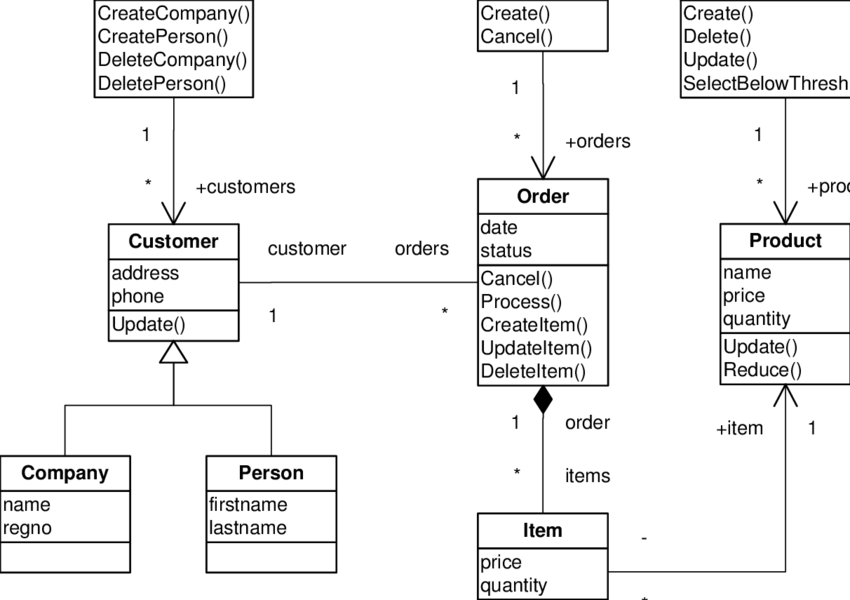


1. Purpose of Class Diagrams

The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. It is one of the most popular UML diagrams. Following are the purpose of class diagrams given below:

1. It analyses and designs a static view of an application.
2. It describes the major responsibilities of a system.
3. It is a base for component and deployment diagrams.
4. It incorporates forward and reverse engineering.



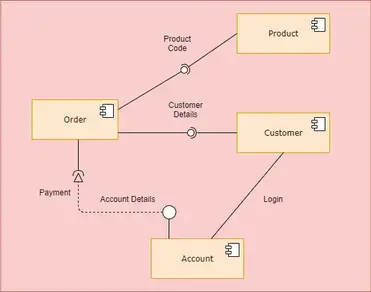
## Purpose of a Component Diagram

Since it is a special kind of a UML diagram, it holds distinct purposes. It describes all the individual components that are used to make the functionalities, but not the functionalities of the system. It visualizes the physical components inside the system. The components can be a library, packages, files, etc.

The component diagram also describes the static view of a system, which includes the organization of components at a particular instant. The collection of component diagrams represents a whole system.

The main purpose of the component diagram are enlisted below:

1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.



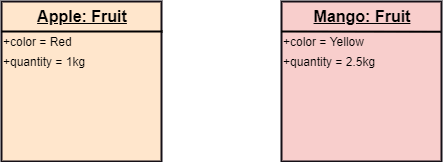
3.Purpose of Object Diagram

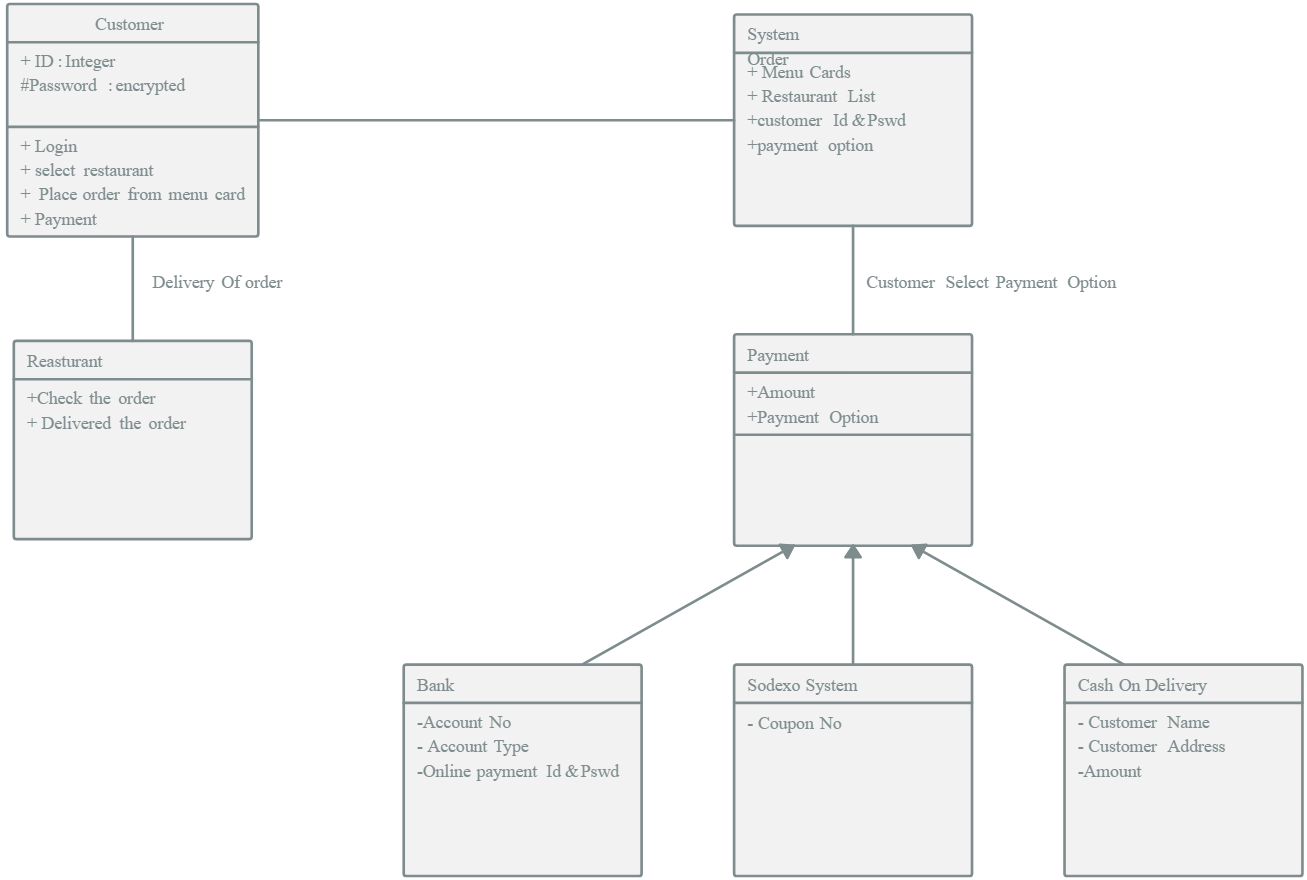
The object diagram holds the same purpose as that of a class diagram. The class diagram provides an abstract view which comprises of classes and their relationships, whereas the object diagram represents an instance at a particular point of time.

The object diagram is actually similar to the concrete (actual) system behavior. The main purpose is to depict a static view of a system.

Following are the purposes enlisted below:

* It is used to perform forward and reverse engineering.
* It is used to understand object behavior and their relationships practically.
* It is used to get a static view of a system.
* It is used to represent an instance of a system.





**4. WHEN TO USE PACKAGE DIAGRAMS**

1. It is used in large scale systems to picture dependencies between major elements in the system.

2. Package diagrams represent a compile time grouping mechanism

**Purpose**: The purpose of the Package Diagram is to group elements of a large system and illustrate the dependencies between them.

**Core concerns**: The Package Diagram can be used to organize a system’s parts and enables you to model Packages, Profiles and Annotations. These can be connected using Dependency, Profile Application, Package Merge, Package Import and Containment.

1. Purpose of Activity Diagrams

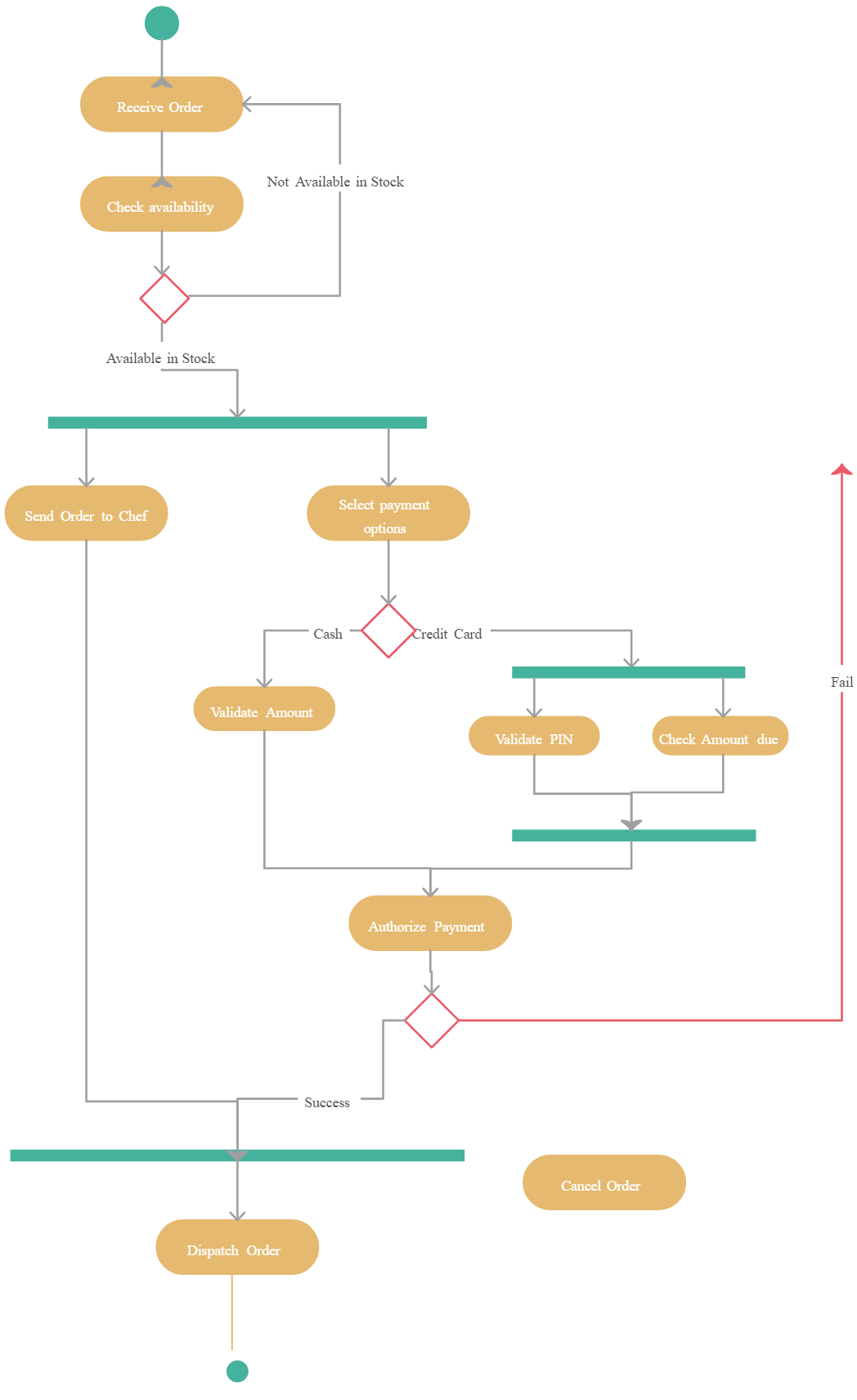
The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as −

* Draw the activity flow of a system.
* Describe the sequence from one activity to another.
* Describe the parallel, branched and concurrent flow of the system.



### The importance of sequence diagrams

Sequence diagrams have a variety of uses, many of them geared toward system design, testing and maintenance. Software developers and architects can use these diagrams in initial planning stages to brainstorm design, establish system requirements and consider how individual aspects of the system contribute to the overall end goal.

Sequence diagrams can increase the value of design meetings by enabling team members to communicate through a common visual. Teams might also spot problems earlier on. This can increase efficiency and troubleshooting to correct the issue before it reaches a later stage of development.

Software development teams can also use sequence diagrams to model and analyze specific [use cases](https://www.techtarget.com/searchsoftwarequality/definition/use-case). By visually mapping out the order in which actions occur in the use case, teams can spot the dynamic sequence of events. This can help solve interaction problems that arise and enhance team understanding of the use case.

Visually documenting systems can help maintain existing systems and explain how certain objects interact with each other. This makes a sequence diagram especially beneficial when control of a software system transfers from one team to another. This can also be applied to business models or other non software systems; sequence diagrams can establish the interactions that take place in, for example, a business collaboration workflow.

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

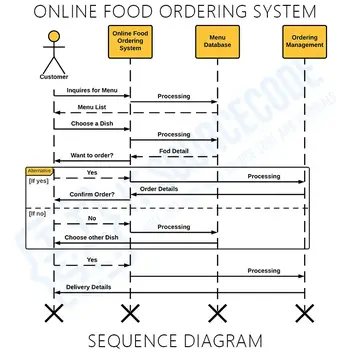
1. Place an order.
2. Pay money to the cash counter.
3. Order Confirmation.
4. Order preparation.
5. Order serving.

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.

You must be careful while selecting the notation of a message for any particular interaction. The notation must match with the particular sequence inside the diagram.

### Benefits of a Sequence Diagram

* Sequence diagrams are used to explore any real application or a system.
* Sequence diagrams are used to represent message flow from one object to another object.
* Sequence diagrams are easier to maintain.
* Sequence diagrams are easier to generate.
* Sequence diagrams can be easily updated according to the changes within a system.
* Sequence diagram allows reverse as well as forward engineering.



## 7.Why State Machine Diagram?

Since it records the dynamic view of a system, it portrays the behavior of a software application. During a lifespan, an object underwent several states, such that the lifespan exist until the program is executing. Each state depicts some useful information about the object.

It blueprints an interactive system that response back to either the internal events or the external ones. The execution flow from one state to another is represented by a state machine diagram. It visualizes an object state from its creation to its termination.

The main purpose is to depict each state of an individual object. It represents an interactive system and the entities inside the system. It records the dynamic behavior of the system.

## When to use a State Machine Diagram?

The state machine diagram implements the real-world models as well as the object-oriented systems. It records the dynamic behavior of the system, which is used to differentiate between the dynamic and static behavior of a system.

It portrays the changes underwent by an object from the start to the end. It basically envisions how triggering an event can cause a change within the system.

