Chapter 6: Object oriented analysis and design

Object oriented development life cycle

- Object oriented methodology of building system takes the object as basis. For ex: in case of banking system, customer is an object, a chequebook is object and even account is object.

- Object oriented life cycle contains:

1. System analysis: developer interacts with user of system to find out the user requirement and analyze the system to understand its functioning. Different model for analysis are as follows.

    - Object Model: it describes object in a system and their interrelationships. Only observes static part

    - Dynamic Model: this model depicts dynamic aspects of the system. It portrays the changes occurring in the states of various objects.

    - Functional model: This describes the flow of data and the changes that occur to the data throughout the system

2. System design
3. Object design
4. Implementation

Characteristics of OOS

- Class/Object
- Abstraction
- Data hiding
- Inheritance
- Polymorphism
- Reusability

Advantages of OOAD

- **Reuse of code through inheritance:** Objects created for Object Oriented Programs can easily be reused in other programs.
- **Flexibility through polymorphism:** makesound() function will output bark if it calls dog class and meow if it is cat class.
- **Software Maintenance:** Programs are not disposable. Legacy code must be dealt with on a daily basis, either to be improved upon (for a new version of an exist piece of software) or made to work with newer computers and software. An Object Oriented Program is much easier to modify and maintain than a non-Object Oriented Program. So although a lot of work is spent before the program is written, less work is needed to maintain it over time.
- **Design Benefits**: Large programs are very difficult to write. Object Oriented Programs force designers to go through an extensive planning phase, which makes for better designs with less flaws. In addition, once a program reaches a certain size, Object Oriented Programs are actually *easier* to program than non-Object Oriented ones.

UML

- The Unified Modeling Language (UML) is a general-purpose modeling language
- Intended to provide a standard way to visualize the design of a system
- Developed by Grady Booch, Ivar Jacobson and James Rumbaugh at Rational Software in 1994–1995, with further development led by them through 1996
- In 1997, UML was adopted as a standard by the Object Management Group (OMG), and has been managed by this organization ever since.
- In 2005, UML was also published by the International Organization for Standardization (ISO) as an approved ISO standard.

UML

- Behavior Diagram
  - Behavior diagrams emphasize what must happen in the system being modeled
  - Represents dynamic view of the system
  - Ex: Use Case Diagram, Activity Diagram, State Transition Diagram, Sequence Diagram, collaboration diagram
- Structure Diagrams
  - Structure diagrams emphasize the things that must be present in the system being modeled
  - Represents static view of the system
  - Ex: Class diagram, Object diagram, Component diagram,Deployment diagram

Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

The purpose of the class diagram can be summarized as –

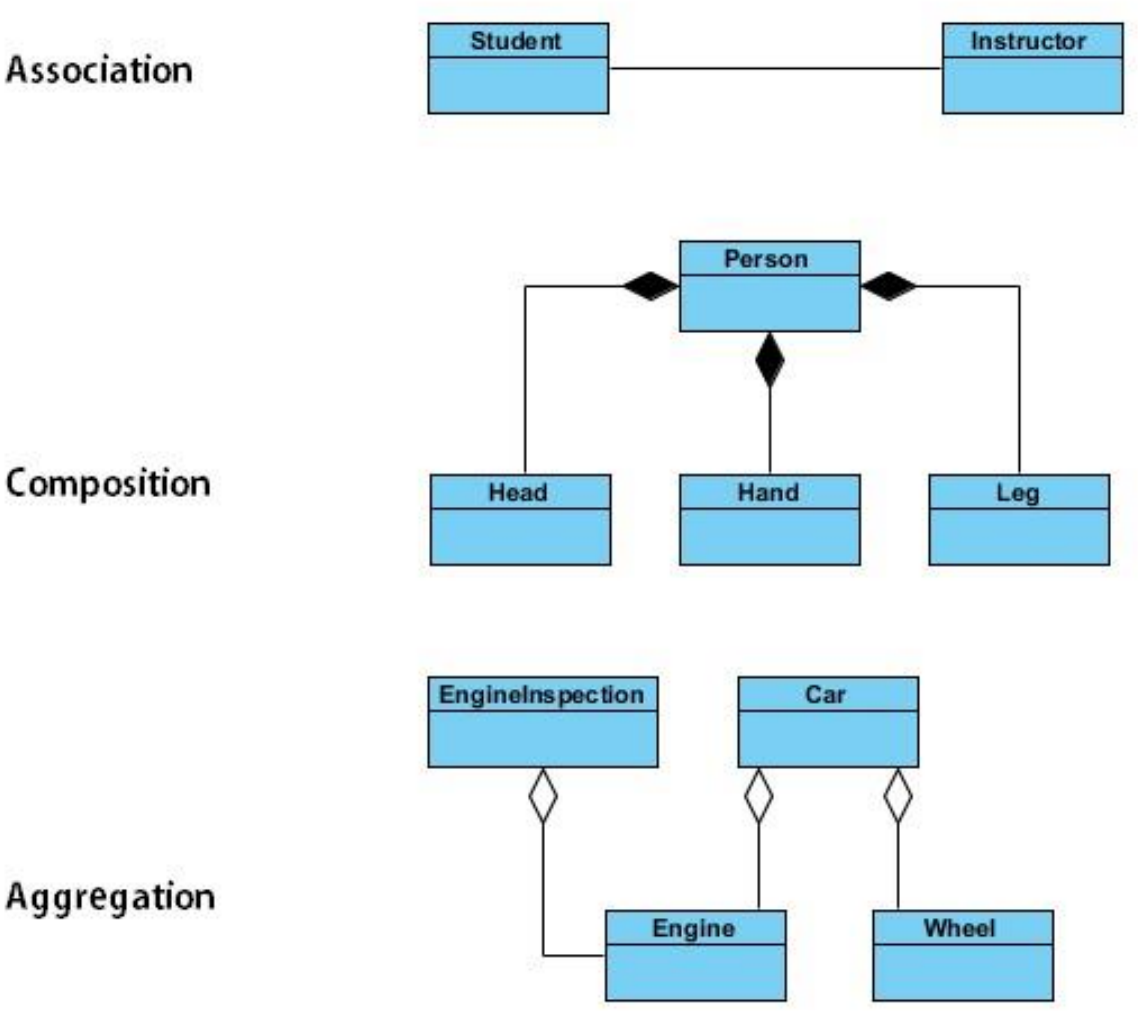
- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.
- Forward and reverse engineering.

The following points should be remembered while drawing a class diagram –

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing it should be understandable to the developer/coder.
- Finally, before making the final version, the diagram should be drawn on plain paper and reworked as many times as possible to make it correct.

Class diagram

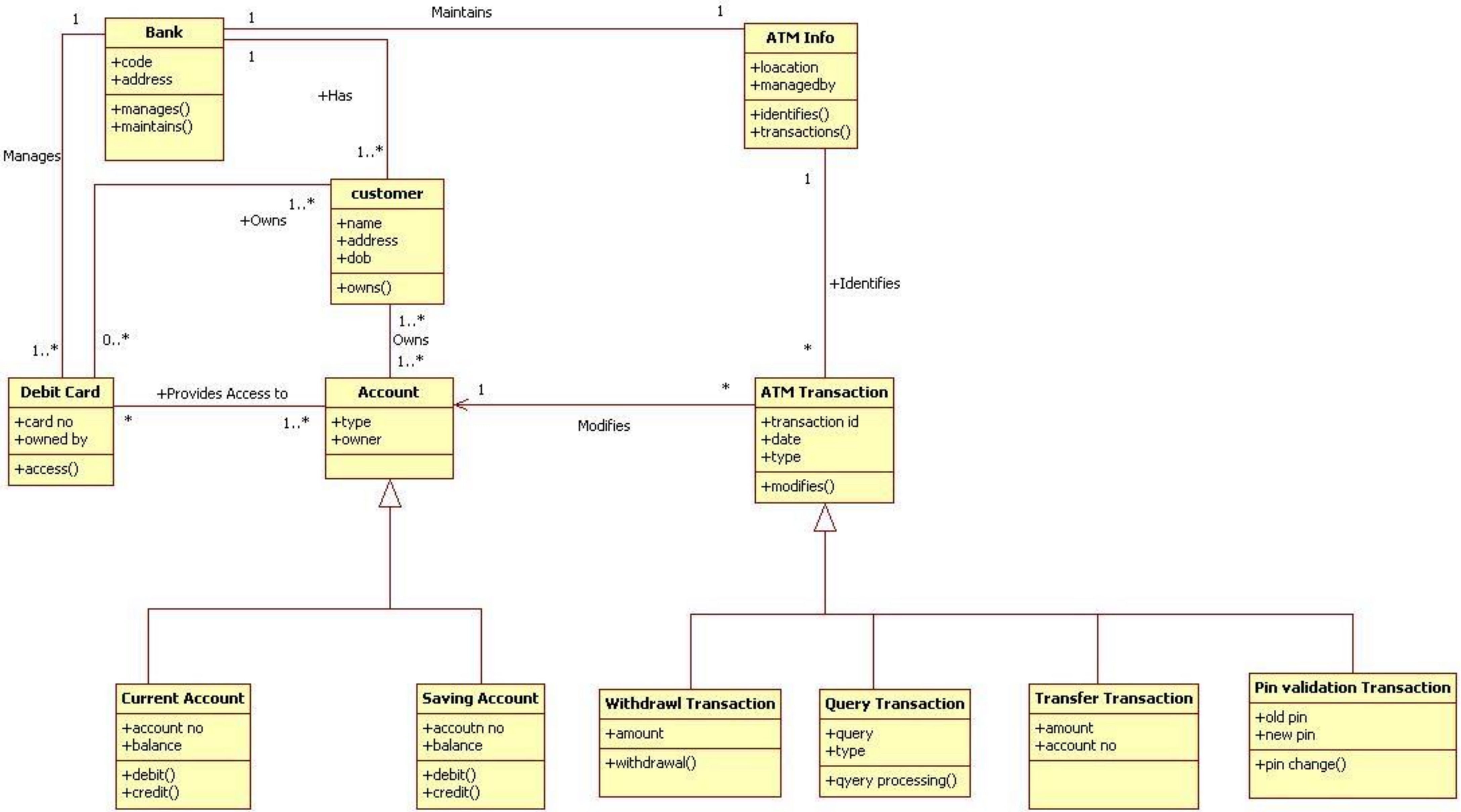
- Relationships



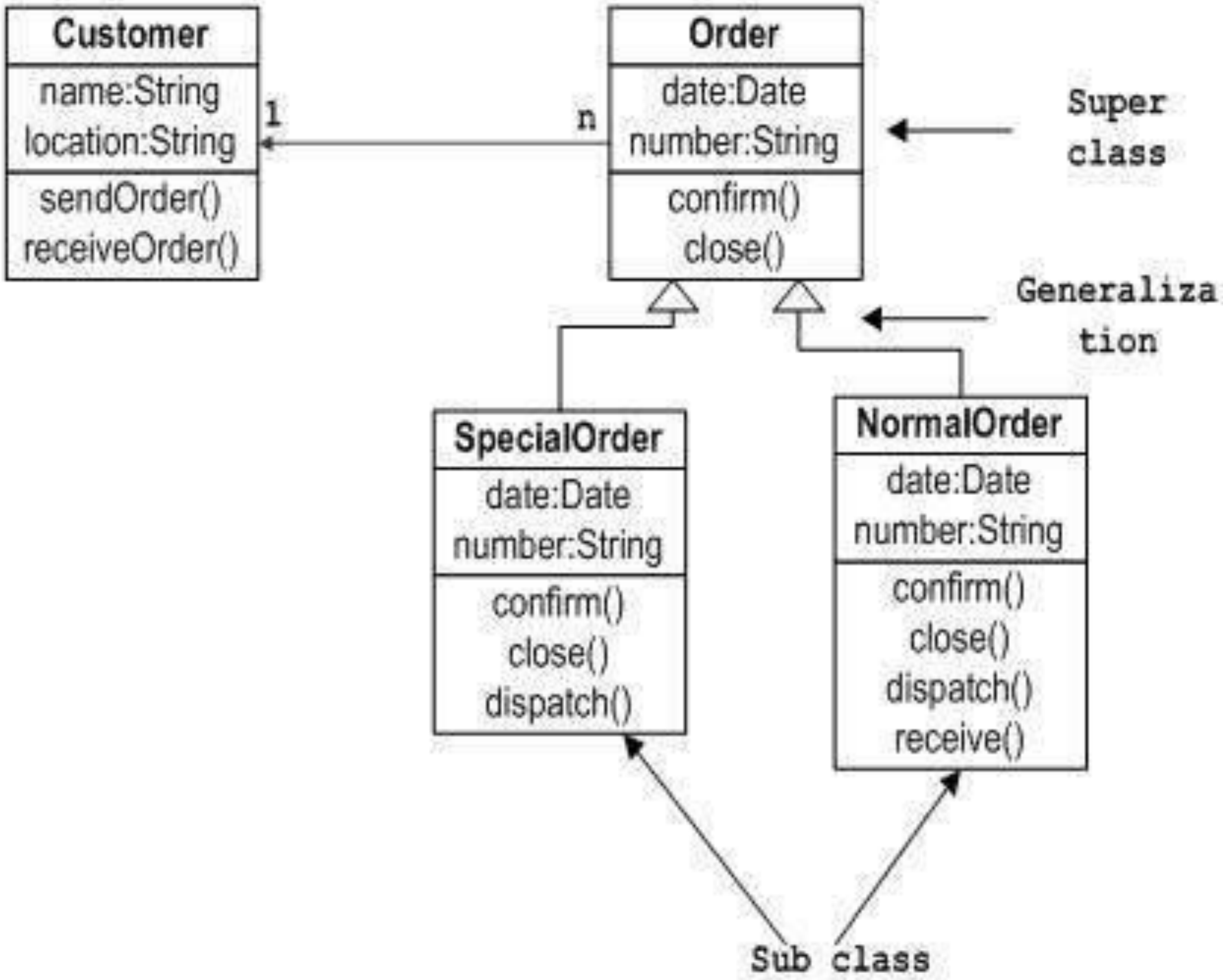
Class diagram

- Visibility
  - Private (-) :
    - A private member is visible only from within the class
  - Protected (#) :
    - A protected member is visible from within the class and from the subclasses inherited from this class
  - Public (+) :
    - A public member is visible from anywhere in the system

Class diagram



Sample Class Diagram



Object Diagram

Object diagrams are dependent on the class diagram as they are derived from the class diagram. It represents an instance of a class diagram. The objects help in portraying a static view of an object-oriented system at a specific instant.

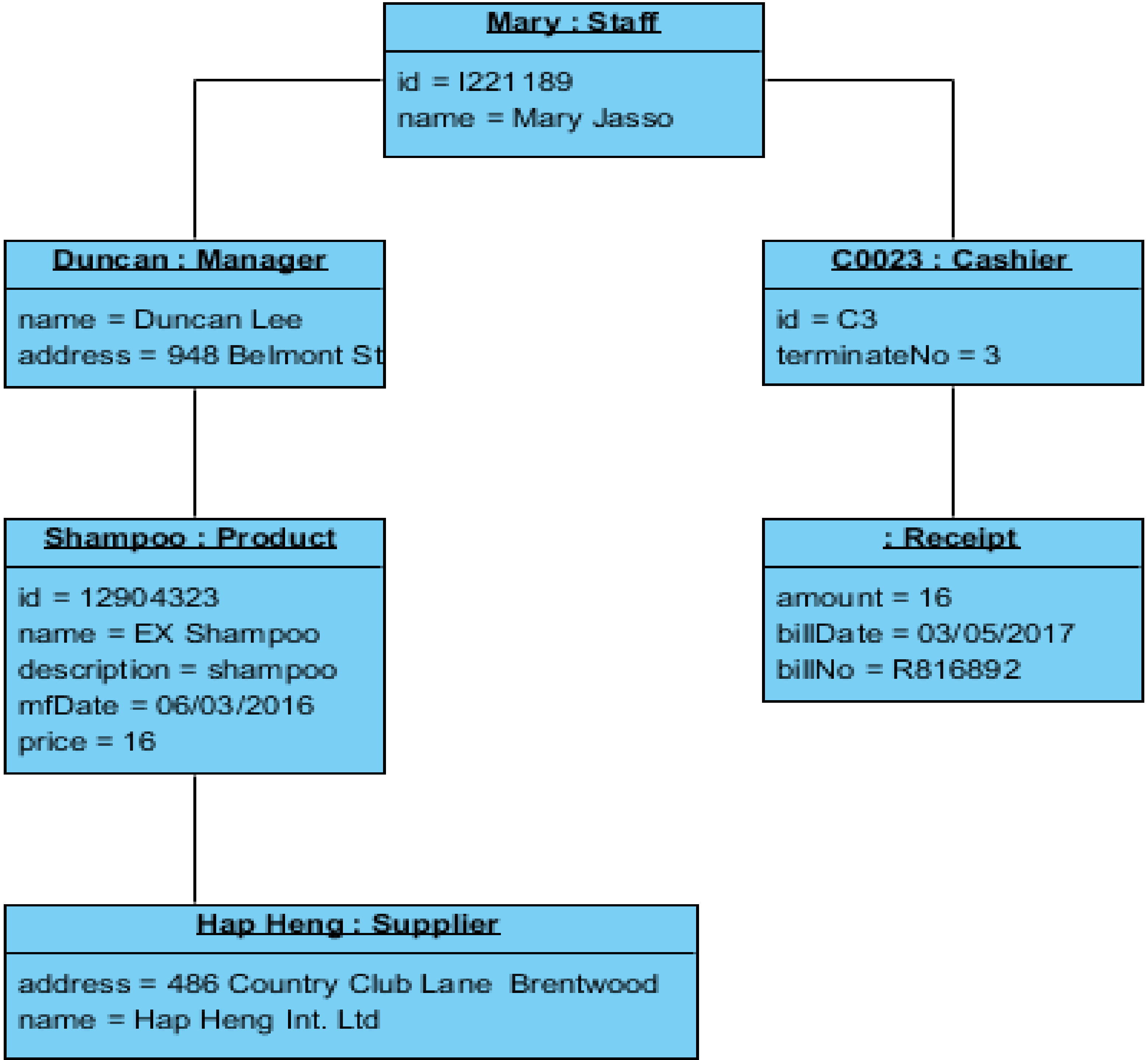
Both the object and class diagram are similar to some extent; the only difference is that the class diagram provides an abstract view of a system. It helps in visualizing a particular functionality of a system.

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.

How to draw an Object Diagram?

1. All the objects present in the system should be examined before start drawing the object diagram.
2. Before creating the object diagram, the relation between the objects must be acknowledged.
3. The association relationship among the entities must be cleared already.
4. To represent the functionality of an object, a proper meaningful name should be assigned.
5. The objects are to be examined to understand its functionality.


Mary : Staff
id = I221189
name = Mary Jasso
Duncan : Manager
name = Duncan Lee
address = 948 Belmont St
C0023 : Cashier
id = C3
terminateNo = 3
Shampoo : Product
id = 12904323
name = EX Shampoo
description = shampoo
mfDate = 06/03/2016
price = 16
: Receipt
amount = 16
billDate = 03/05/2017
billNo = R816892
Hap Heng : Supplier
address = 486 Country Club Lane Brentwood
name = Hap Heng Int. Ltd

Component diagram

A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable. It models the physical view of a system such as executables, files, libraries, etc. that resides within the node.

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:

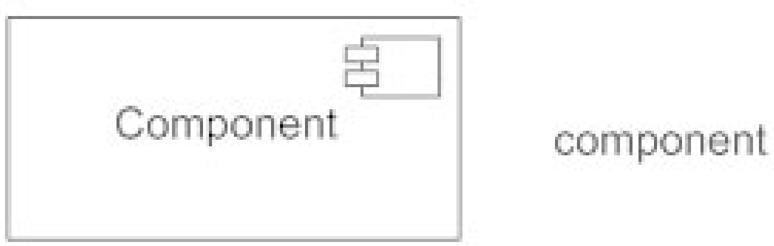
1. It envisions each component of a system.
2. It constructs the executable by incorporating forward and reverse engineering.
3. It depicts the relationships and organization of components.

Following are some reasons for the requirement of the component diagram**:**

1. It portrays the components of a system at the runtime.
2. It is helpful in testing a system.
3. It envisions the links between several connections.

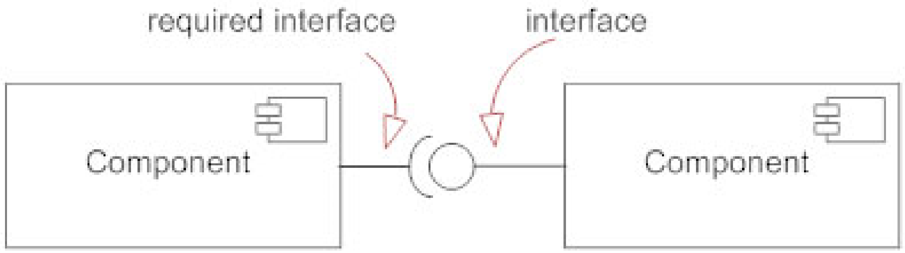
Component:

A component is a logical unit block of the system, a slightly higher abstraction than classes. It is represented as a rectangle with a smaller rectangle in the upper right corner with tabs or the word written above the name of the component to help distinguish it from a class.



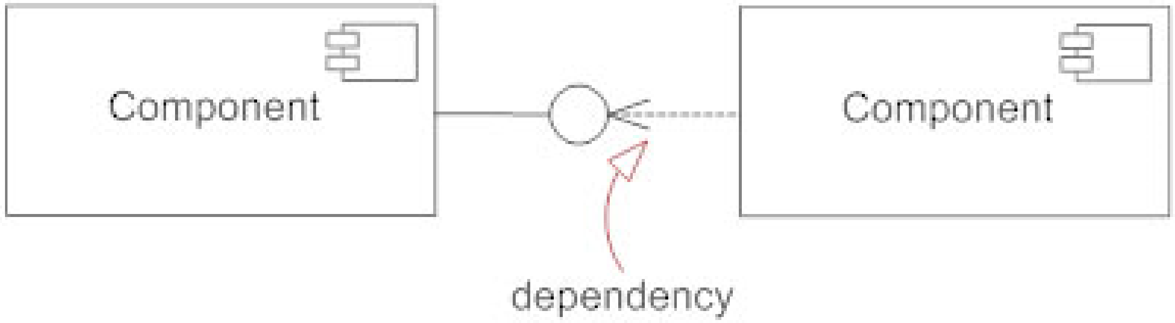
Interface

An interface (small circle or semi-circle on a stick) describes a group of operations used (required) or created (provided) by components. A full circle represents an interface created or provided by the component. A semi-circle represents a required interface, like a person's input.



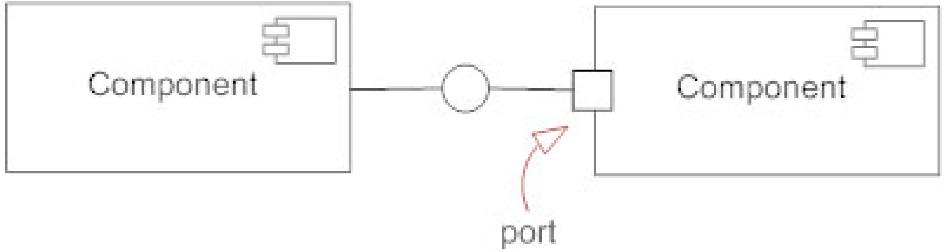
Dependencies:

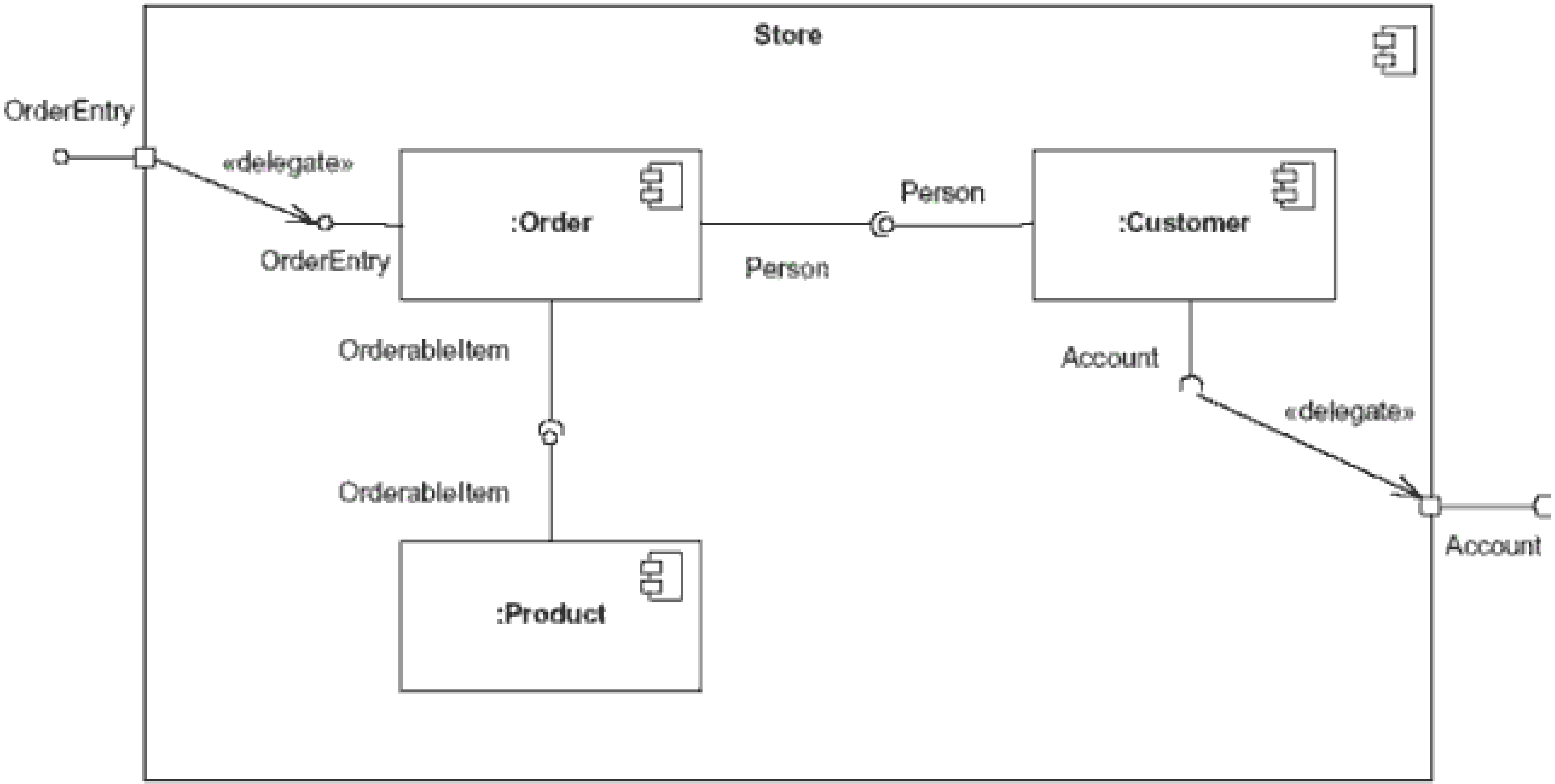
Draw dependencies among components using dashed arrows.



Port

Ports are represented using a square along the edge of the system or a component. A port is often used to help expose required and provided interfaces of a component.




Store
OrderEntry
«delegate»
OrderEntry
:Order
Person
Person
:Customer
OrderableItem
OrderableItem
:Product
Account
«delegate»
Account

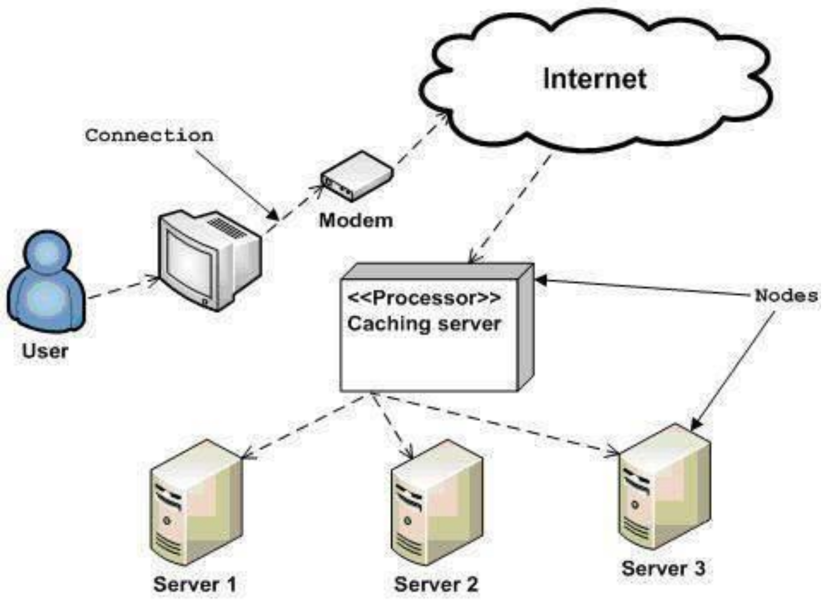
Deployment Diagram

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

The purpose of deployment diagrams can be described as –

- Visualize the hardware topology of a system.
- Describe the hardware components used to deploy software components.
- Describe the runtime processing nodes.

Deployment diagram of an order management system



Use case diagram

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application

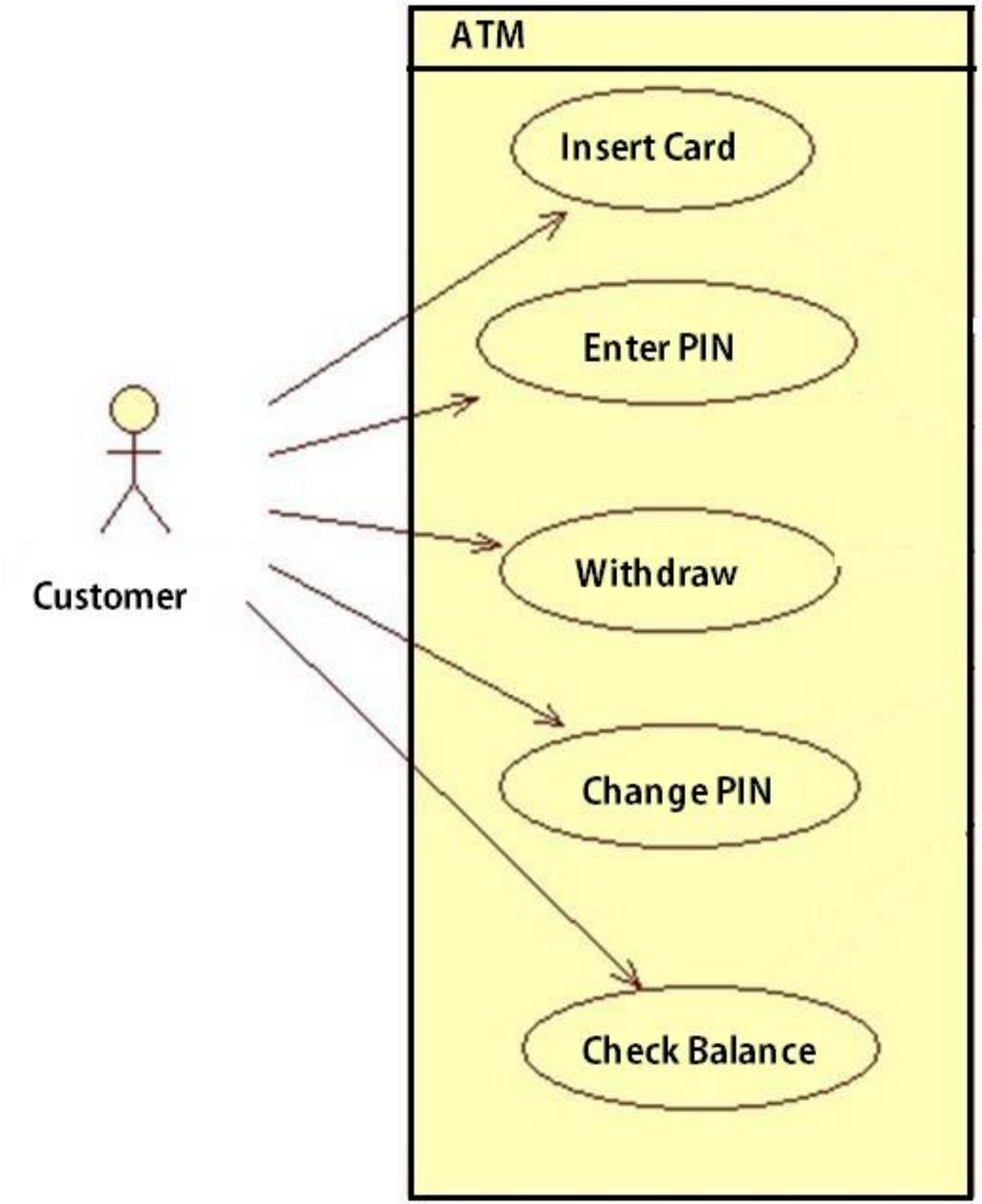
Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Following are some rules that must be followed while drawing a use case diagram:

1. A pertinent and meaningful name should be assigned to the actor or a use case of a system.
2. The communication of an actor with a use case must be defined in an understandable way.
3. Specified notations to be used as and when required.
4. The most significant interactions should be represented among the multiple no of interactions between the use case and actors.

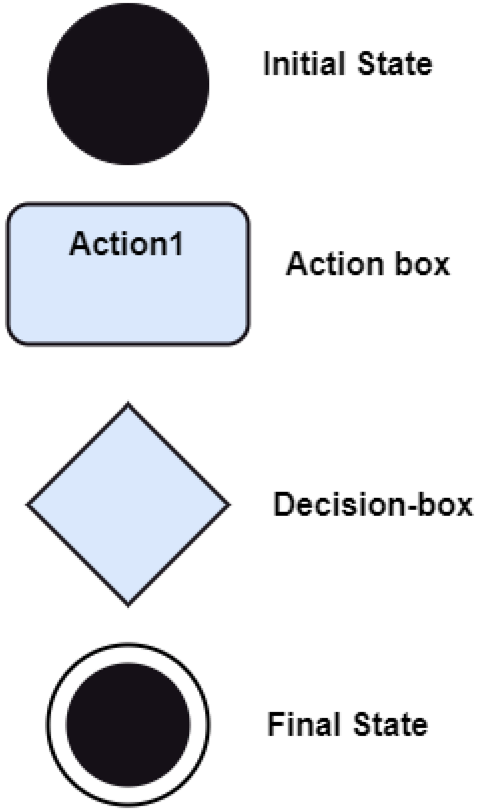
Use case diagram

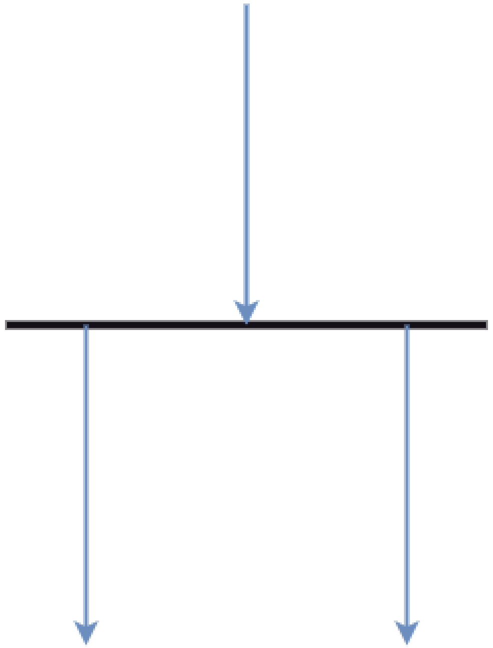


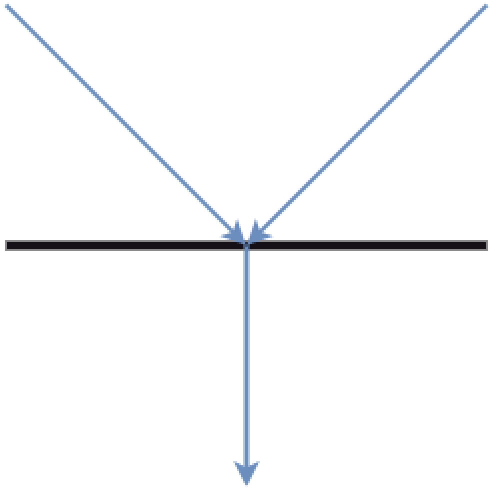
Activity Diagram

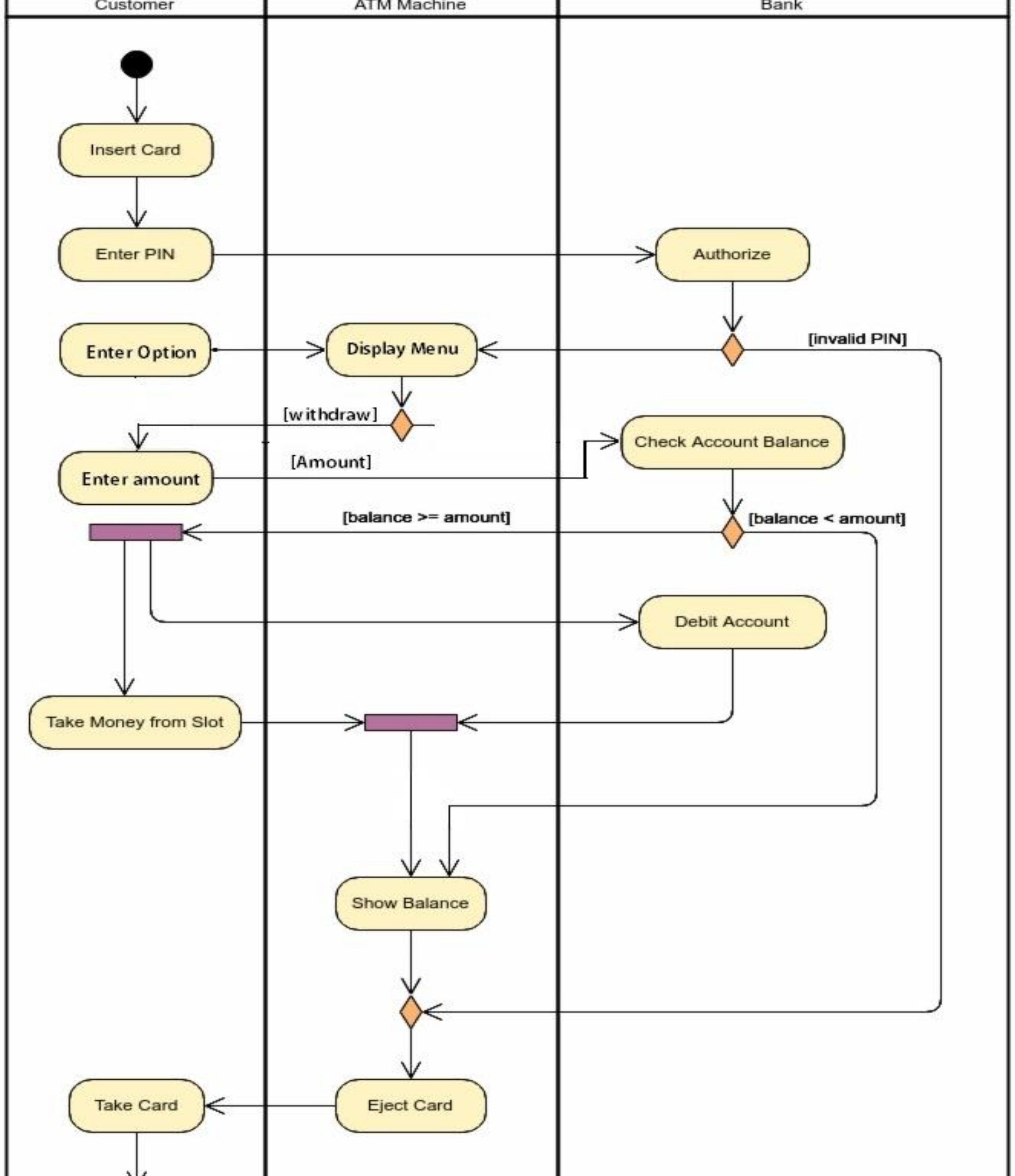
In UML, the activity diagram is used to demonstrate the flow of control within the system rather than the implementation. It models the concurrent and sequential activities.

The activity diagram helps in envisioning the workflow from one activity to another. It put emphasis on the condition of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and to deal with such kinds of flows, the activity diagram has come up with a fork, join, etc.








Customer
ATM Machine
Bank
Insert Card
Enter PIN
Authorize
[invalid PIN]
Enter Option
Display Menu
[withdraw]
Check Account Balance
[Amount]
Enter amount
[balance >= amount]
[balance < amount]
Debit Account
Take Money from Slot
Show Balance
Take Card
Eject Card

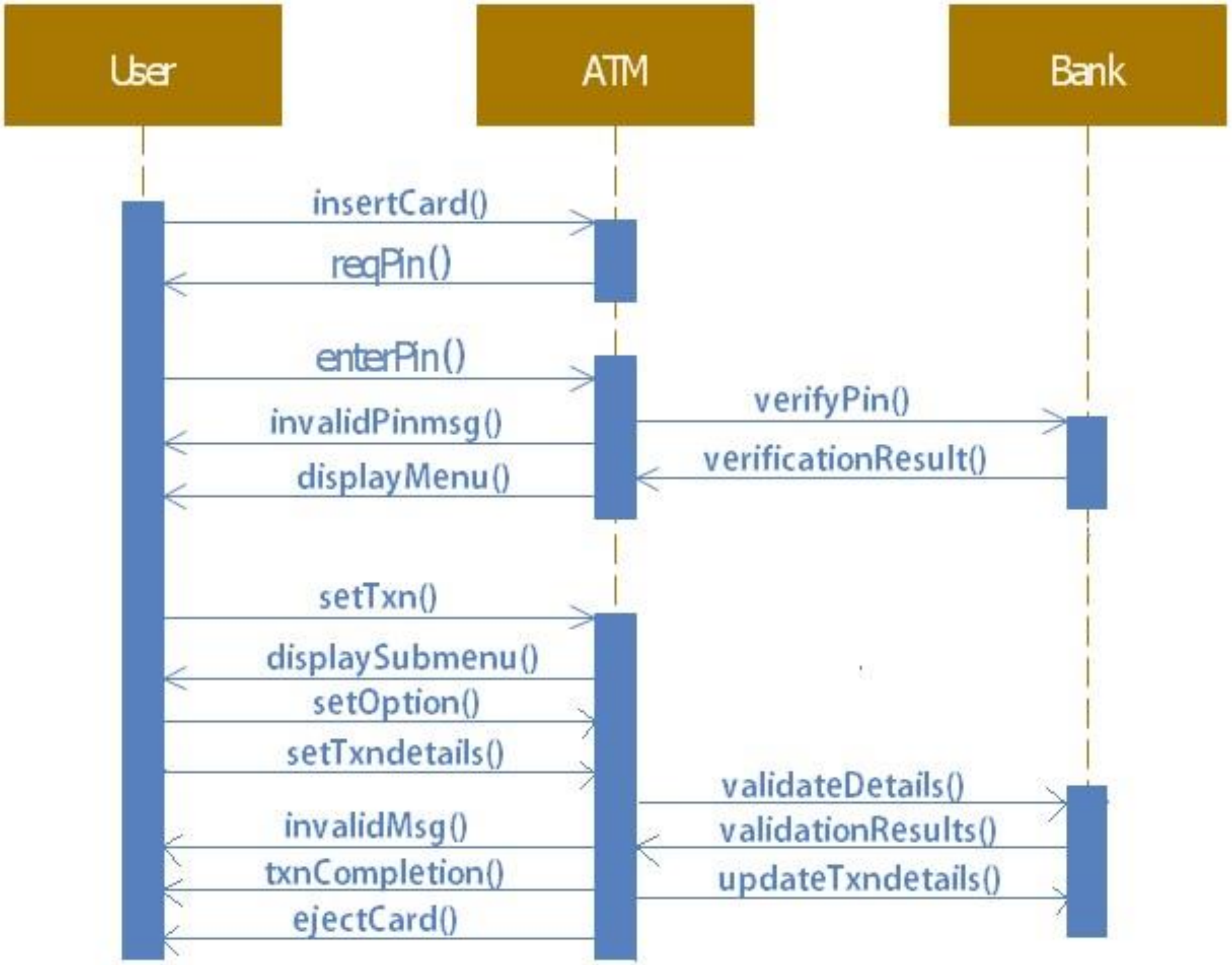
Sequence Diagram

The sequence diagram represents the flow of messages in the system and is also termed as an event diagram. It helps in envisioning several dynamic scenarios. It portrays the communication between any two lifelines as a time-ordered sequence of events, such that these lifelines took part at the run time

Purpose of a Sequence Diagram

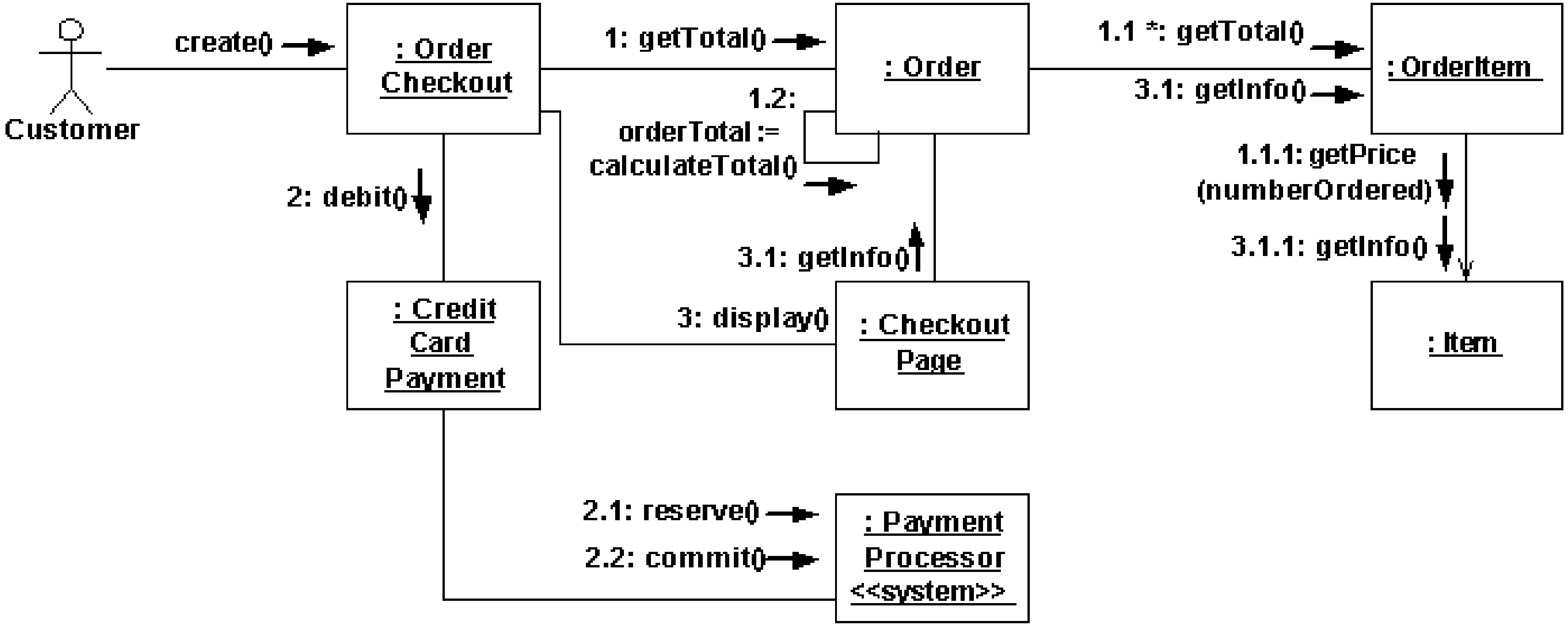
1. To model high-level interaction among active objects within a system.
2. To model interaction among objects inside a collaboration realizing a use case.
3. It either models generic interactions or some certain instances of interaction.

Sequence diagram



Collaboration Diagram

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.



Sequence Diagrams	Collaboration Diagrams
The sequence diagram represents the UML, which is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	The collaboration diagram also comes under the UML representation which is used to visualize the organization of the objects and their interaction.
The sequence diagram are used to represent the sequence of messages that are flowing from one object to another.	The collaboration diagram are used to represent the structural organization of the system and the messages that are sent and received.
The sequence diagram is used when time sequence is main focus.	The collaboration diagram is used when object organization is main focus.
The sequence diagrams are better suited of analysis activities.	The collaboration diagrams are better suited for depicting simpler interactions of the smaller number of objects.