Unified Modelling Language (UML)

- Systems and Models in UML
  - System − A set of elements organized to achieve certain objectives form a system. Systems are often divided into subsystems and described by a set of models.
  - Model − Model is a simplified, complete, and consistent abstraction of a system, created for better understanding of the system.
  - View − A view is a projection of a system's model from a specific perspective.

Conceptual Model of UML

- Three major elements −
  - Basic building blocks
  - Rules
  - Common mechanisms
- Basic Building Blocks
  - Three building blocks of UML are −
    - Things
    - Relationships
    - Diagrams

Things

- There are four kinds of things in UML
  - Structural Things − These are the nouns of the UML models representing the static elements which is either physical or conceptual. The structural things are class, interface, collaboration, use case, active class, components, and nodes.
  - Behavioural Things − These are the verbs of the UML models representing the dynamic behaviour over time and space. The two types of behavioural things are interaction and state machine.
  - Grouping Things − They comprise the organizational parts of the UML models. There is only one kind of grouping thing, i.e., package.
  - Annotational Things − These are the explanations in the UML models representing the comments applied to describe elements.

Relationships

- Relationships are the connection between things.
- The four types of relationships that can be represented in UML
  - Dependency − This is a semantic relationship between two things such that a change in one thing brings a change in the other.
  - Association − This is a structural relationship that represents a group of links having common structure and common behaviour.
  - Generalization − This represents a generalization/specialization relationship in which subclasses inherit structure and behaviour from super-classes.
  - Realization − This is a semantic relationship between two or more classifiers such that one classifier lays down a contract that the other classifiers ensure to abide by.

Diagrams

- A diagram is a graphical representation of a system
- UML includes nine diagrams in all, namely −
  - Class Diagram
  - Object Diagram
  - Use Case Diagram
  - Sequence Diagram
  - Collaboration Diagram
  - State Chart Diagram
  - Activity Diagram
  - Component Diagram
  - Deployment Diagram

Rules

- UML has a number of rules so that the models are semantically self-consistent and related to other models in the system harmoniously. UML has semantic rules for the following −
  - Names
  - Scope
  - Visibility
  - Integrity
  - Execution
- Common Mechanisms
  - Specifications
  - Adornments
  - Common Divisions
  - Extensibility Mechanisms

Contd.

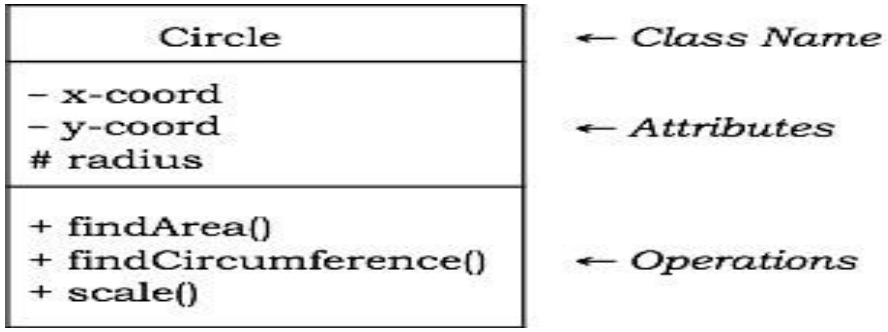
- Specifications
  - The specifications provide a semantic backplane that contains all the parts of a system and the relationship among the different paths.
- Adornments
  - Each element in UML has a unique graphical notation.
- Common Divisions
  - Division of classes and objects − A class is an abstraction of a group of similar objects. An object is the concrete instance that has actual existence in the system.
  - Division of Interface and Implementation − An interface defines the rules for interaction. Implementation is the concrete realization of the rules defined in the interface.

Contd.

- Extensibility Mechanisms
  - Stereotypes − It extends the vocabulary of the UML, through which new building blocks can be created out of existing ones.
  - Tagged Values − It extends the properties of UML building blocks.
  - Constraints − It extends the semantics of UML building blocks.

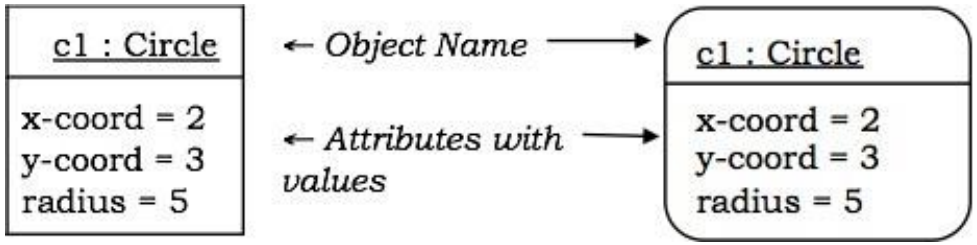
Basic Notations

- Class
  - A class is represented by a rectangle having three sections −
  - the top section containing the name of the class
  - the middle section containing class attributes
  - the bottom section representing operations of the class
- The visibility of the attributes and operations are represented as,
  - Public − A public member is visible from anywhere in the system. In class diagram, it is prefixed by the symbol '+'.
  - Private − A private member is visible only from within the class. It cannot be accessed from outside the class. A private member is prefixed by the symbol '−'.
  - Protected − A protected member is visible from within the class and from the subclasses inherited from this class, but not from outside. It is prefixed by the symbol '#'.
  - An abstract class has the class name written in italics.



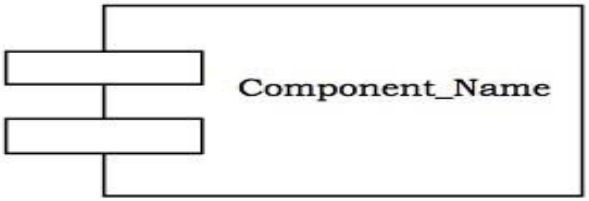
Contd.

- Object
  - An object is represented as a rectangle with two sections −
  - The top section contains the name of the object with the name of the class or package of which it is an instance of. The name takes the following forms −
    - **object-name** − class-name
    - **object-name** − class-name :: package-name
    - **class-name** − in case of anonymous objects
  - The bottom section represents the values of the attributes. It takes the form attribute-name = value.
  - Sometimes objects are represented using rounded rectangles.

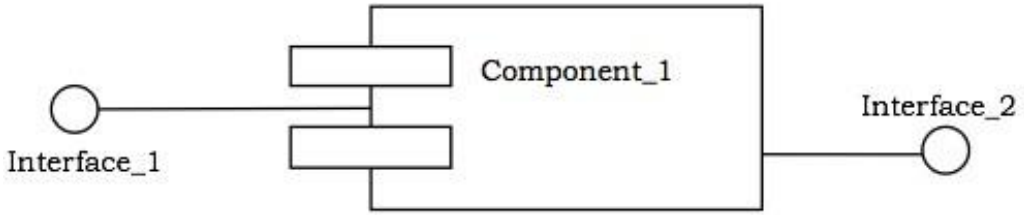


Contd.

- Component
  - A component is a physical and replaceable part of the system that provides the realization of a set of interfaces. is represented by a rectangle with tabs as shown in the figure below.

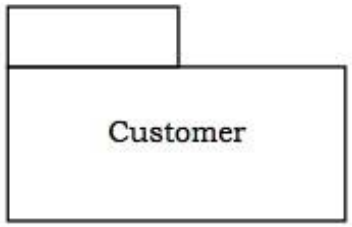


- Interface
  - Interface is a collection of methods of a class or component

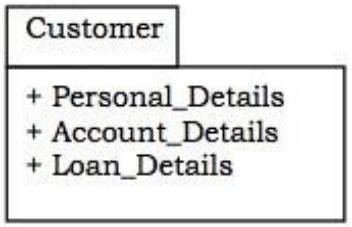


Contd.

- Package
  - A package is an organized group of elements. A package is represented by a tabbed folder.

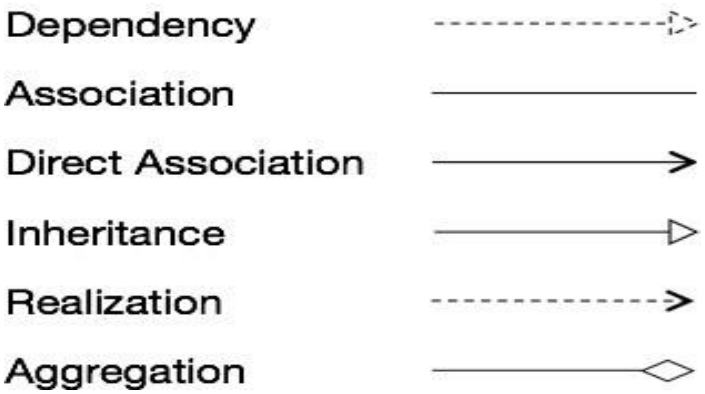


(a)



(b)

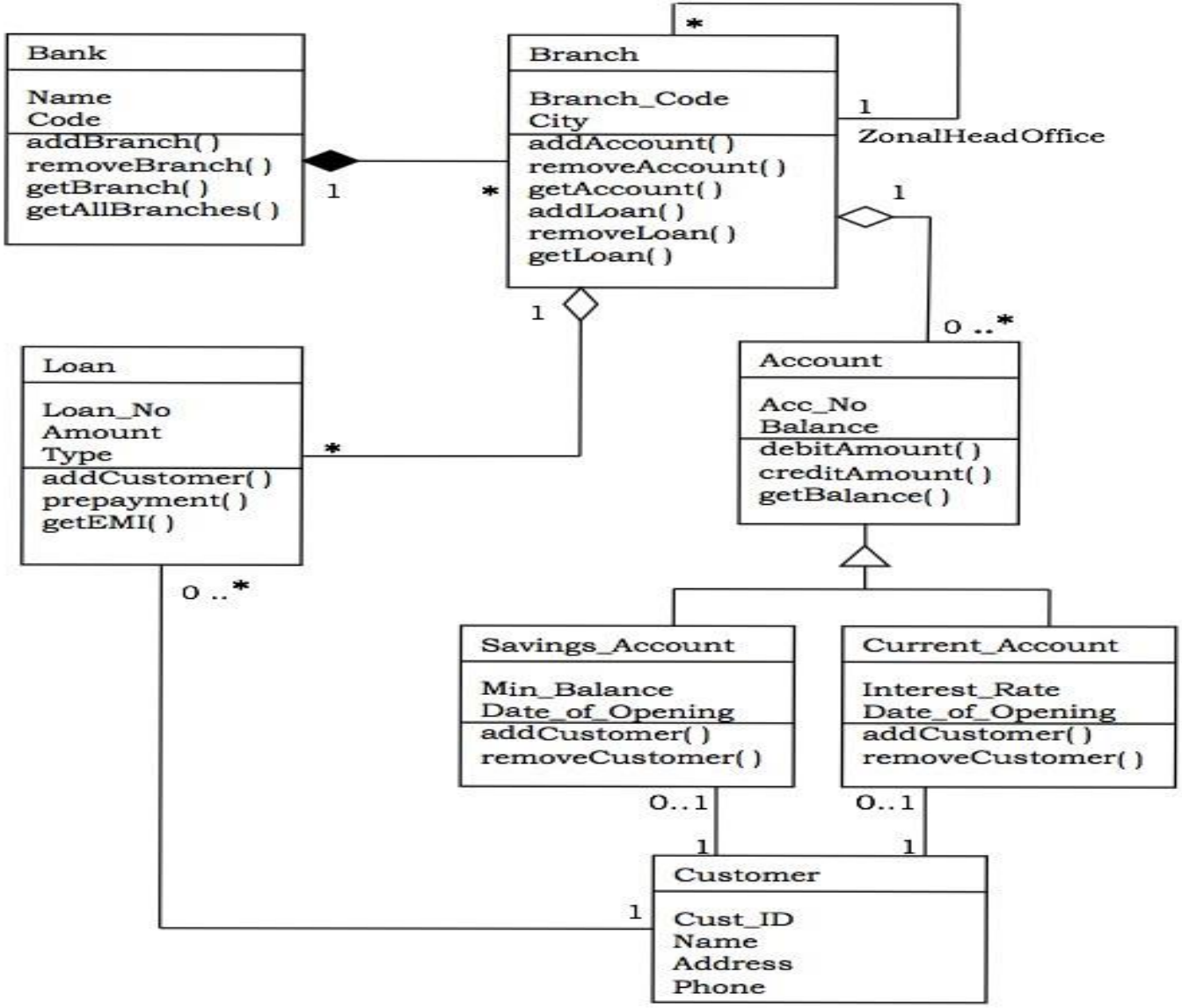
- Relationship



Class Diagram

- A class diagram models the static view of a system. It comprises of the classes, interfaces, and collaborations of a system; and the relationships between them.
- Example Prerequisite
- Classes in the system
- Bank, Branch, Account, Savings Account, Current Account, Loan, and Customer.

Class Diagram Example



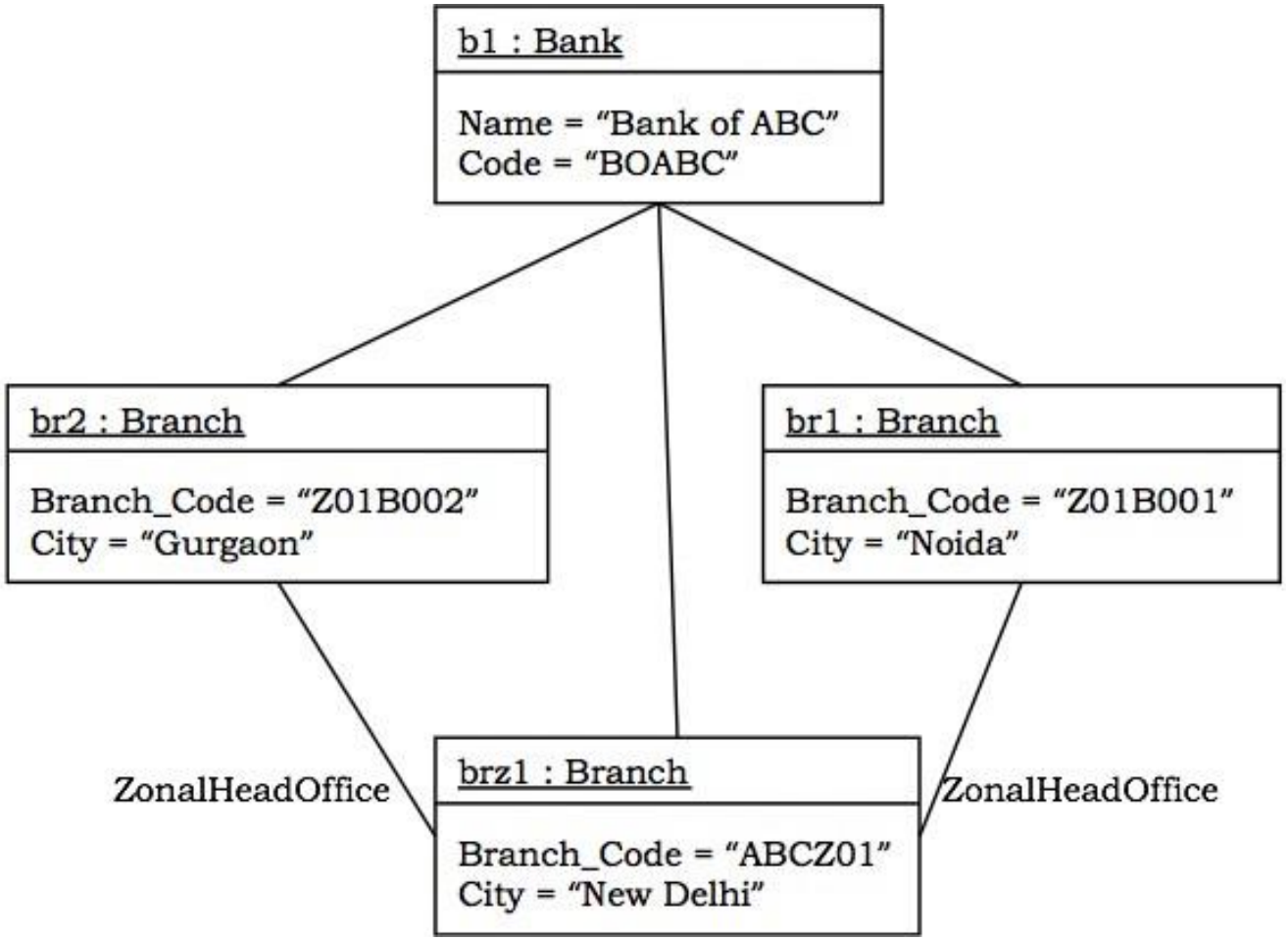
Contd.

- Relationships
- A Bank **“has–a”** number of Branches − composition, one–to–many
- A Branch with role Zonal Head Office **supervises** other Branches − unary association, one–to-many
- A Branch **“has–a”** number of accounts − aggregation, one–to–many
- From the class Account, two classes have inherited, namely, Savings Account and Current Account.
- A Customer can have one Current Account − association, one–to–one
- A Customer can have one Savings Account − association, one–to–one
- A Branch “**has–a**” number of Loans − aggregation, one–to–many
- A Customer can take many loans − association, one–to–many

Object Diagram

- An object diagram models a group of objects and their links at a point of time.
- the instances are shown of the things in a class diagram.
- Object diagram is the static part of an interaction diagram.

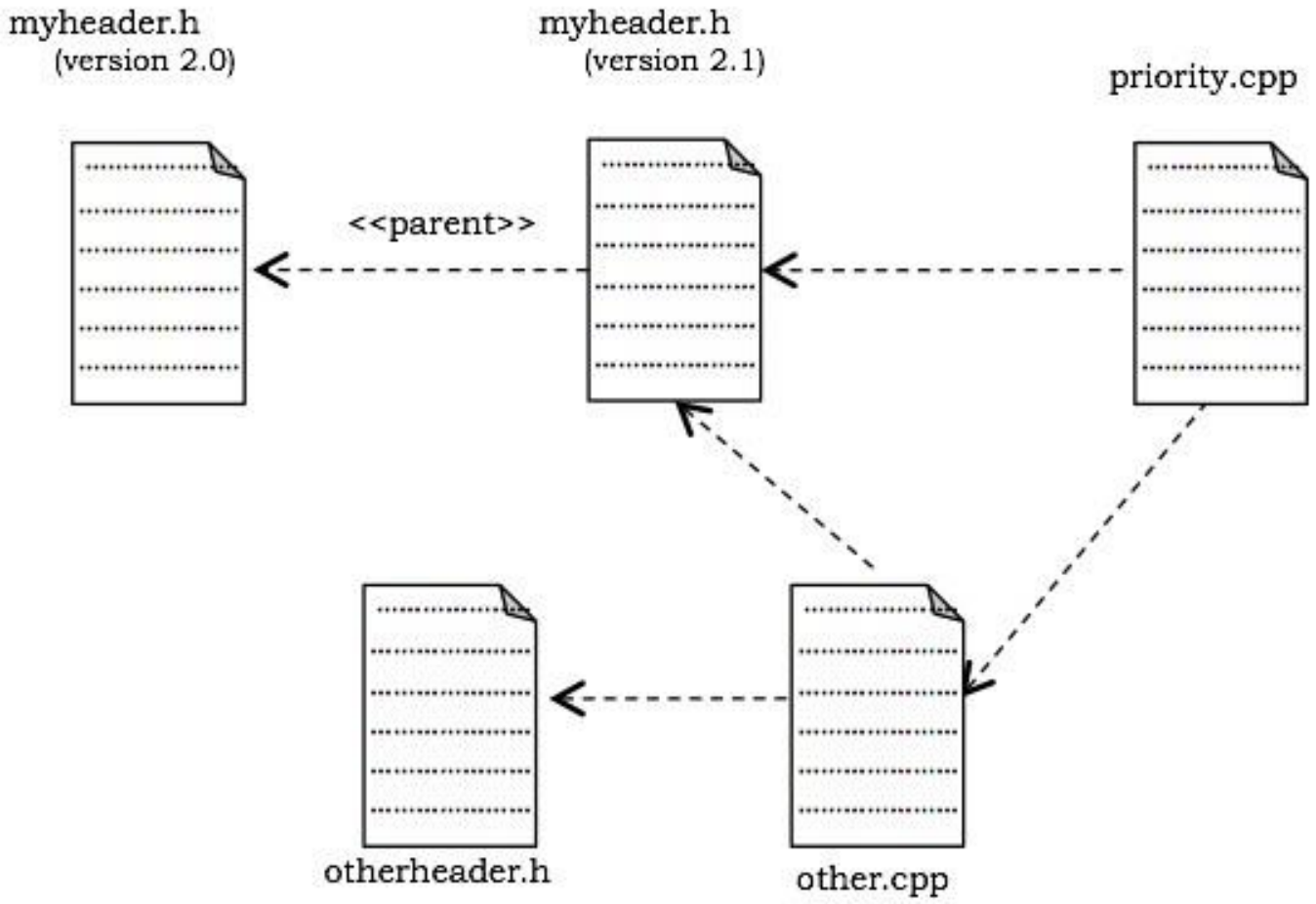
Example of Object Diagram



Component Diagram

- Component diagrams show the organization and dependencies among a group of components.
- Component diagrams comprise of,
  - Components
  - Interfaces
  - Relationships
  - Packages and Subsystems (optional)
- Applications
  - constructing systems through forward and reverse engineering.
  - modelling configuration management of source code files while developing a system using an object-oriented programming language.
  - representing schemas in modelling databases.
  - modelling behaviours of dynamic systems.

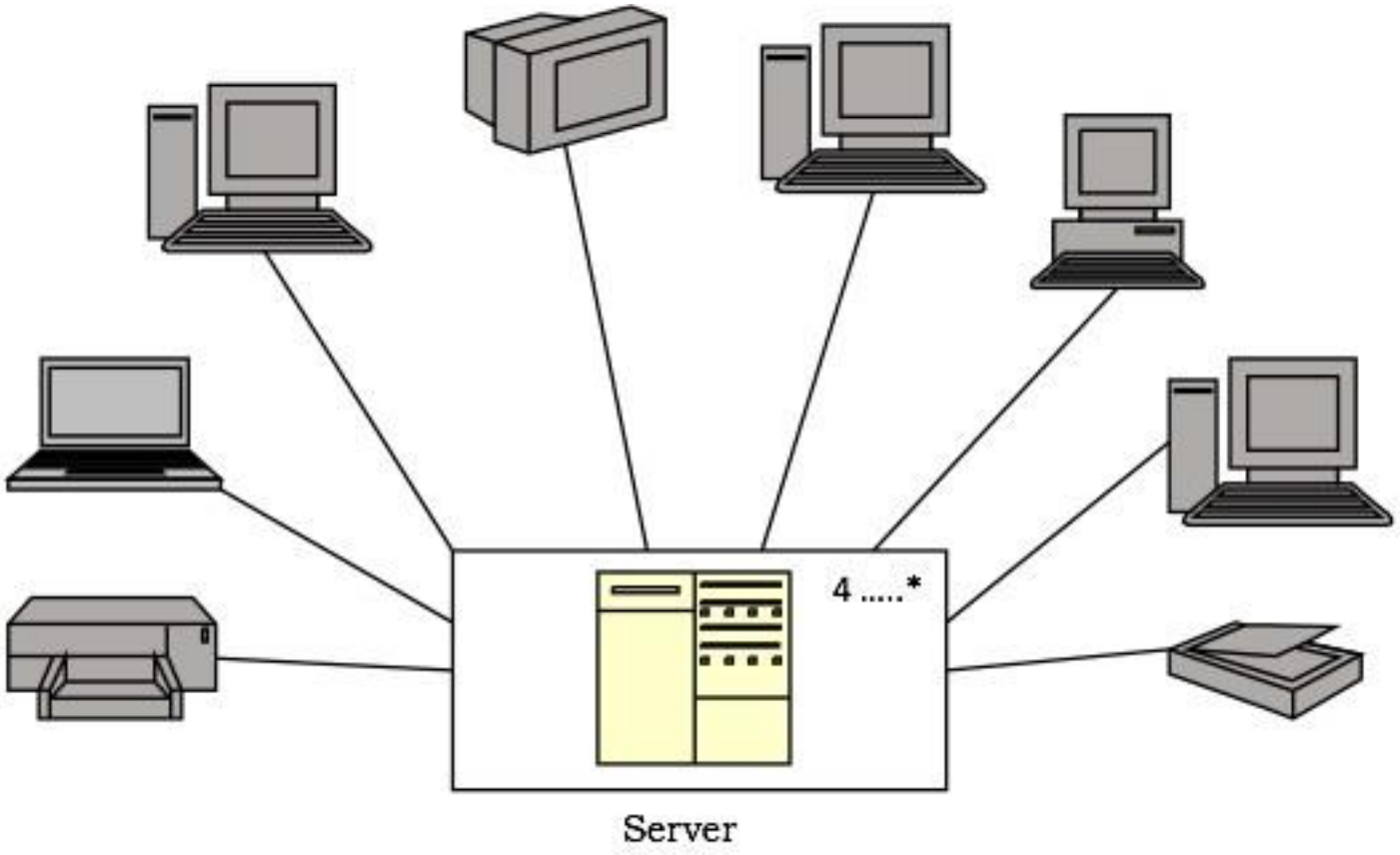
Example of Component Diagram



Deployment Diagram

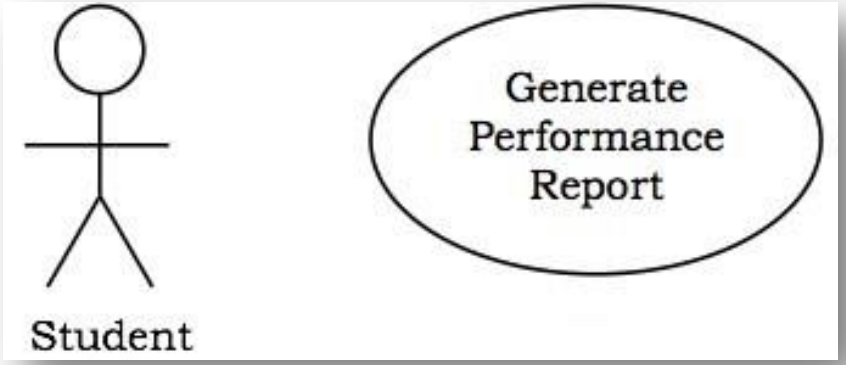
- It reflects the configuration of runtime processing nodes and their components that live on them; which is commonly comprised of nodes and dependencies, or associations between the nodes.
- Applications
  - model devices in embedded systems that typically comprise of software-intensive collection of hardware.
  - represent the topologies of client/server systems.
  - model fully distributed systems.

Example of Deployment Diagram



Use Case Model

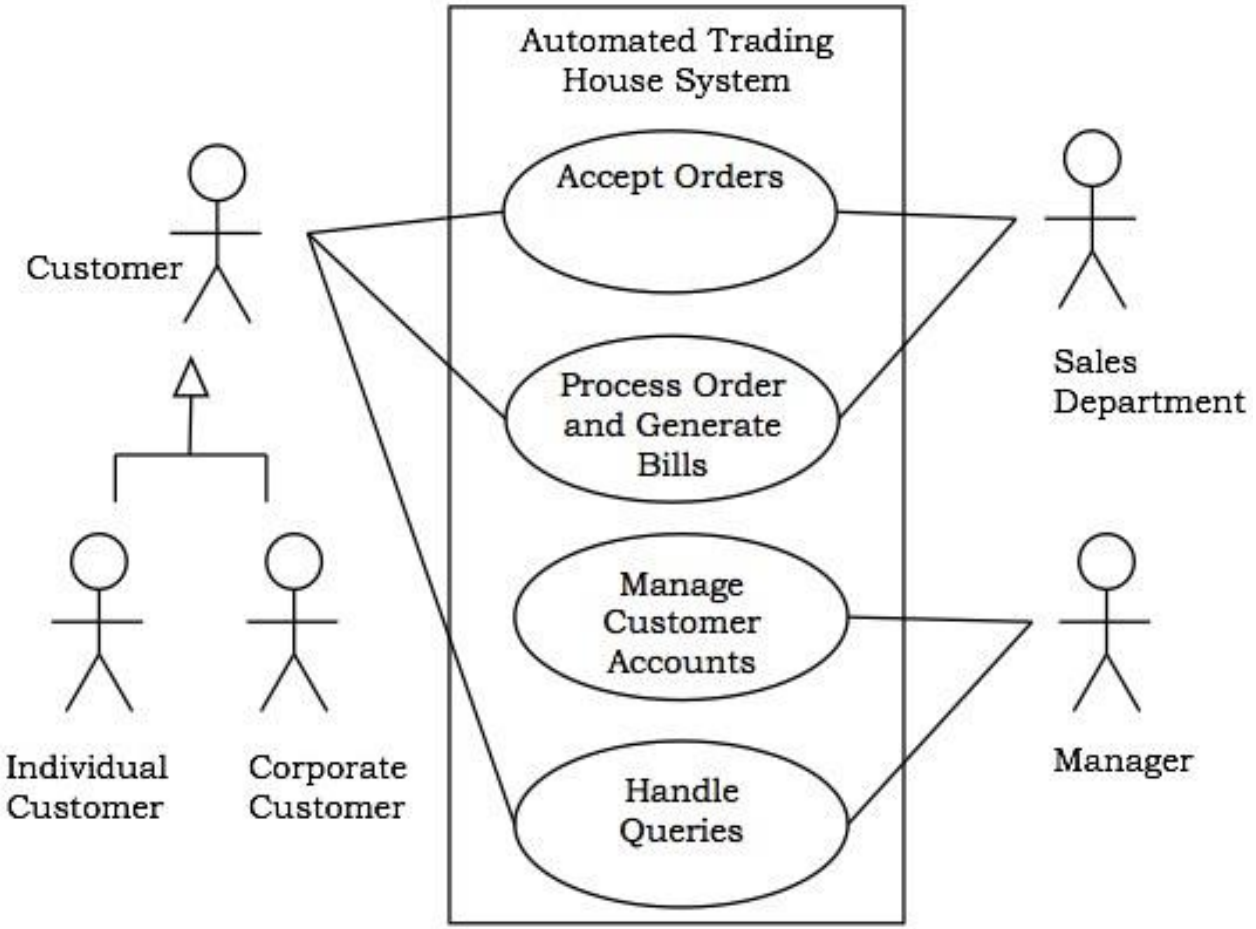
- It describes the sequence of actions on a system performs yielding visible results.
- It shows the interaction of things outside the system with the system itself.
- This may be applied to the whole system as well as a part of the system.
- Actor
  - An actor represents the roles of the users



Use case diagrams

- Components of Use case diagrams
  - Use cases
  - Actors
  - Relationships like dependency, generalization, and association
- Applications
  - To model the context of a system by enclosing all the activities of a system within a rectangle and focusing on the actors outside the system by interacting with it.
  - To model the requirements of a system from the outside point of view.

Example of Use Case Diagram

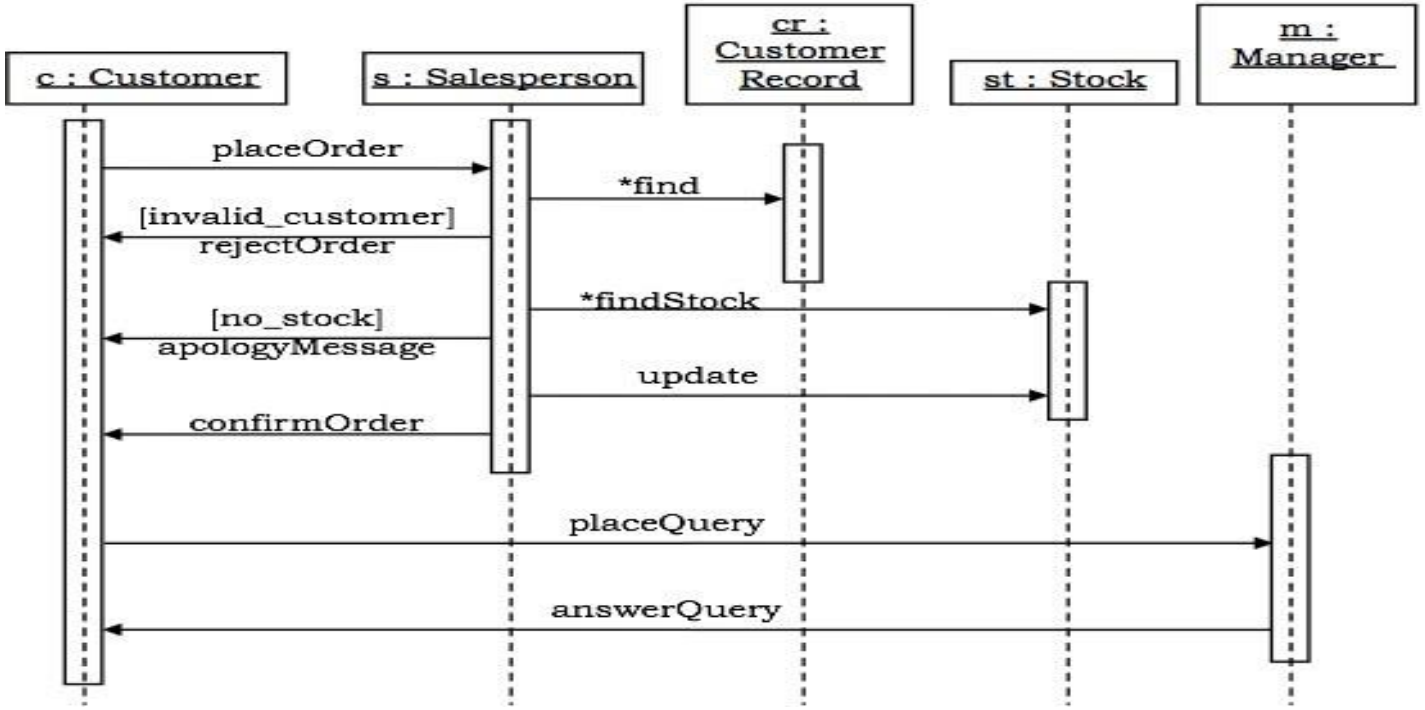


Interaction Diagrams

- It depicts the interactions of objects and their relationships. They also include the messages passed between them.
- There are two types of interaction diagrams,
  - Sequence Diagrams
  - Collaboration Diagrams
- Application
  - the control flow by time ordering using sequence diagrams.
  - the control flow of organization using collaboration diagrams.

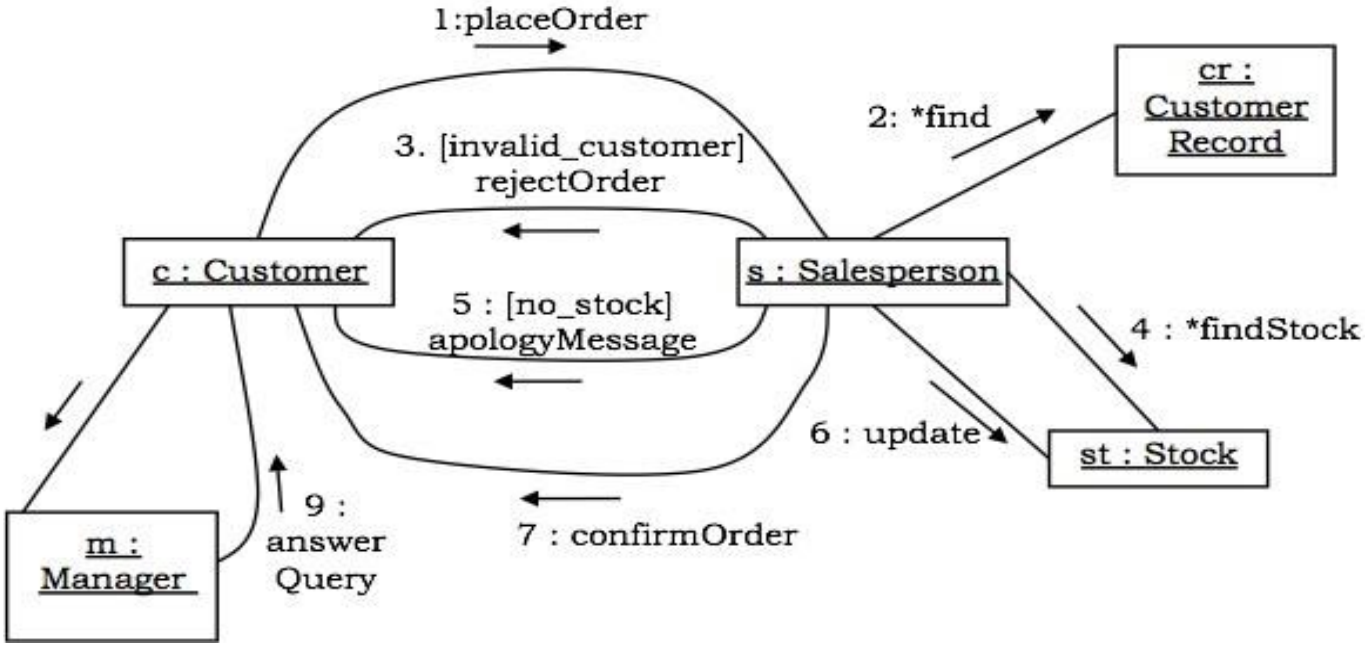
Sequence Diagrams

- These diagrams are in the form of two-dimensional charts. The objects that initiate the interaction are placed on the x–axis. The messages that these objects send and receive are placed along the y–axis, in the order of increasing time from top to bottom.



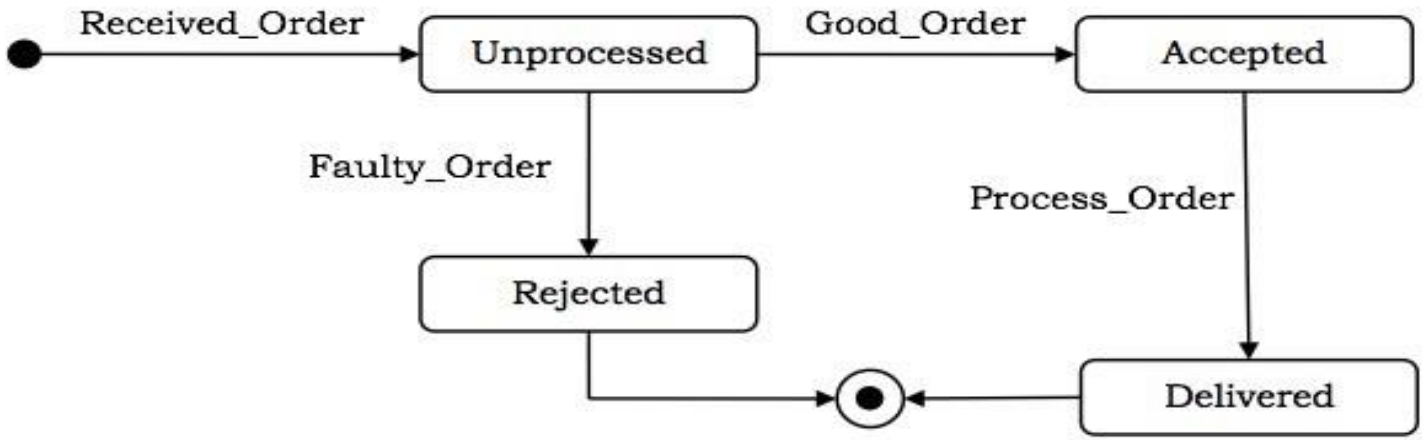
Collaboration Diagrams

- These diagrams, the objects that participate in the interaction are shown using vertices. The links that connect the objects are used to send and receive messages. The message is shown as a labelled arrow.



State–Chart Diagrams

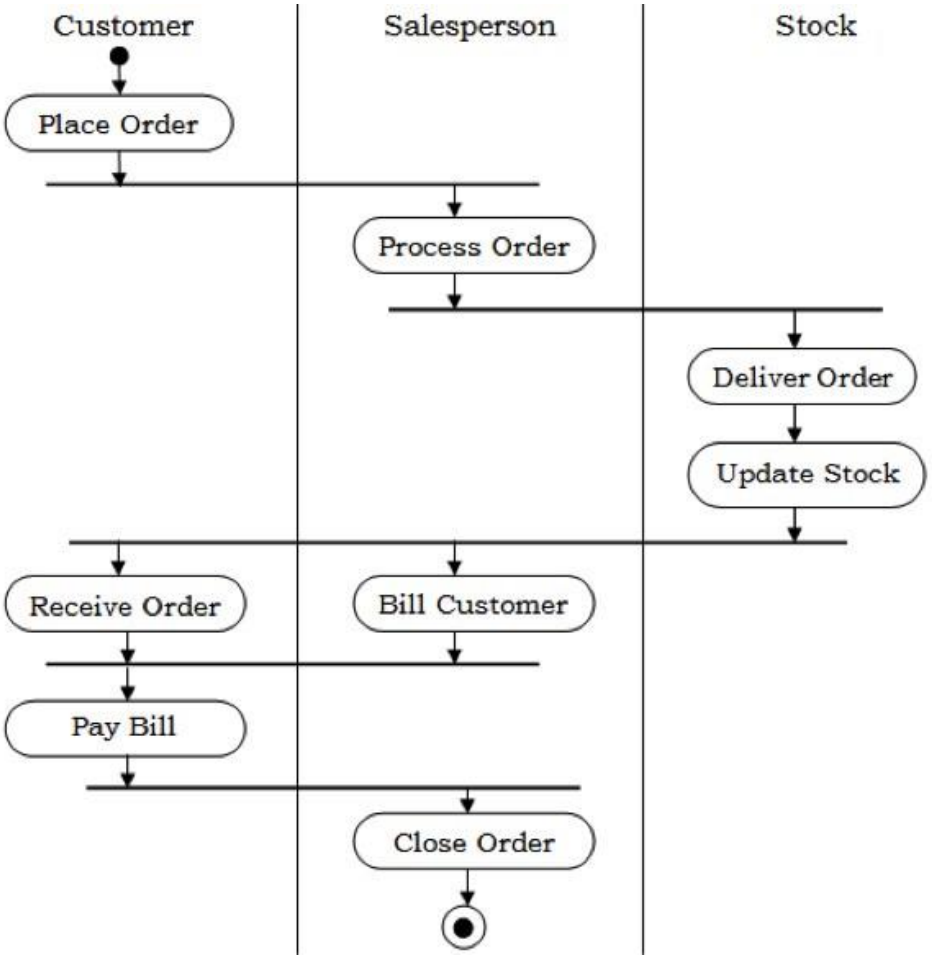
- It shows a state machine that depicts the control flow of an object from one state to another.
- State–Chart Diagrams comprise of −
  - States: Simple or Composite
  - Transitions between states
  - Events causing transitions
  - Actions due to the events



Activity Diagrams

- It depicts the flow of activities which are on-going non-atomic operations in a state machine. Activities result in actions which are atomic operations.
- Activity diagrams comprise of −
  - Activity states and action states
  - Transitions
  - Objects
- Applications
  - workflows as viewed by actors, interacting with the system.
  - details of operations or computations using flowcharts.

Example of Activity Diagram



Steps

- Conclusion of System Analysis Phase
- Conceptual Design using OOD
- The technology-independent concepts in the analysis model are mapped onto implementing classes
- Constraints are identified & interfaces are designed
- Resulting in a model for the solution domain. In a nutshell, a detailed description is constructed specifying how the system is to be built on concrete technologies
- The stages for object–oriented design ,
  - Definition of the context of the system
  - Designing system architecture
  - Identification of the objects in the system
  - Construction of design models
  - Specification of object interfaces