AMITY UNIVERSITY
UTTAR PRADESH

Software Engineering [IT301]

Practical File

Submitted By:

Anchal kumari (A12405218083)

Submitted To:

Rakesh Garg

AMITY SCHOOL OF ENGINEERING & TECHNOLOGY AMITY UNIVERSITY

CAMPUS, SECTOR- 125, NOIDA- 201303

Draw class, sequence, Activity diagram of bankingsystem

Class Diagram

A **Class Diagram** in the Unified Modelling Language (UML) is a type of **static structure** diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

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.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. The diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

A UML class diagram is made up of:

- A set of **classes** and
- A set of **relationships** between classes

A class notation consists of three parts:

- **Class Name:** The name of the class appears in the first partition.
- **Class Attributes:** Attributes are shown in the second partition, and attribute type is shown after the colon. Attributes map onto member variables (data members) in code.
- **Class Operations** (Methods)**:** Operations are shown in the third partition. They are the services that the class provides. The return type of a method is shown after the colon at the end of the method signature. The return type of method parameters is shown after the colon following the parameter name. Operations map onto class methods in code

A class may be involved in one or more relationships with other classes. A relationship can be one of the following types:

- Inheritance or Generalization
- Simple Association
- Aggregation
- Composition
- Dependency

Diagram:

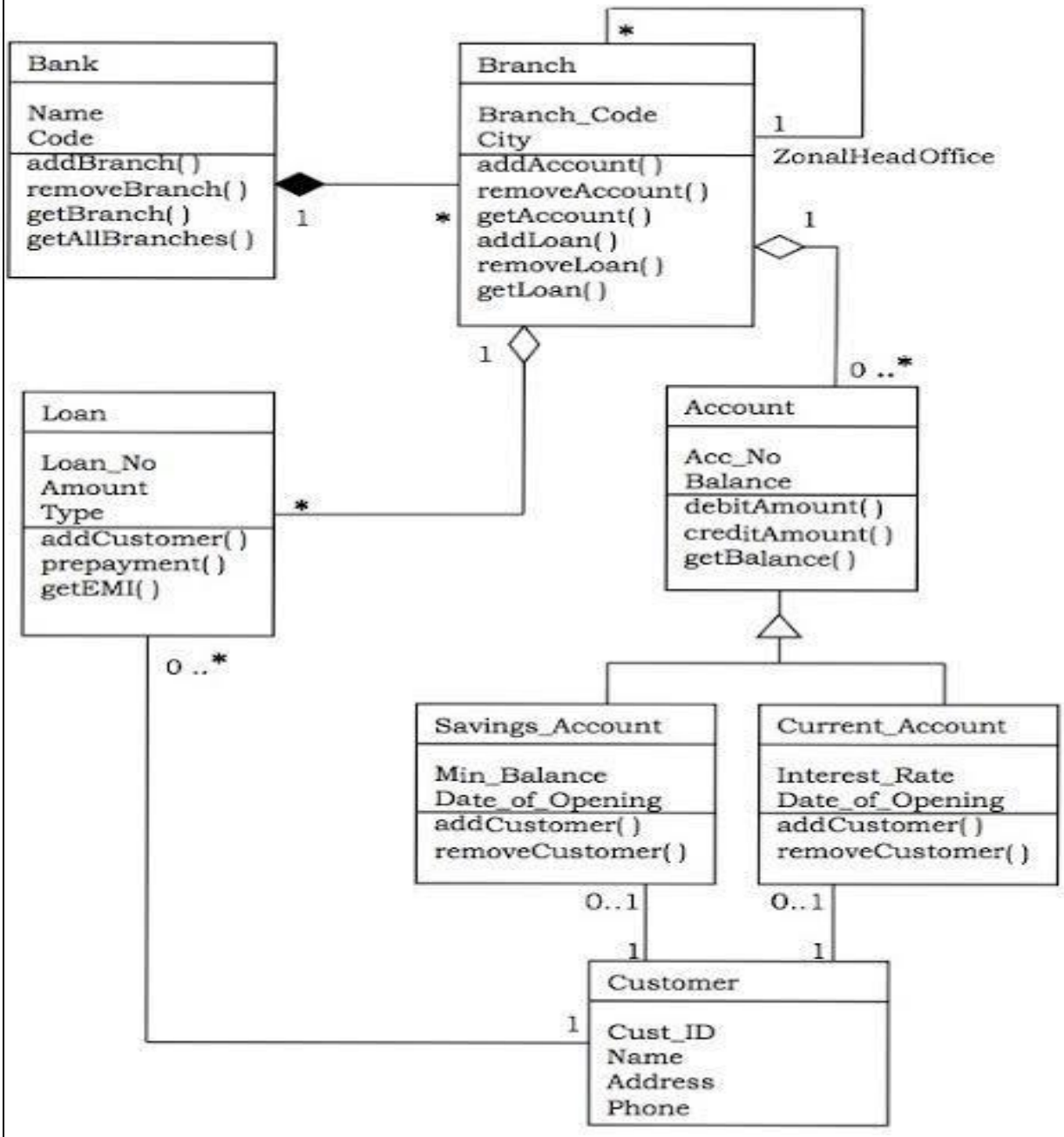

Fig : 1

Sequence Diagram

Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are **time focus** and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

The sequence diagram represents the flow of messages in the system. 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.

An activity diagram notation consists of the following components:

- **Actor:** A role played by an entity that interacts with the subject is called as an actor. It is out of the scope of the system. It represents the role, which involves human users and external hardware or subjects. An actor may or may not represent a physical entity, but it purely depicts the role of an entity. Several distinct roles can be played by an actor or vice versa.
- **Lifeline:** An individual participant in the sequence diagram is represented by a lifeline. It is positioned at the top of the diagram. In UML, the lifeline is represented by a vertical bar, whereas the message flow is represented by a vertical dotted line that extends across the bottom of the page. It incorporates the iterations as well as branching.
- **Activation:** It is represented by a thin rectangle on the lifeline. It describes that time period in which an operation is performed by an element, such that the top and the bottom of the rectangle is associated with the initiation and the completion time, each respectively.
- **Messages:** The messages depict the interaction between the objects and are represented by arrows. They are in the sequential order on the lifeline. The core of the sequence diagram is formed by messages and lifelines.

  Following are the types of messages:

  - **Call Message:** It defines a particular communication between the lifelines of an interaction, which represents that the target lifeline has invoked an operation.
  - **Return Message:** It defines a particular communication between the lifelines of interaction that represent the flow of information from the receiver of the corresponding caller message.
  - **Self-Message:** It describes a communication, particularly between the lifelines of an interaction that represents a message of the same lifeline, has been invoked.
  - **Recursive Message:** A self-message sent for recursive purpose is called a recursive message. In other words, it can be said that the recursive message is a special case of the self-message as it represents the recursive calls.
  - **Create Message:** It describes a communication, particularly between the lifelines of an interaction describing that the target (lifeline) has been instantiated.
  - **Destroy Message:** It describes a communication, particularly between the lifelines of an interaction that depicts a request to destroy the lifecycle of the target.
  - **Duration Message:** It describes a communication particularly between the lifelines of an interaction, which portrays the time passage of the message while modelling a system.

- **Note:** A note is the capability of attaching several remarks to the element. It basically carries useful information for the mode

Diagram:

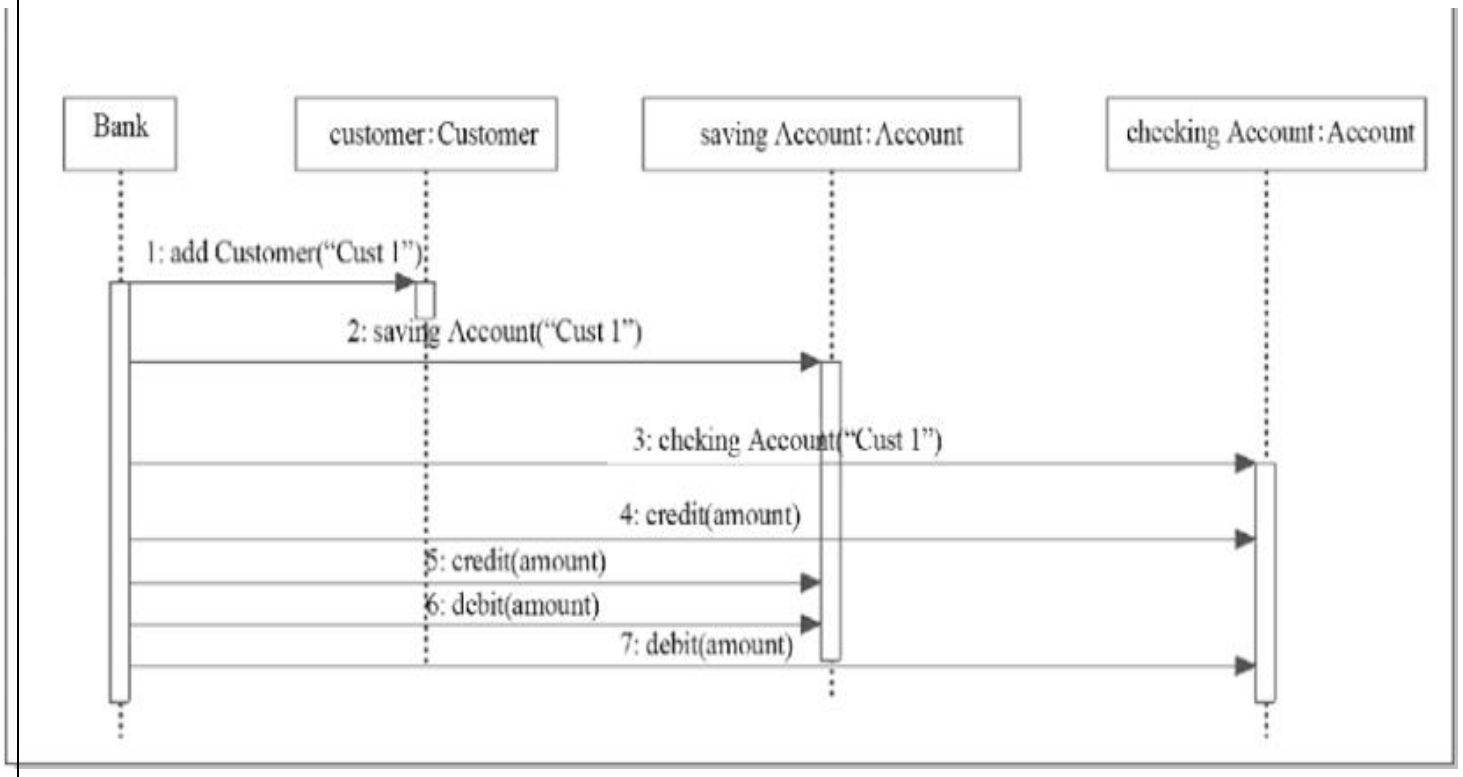

Fig: 2

Activity Diagram

Activity diagram is another important behavioural diagram in UML diagram to describe **dynamic aspects** of the system. Activity diagram is essentially an advanced version of flow chart that modelling the flow from one activity to another activity.

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another,in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases co-ordinate to represent business workflows

An activity diagram notation consists of the following components:

- **Activity:** The categorization of behaviour into one or more actions is termed as an activity. In other words, it can be said that an activity is a network of nodes that are connected by edges. The edges depict the flow of execution. It may contain action nodes, control nodes, or object nodes.
- **Swimlane and Partition:** The swimlane is used to cluster or group all the related

activities in one column or one row. It can be either vertical or horizontal. It is used to add modularity to the activity diagram. It is not necessary to incorporate swimlane in the activity diagram. But it is used to add more transparency to the activity diagram.

- **Fork Node:** A fork node consists of one inward edge and several outward edges. It is the same as that of various decision parameters. Whenever a data is received at an inward edge, it gets copied and split crossways various outward edges. It split a single inward flow into multiple parallel flows.
- **Join Node:** Join nodes are the opposite of fork nodes. A Logical AND operation is performed on all of the inward edges as it synchronizes the flow of input across one single output (outward) edge.

An Activity diagram constitutes the following notations:

- **Initial Node:** It depicts the initial stage or beginning of a set of actions or activities.
- **Activity Final Node:** It is the stage where all the control flows and object flows end.
- **Object Node:** It represents an object that is connected to a set of Object Flows
- **Decision Node:** It represents a test condition to ensure that the control flow or object flow only goes down one path.

Action: It represents the set of actions or tasks that are to be performed.

- **Control Flow:** Shows the sequence of execution
- **Object Flow:** Show the flow of an object from one activity (or action) to another activity (or action).