list-style-type: none"> <li>• Directed Association represents the one-way relationship where an actor takes the responsibility of influencing a usecase.</li> </ul>
3	Include	 <ul style="list-style-type: none"> <li>• Include relationship represents the situation where one usecase includes the functionality of one other usecase.</li> </ul>
4	Extend	 <ul style="list-style-type: none"> <li>• Extend relationship between any two usecases implies a meaningful relationship that the extended usecase adds up additional behaviour towards the existing functionality of the base usecase.</li> </ul>

(Continued)

**Table 1.3 (Continued) Usecase Relationships**

S. No	Usecase Relationship	UML Notation and Its Functionality
5	Generalization	 <ul style="list-style-type: none"> <li>Generalization is the relationship between a parent usecase and one or more child usecases.</li> </ul>
6	Dependency	 <ul style="list-style-type: none"> <li>Dependency defines the relationship in which the existence of one usecase is dependent on the existence of another usecase.</li> </ul>

### 1.1.2.1 Usecase Template



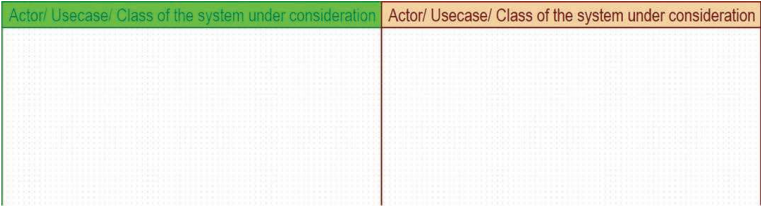

Every usecase is explained in detail through a standard template called as usecase template.

<i>Usecase ID</i> – represents a unique ID for a usecase.
<i>Usecase Name</i> – represents a meaningful name of the usecase.
<i>Actors Involved</i> – actors interacting with the usecase.
<i>Description</i> – the brief explanation of the functionality of the usecase.
<i>Preconditions</i> – state of the system before executing this functionality of the usecase.
<i>Main Flow</i> – description of how the functionality is implemented.
<i>Post Conditions</i> – state of the system after executing this functionality of the usecase.
<i>Alternate Flow</i> – other possibilities available.

### 1.1.3 UML Activity Diagram


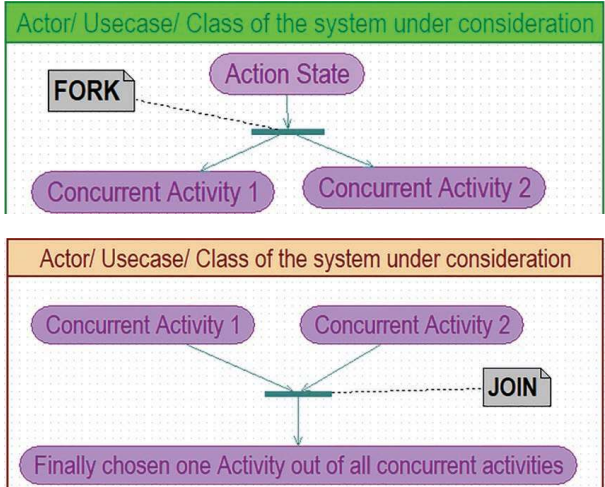
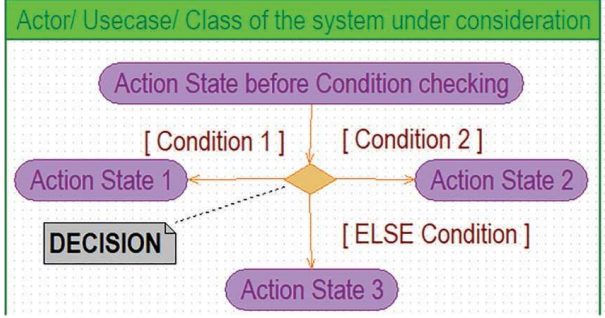
Table 1.4 contains the details of components of the UML Activity Diagram.

**Table 1.4 Components of UML Activity Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Initial state	 <ul style="list-style-type: none"> <li>It represents the start state of the system under consideration.</li> </ul>
2	Final state	 <ul style="list-style-type: none"> <li>It represents the termination state of the system under consideration.</li> </ul>
3	Swimlanes	<ul style="list-style-type: none"> <li>A swimlane contains two partitions namely a top partition representing entities like an actor/a usecase/a class, etc. and the second partition focuses on the set of activities involved.</li> <li>Swimlane is of two types namely vertical swimlane and horizontal swimlane.</li> <li>Vertical swimlane – represents parallel activities of a particular scenario of the system under consideration.</li> </ul>  <ul style="list-style-type: none"> <li>Horizontal swimlane – represents sequential activities of a particular scenario of the system under consideration.</li> </ul>
4	Action state	<ul style="list-style-type: none"> <li>An action state represents an operation or a business activity or a process.</li> </ul> 

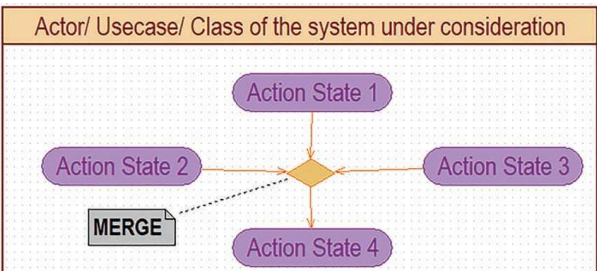




(Continued)

**Table 1.4 (Continued) Components of UML Activity Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
5	Object	<ul style="list-style-type: none"> <li>An entity that carries data between any two action states.</li> </ul> 
6	Synchronization	<ul style="list-style-type: none"> <li>Synchronization represents two or more activities happening at same time or rate.</li> <li>It is of two types namely: Fork – divides a single activity flow into two or more concurrent activities and Join – combines two or more concurrent activities into a single flow by ensuring that only one of the activities at a time.</li> </ul> 
7	Decision	<ul style="list-style-type: none"> <li>A decision node has one input or two or more outputs depending on the condition for which it is designed.</li> <li>Each output flow has a condition attached to it.</li> <li>If a condition is met, the flow proceeds along with the appropriate output.</li> <li>An 'else' output can be defined along which the flow can proceed if no other condition is met.</li> </ul> 

(Continued)



**Table 1.4 (Continued) Components of UML Activity Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
8	Merge	<ul style="list-style-type: none"> <li>The purpose is to merge the input flows.</li> <li>The inputs are not synchronized; if a flow reaches such a node it proceeds at the output without waiting for the arrival of other flows.</li> </ul>  <p>The diagram shows a rectangular frame labeled 'Actor/ Usecase/ Class of the system under consideration'. Inside, four rounded rectangular nodes labeled 'Action State 1', 'Action State 2', 'Action State 3', and 'Action State 4' are arranged around a central yellow diamond-shaped merge node. Arrows point from each of the four action states to the diamond. A label 'MERGE' in a box points to the diamond with a dashed line.</p>
9	Flow Final	<ul style="list-style-type: none"> <li>It represents abnormal termination of a path in an activity diagram that is not considered as a part of the system under development.</li> </ul>  <p>The diagram shows a red circle with a red 'X' inside, representing a flow final node.</p>
10	Transition	<ul style="list-style-type: none"> <li>Transition are arrows that represent a movement from the source activity state to the target activity state which gets triggered by the completion of the activity of the source activity state.</li> </ul>  <p>The diagram shows two rounded rectangular nodes labeled 'Source Activity State' and 'Target Activity State' connected by a solid arrow pointing from the source to the target.</p>
11	Self-Transition	<ul style="list-style-type: none"> <li>It represents the internal transition to an actions state itself.</li> </ul>  <p>The diagram shows a rounded rectangular node labeled 'Activity with Self Transition'. A solid arrow originates from the top of the node, loops back, and points to the top of the same node, representing a self-transition.</p>
12	Signal Send State	<ul style="list-style-type: none"> <li>Signals represent how activities can be influenced externally by the system.</li> <li>They usually appear in pairs of sent and received signals.</li> <li>Signal Send State represents sending signal action outside of the activity.</li> <li>The signal sends state does not wait for any responses from the receiver of the signal.</li> <li>It ends itself and passes the execution control to the next action.</li> <li>A Signal Send State takes its notation as a convex pentagon.</li> </ul>  <p>The diagram shows a convex pentagon shape labeled 'Signal Send State'.</p>

(Continued)



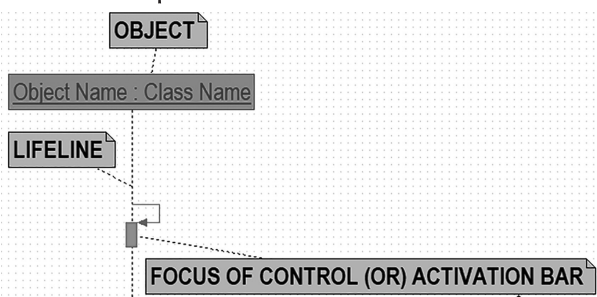
**Table 1.4 (Continued) Components of UML Activity Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
13	Signal Accept State	<ul style="list-style-type: none"> <li>• An action state whose trigger is a signal event is informally called Signal Accept State.</li> <li>• It corresponds to Signal Send State.</li> <li>• Signal Accept State with incoming edges means that the action starts after the previous action state completes.</li> <li>• Signal Accept State with no incoming edges remains enabled after it accepts an event.</li> <li>• It does not terminate after accepting an event and outputting a value, but continues to wait for other events.</li> </ul> 
14	Subactivity	<ul style="list-style-type: none"> <li>• A subactivity is an action state that can become as another main activity scenario in future.</li> </ul> 

### 1.1.4 UML Sequence Diagram

Table 1.5 contains the details of components of the UML Sequence Diagram.



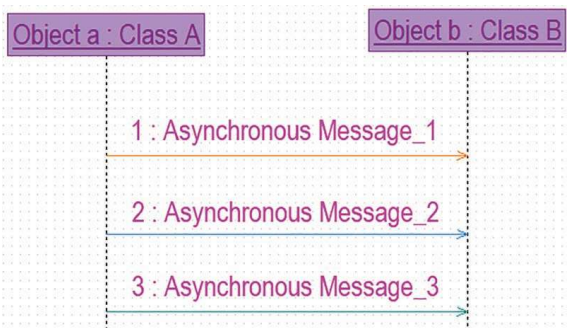
**Table 1.5 Components of UML Sequence Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Object	<ul style="list-style-type: none"> <li>• It represents an entity of the system that gets involved in interaction with other entities of the system via messages.</li> <li>• An object has three related information namely object name, lifeline and focus of control.</li> <li>• The standard syntax of naming an object is Object_name: Class_name.</li> <li>• Lifeline represents the complete existence of a particular object involved in a scenario.</li> <li>• Focus of control represents the active session of the object.</li> </ul> 

(Continued)

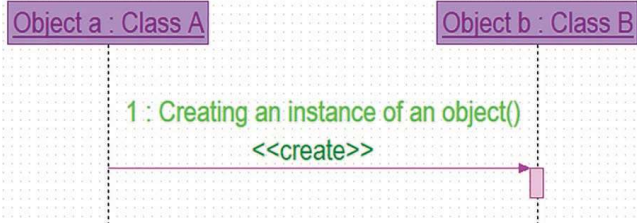
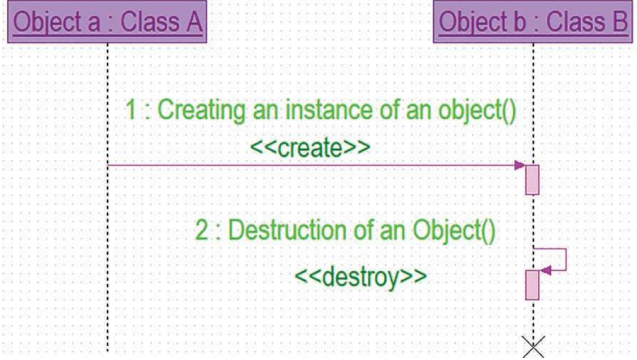
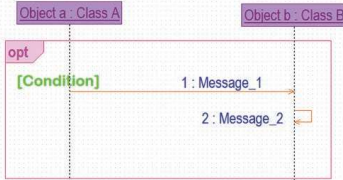


**Table 1.5 (Continued) Components of UML Sequence Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
2	Message & its types	<ul style="list-style-type: none"> <li>The communication between objects of any scenario is done through the concept of message passing with the help of transition component.</li> <li>Messages are numbered over the transition arrows.</li> <li>Syntax of message is Message_no: Message()</li> <li>There are five types of messages applicable to any transition between any two objects in a sequence diagram.</li> </ul>
2.1	Synchronous Message	<ul style="list-style-type: none"> <li>The messages sent from the sender are based on wait semantics.</li> <li>Wait semantics is a situation where the sender needs an acknowledgement of each message sent from the receiver before transmitting the next successive message.</li> <li>The notation of a synchronous message is a solid line with a shaded arrow head.</li> </ul>  <pre> sequenceDiagram     participant A as Object a : Class A     participant B as Object b : Class B     A-&gt;&gt;B: 1 : Synchronous Message()     activate B     B--&gt;&gt;A:      deactivate B   </pre>
2.2	Return Message	<ul style="list-style-type: none"> <li>It represents a reply message for a synchronous message.</li> </ul>  <pre> sequenceDiagram     participant A as Object a : Class A     participant B as Object b : Class B     A-&gt;&gt;B: 1 : Synchronous Message()     activate B     B--&gt;&gt;A: 2 : Reply Message     deactivate B   </pre>
2.3	Asynchronous Message	<ul style="list-style-type: none"> <li>These type of messages are sent by the sender to the receiver which does not require an acknowledgement from the receiver.</li> <li>This feature enables the sender to send any number of messages to the receiver.</li> </ul>  <pre> sequenceDiagram     participant A as Object a : Class A     participant B as Object b : Class B     A-&gt;&gt;B: 1 : Asynchronous Message_1     A-&gt;&gt;B: 2 : Asynchronous Message_2     A-&gt;&gt;B: 3 : Asynchronous Message_3   </pre>

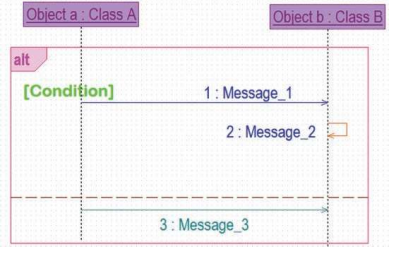
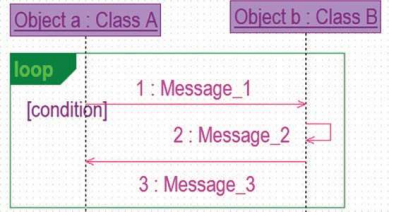
(Continued)

**Table 1.5 (Continued) Components of UML Sequence Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
2.4	Create Message	<ul style="list-style-type: none"> <li>Create message represents the instantiation of objects in a scenario.</li> <li>It includes the stereotype called &lt;&lt;create&gt;&gt;.</li> </ul> 
2.5	Destroy Message	<ul style="list-style-type: none"> <li>Destroy message represents the destruction of the lifecycle of an object in a scenario.</li> <li>It includes the stereotype called &lt;&lt;destroy&gt;&gt;.</li> <li>Destruction is pictorially represented by a large X pointing at the end of the lifeline of an object.</li> </ul> 
3	Fragments	<ul style="list-style-type: none"> <li>Fragments represent a set of conditional messages or looping messages in a scenario.</li> <li>The most vital fragments are opt, alt, loop.</li> <li>Each fragment is represented as a rectangular box encapsulating those set of messages with a label in the top left corner representing the fragment type.</li> </ul>
3.1	Opt	<ul style="list-style-type: none"> <li>This fragment represents a simple IF scenario.</li> <li>Messages get executed only if the condition defined in the opt fragment remains true.</li> <li>In case of condition does not get satisfied, the control jumps out of the fragment.</li> <li>Conditions are technically called as Guard.</li> </ul> 

(Continued)



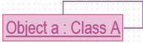

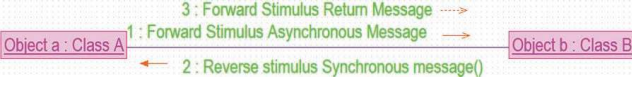
**Table 1.5 (Continued) Components of UML Sequence Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
3.2	Alt	<ul style="list-style-type: none"> <li>This fragment represents IF ELSE scenario.</li> <li>This fragment has two partitions inside the rectangular area.</li> <li>Messages in the first compartment get executed only if the condition defined in the first partition of the alt fragment remains true.</li> <li>Otherwise, the messages in the second partition of the alt fragment get executed.</li> </ul> 
3.3	Loop	<ul style="list-style-type: none"> <li>This fragment represents those set of messages that gets executed repeatedly until the condition defined in the loop fragment remains true.</li> <li>This fragment has two partitions inside the rectangular area.</li> <li>Messages in the first compartment get executed only if the condition defined in the first partition of the alt fragment remains true.</li> </ul> 
4	Nesting of fragments	<ul style="list-style-type: none"> <li>Depending on the demand of multiple conditions to get verified or repeated execution of the set of nested conditions, this concept of nesting can be applied to different types of fragments.</li> </ul>

### 1.1.5 UML Collaboration Diagram

Majority of the components are common between UML Sequence Diagram and UML Collaboration Diagram. Table 1.6 contains the details of the unique components of the UML Collaboration Diagram.





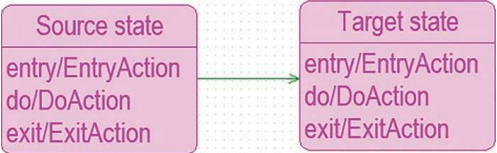
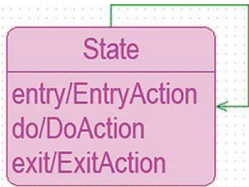
**Table 1.6 Unique Components of UML Collaboration Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Object	<ul style="list-style-type: none"> <li>It represents an entity of the system that gets involved in interaction with other entities of the system via messages.</li> <li>In the case of UML Collaboration diagrams, an object does not have any lifeline or focus of control.</li> </ul> 
2	Link	<ul style="list-style-type: none"> <li>Link enables the communication between any two objects.</li> <li>It is represented as a solid line without any arrow heads, so that it represents bidirectional communication.</li> <li>Any number of bidirectional messages between two objects can be communicated via a same link.</li> </ul> 
3	Self Link	<ul style="list-style-type: none"> <li>A self-link connects an object to itself.</li> </ul> 
4	Forward stimulus	<ul style="list-style-type: none"> <li>It represents messages from sender (first object) to receiver (second object).</li> <li>It can be any of the five types of messages as discussed in UML Sequence Diagram like Synchronous messages, Return messages, Asynchronous messages, Create messages and Destroy messages.</li> </ul> 
5	Reverse stimulus	<ul style="list-style-type: none"> <li>It represents response messages from the receiver (second object) to sender (first object).</li> <li>It can be any of the five types of messages as discussed in UML Sequence Diagram like Synchronous messages, Return messages, Asynchronous messages, Create messages and Destroy messages.</li> </ul> 

### 1.1.6 UML State Chart Diagram

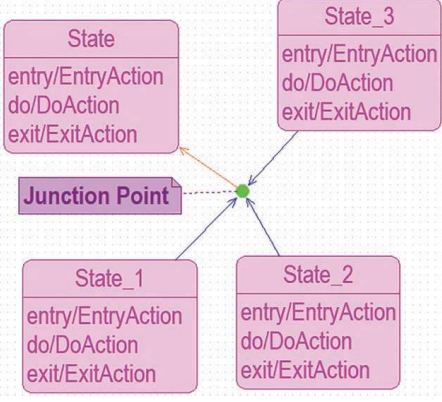
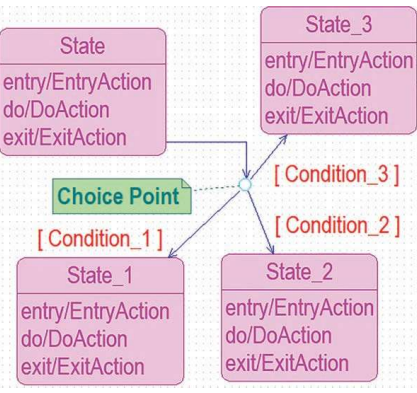


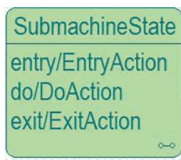
Table 1.7 contains the details of components of the UML State Machine Diagram.

**Table 1.7 Components of UML State Machine Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Initial State	<ul style="list-style-type: none"> <li>It represents the start state of a system under consideration.</li> </ul> 
2	Final State	<ul style="list-style-type: none"> <li>It represents the end state of a system under consideration.</li> </ul> 
3	Flow final	<ul style="list-style-type: none"> <li>It represents abnormal termination of a path in a state machine diagram which is not considered as a part of the system under development.</li> </ul> 
4	State	<ul style="list-style-type: none"> <li>It represents the state of the object in the system under consideration.</li> <li>It is described using the following attributes namely:               <ol style="list-style-type: none"> <li>Name – Name of the object.</li> <li>Entry – Action to be done before to enter the system.</li> <li>Do – Action to be done in that state.</li> <li>Exit – Action to be required to exit the system.</li> </ol> </li> </ul> 
5	Transition	<ul style="list-style-type: none"> <li>Transition are arrows that represent a movement from one state to the other state which gets triggered by the completion of the activity of the source state.</li> </ul> 
6	Self-Transition	<ul style="list-style-type: none"> <li>It represents internal transition to a state itself.</li> </ul> 
7	Junction Point	<ul style="list-style-type: none"> <li>The purpose is to merge the input flows from different states</li> <li>The inputs are not synchronized; if a flow reaches such a state it proceeds at the output without waiting for the arrival of other flows from other states.</li> </ul>

(Continued)

**Table 1.7 (Continued) Components of UML State Machine Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
		
8	Choice Point	<ul style="list-style-type: none"> <li>• A Choice point has one input or two or more outputs depending on the condition for which it is designed.</li> <li>• Each output flow has a condition attached to it.</li> </ul> <p>If a condition is met, the flow proceeds along with the appropriate output.</p> 
9	Shallow History	<ul style="list-style-type: none"> <li>• Shallow history of a state is a reference to the most recently visited state on the same hierarchy level.</li> </ul> 
10	Deep History	<ul style="list-style-type: none"> <li>• Deep history of a state is a reference to the most recently visited simple state.</li> </ul> 
11	Submachine state	<ul style="list-style-type: none"> <li>• A submachine is a state that can become as another main state chart diagram scenario in future.</li> </ul> 



### 1.1.7 UML Class Diagram

Table 1.8 contains the details of components of the UML Class Diagram.




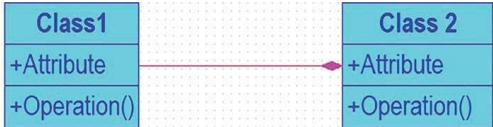
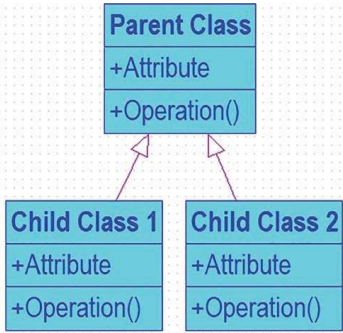
**Table 1.8 Components of UML Class Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component											
1	Class	<div><ul style="list-style-type: none"><li>Each class has three compartments namely:<ul style="list-style-type: none"><li>i. Top compartment represents the name of class,</li><li>ii. Middle compartment represents structure of the class (attributes) and</li><li>iii. Bottom compartment represents behaviour of the class (operations)</li></ul></li></ul></div> <div><table><tr><td>Class_Name</td></tr><tr><td>+Attribute</td></tr><tr><td>+Operation()</td></tr></table></div> <div><ul style="list-style-type: none"><li>Attribute represents properties that describe the state of the object.</li><li>Syntax: Visibility Attribute_Name : Data_Type</li><li>Examples of data types are: Int, Char, Float, String, Date, Time, Money, Dollars, Colour, City, Country, Postal Code, Address, Boolean, etc.</li></ul></div> <div><table><tr><th>Visibility Notation</th><th>Access Specifier</th></tr><tr><td>+</td><td>Public</td></tr><tr><td>#</td><td>Protected</td></tr><tr><td>-</td><td>Private</td></tr></table></div> <div><ul style="list-style-type: none"><li>Derived attributes are those attributes that are calculated or derived from other attributes denoted by placing slash (/) before name.</li><li>Syntax:/Visibility Attribute_Name: Data_Type</li><li>Operations represents the actions or functions that a class can perform.</li><li>Syntax: Visibility Operation_Name (Input Arg): Data_Type</li><li>Input arguments take the syntax of an attribute.</li><li>Methods that don't return a value (i.e. void methods) should give a return type of void.</li></ul></div>	Class_Name	+Attribute	+Operation()	Visibility Notation	Access Specifier	+	Public	#	Protected	-	Private
Class_Name													
+Attribute													
+Operation()													
Visibility Notation	Access Specifier												
+	Public												
#	Protected												
-	Private												
2	Class Relationships	<ul style="list-style-type: none"><li>There are seven types of class relationship namely association, directed association, dependency, aggregation, composition, generalization, realization.</li></ul>											
2.1	Association	<div><ul style="list-style-type: none"><li>When two classes communicate with each other, an association is used to connect those two classes.</li></ul></div> <div><table><tr><td>Class1</td><td rowspan="3">Association Name</td><td>Class 2</td></tr><tr><td>+Attribute</td><td>+Attribute</td></tr><tr><td>+Operation()</td><td>+Operation()</td></tr></table></div>	Class1	Association Name	Class 2	+Attribute	+Attribute	+Operation()	+Operation()				
Class1	Association Name	Class 2											
+Attribute		+Attribute											
+Operation()		+Operation()											

(Continued)



**Table 1.8 (Continued) Components of UML Class Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
2.2	Directed Association	<ul style="list-style-type: none"> <li>When two classes get connected in such way by giving priority to one-way communication, they are connected by directed association.</li> </ul> 
2.3	Dependency	<ul style="list-style-type: none"> <li>When the existence of one class is dependent on the existence of another class, then they are connected using dependency.</li> </ul> 
2.4	Aggregation	<ul style="list-style-type: none"> <li>It represents a 'part of' relationship.</li> <li>Many instances of Class1 can be associated with Class2.</li> <li>Aggregation implies a relationship where the child class can exist independently of the parent class.</li> </ul> 
2.5	Composition	<ul style="list-style-type: none"> <li>It represents a 'whole' relationship.</li> <li>Composition implies a relationship where the child class cannot exist independent of the parent class.</li> </ul> 
2.6	Generalization	<ul style="list-style-type: none"> <li>Generalization represents the concept of inheritance.</li> <li>It represents the relationship between parent class and its child classes.</li> </ul> 

(Continued)


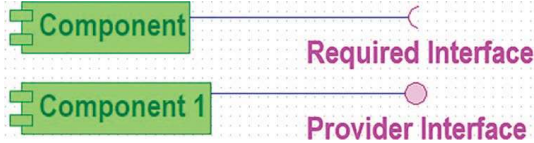
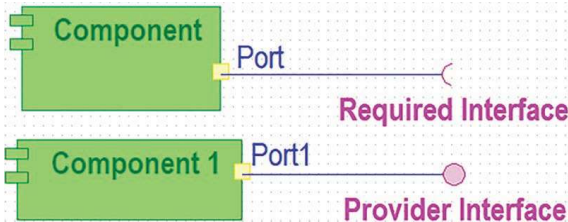
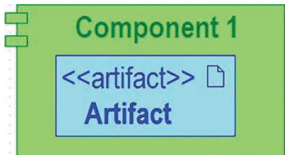

**Table 1.8 (Continued) Components of UML Class Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
2.7	Interface & role of Realization	<ul style="list-style-type: none"> <li>• Interfaces can be of two types namely required interfaces and provider interfaces.</li> <li>• Provider interface describes the functionality offered by a class.</li> <li>• Required interfaces describe the functionality needed by another class.</li> <li>• Provided interface is the interface implemented by a class, i.e a class implements an interface.</li> <li>• The required interface would be any use of an interface by a component, i.e if a class defines a method that has the interface as a parameter.</li> </ul> <p><b>Required Interface (Dependency between the class and interface)</b></p> <p>Class 1 (with +Attribute and +Operation()) is connected to Interface 1 (with a hollow semi-circle) by a dashed line.</p> <p>Class 2 (with +Attribute and +Operation()) is connected to Interface 2 (with a solid circle) by a solid line.</p> <p><b>Provider Interface (Realization between the class and interface)</b></p>
3	Association Class	<ul style="list-style-type: none"> <li>• An association class is an association that is also a class.</li> <li>• It not only connects a set of classifiers but also defines a set of features that belong to the relationship itself and not any of the classifiers.</li> </ul> <p>Class 1 (with +Attribute and +Operation()) and Class 2 (with +Attribute and +Operation()) are connected by a solid line. An Association Class (with +Attribute and +Operation()) is connected to this association line by a dashed line.</p>

### 1.1.8 UML Component Diagram

Table 1.9 contains the details of components of the UML Component Diagram.

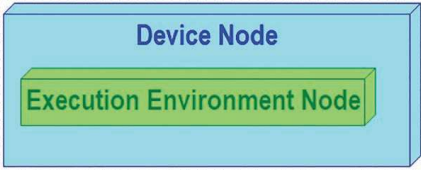
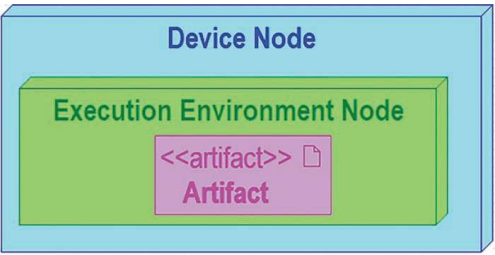
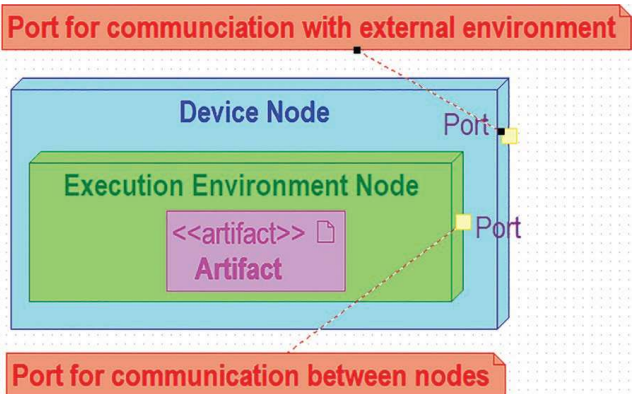
**Table 1.9 Components of UML Component Diagram**

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Component	<ul style="list-style-type: none"> <li>It is a modular part of a system that encapsulates its contents.</li> <li>They are the logical elements of a system that plays an essential role during the execution of a system.</li> <li>A component is a replaceable and executable piece of a system.</li> </ul> 
2	Interface	<ul style="list-style-type: none"> <li>There are two types of component interfaces, namely provider interface and required interface.</li> <li>Provider interface describes the functionality offered by a component.</li> <li>Required interfaces describe the functionality needed by another component.</li> </ul> 
3	Port	<ul style="list-style-type: none"> <li>A port enables better communication between an interface and a component.</li> </ul> 
4	Artifact	<ul style="list-style-type: none"> <li>Scripting files like PHP, database files, configuration files, rar files, executable files.</li> </ul> 
5	Association	<ul style="list-style-type: none"> <li>It represents the two-way communication between any two components involved in a relationship.</li> </ul> 

### 1.1.9 UML Deployment Diagram

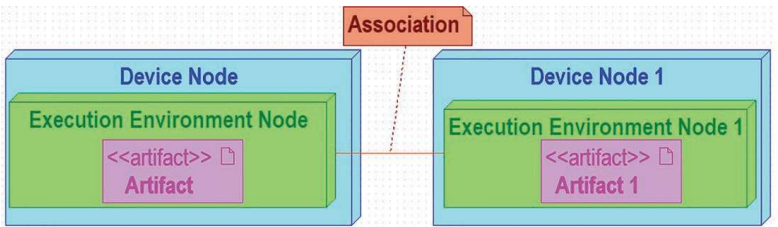
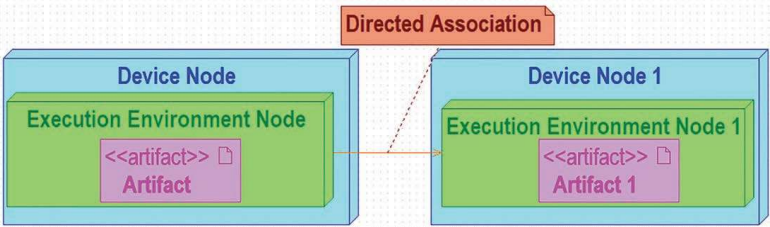
Table 1.10 contains the details of components of the UML Deployment Diagram.

**Table 1.10** Components of UML Deployment Diagram

S. No	Name of the Component	UML Notation and the Purpose of the Component
1	Node	<ul style="list-style-type: none"> <li>Any computational resource with memory and processing capability.</li> <li>There are two types of nodes namely Device node and Execution Environment Node.</li> <li>A Device node represents a hardware device like Application server, workstations, mobile device, embedded device, etc.</li> <li>An Execution Environment node represents a software or a program like Operating system, workflow engine, browser, Container, web server, database system, etc.</li> </ul> 
2	Artifact	<ul style="list-style-type: none"> <li>Scripting files like PHP, database files, configuration files, rar files, executable files.</li> </ul> 
3	Port	<ul style="list-style-type: none"> <li>Port enables the communication between a device node and an execution environment node.</li> <li>It also enables the communication between a node and an external environment.</li> </ul> 

(Continued)

**Table 1.10 (Continued)** Components of UML Deployment Diagram

S. No	Name of the Component	UML Notation and the Purpose of the Component
4	Association	<ul style="list-style-type: none"> <li>It represents the two-way communication between any two nodes involved in a relationship.</li> </ul> 
5	Directed Association	<ul style="list-style-type: none"> <li>It represents the one-way communication between any two nodes involved in a relationship.</li> </ul> 
6	Dependency	<ul style="list-style-type: none"> <li>It represents how an existence of a node is dependent on another node involved in a relationship.</li> </ul> 