

Slide 4

UML DIAGRAMS

USE CASE

ACTIVITY

USE CASE DIAGRAM

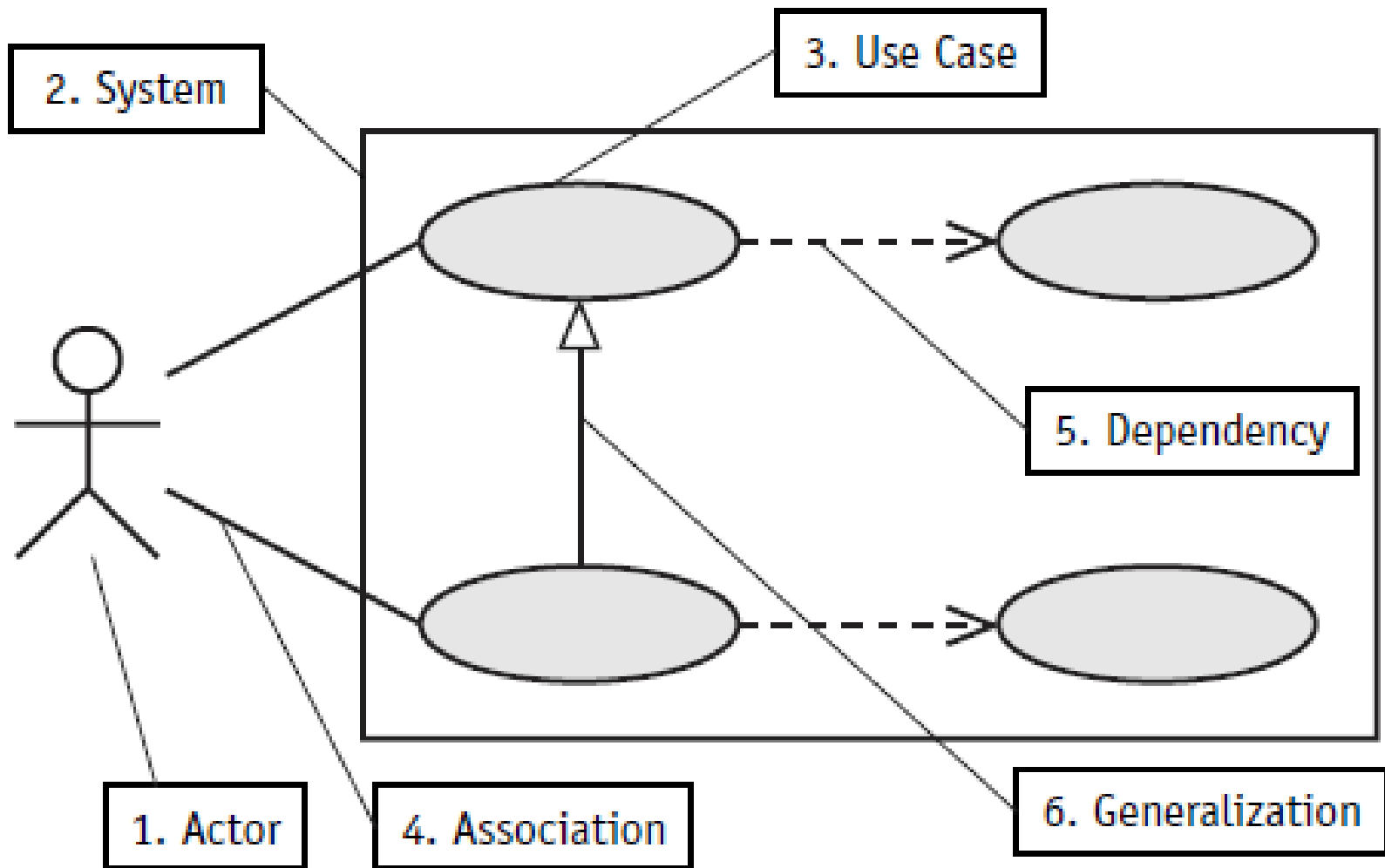
- The use case diagram is used to identify the functionality provided by the system (use cases), the users who interact with the system (actors) and the association between the users and the functionality.
- Use case diagrams are used in the analysis phase to articulate the high level requirements of the system.

USE CASE DIAGRAM

Purpose

- Provides a high level view of the system- It describes the proposed functionality of the new system.
- Identify the users of the system
- Determine areas which require Human-Computer Interaction - It shows the interaction of people or external device with the system under design.

ELEMENTS OF A USE CASE DIAGRAM



Six modeling elements make up the Use Case diagram: systems, actors, Use Cases, associations, dependencies, and generalizations.

System: The system is used to define the scope of the use case and drawn as a rectangle. Sets the boundary of the system in relation to the actors who use it (outside the system) and the features it must provide (inside the system).

Actor: is any entity that performs a role in one given system. This could be a person, organization or an external system that has a stake in the successful operation of the system.

Use Case: Identifies a key feature of the system, which fulfill the user/actor requirements. Each Use Case expresses a goal that the system must achieve.

Association: Identifies an interaction between actors and Use Cases. Each association becomes a dialog that must be explained in a Use Case narrative.

Dependency: Identifies a communication relationship between two Use Cases.

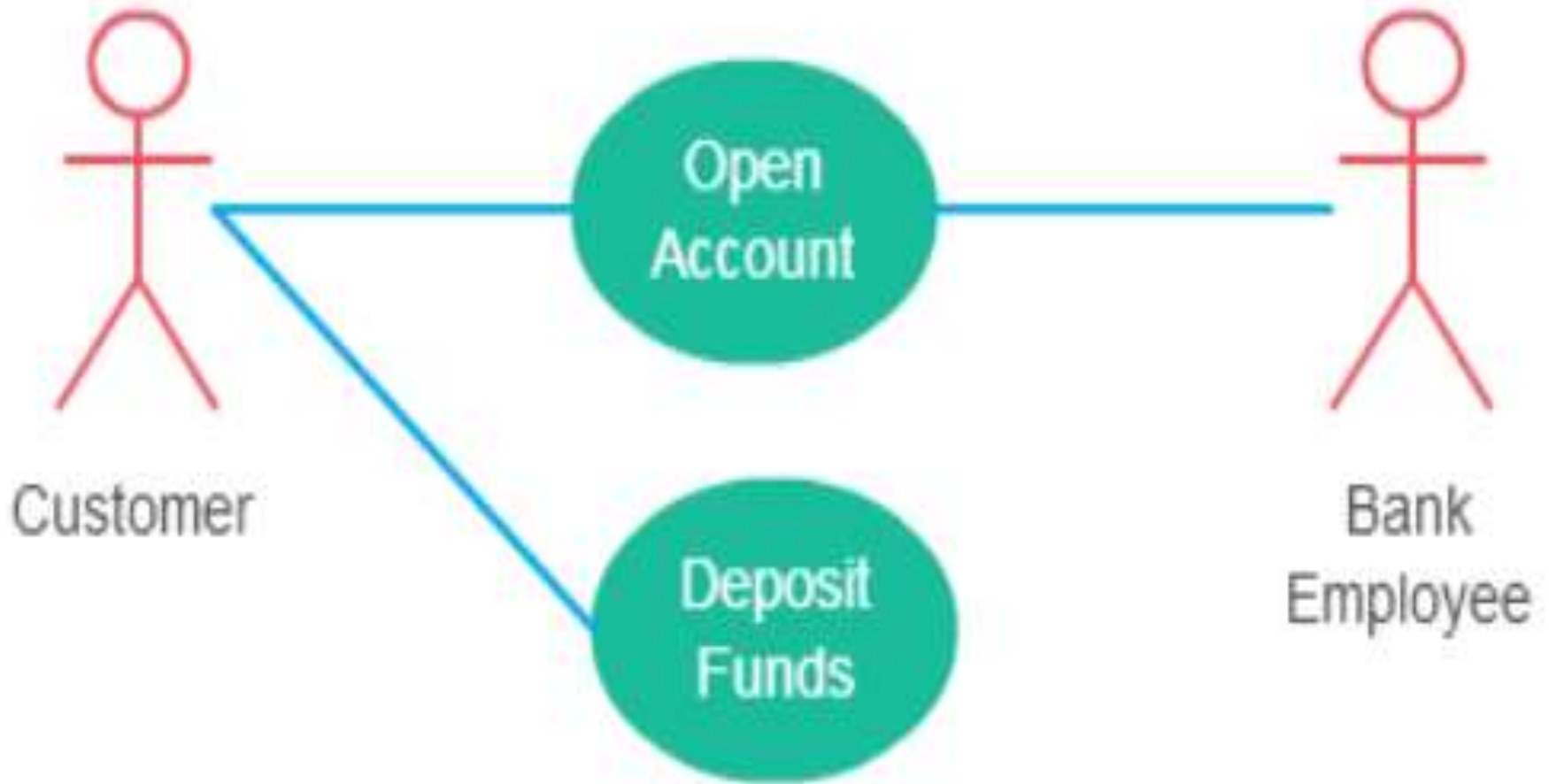
Generalization: Defines a relationship between two actors or two Use Cases where one Use Case inherits and adds to or overrides the properties of the other.

Relationships in Use Case Diagrams

There are five types of relationships in a use case diagram. They are

- Association between an actor and a use case
- Generalization of an actor
- Extend relationship between two use cases
- Include relationship between two use cases
- Generalization of a use case

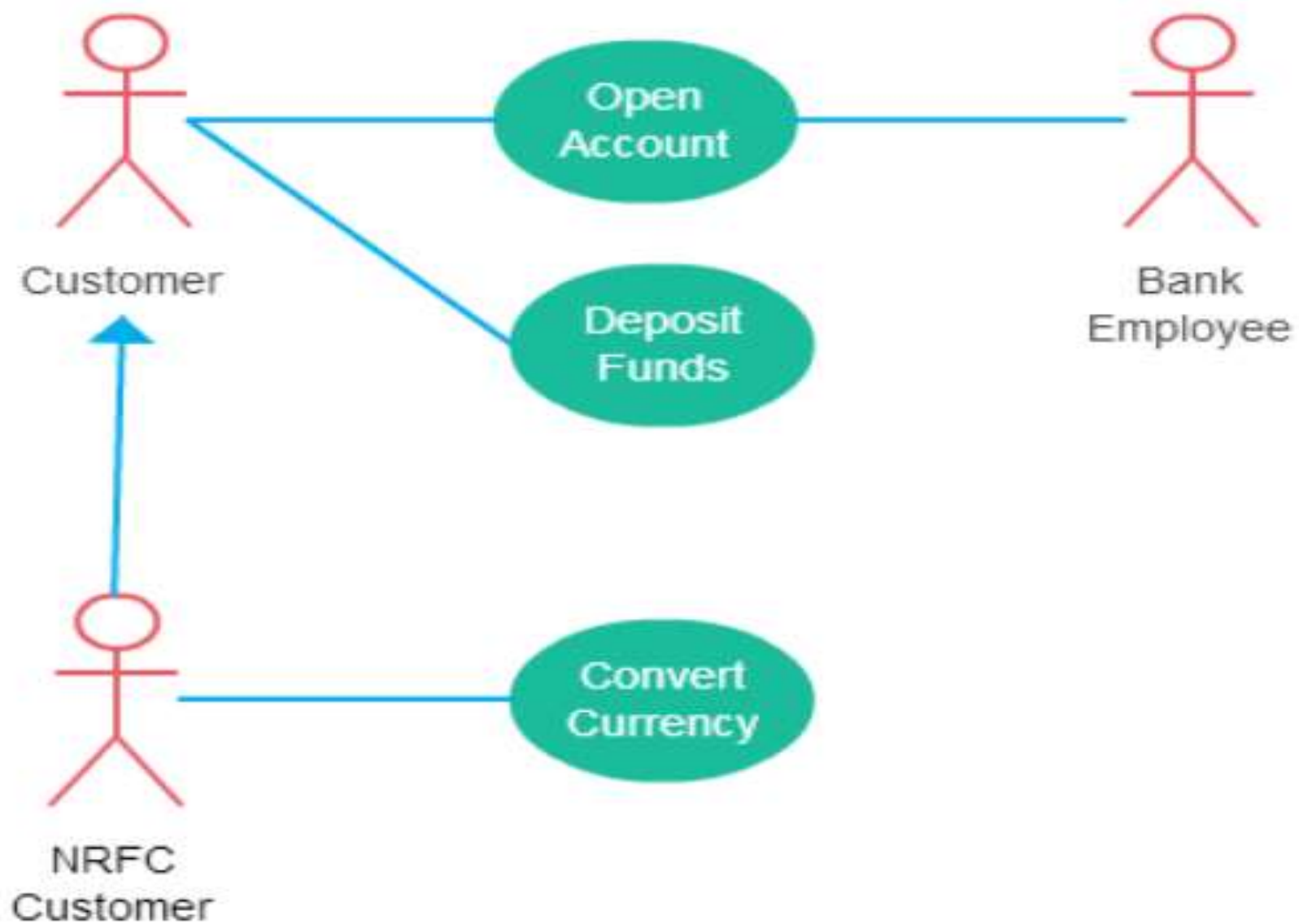
Association Between Actor and Use Case



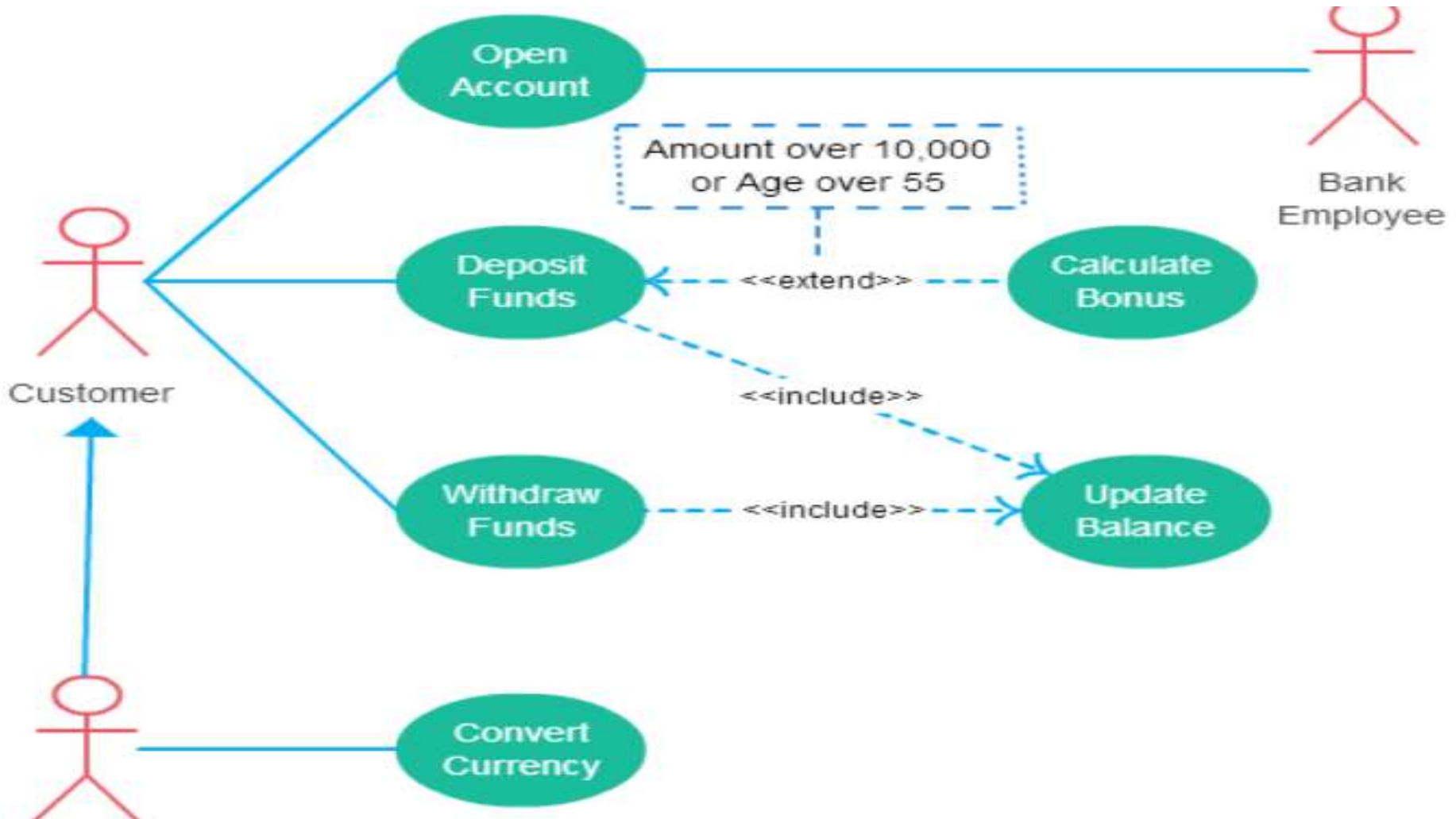
- An association indicates an interaction between an actor and use case
 - An actor must be associated with at least one use case.
 - An actor can be associated with multiple use cases.
 - Multiple actors can be associated with a single use case.

Generalization of an Actor

- Generalization of an actor means that one actor can inherit the role of the other actor.
- The descendant inherits all the use cases of the ancestor.
- The descendant has one or more use cases that are specific to that role.



INCLUDE AND EXTEND RELATIONSHIP




Include Relationship

- Include relationship is modeled between the use cases when a use case **includes** the behavior sequence of another use case. The use cases that need to be described repeatedly are modeled once and included in the other use cases when required.
- Analogous to a function call

Extend relationship

- Specify that one use case adds (extends) the behavior of another use case (base). While the base use case is defined independently and is meaningful by itself, the extension use case is not meaningful on its own.
- The extension use case consists of one or several behavior sequences (segments) that describe additional behavior that can incrementally augment the behavior of the base use case.

<<include>>

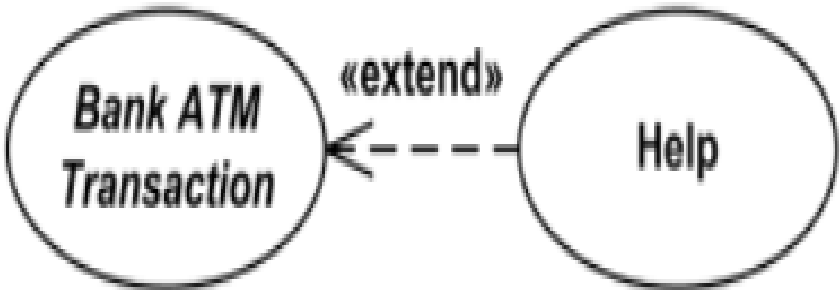
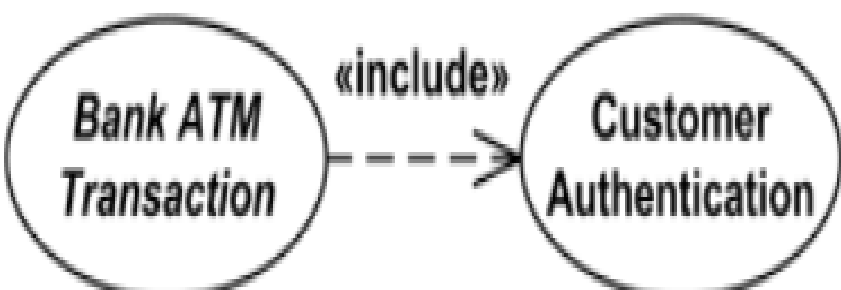


Include relationship between Use Cases (one UC must call another; e.g., Login UC includes User Authentication UC)

<<extend>>

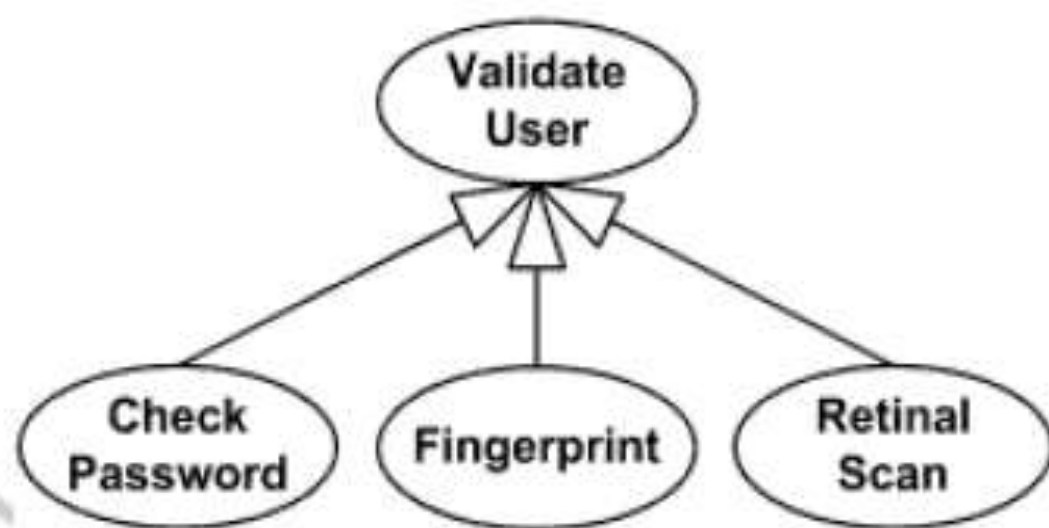


Extend relationship between Use Cases (one UC calls Another under certain condition; think of if-then decision points)

Extend	Include
 <pre> graph LR Help((Help)) -.-> "«extend»" BankATMTransaction((Bank ATM Transaction)) </pre>	 <pre> graph LR BankATMTransaction((Bank ATM Transaction)) -.-> "«include»" CustomerAuthentication((Customer Authentication)) </pre>
Base use case is complete (concrete) by itself, defined independently.	Base use case is incomplete (abstract use case).
Extending use case is optional, supplementary.	Included use case required, not optional.
Has at least one explicit extension location.	No explicit inclusion location but is included at some location.
Could have optional extension condition.	No explicit inclusion condition.

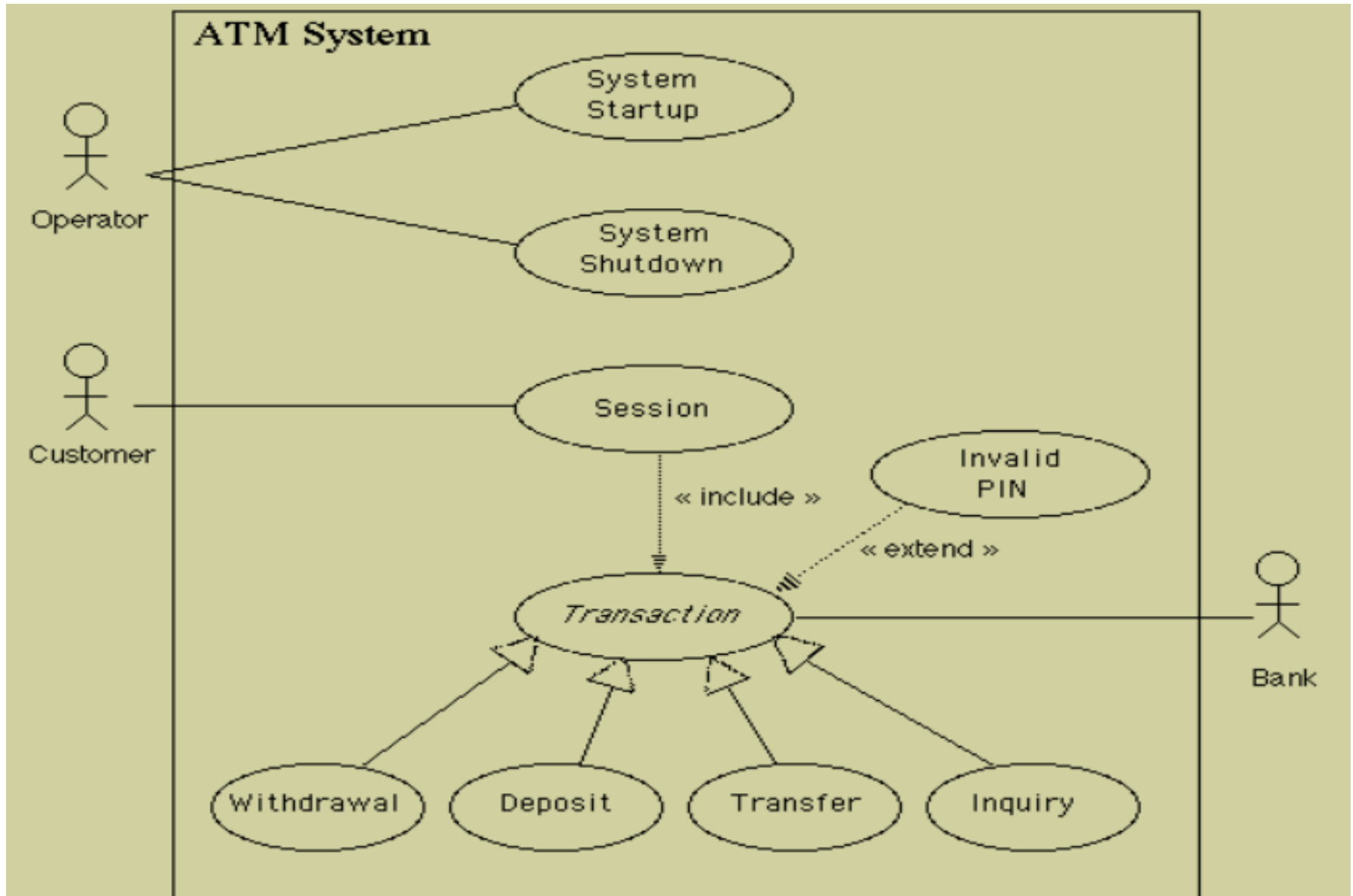
Generalization of a Use Case

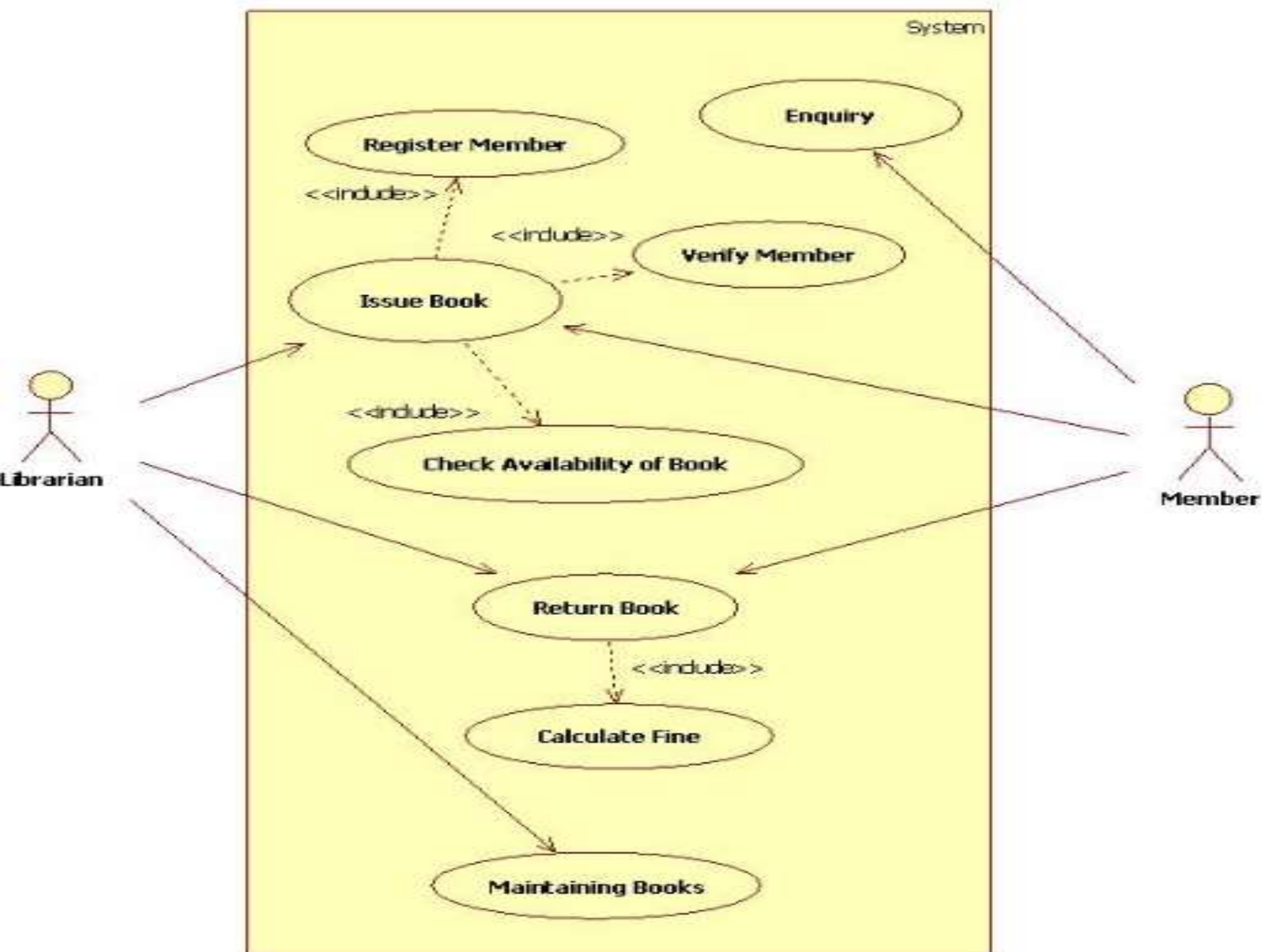
- This is similar to the generalization of an actor. The behavior of the ancestor is inherited by the descendant.
- This is used when there is common behavior between two or more use cases and also specialized behavior specific to each use case.



Generalization between use cases

Sample Use Case Diagram





ACTIVITY DIAGRAMS

- Describes the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.
- Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

- Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram.
- The main element of an activity diagram is the activity itself.
- An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

The purpose of an activity diagram can be described as -

- a) Capture the activity flow of a system.
- b) Describe the sequence from one activity to another.
- c) Describe the parallel, branched and concurrent flow of the system.

Activity Diagram Notations

- Activity diagrams symbols can be generated by using the following notations:
- Initial states: The starting stage before an activity takes place is depicted as the initial state
- Final states: The state which the system reaches when a specific process ends is known as a Final State
- State or an activity box:

- Decision box: It is a diamond shape box which represents a decision with alternate paths. It represents the flow of control.



initial-state



action-box



decision-box

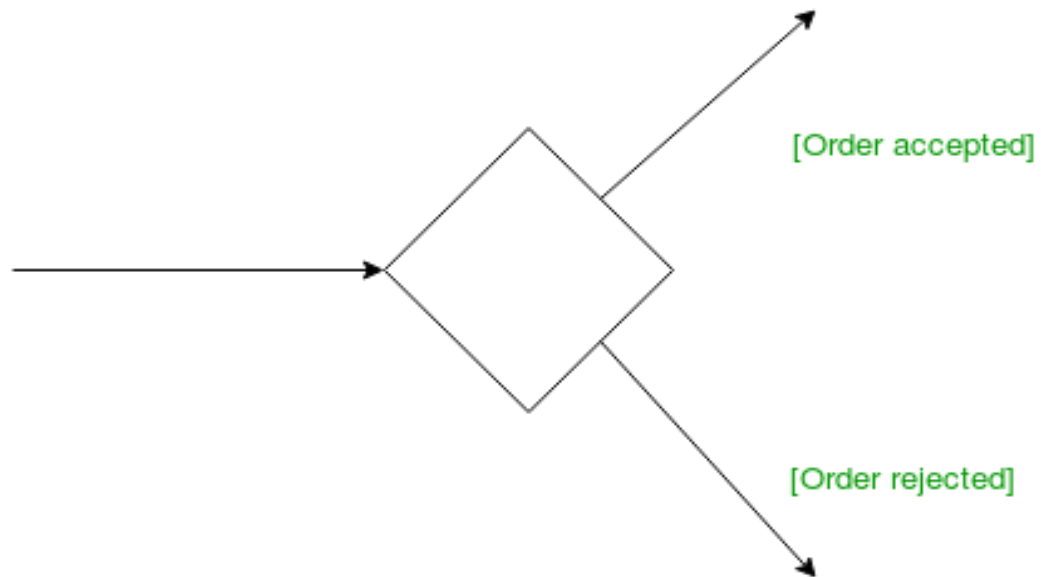


final-state

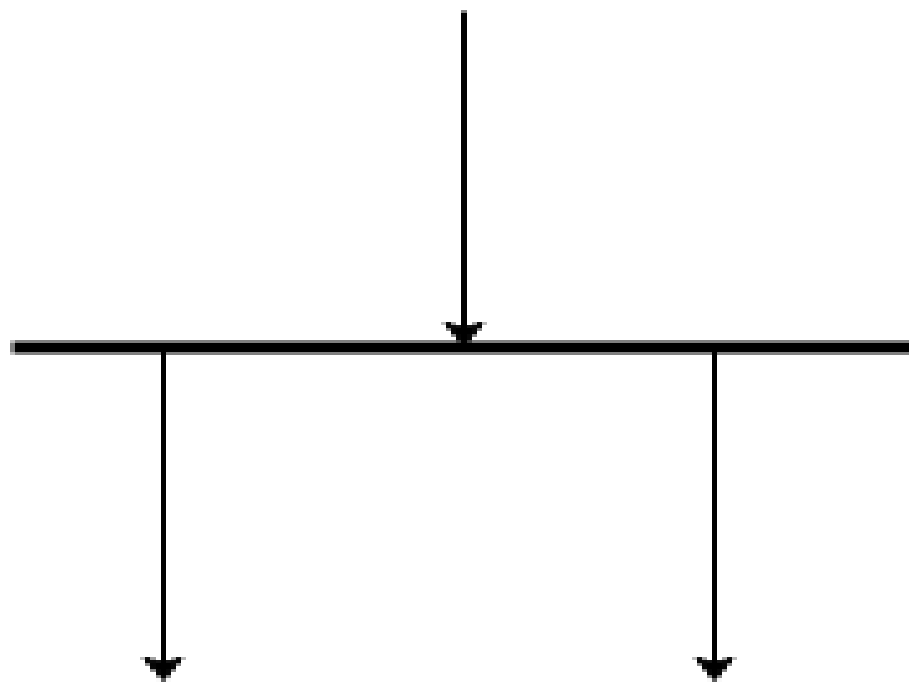
- Fork nodes are used to generate concurrent flows within an activity.

- **Action Flow or Control flows -**
Action flows or Control flows are also referred to as paths and edges. They are used to show the transition from one activity state to another.



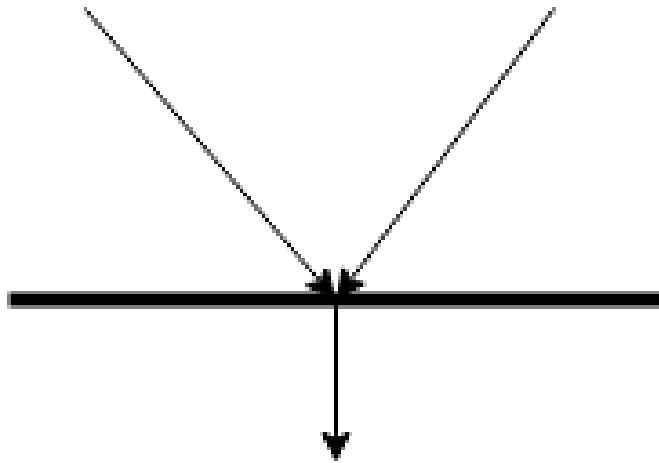


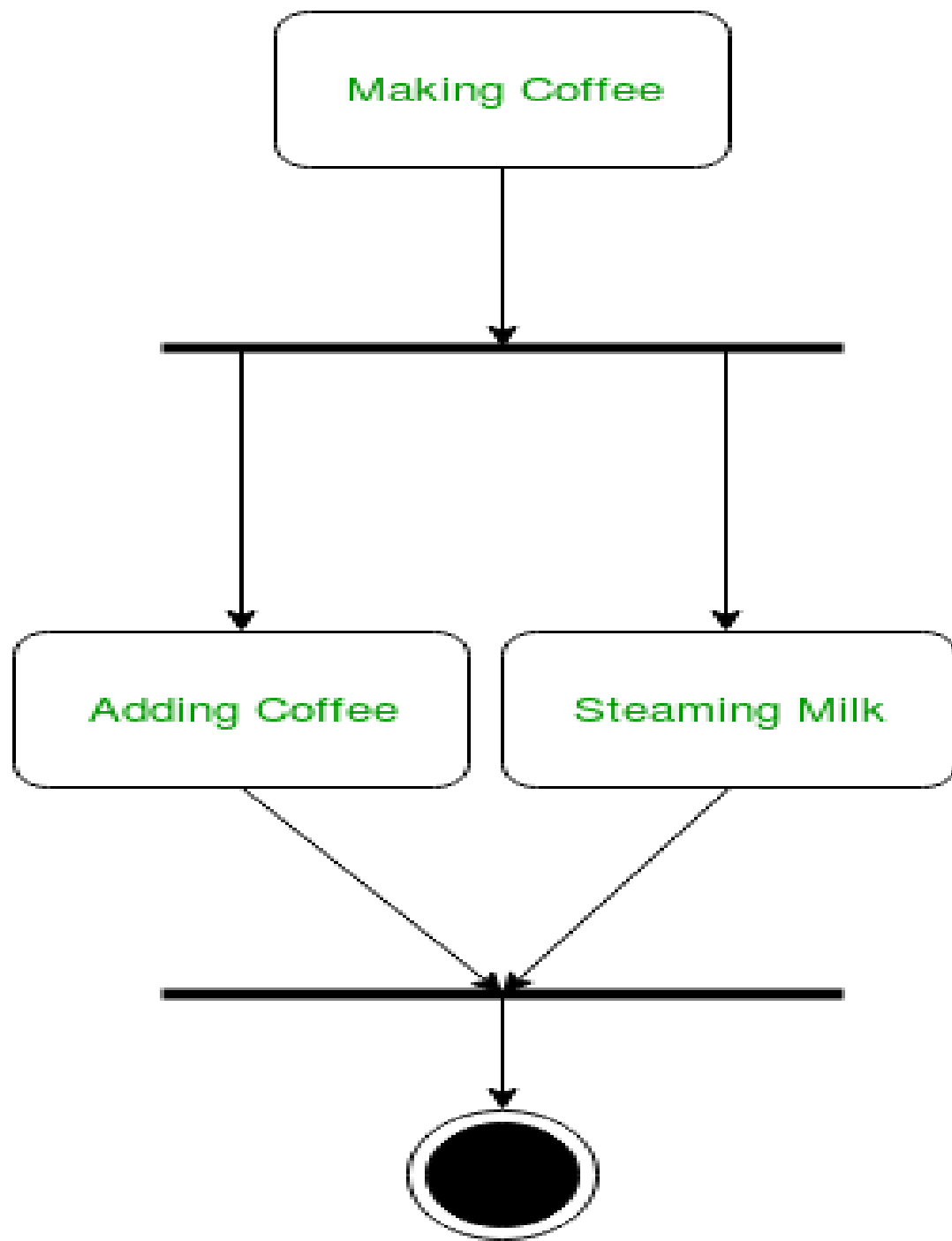
Decision



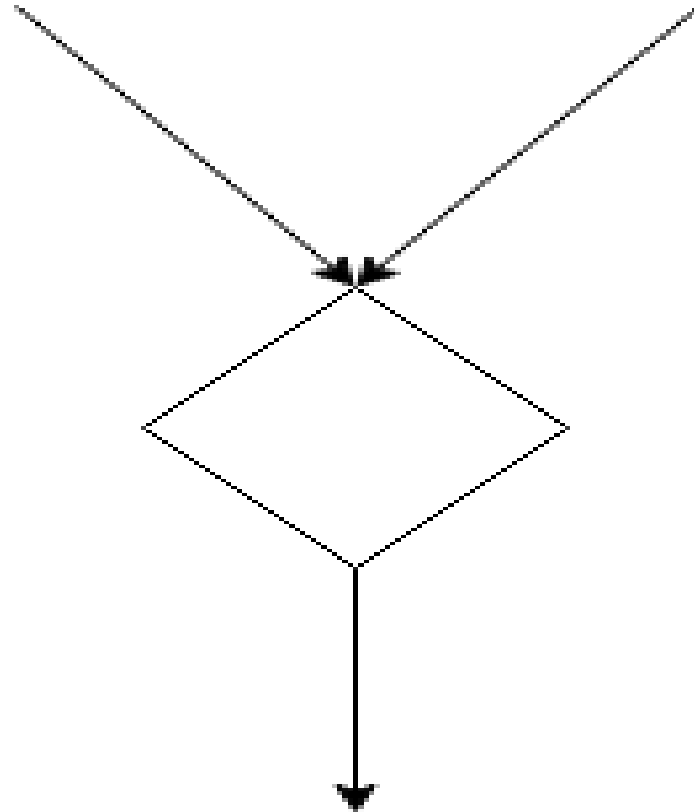
Fork

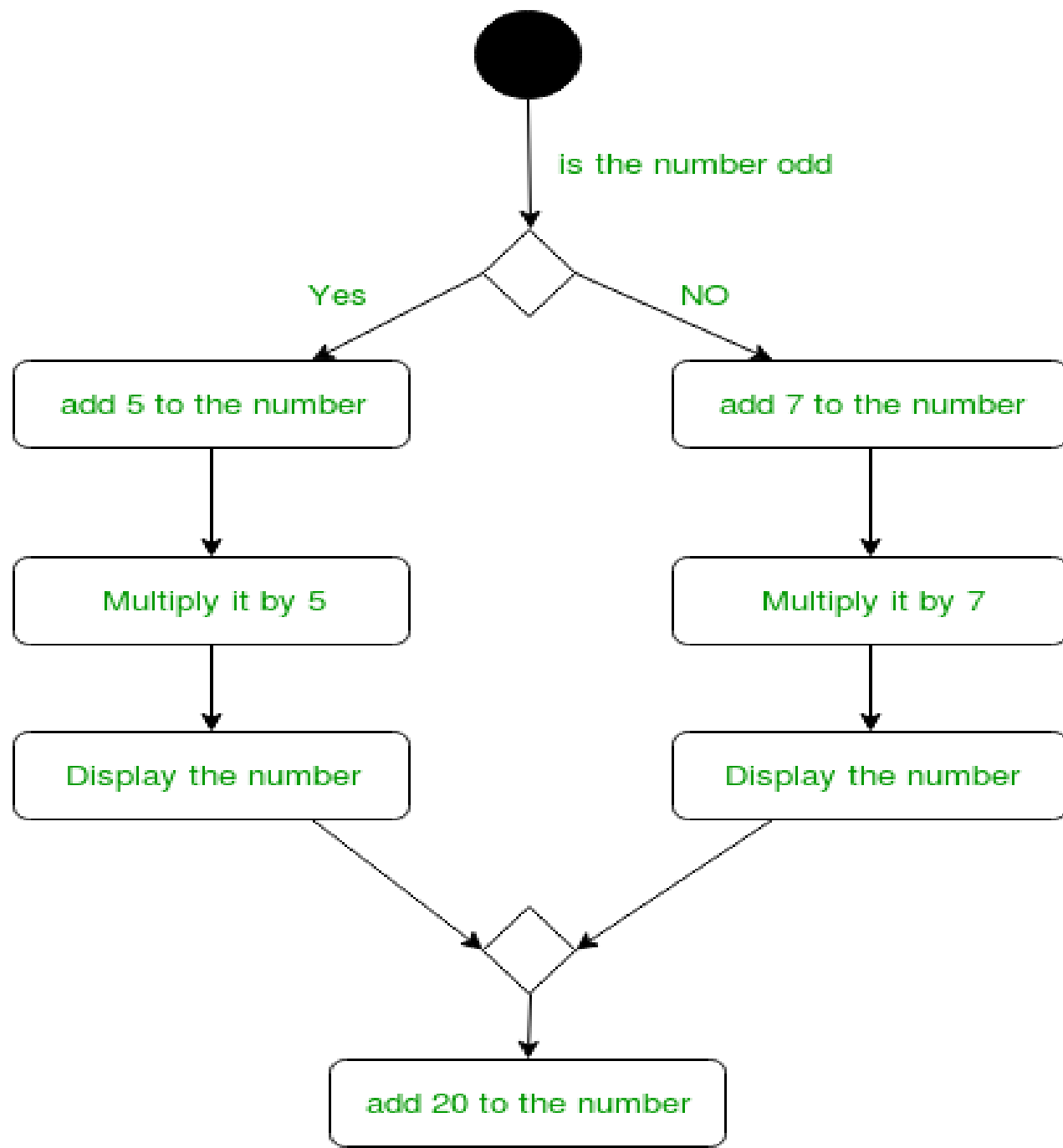
Join - Join nodes are used to support concurrent activities converging into one. For join notations we have two or more incoming edges and one outgoing edge.





Merge or Merge Event – Scenarios arise when activities which are not being executed concurrently have to be merged. We use the merge notation for such scenarios. We can merge two or more activities into one if the control proceeds onto the next activity irrespective of the path chosen





Following is an example of an activity diagram for order management system. In the diagram, four activities are identified which are associated with conditions. One important point should be clearly understood that an activity diagram cannot be exactly matched with the code. The activity diagram is made to understand the flow of activities and is mainly used by the business users

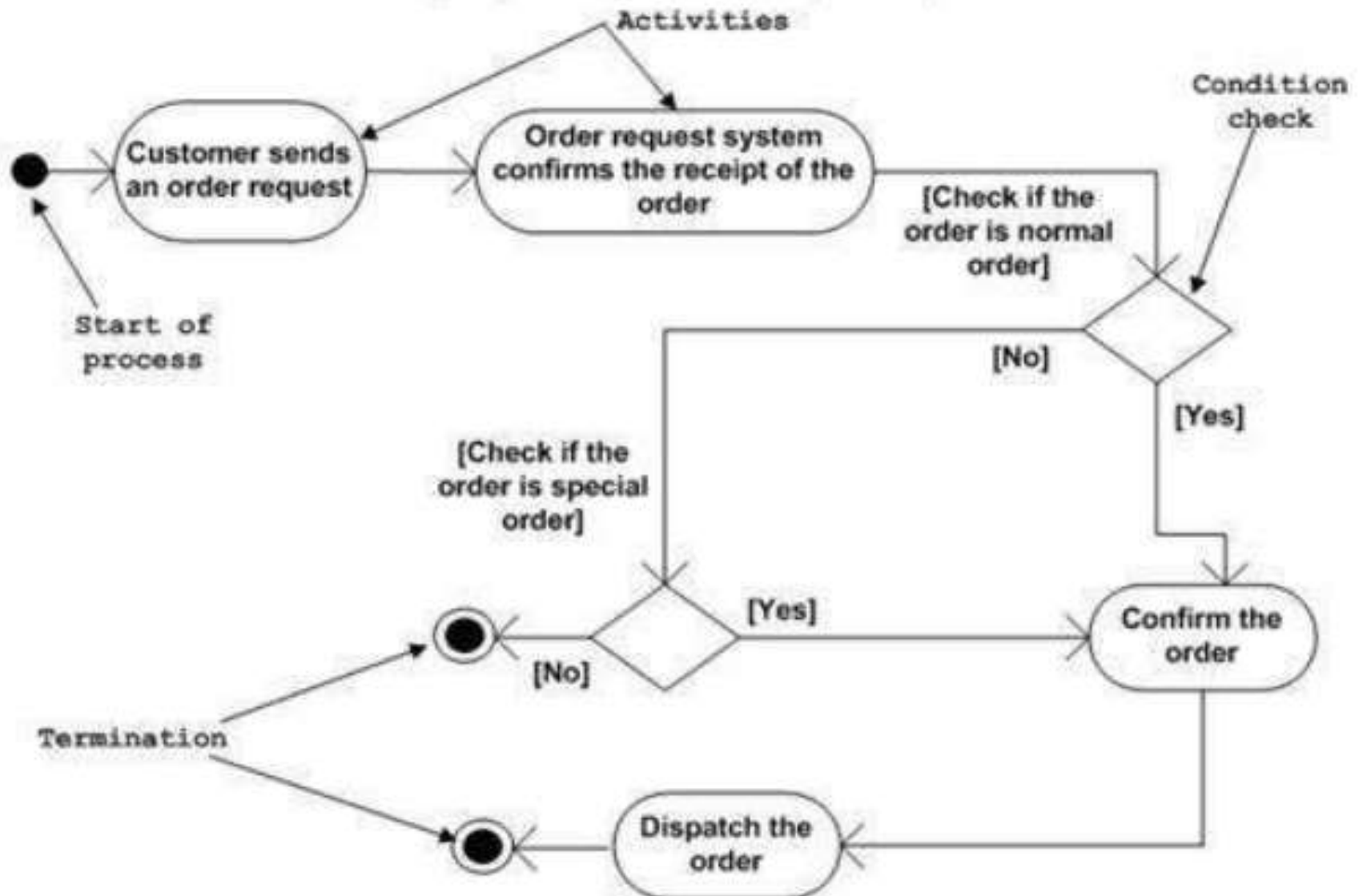
Following diagram is drawn with the four main activities -

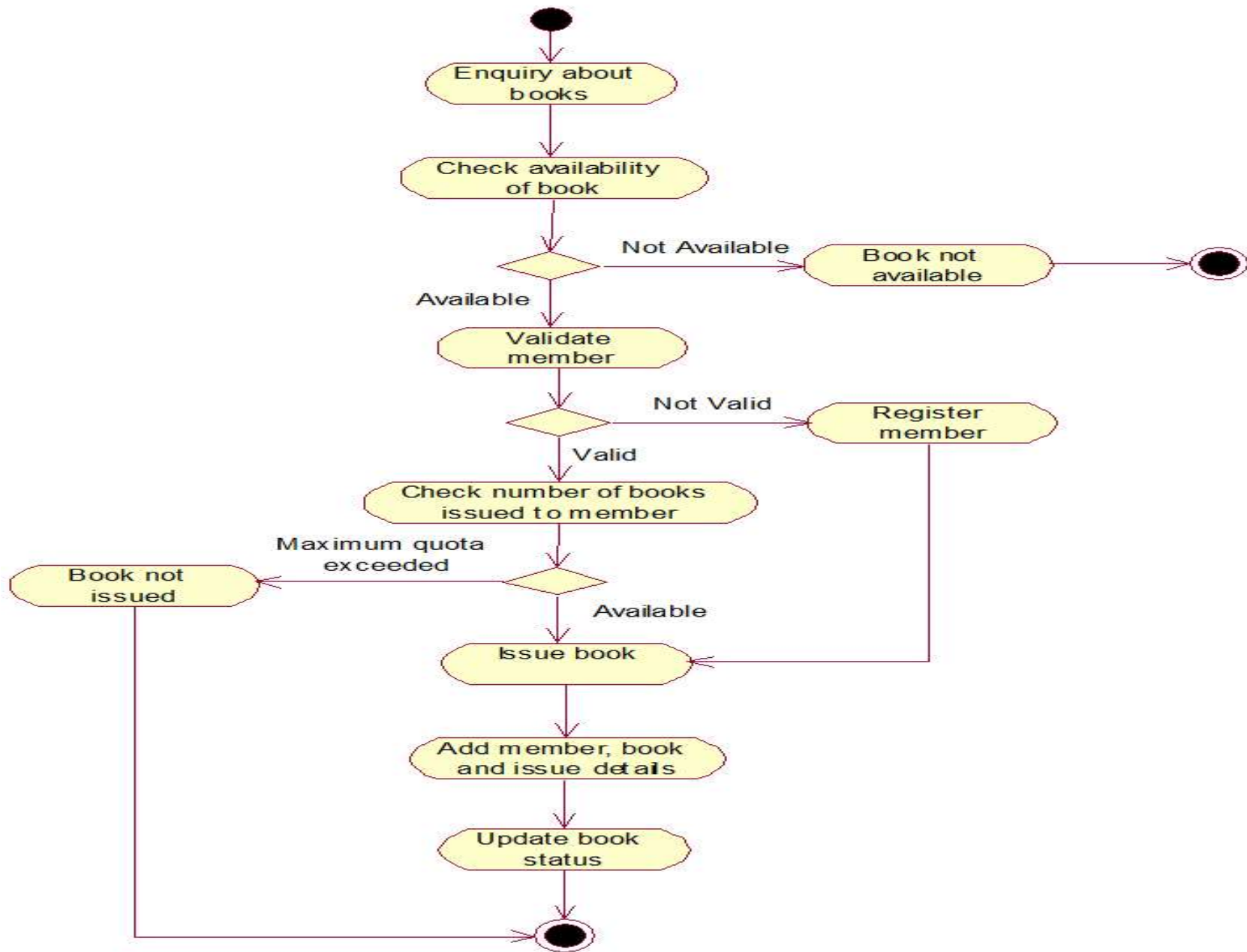
- Send order by the customer
- Receipt of the order
- Confirm the order

Dispatch the order

After receiving the order request, condition checks are performed to check if it is normal or special order. After the type of order is identified, dispatch activity is performed and that is marked as the termination of the process

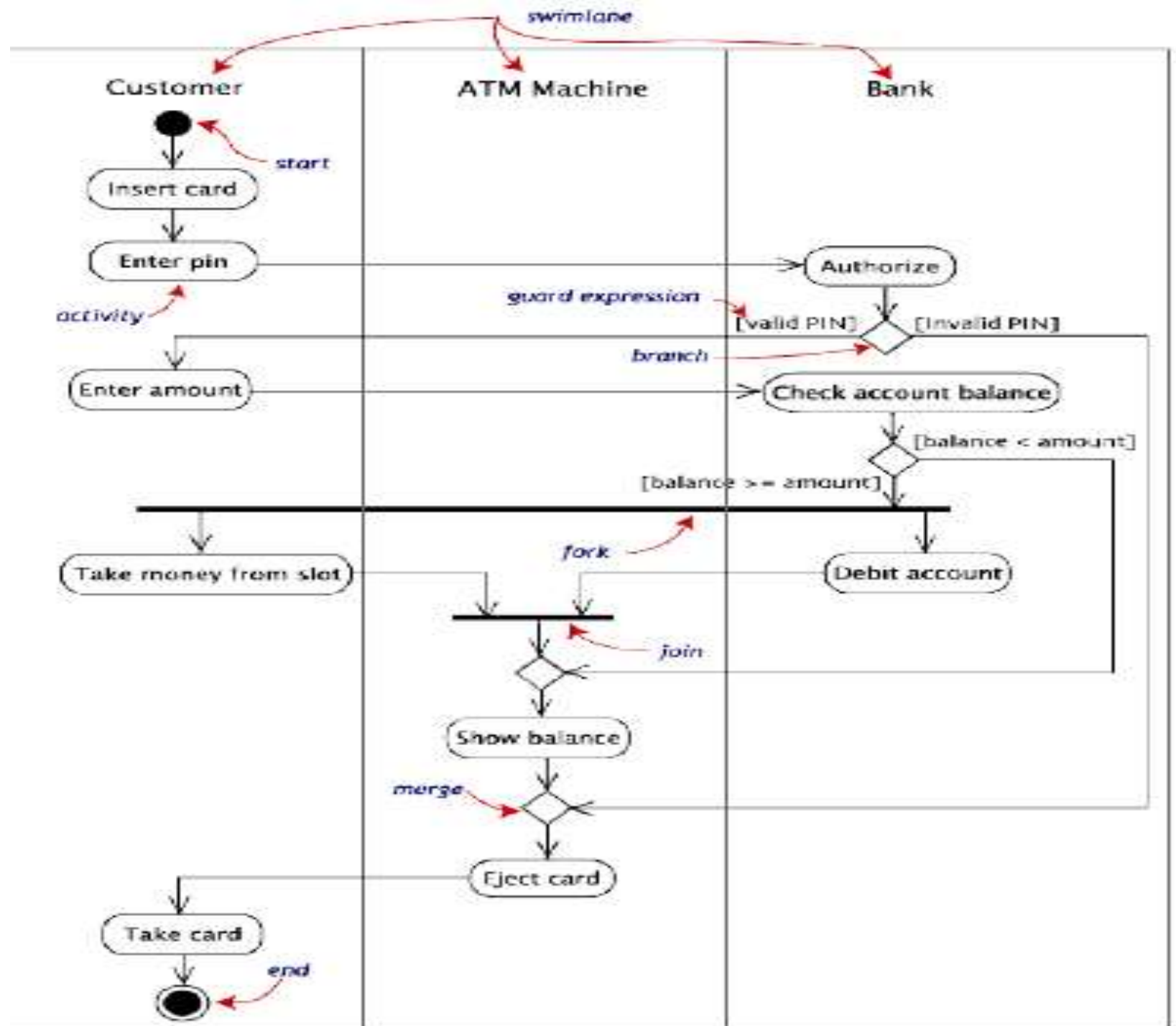
Activity diagram of an order management system





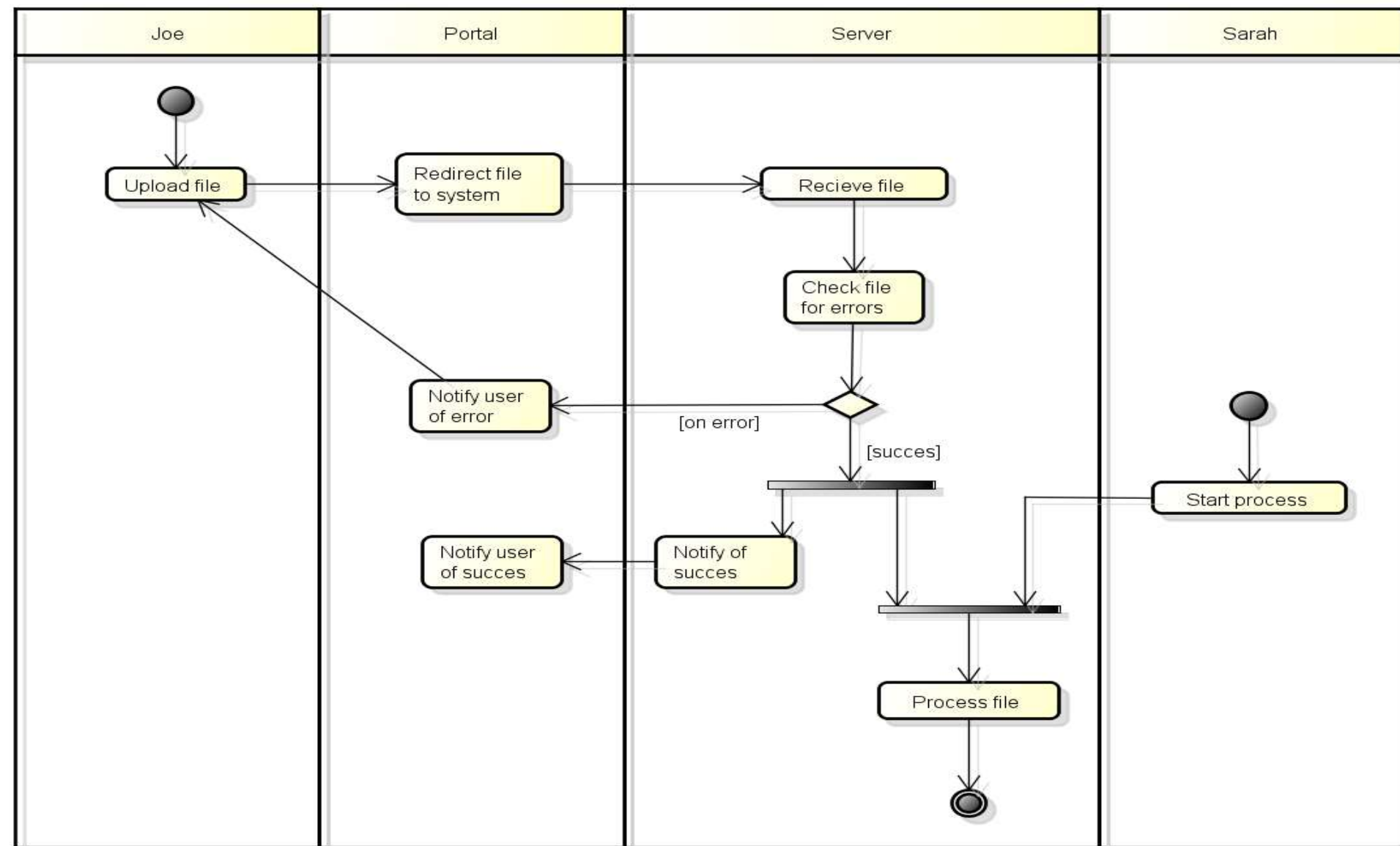
Activity Diagram - Swimlane

- A swimlane is a way to group activities performed by the same actor on an activity diagram . Here is an example of a swimlane activity diagram.



Sample Activity Diagram

- Joe uploads a file into a portal, this portal transfers the file to our server where it is checked for errors. In case of an error the server sends a message to the portal where Joe can see this (if he is still logged in or on his next visit) and upload his file again (hopefully without the errors this time). In case of success the server will also notify the portal but Joe doesn't have to take any action so we are not interested in the result. As the file is okay the system now wait for Sarah who has to start the processing of the file manually after which the process completes. Draw an activity diagram for the system



References



The Unified Modeling Language User Guide

Grady Booch, James Rumbaugh, Ivar Jacobson

Addison-Wesley (International Student Edition)



UML Distilled

Martin Fowler (with Kendall Scott)

Addison-Wesley

END.

Wairagu G.R