BACHELOR OF TECHNOLOGY YEAR 3

OBJECT ORIENTED ANALYSIS AND DESIGN

WEEK 8:

1. UML STRUCTURAL DIAGRAMS

UML structural diagrams are categorized as follows: class diagram, object diagram, component diagram, and deployment diagram.

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.

CLASS DIAGRAM OF A SYSTEM

Illustration using a Banking System

- A bank has many *branches*. In *each zone*, one branch is designated as the *zonal head office* that supervises the other branches in that zone.
- Each branch can have *multiple accounts* and *loans*.
- An *account* may be either a *savings* account or a *current* account.
- A customer may open *both* a savings account and a current account. However, a customer must not have *more than one* savings account or current account.
- A customer may also procure loans from the bank.

CLASSES IN THE SYSTEM

Bank, Branch, Account, Savings Account, Current Account, Loan, and Customer

RELATIONSHIPS

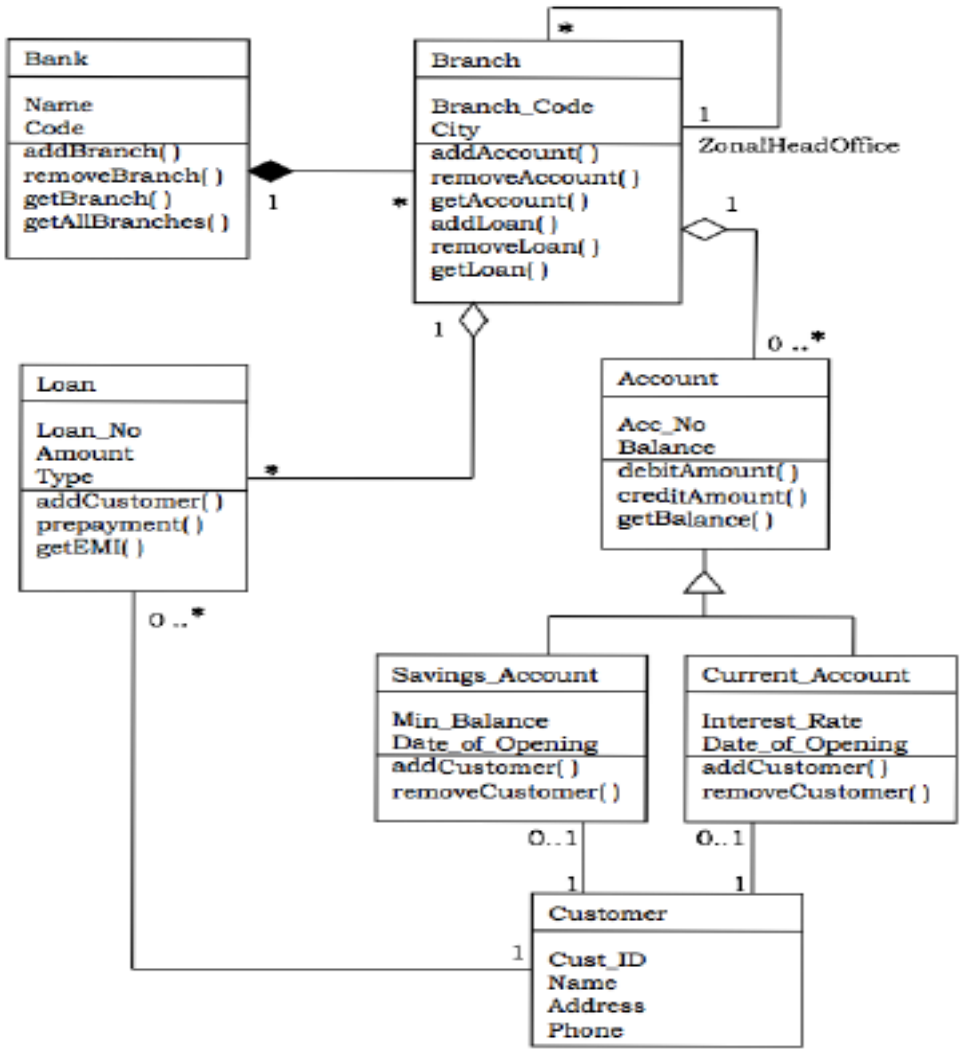
- A Bank "*has–a*" number of Branches: *composition*, *one–to–many*
- A Branch with role Zonal Head Office supervises other Branches: *unary association*, *one–to-many*
- A Branch "*has–a*" number of accounts: *aggregation*, *one–to–many*

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

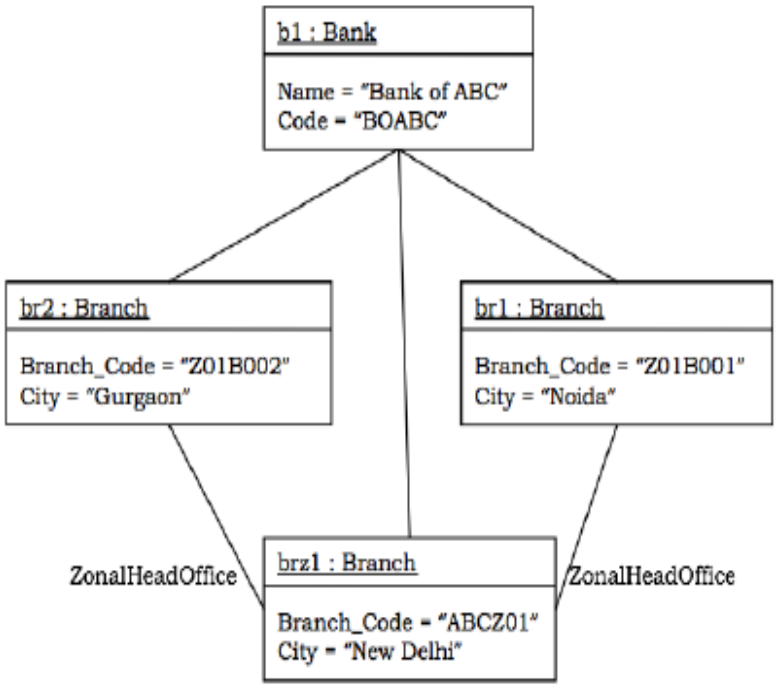
The following figure shows the corresponding class diagram.



OBJECT DIAGRAM

An *object diagram* models a group of *objects* and their *links* at a point of time. It shows the *instances* of the items in a class diagram. Object diagram is the *static* part.

Example: The figure below shows an object diagram of a portion of the class diagram of the Banking System.



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)

Component diagrams are used for:-

- *Constructing systems* through forward and reverse engineering.
- Modeling *configuration management* of source code files while developing a system using an object-oriented programming language
- Representing *schemas* in modeling databases
- Modeling *behaviors* of dynamic systems

Example

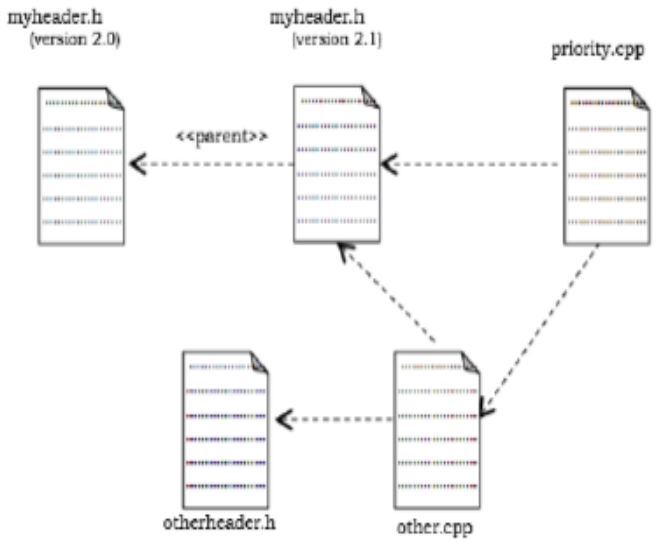
The figure below shows a component diagram to model a system's source code that is developed using C++.

It shows four source code files, namely:-

- myheader.h
- otherheader.h
- priority.cpp
- other.cpp.

Two versions of myheader.h are shown, tracing from the recent version to its ancestor.

The file priority.cpp has compilation dependency on other.cpp. The file other.cpp has compilation dependency on otherheader.h.



DEPLOYMENT DIAGRAM

A *deployment diagram* puts emphasis on the configuration of runtime processing nodes and their components that live on them. They are commonly comprised of *nodes* and *dependencies*, or *associations* between the nodes.

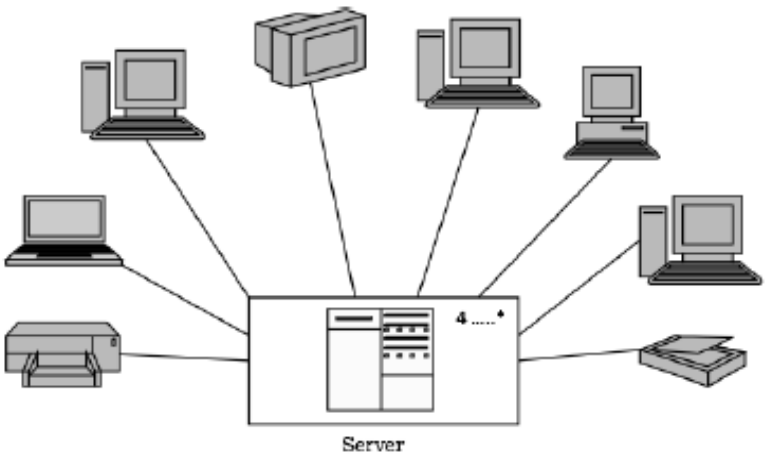
Deployment diagrams are used to:-

- *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

The figure below shows the topology of a computer system that follows client/server architecture. The figure illustrates a node stereotyped as *server* that comprises of processors.

The figure indicates that four or more servers are deployed at the system. Connected to the server are the client nodes, where each node represents a terminal device such as workstation, laptop, scanner, or printer. The *nodes* are represented using icons that clearly depict the real-world equivalent.



2. UML BEHAVIORAL DIAGRAMS

UML *behavioral diagrams* ***visualize***, ***specify***, *construct*, and ***document*** the dynamic aspects of a system. The behavioral diagrams are categorized as follows:-

- Use case diagrams
- Interaction diagrams
- State–chart diagrams
- Activity diagrams.

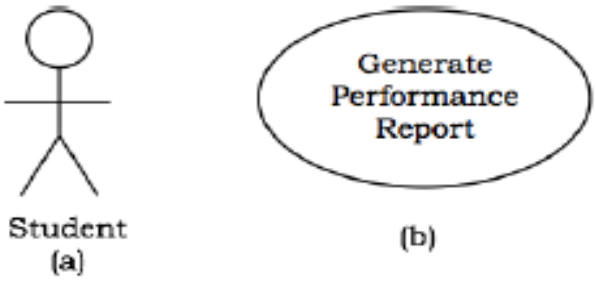
USE CASE MODEL

USE CASE

A use case describes the *sequence of actions* a system performs to achieve visible results. It shows the *interaction* of things outside the system. Use cases may be applied to the whole system as well as a part of the system.

ACTOR

An *actor* represents the roles that the *users* of the use cases play. An actor may be a *person* (e.g. student, customer), a *device* (e.g. workstation), or *another system* (e.g. banking systems). The figure below shows the notations of an actor named *Student* and a use case called *Generate Performance Report.*



A) USE CASE DIAGRAMS

Use case diagrams present an *outside view* of the manner the elements in a system *behave* and *how* they can be used in the context.

Use case diagrams comprise of the following:-

- Use cases
- Actors
- Relationships like dependency, generalization, and association

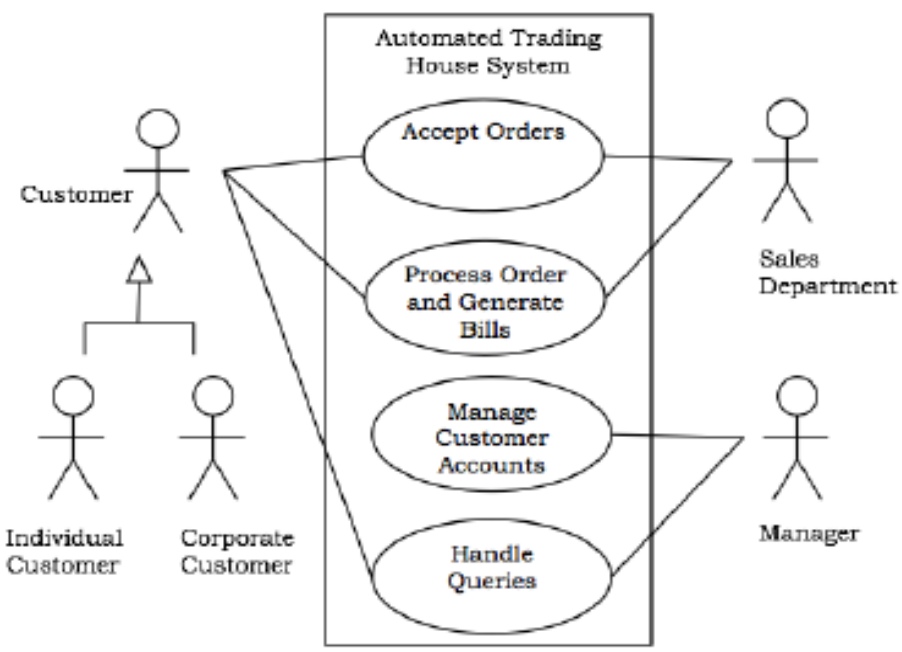
Use case diagrams are used to:-

- 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.
- Model the *requirements* of a system from the outside point of view.

Example

Let us consider an Automated Trading House System. We assume the following features of the system:-

- The trading house has transactions with two types of customers, *individual* customers and *corporate* customers.
- Once the customer places an *order*, it is processed by the *sales department* and the customer is given the *bill*.
- The system allows the *manager* to manage *customer accounts* and responds to any *queries* posted by the customer.



B) Interaction Diagrams

Interaction diagrams depict *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

Interaction diagrams are used for modeling:-

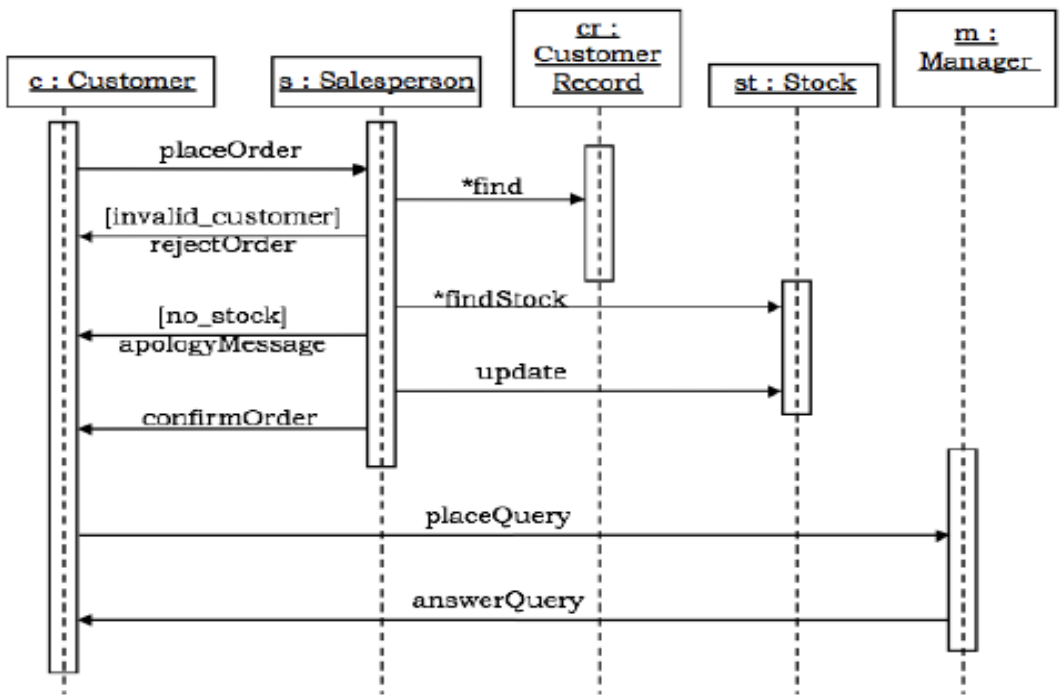
- The control flow by *time ordering* using *sequence diagrams*
- The control flow of *organization* using *collaboration diagrams*

I) Sequence Diagrams

Sequence diagrams are *interaction diagrams* that illustrate the *ordering of messages* according to time.

Notations: These diagrams are in the form of *two-dimensional* charts i.e. x and y axis. 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.

Example: A *sequence diagram* for the Automated Trading House System is shown in the following figure.

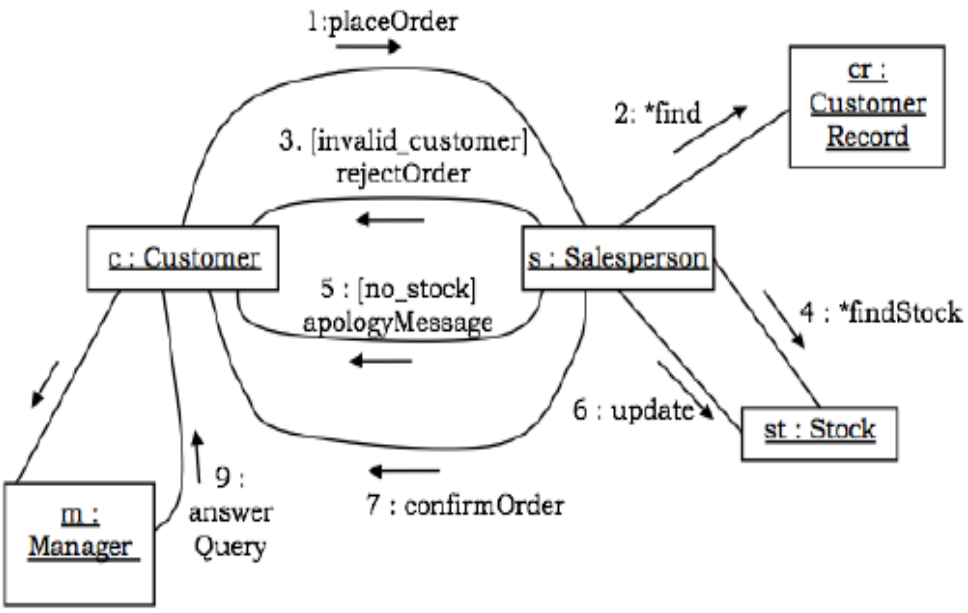


II) COLLABORATION DIAGRAMS

Collaboration diagrams are interaction diagrams that illustrate the structure of the *objects* that *send* and *receive* messages.

Notations: In 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 illustrates as a labeled arrow.

Example: Collaboration diagram for the Automated Trading House System is illustrated in the figure below.



C) STATE–CHART DIAGRAMS

A *state–chart diagram* shows a state machine that *depicts* the *control flow* of an object from one state to another. A *state machine* portrays the *sequences of states* which an object undergoes due to *events* and their *responses* to events.

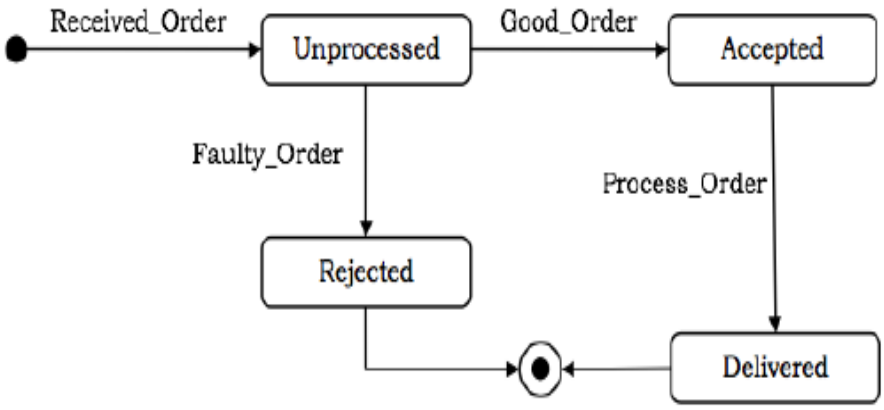
State–Chart Diagrams comprise of:-

- *States*: Simple or Composite
- *Transitions* between states
- *Events* causing transitions
- *Actions* due to the events

State-chart diagrams are used for *modeling objects* which are *reactive* in nature.

Example

In the Automated Trading House System, let us model Order as an object and trace its sequence. The figure below illustrates the corresponding state–chart diagram.



D) ACTIVITY DIAGRAMS

An *activity diagram* depicts the *flow of activities* which are ongoing *non-atomic* operations in a state machine. Activities result in actions which are atomic operations. Activity diagrams comprise of:-

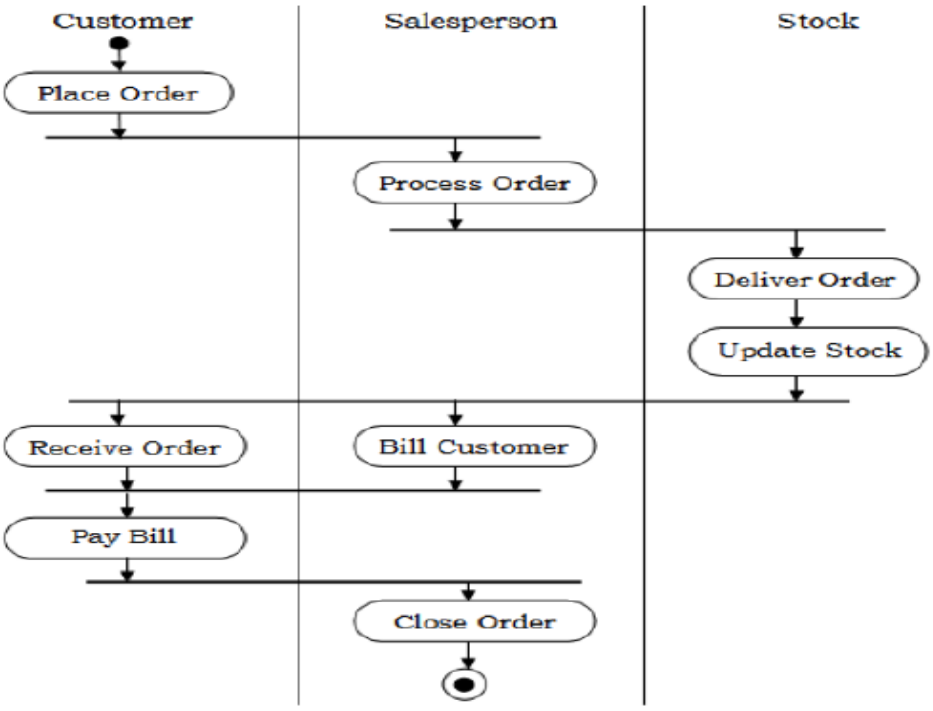
- Activity states and action states
- Transitions
- Objects

Activity diagrams are used for modeling:-

- Workflows as viewed by actors, interacting with the system.
- Details of operations or computations using flowcharts

Example

The figure below shows an *activity diagram* of a portion of the Automated Trading House System.


Customer
Salesperson
Stock
Place Order
Process Order
Deliver Order
Update Stock
Receive Order
Bill Customer
Pay Bill
Close Order