Parul® University

Object Oriented Concepts with UML 203105207

Prof. Shaleen Shukla, Assistant Professor
Information Technology Department

Interaction Models

- The class model describes the class & objects in a system and their relationship.
- The state model describes the life cycles of the objects.
- The interaction model describes how the objects interact.
- The interaction model starts with use cases that are then elaborated with sequence and activity diagrams

Interaction Models

Use case: focuses on functionality of a system- i.e, what a system does for users

Sequence diagrams: shows the object that interact and the time sequence of their interactions

Activity diagrams: elaborates important processing steps

Parul®
University

DIGITAL LEARNING CONTENT

CHAPTER-3

Use Case Models

Introduction

- Use cases are description of the functionality of a system from user perspective.
- It shows the relationships between the actors(Users) that use the system, the use cases(functionalities) they use and the relationships between different use cases.

Use Case Diagrams

- Use Case diagrams show the various activities the users can perform on the system.
  - System is something that performs a function.
- They model the dynamic aspects of the system.
- Provides a *user's* perspective of the system

Use-Case Diagrams

A use case is a model of the interaction between External users of a software product (actors) and The software product itself

➢More precisely, an actor is a user playing a specific role describing a set of user scenarios capturing user requirements contract between end user and software developers

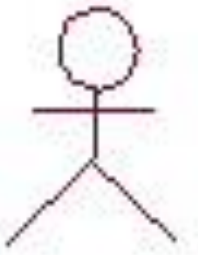
Using Use Case Diagrams

- Use case diagrams are used to visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
- Provide a way for developers, domain experts and end-users to Communicate.
- Serve as basis for testing.
- Use case diagrams contain use cases, actors, and their relationships.

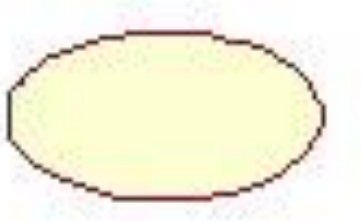
Use-Case Diagrams

- **Actor:** A role that a user plays with respect to the system, including human users and other systems. e.g., inanimate physical objects (e.g. robot); an external system that needs some information from the current system.
- **Use case:** A set of scenarios that describing an interaction between a user and a system, including alternatives.
- **System boundary**: rectangle diagram representing the boundary between the actors and the system.
- **Association:** Communication between an actor and a use case; Represented by a solid line.

Use-Case Diagrams



Actor



Use Case

Actors

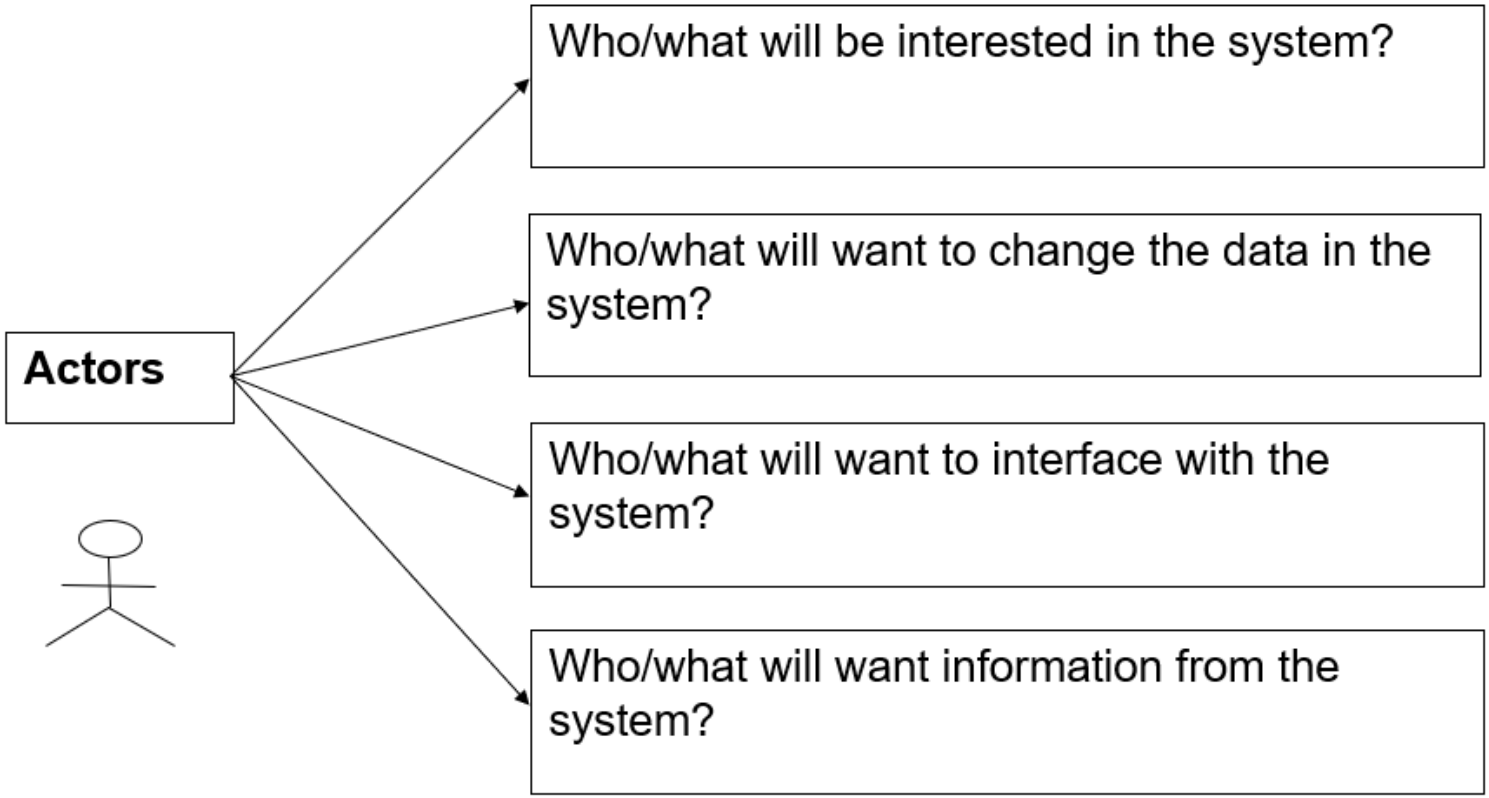
Actors

Could be human beings, other systems, timers and clocks or hardware devices.

Actors that stimulate the system and are the initiators of events are called primary actors (active)

Actors that only receive stimuli from the system are called secondary actors (passive)

Actors


Actors
Who/what will be interested in the system?
Who/what will want to change the data in the system?
Who/what will want to interface with the system?
Who/what will want information from the system?

Use Case Diagram– Guidelines & Caution

1. Avoid showing communication between actors.
2. Actors should be named as singular. i.e student and NOT students. NO names should be used – i.e John, Sam, etc.

Use-Case Diagrams: Actors and Goals

1. Start by identifying the actors of the system

2. Define the goals of the system and how they can be achieved using the systems' actors

3. Illustrate these goals and actors actions using use-case diagram(s)

Use-case Diagram: Use-case

1. A use case describes a sequence of actions a system performs to yield an observable result or value to a particular actor
2. Naming convention = verb + noun (or) verb + noun-phrase,

   e.g. withdraw cash
3. A good use case should:

   Describe a sequence of transactions performed by a system that produces a measurable result (goal) for a particular actor

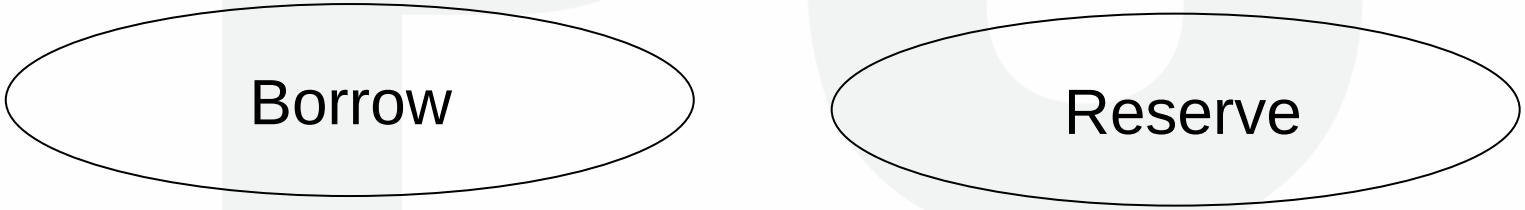
   Describe the behavior expected of a system from a user's perspective

   Enable the system analyst to understand and model a system from a high-level business viewpoint

   Represent the interfaces that a system makes visible to the external entities and the interrelationships between the actors and the system

Use Case Diagrams – Use Cases

1. Use case is a particular activity a user can do on the system.
2. Is represented by an ellipse.
3. Following are two use cases for a library system.



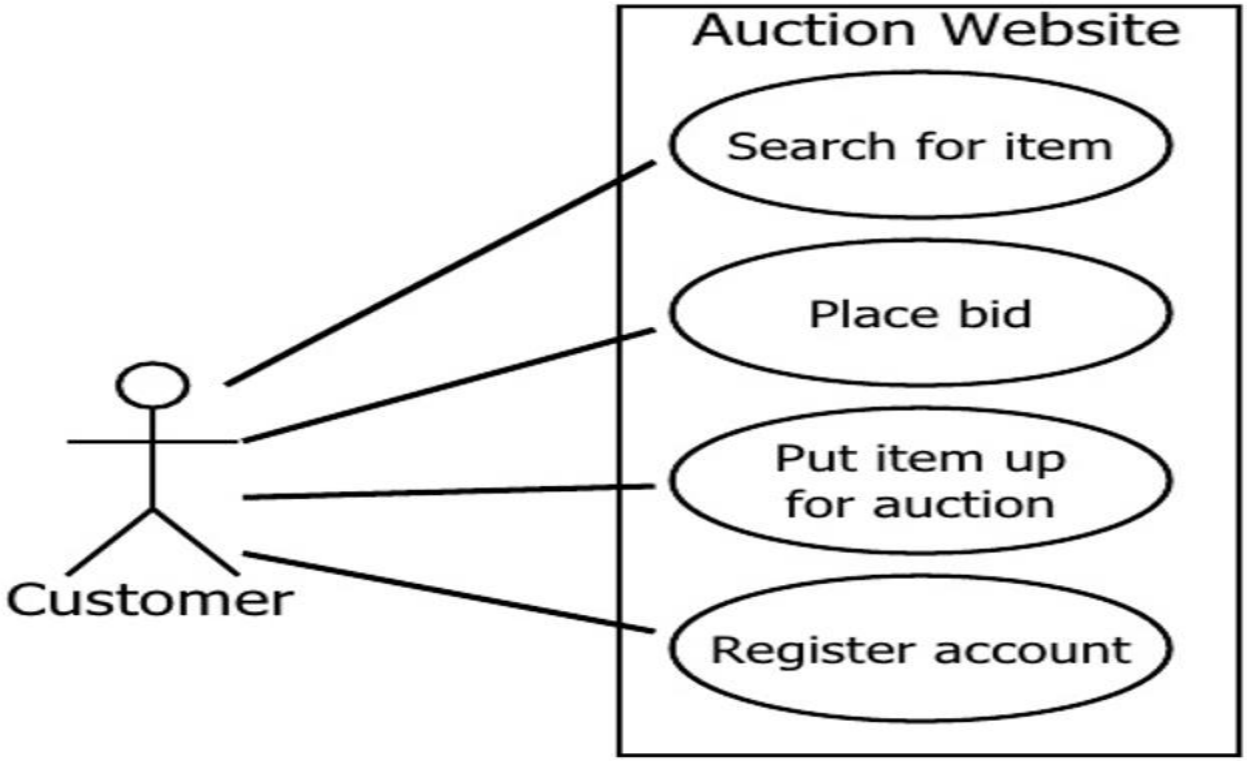
Identifying use cases for a system

1. What are the tasks of each actor?
2. Will any actor create, store, change, remove, or read the information?
3. Will any actor need to inform the system about the sudden, external changes?
4. Does any actor need to informed about certain occurrences in the system?
5. What use cases will support and maintain the system?
6. Can all functional requirements be performed by the use cases?

Summary of Notations

Construct	Description	Notation
Use-case	A sequence of transactions performed by a system that produces a measurable result for a particular actor	
Actor	A coherent set of roles that users play when interacting with these use cases	
System Boundary	The boundary between the physical system and the actors who interact with the physical system	

Auction Website Use Cases

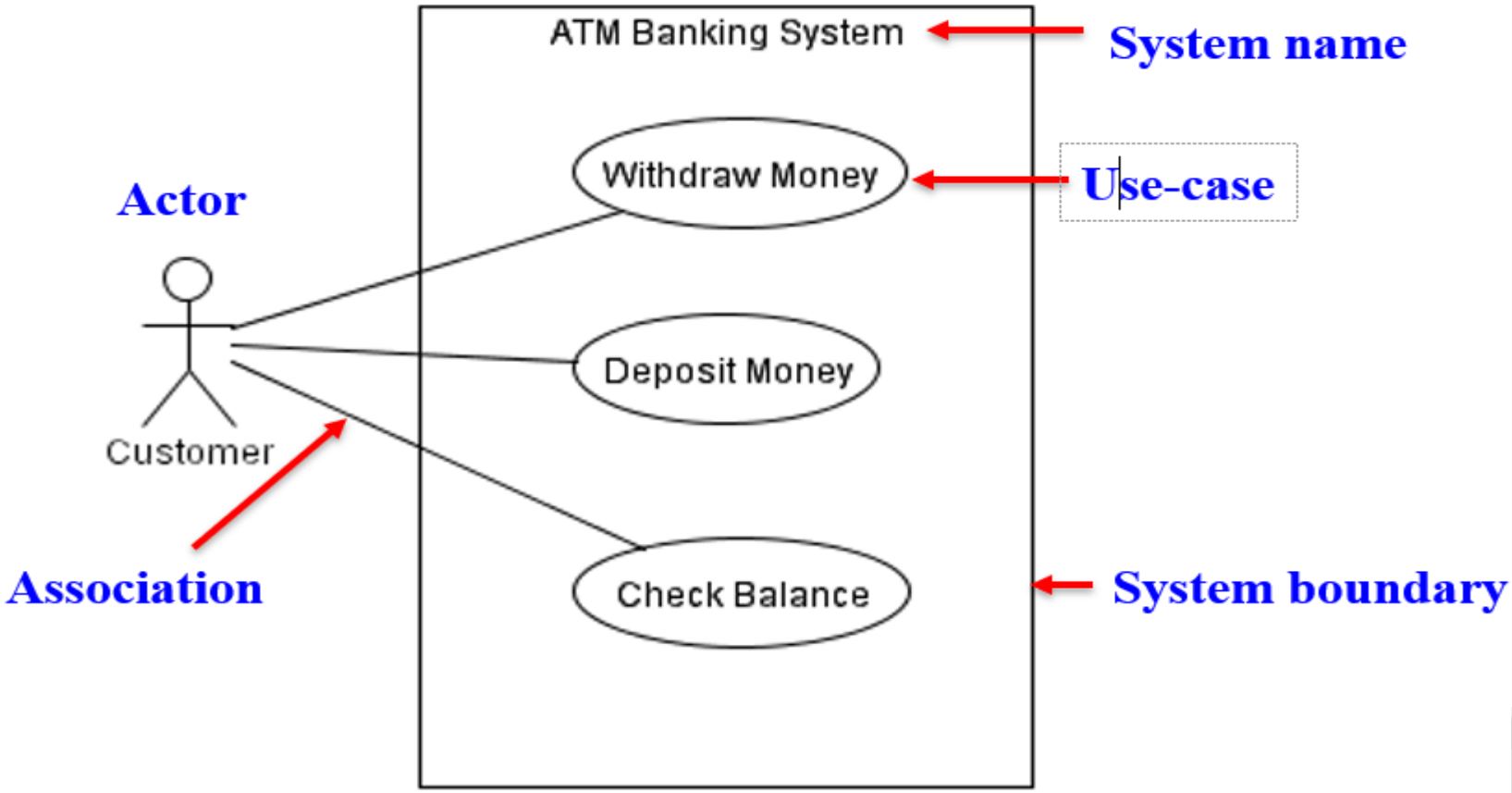


Use cases

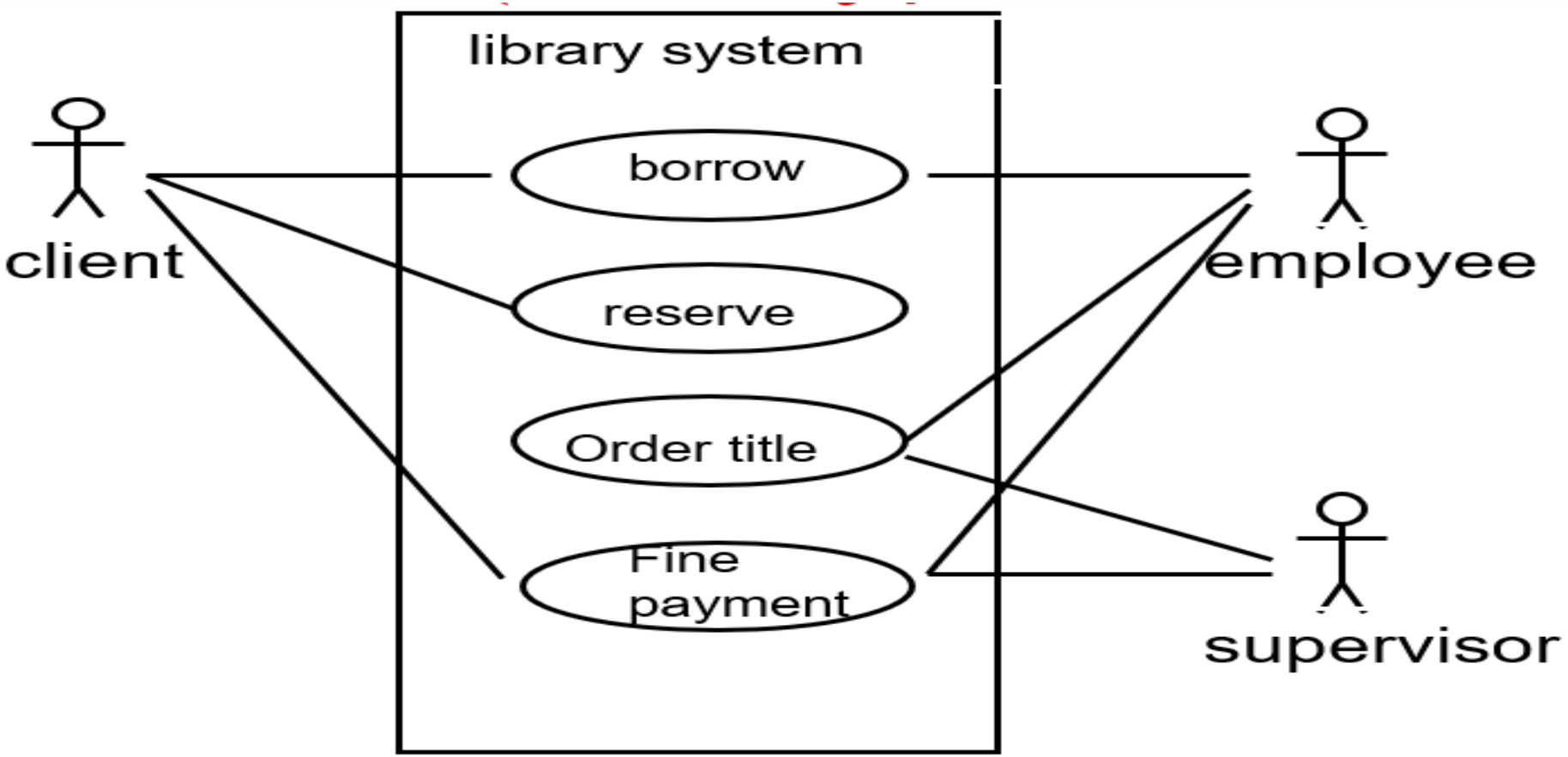
1. Functionality provided by the system
2. Consist of a series of steps which collectively add value to the user of the system
3. Examples

Issue a book to a member
Receive a book back from a member
Query the current location of a book
Maintain member's information
Maintain book's information

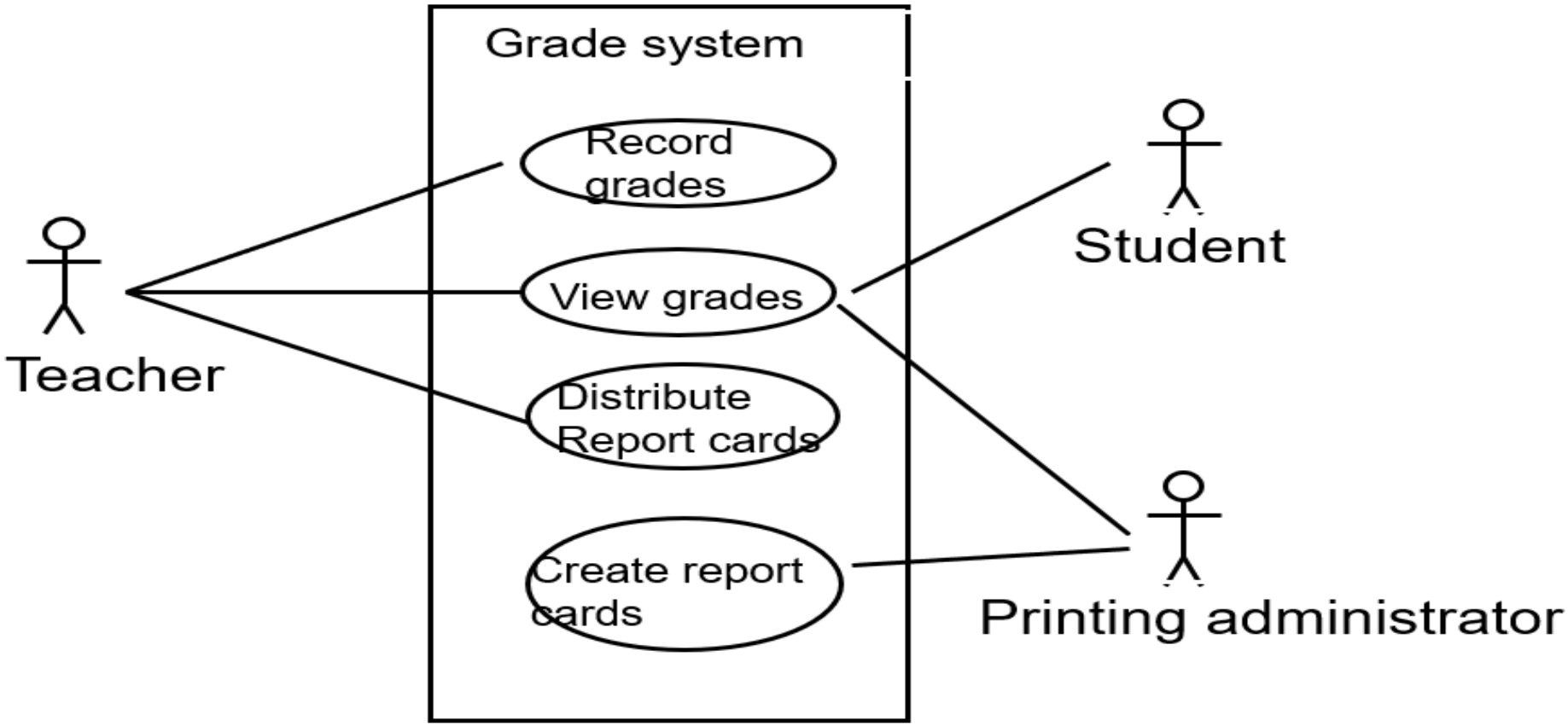
Use-Case Diagram: Example



Use Case Diagram – Example1 (Library)



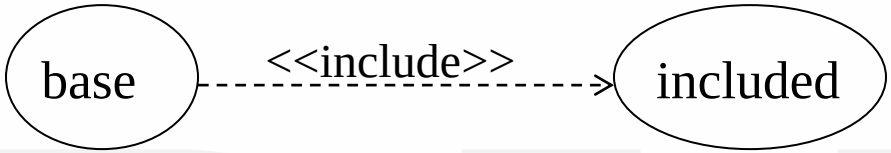
Use Case Diagram for Student Assessment Management System



Use-Case Diagrams

1. Include: a dotted line labeled <<include>> beginning at base use case and ending with an arrows pointing to the include use case. The include relationship occurs when a chunk of behavior is similar across more than one use case. Use “include” in stead of copying the description of that behavior.
2. Extend: a dotted line labeled <<extend>> with an arrow toward the base case. The extending use case may add behavior to the base use case. The base class declares “extension points”.

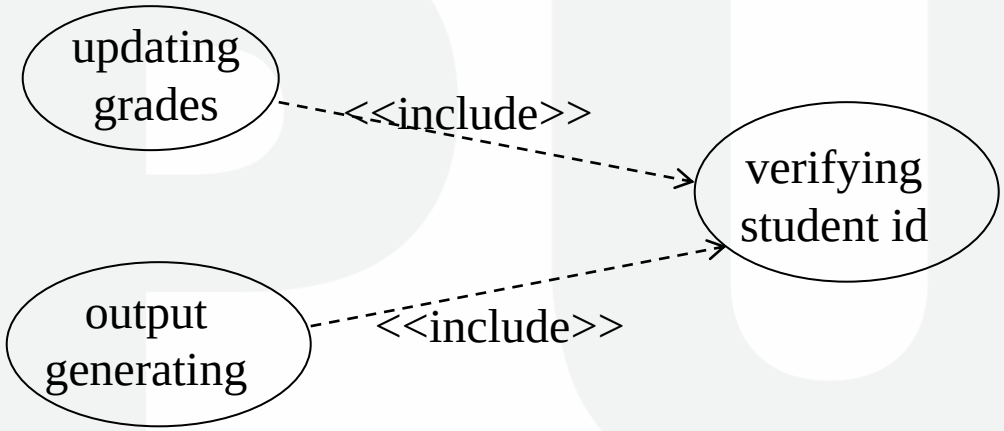
Include



- The base use case explicitly incorporates the behavior of another use case at a location specified in the base.
- The included use case never stands alone. It only occurs as a part of some larger base that includes it.

More about Include

- Enables to avoid describing the same flow of events several times by putting the common behavior in a use case of its own.

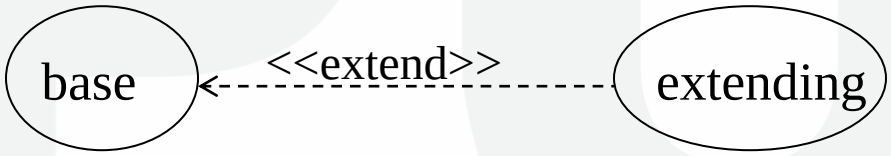


The <<include>> Relationship

- Include relationships are used when two or more use cases share some common portion in a flow of events
- This common portion is then grouped and extracted to form an inclusion use case for sharing among two or more use cases
- Most use cases in the ATM system example, such as Withdraw Money, Deposit Money or Check Balance, share the inclusion use-case Login Account

Extend

- The base use case implicitly incorporates the behavior of another use case at certain points called extension points.
- The base use case may stand alone, but under certain conditions its behavior may be extended by the behavior of another use case.

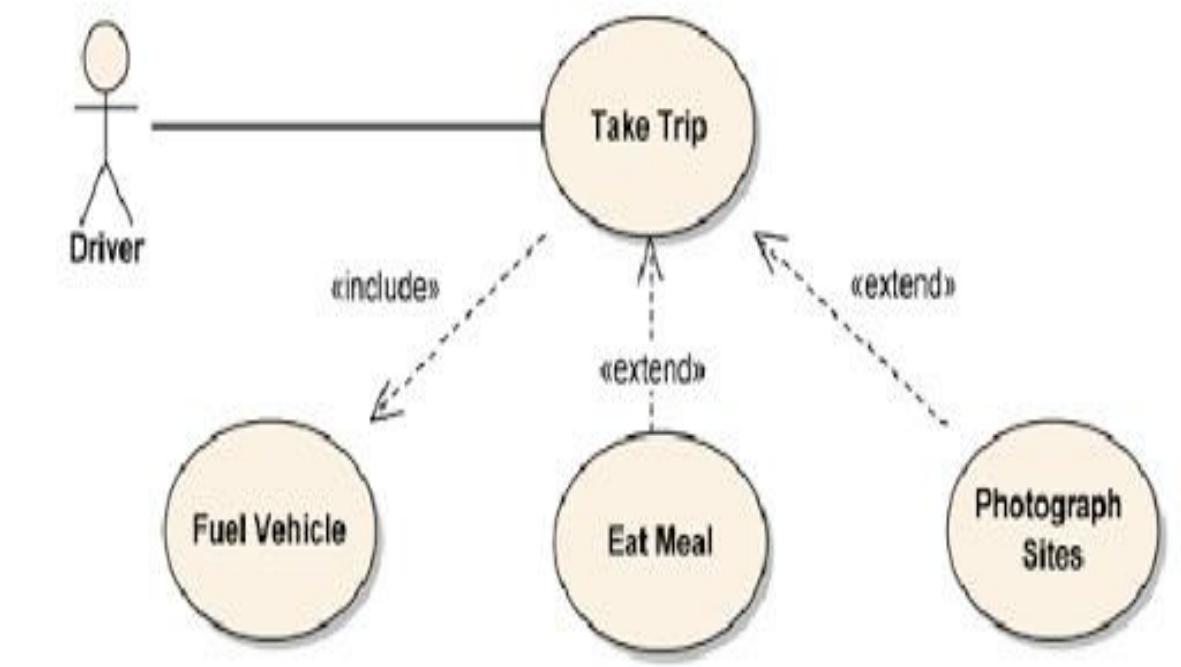


The <<extend>> Relationship

- In UML modeling, you can use an extend relationship to specify that one use case (extension) extends the behavior of another use case (base)

- This type of relationship reveals details about a system or application that are typically hidden in a use case

The <<extend>> Relationship



Parul®
University

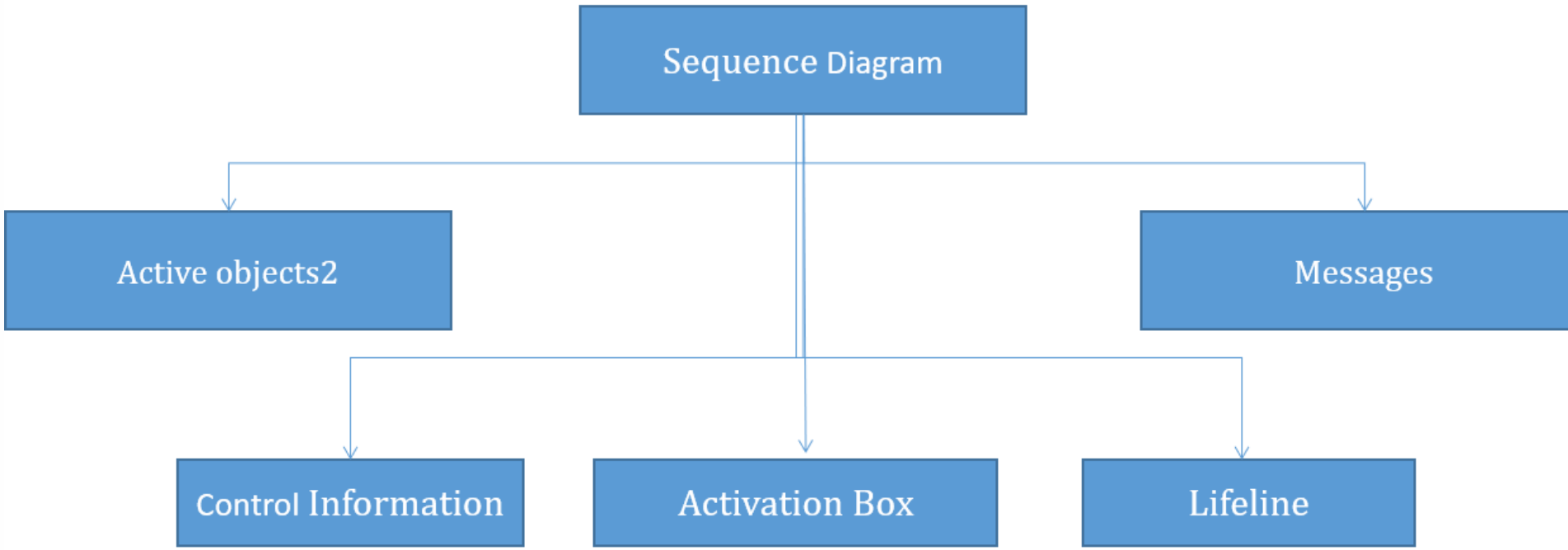
DIGITAL LEARNING CONTENT

Sequence Models

Sequence Diagrams

- Sequence diagrams, also known as event diagrams or event scenarios, illustrate how processes interact with each other by showing calls between different objects in a sequence.
- These diagrams have two dimensions:
- The vertical lines show the sequence of messages and calls in chronological order
- Horizontal elements show object instances where the messages are relayed.

Sequence Diagram



SEQUENCE DIAGRAM (Important Components)

- Active Objects:
  - Any objects that play a role in the system
  - Can be any object or class that is valid within the system
  - Can be an Actor that is external to the system and derives benefits from the system
- Messages:
  - Used to illustrate communication between different active objects.
  - Used when an object needs
    - To activate a process of a different object
    - To give information to another object

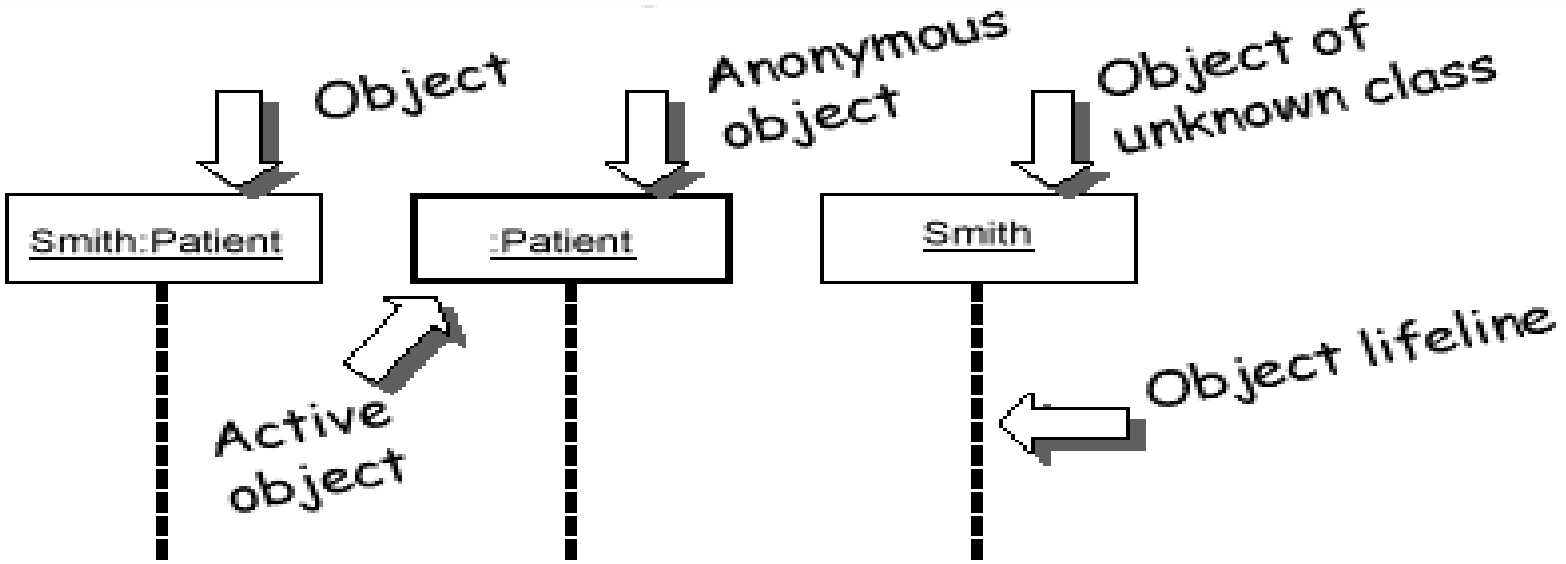
SEQUENCE DIAGRAM (other Components)

- Lifeline
  - Denotes the life of actors/objects over time during a sequence
- Focus of control (activation box)
  - Means the object is active and using resources during that time period
- Control information
  - Shows the control flow in the system
  - Creation and destruction of an object through <<create>> and <<destroy>>

Representing Objects

- Squares with object type, optionally preceded by object name and colon
- Write object's name if it clarifies the diagram
- Object's "life line" represented by dashed vertical line

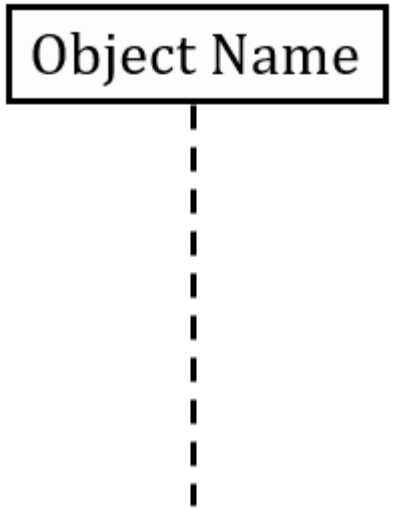
Representing Objects



Name syntax: <objectname>:<classname>

Lifeline

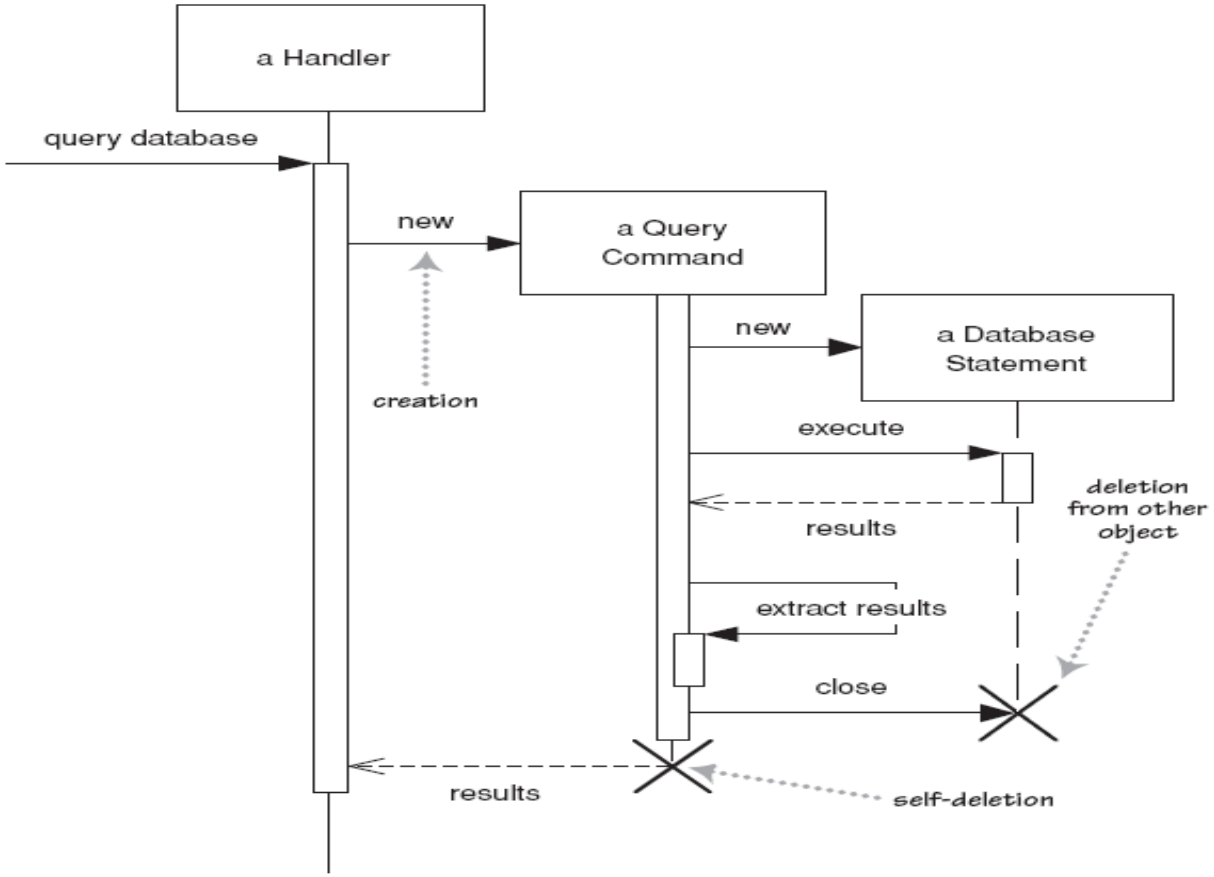
- Objects are displayed at the top of the diagram
- The vertical dimension represents time
- Each object has a dashed line – lifeline – extending below it – to indicate the period of time during which objects playing that role actually exist



Lifetime of objects

- Creation: An arrow with 'new' written above it
- Deletion: An X at bottom of object's lifeline

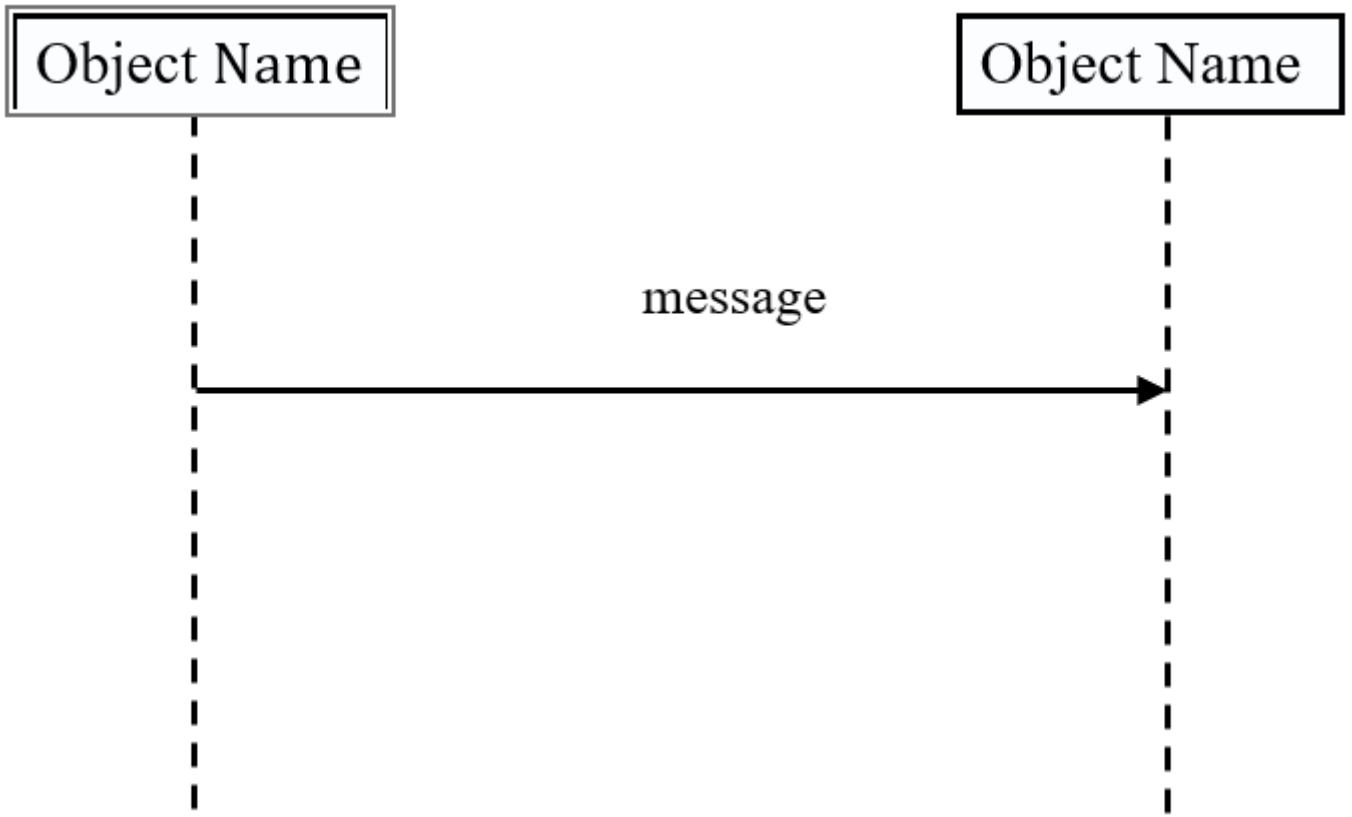
Lifetime of objects



Message

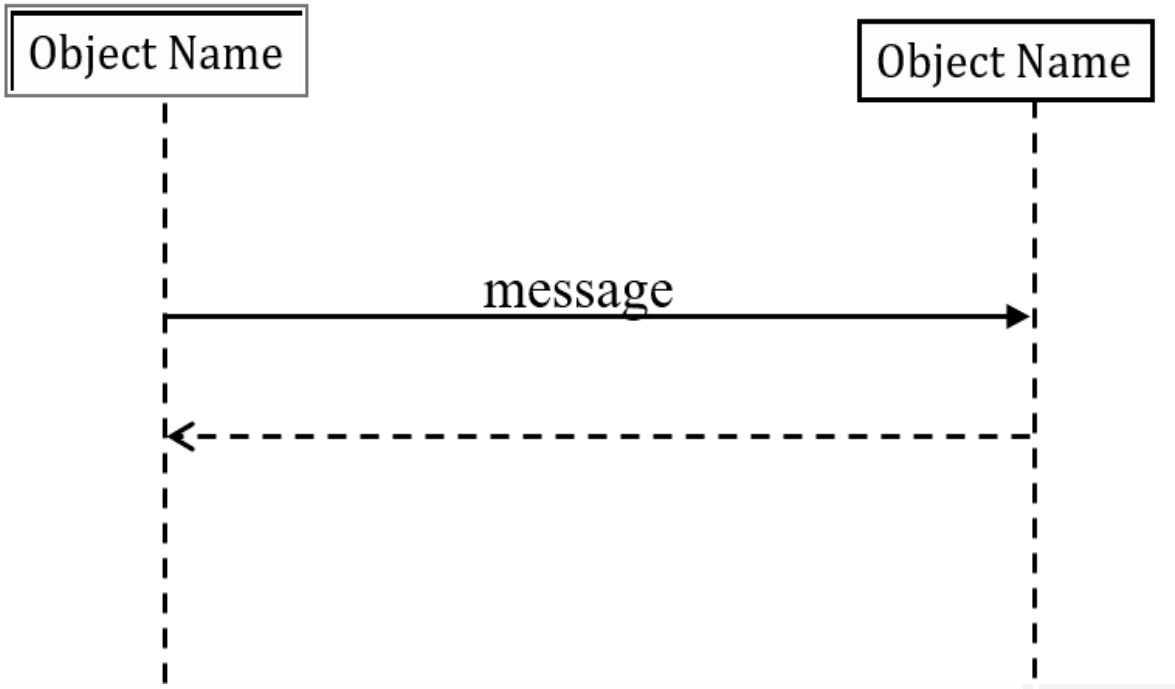
- The messages in an interaction are drawn from top to bottom, in the order that they are sent.
- Messages are shown as arrows pointing from the lifeline of the role sending the message to the lifeline of the receiver.
- When a message is sent, control passes from the sender of the message to the receiver.

Message



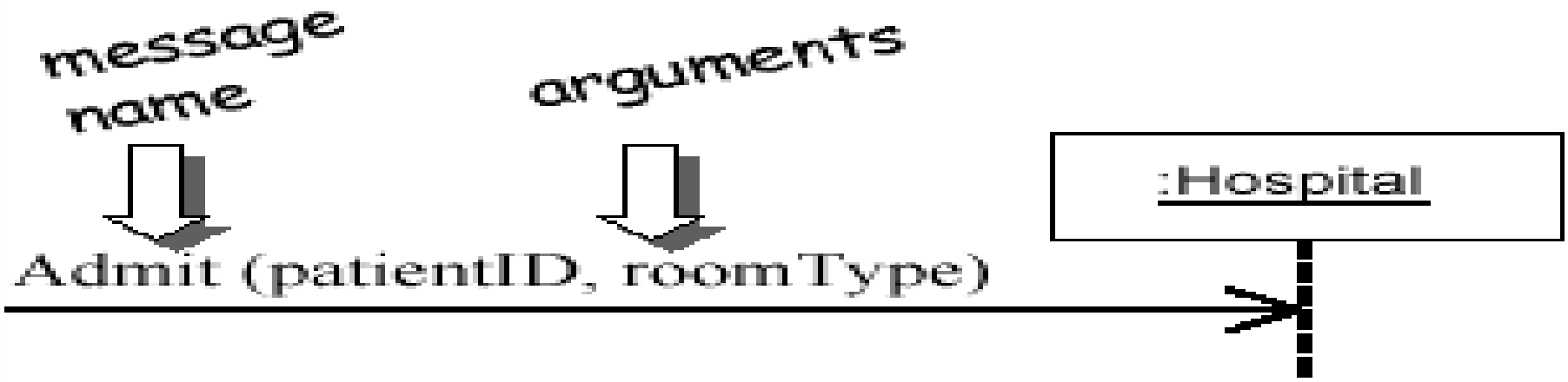
Return

- Return of control is shown using dashed arrow returning to the calling object.



Messages between Objects

- Message (method call) indicated by horizontal arrow to other object
- Write message name and arguments above arrow



Indicating method calls

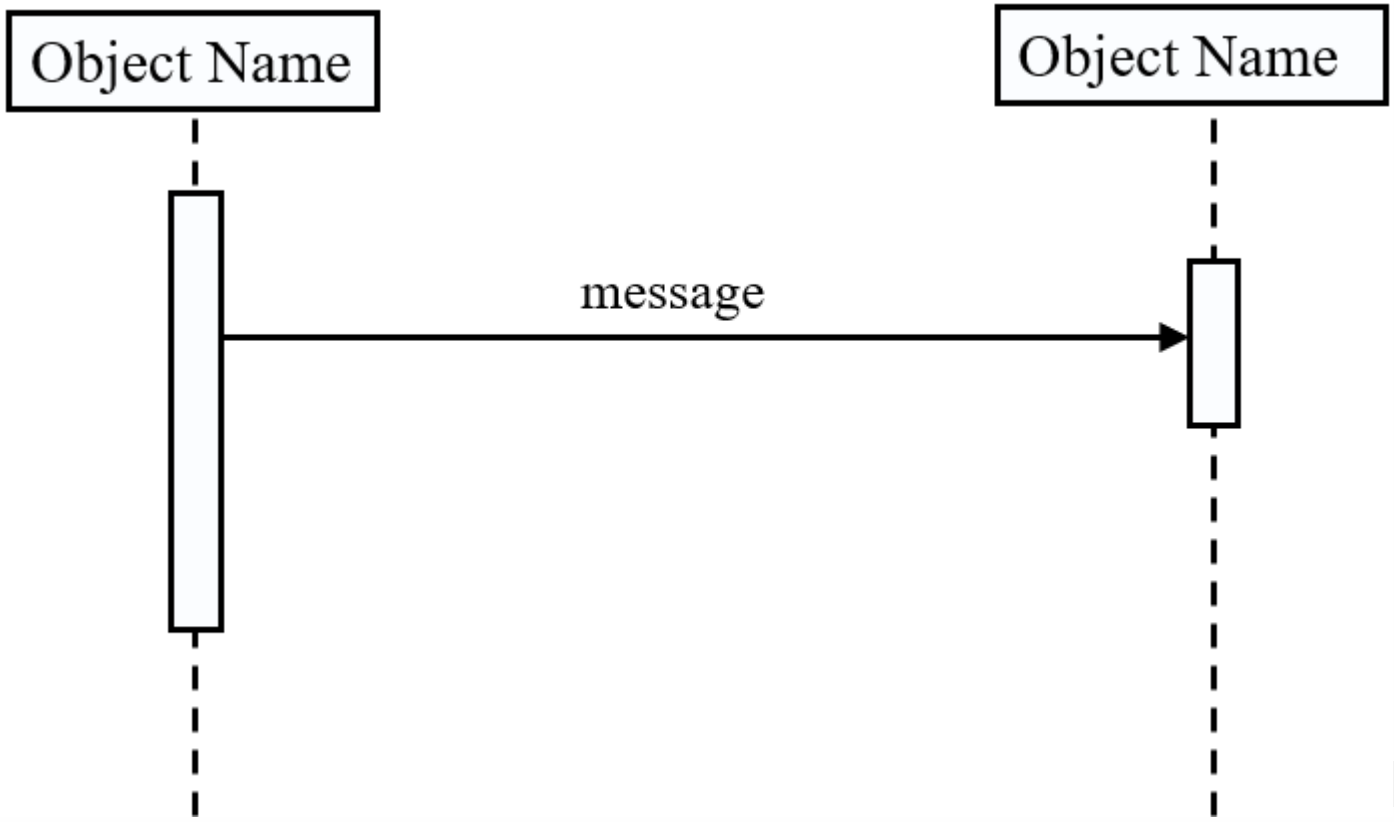
- Activations - show when a method is active – either executing or waiting for a subroutine to return
- Either that object is running its code, or it is on the stack waiting for another object's method to finish

Activation

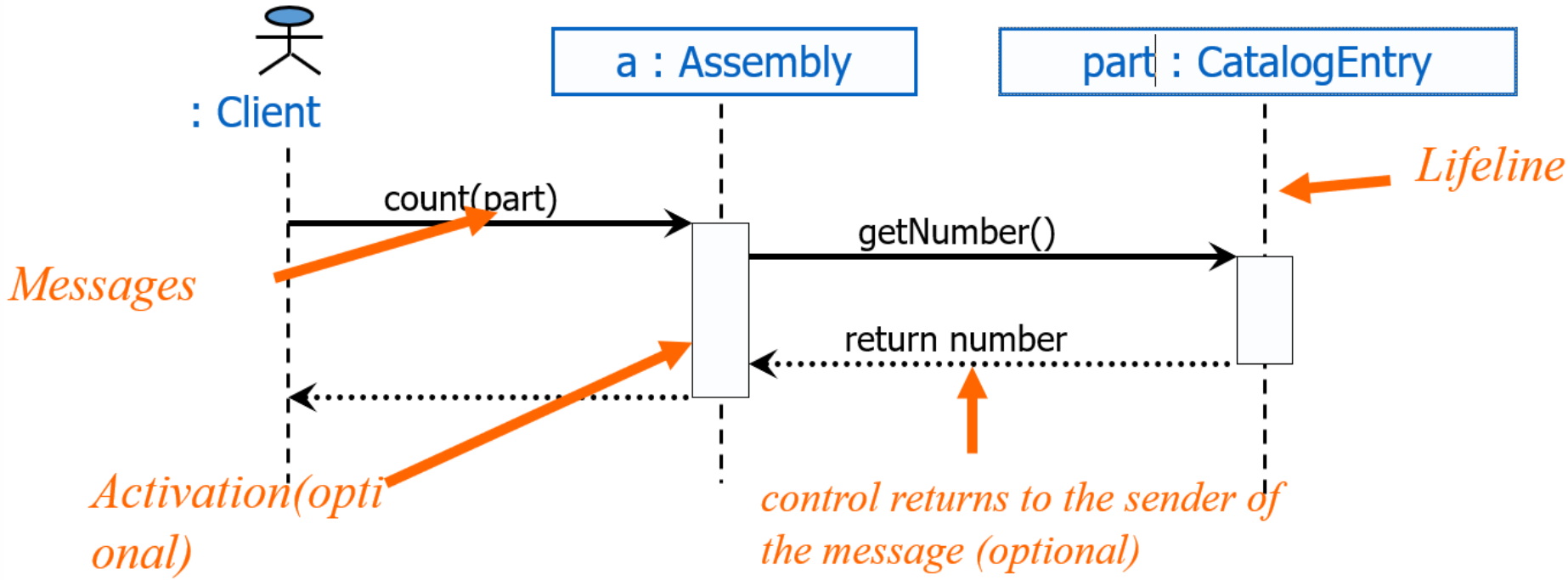
- Period of time during which an object is processing a message, Shown on a lifeline as a narrow rectangle whose top is connected to a message.

- When an object finishes processing a message, control returns to the sender of the message

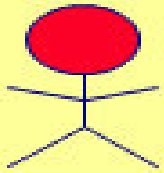
Activation



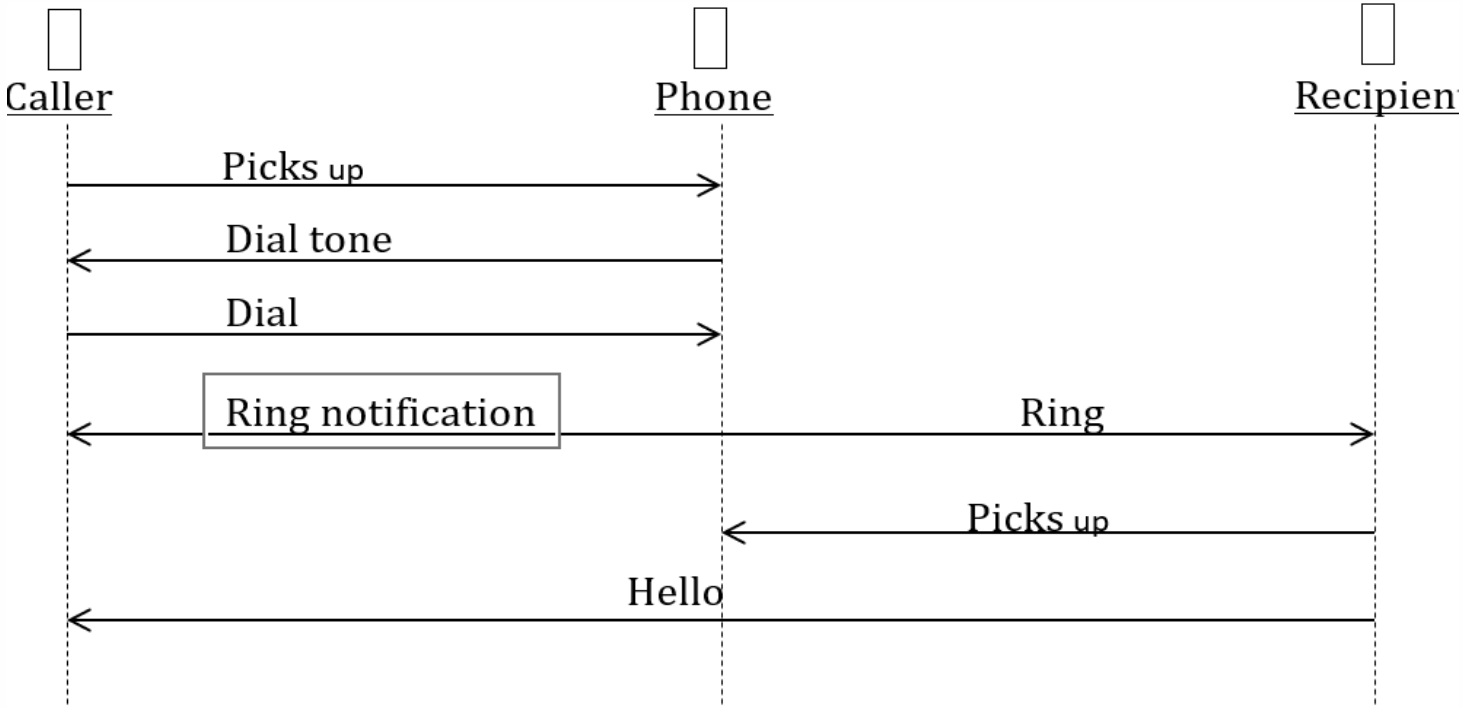
Sequence Diagram

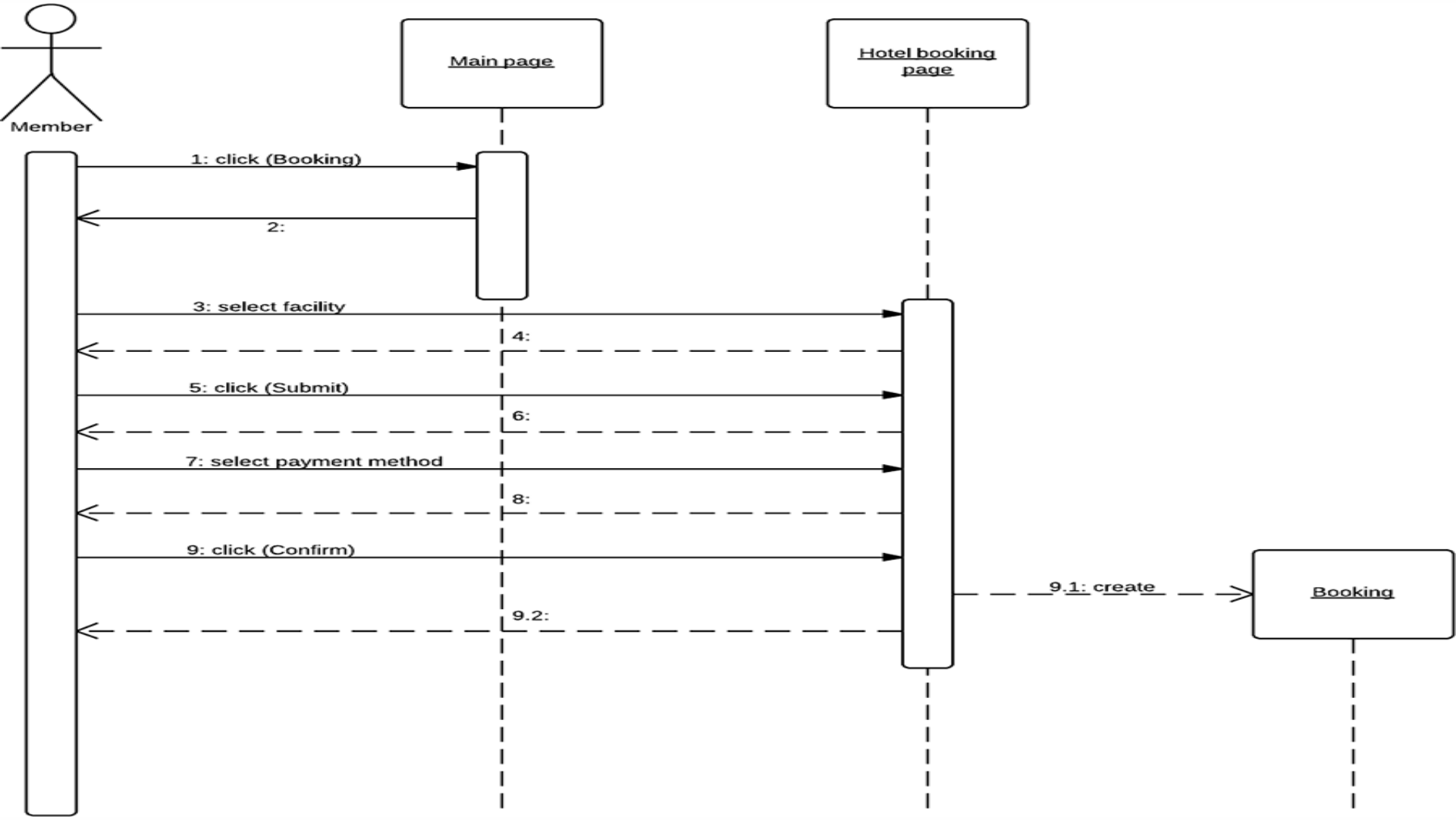


Sequence Diagram Syntax

AN ACTOR	
AN OBJECT	anObject:aClass
A LIFELINE	
A FOCUS OF CONTROL	
A MESSAGE	aMessage()
OBJECT DESTRUCTION	x

Sequence Diagram(make a phone call)




Member
Main page
Hotel booking page
1: click (Booking)
2:
3: select facility
4:
5: click (Submit)
6:
7: select payment method
8:
9: click (Confirm)
9.1: create
Booking
9.2:

Parul®
University

DIGITAL LEARNING CONTENT

Activity Models

What Is an Activity Diagram?

- Activity diagrams and use cases are logical model which describe the business domain's activities without suggesting how they are conduct.
- Shows the sequence of steps that make up a complex process.
- Shows flow of control, similarly sequence diagram but focus on operations.
- A diagram that emphasizes the flow of control from activity to activity in an object.
- Similar to the traditional program flowchart.
- Used to provide detail for complex algorithms.
- Primary activities and the relationships among the activities in a process.

Drawing Activity Diagrams

Purpose

to model a task (for example in business modelling)

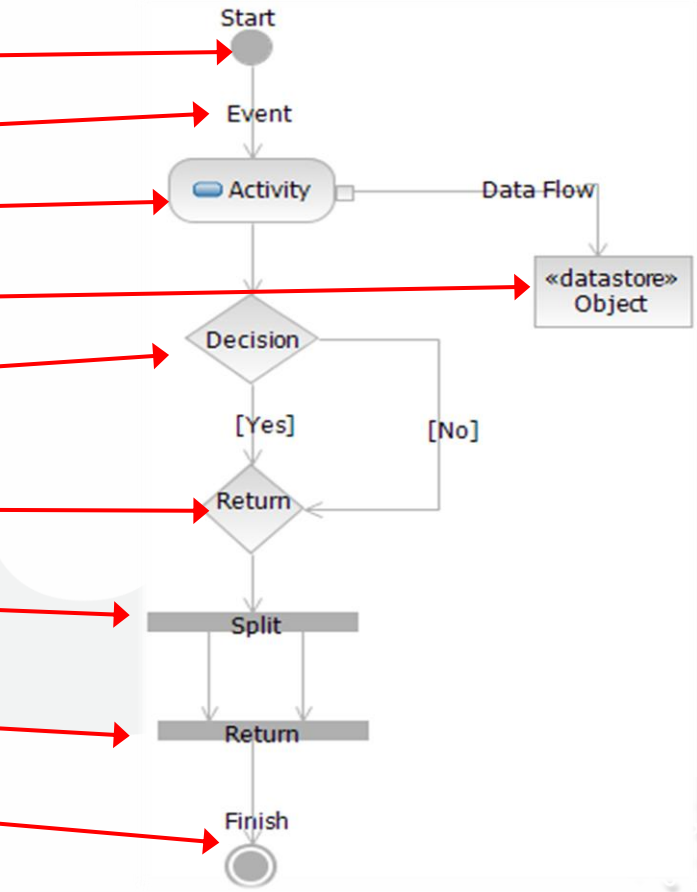
to describe a function of a system represented by a use case

to describe the logic of an operation

to model the activities that make up the life cycle in the Unified Process

The Activity Diagram Components

- Initial Node
- Control Flow
- Action or Activity
- Object Flow
- Branch
- Merge
- Fork
- Join
- Final Node

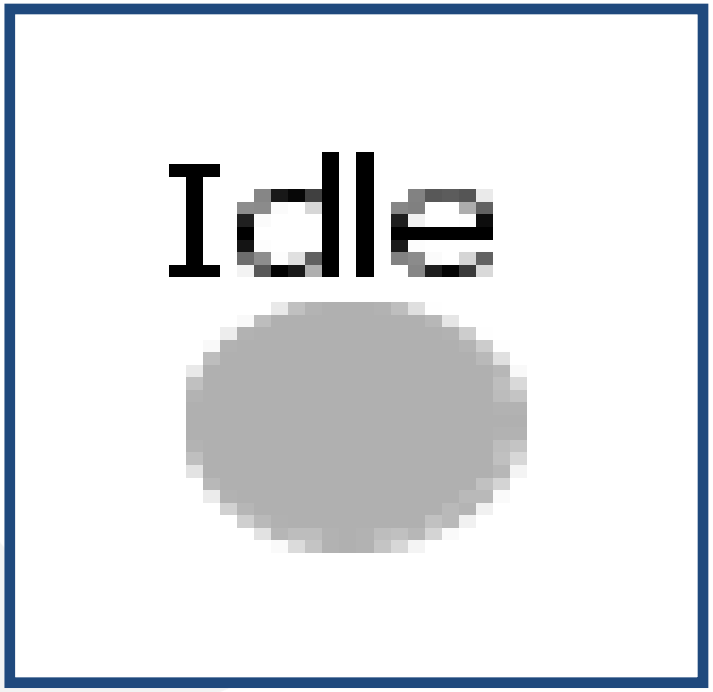


Initial Node

This represents the start of the flow of an activity diagram.

An activity diagram contains a single start node.

The name of the initial node is entered on the node. It takes the form of an adjective.



Control Flow

A control flow connects any combination of:

activities branches merges forks joins

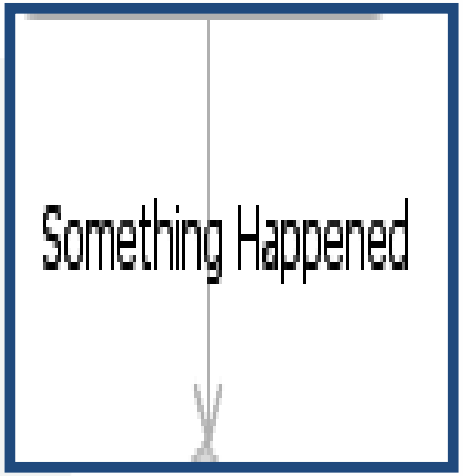
A control flow has direction, which is indicated by the arrow head – you may only traverse the control flow in the direction of the arrow.

A control flow may not enter an initial state.

A control flow may not exit a final node.

A control flow is the representation of an occurrence of an event.

The name of the event is entered on the control flow. It takes the form of something has been done, noun-verb(past-tense)



Activity And Action

The activity represents the actions that occur as a result of an incoming event from a control flow.

The name of the activity is entered on the activity and takes the form of something being done, present tense verb-nounj

Branch

The branch is used to show alternative paths in the activity diagram.

Label the decision node with a question(?).

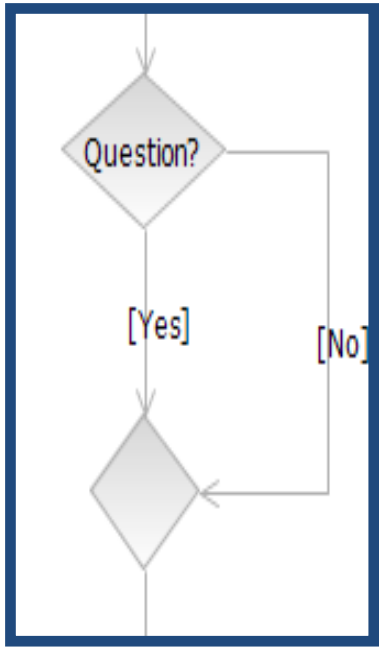
Do not label the merge,

(unless you have a good reason to).

One control flow enters the decision node and two or more alternative control flows exit the decision node.

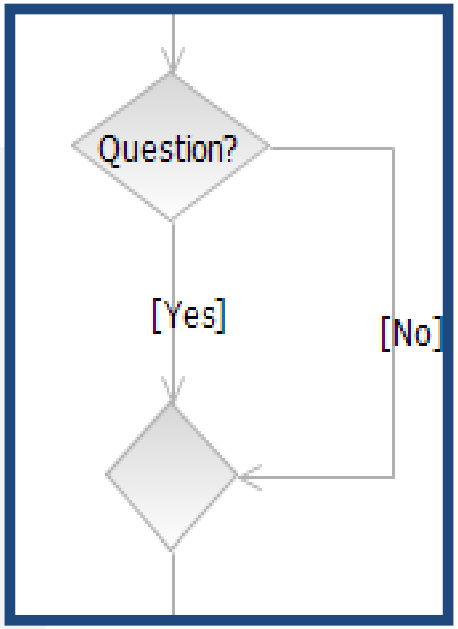
Only one of the paths may be transitioned as the result of an event occurring.

Each exiting control flow contains the condition under which it is taken (called a guard), dependent upon the answer to the question. These guards must be mutually exclusive.



Branch

The guards on exiting control flows must cover all possible outcomes of the question being asked by the branch.

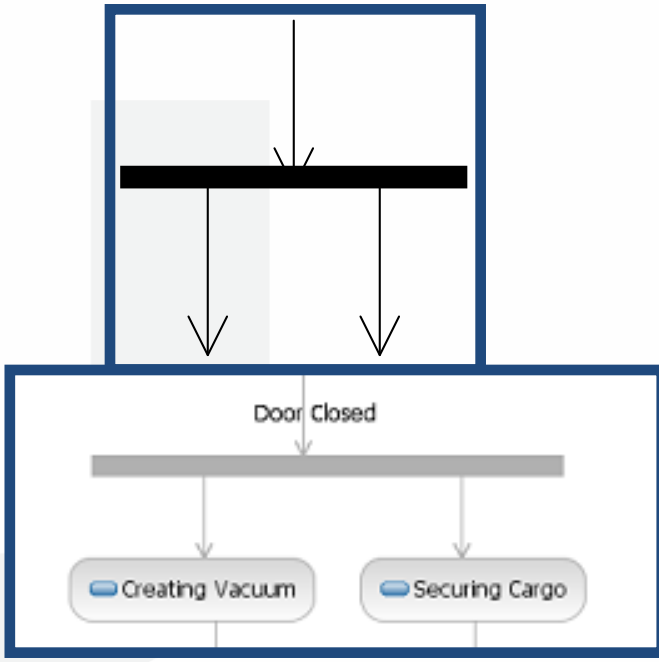


The simplest way to ensure all possible outcomes are covered is to phrase the branch question such that the only possible answers are ‘Yes’ or ‘No’. Note, this can add extra branches to the diagram.

Two or more control flows enter the merge node and one control flow exits.

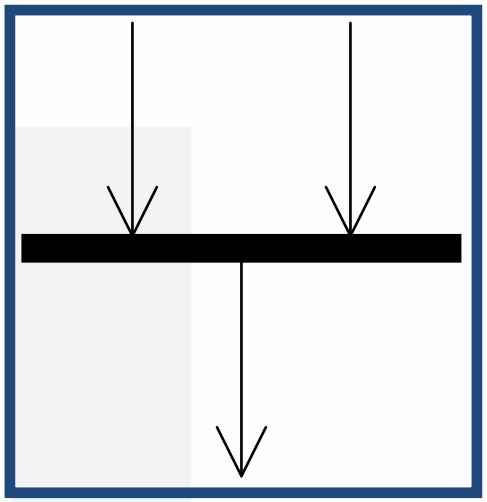
Fork

- The fork may be represented by vertical or horizontal bars.
- The fork represents that the flow through the diagram has split into 2 paths that are running in parallel (multitasking).
- The fork has a single control flow on entry and several control flows exiting.
- Use a fork when there is no requirement on the order of activities in a flow.
    - For example, the DE materializer receives an event that the door is shut. It now suspends the cargo and creates a vacuum, but these actions may be performed in parallel, so we model them with a fork.



Join

- For every fork there should be a join (if not your activity diagram is broken).
- The join may be represented by vertical or horizontal bars.
- A join simply shows that when the parallel activities have finished that they then come back to join a single flow again.
- The join has several control flows entering and a single control flow on exit.
- The exiting control flow cannot be executed until every incoming control flow has completed.
- There is no need to label the fork or join.



Final Node

The final node represents the termination of the activity diagram.

There may be several termination states in a single diagram.

Label the final node with an adjective.

Parul®
University

DIGITAL LEARNING CONTENT

Let’s review the shapes

START POINT

DECISION POINT

END POINT

[Condition] GUARD

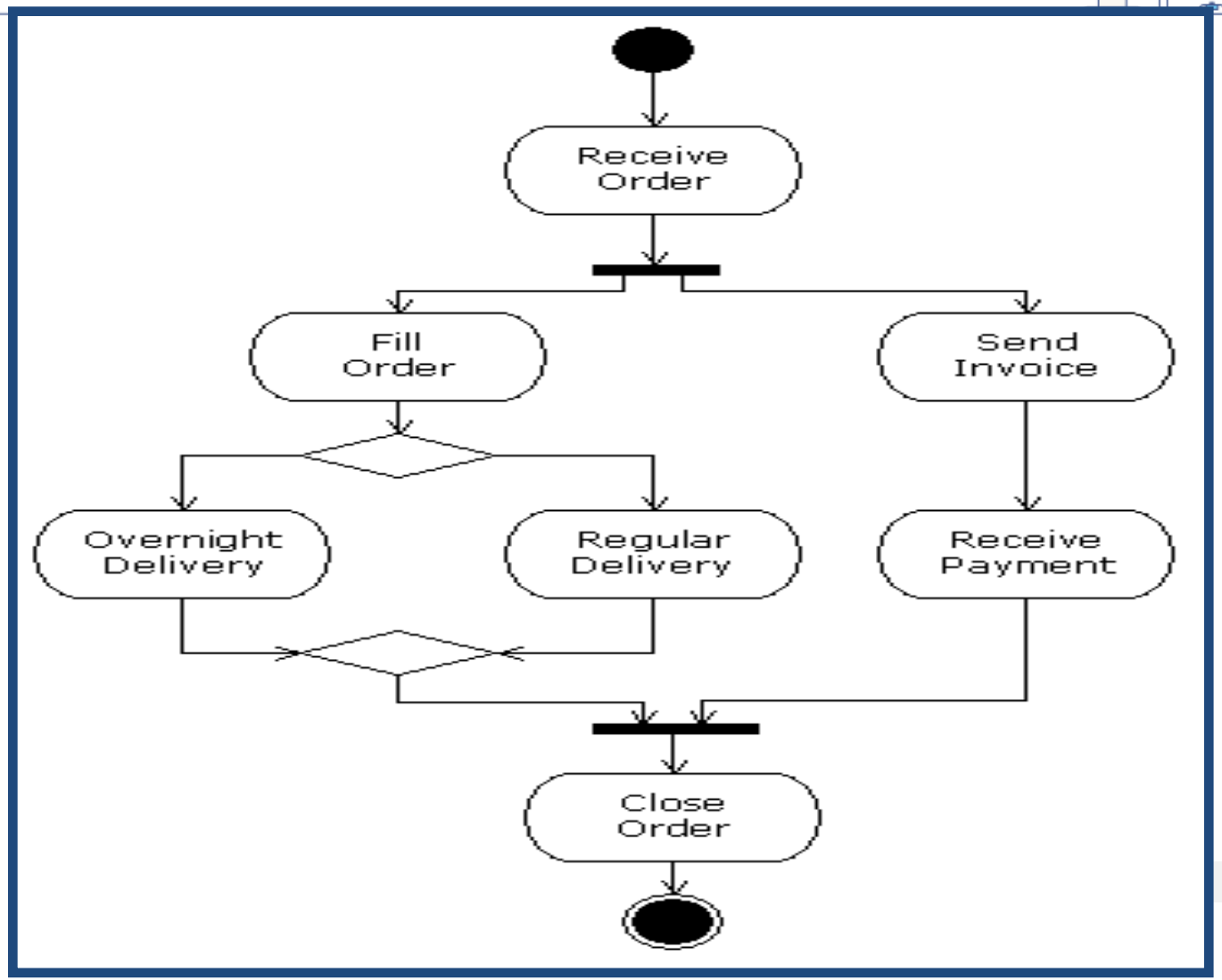
STEP

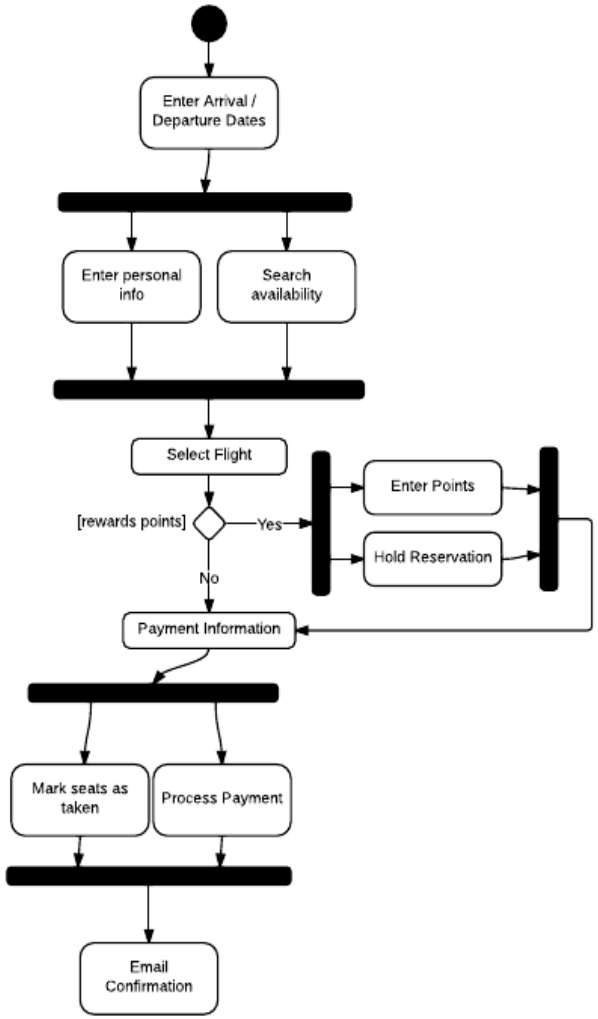
PARALLEL STEPS

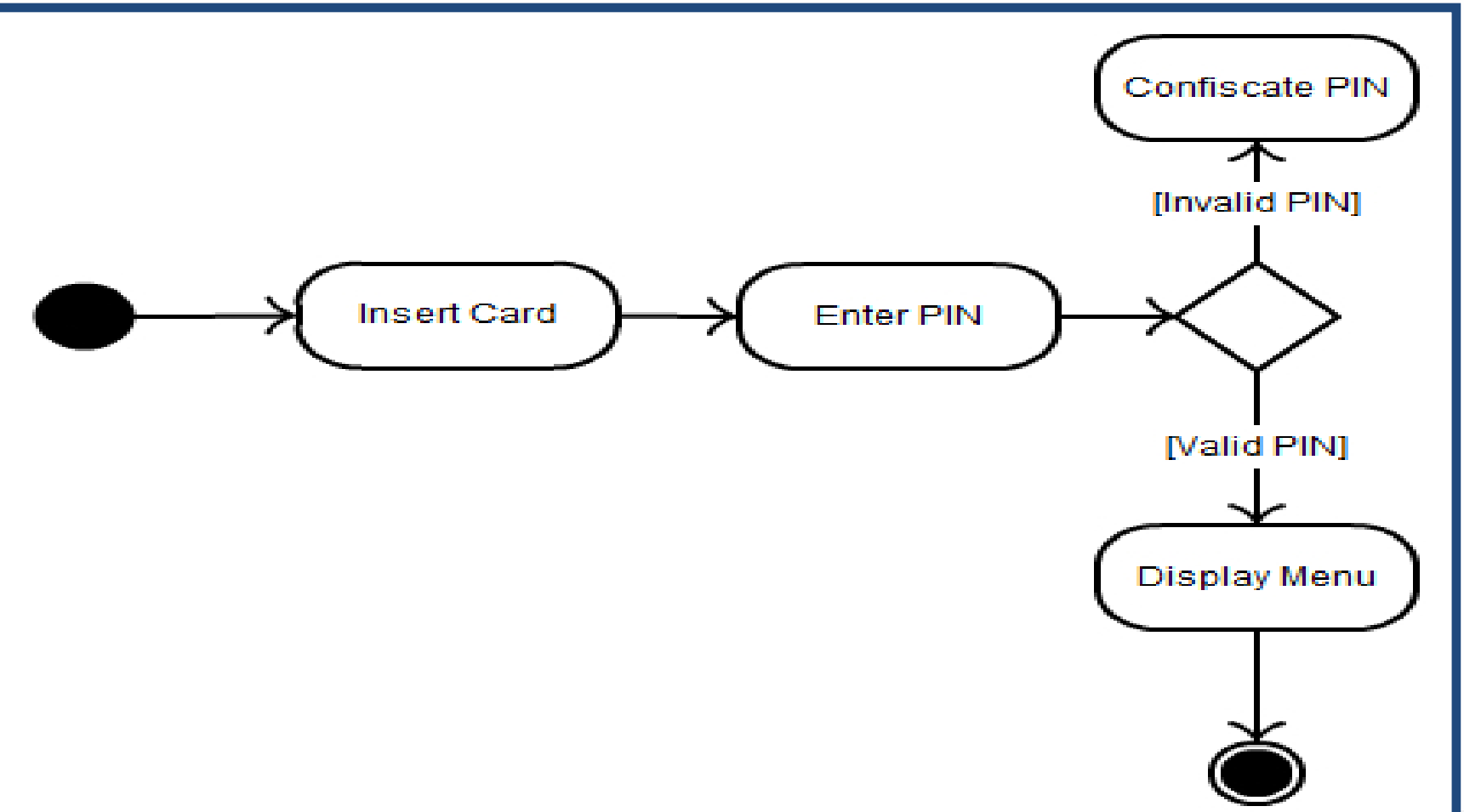
TRANSITION

For each X:

REPEATED STEPS


Receive Order
Fill Order
Send Invoice
Overnight Delivery
Regular Delivery
Receive Payment
Close Order


Enter Arrival / Departure Dates
Enter personal info
Search availability
Select Flight
[rewards points]
Yes
No
Enter Points
Hold Reservation
Payment Information
Mark seats as taken
Process Payment
Email Confirmation


Confiscate PIN
[Invalid PIN]
Insert Card
Enter PIN
[Valid PIN]
Display Menu
[online diagramming & design] creately.com

Parul®
University

DIGITAL LEARNING CONTENT

START
POINT

- The Start Point represents the EVENT that triggers the use case.

- Actor elects to Add Customer

END POINT

- Actor elects to Add Customer

Label the End Point to EXPLICITLY confirm that the intent of the use case has been achieved.

- Actor elects to Add Customer
- Customer added

- Actor elects to Add Customer

This makes it clear to the reader that the use case is complete and that nothing further is needed in order to fulfil the intent.

- Customer added

- Actor elects to Add Customer
- **End of process**

- Actor elects to Add Customer

To reach the End Point…

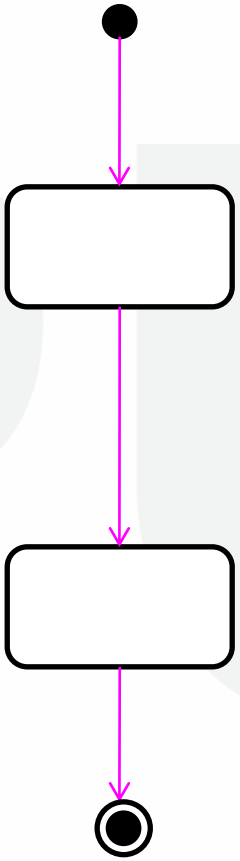
- **End of process**

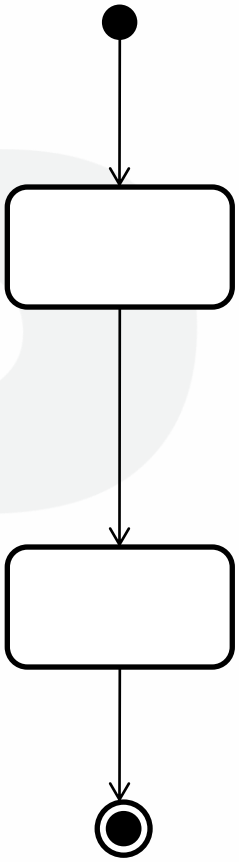
… you need to model STEPS.

Parul®
University

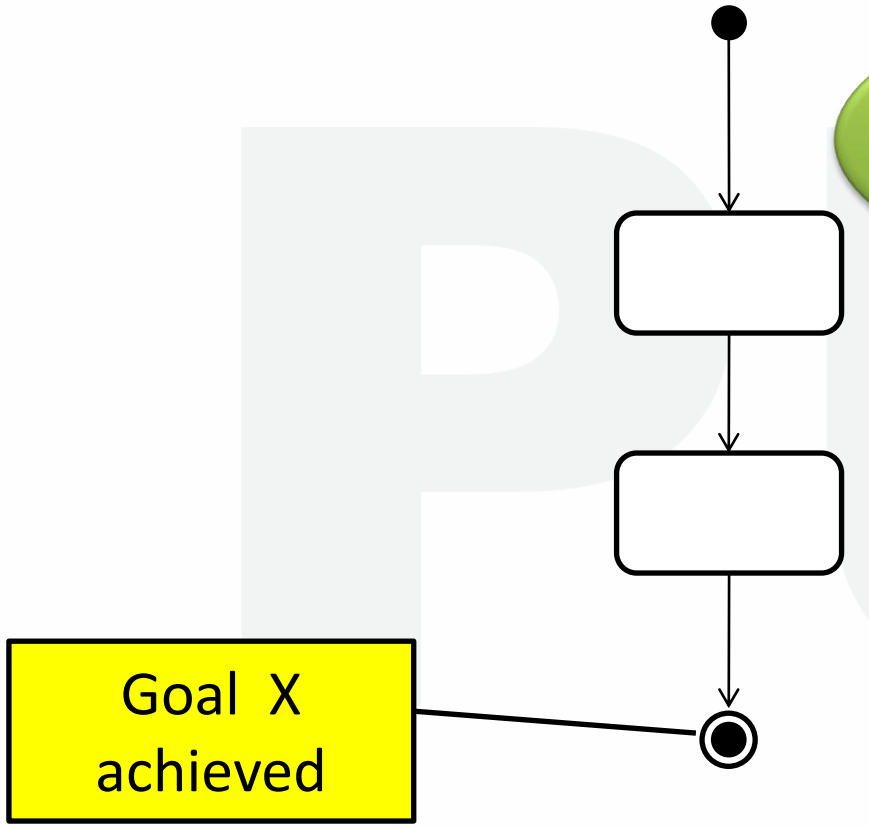
DIGITAL LEARNING CONTENT

Link the steps with TRANSITIONS.

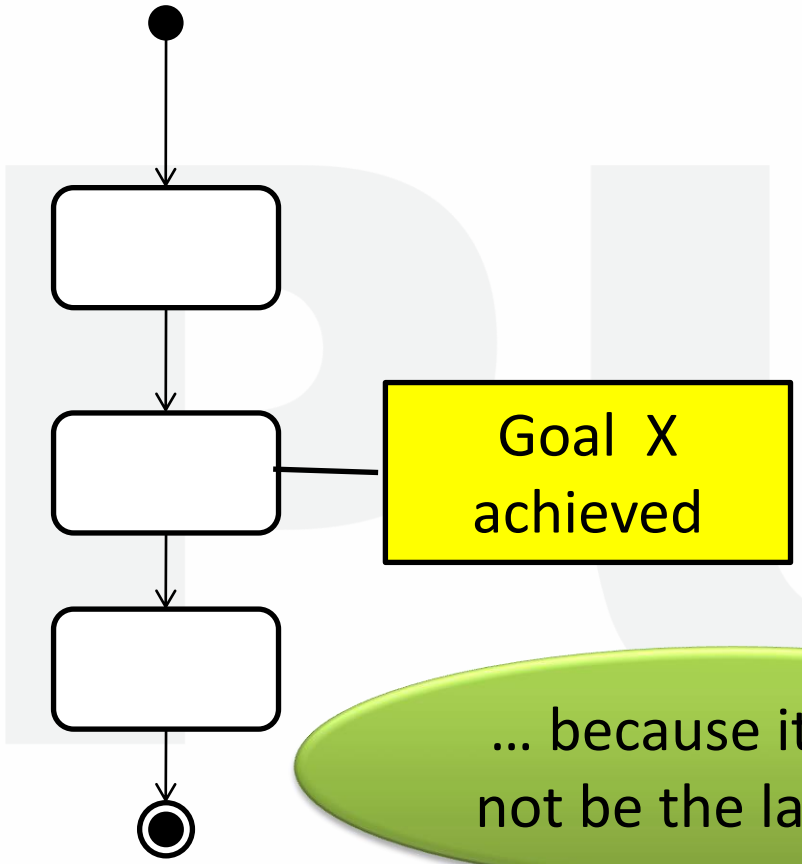




Transitions use arrow heads to show the direction of process flow.

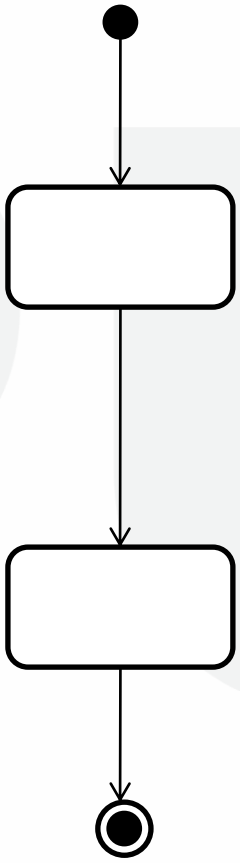


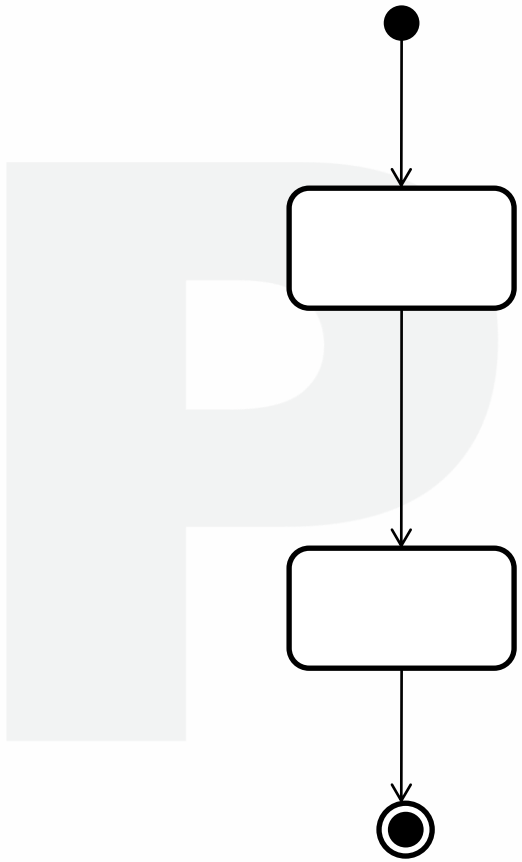
I like to put a note against any step that achieves the goal of the use case.



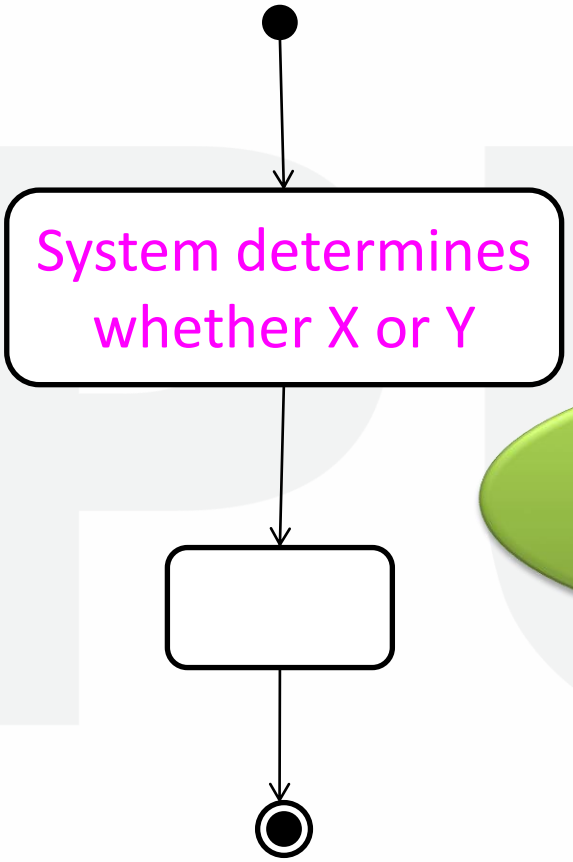
… because it might not be the last step.

Often in a use case the System has to make a decision based on business rules...



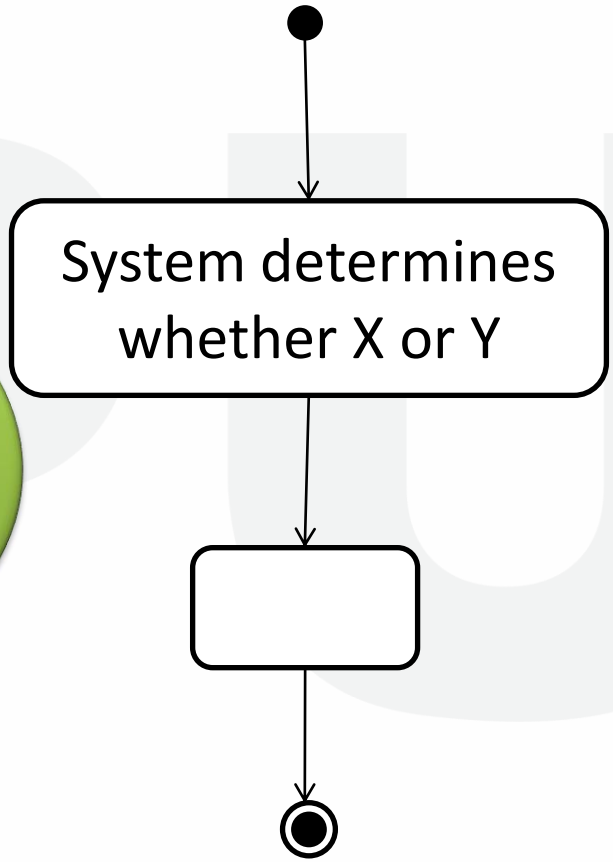


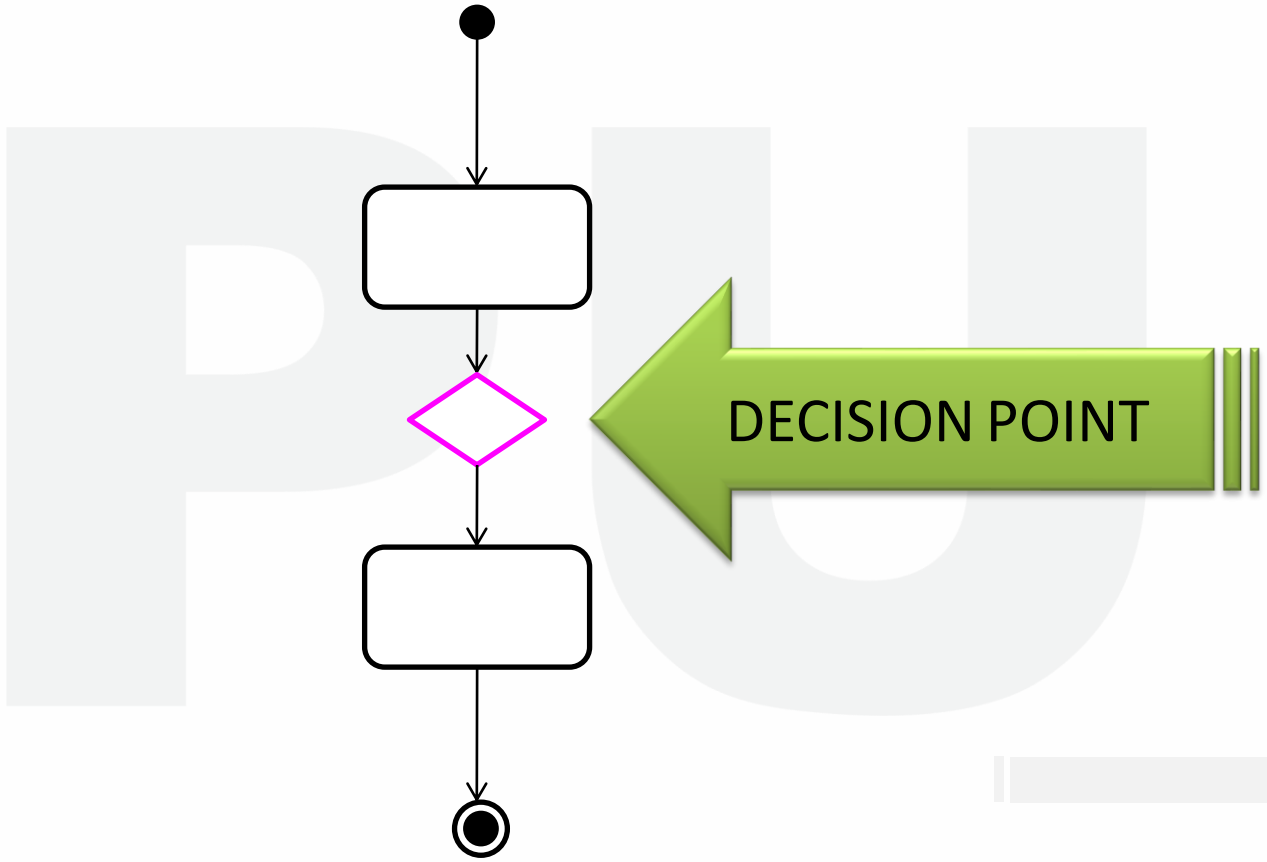
The actual decision takes place within a STEP



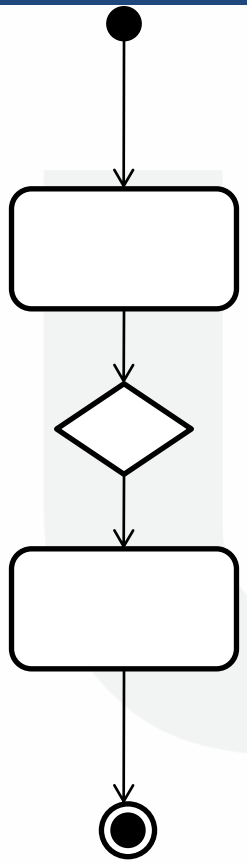
The actual decision takes place within a STEP

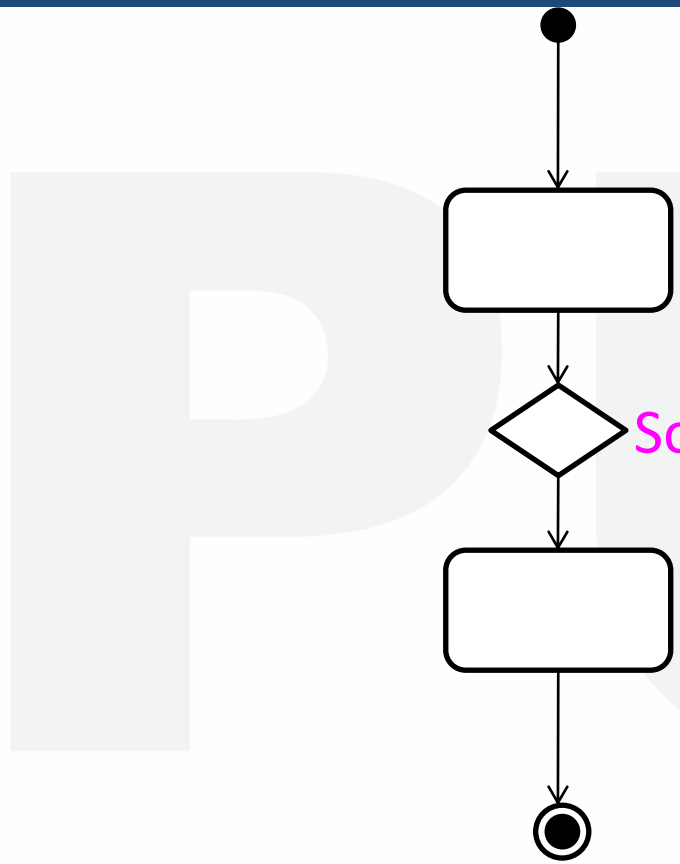
A DECISION POINT is then used to help the reader navigate the diagram.

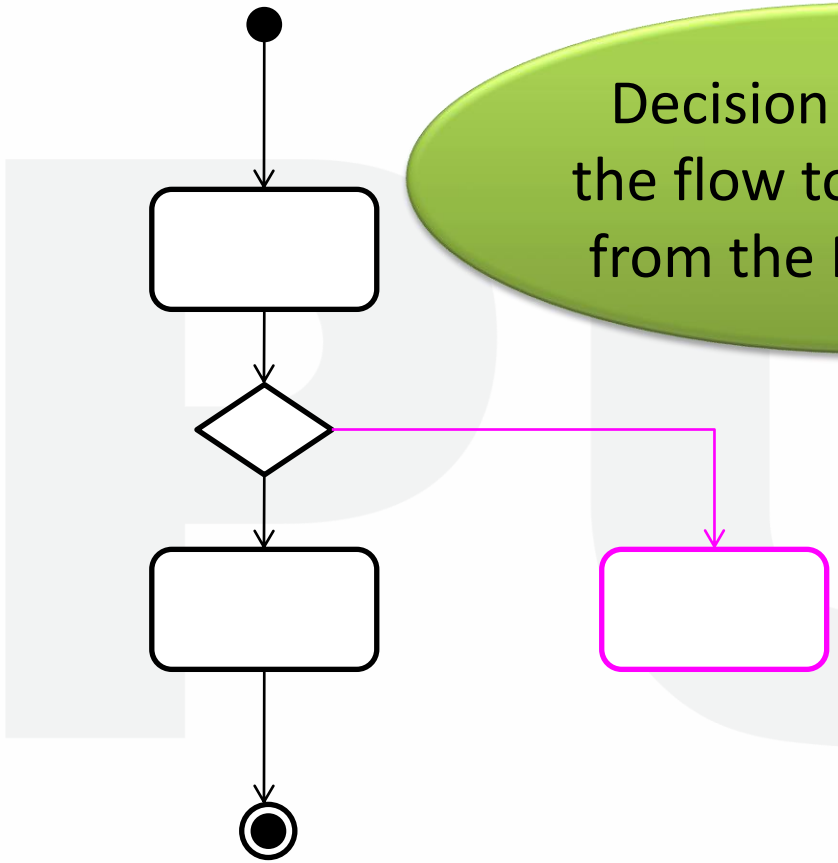



DECISION POINT

Decision Points contain text which describes the nature of the decision to be made.

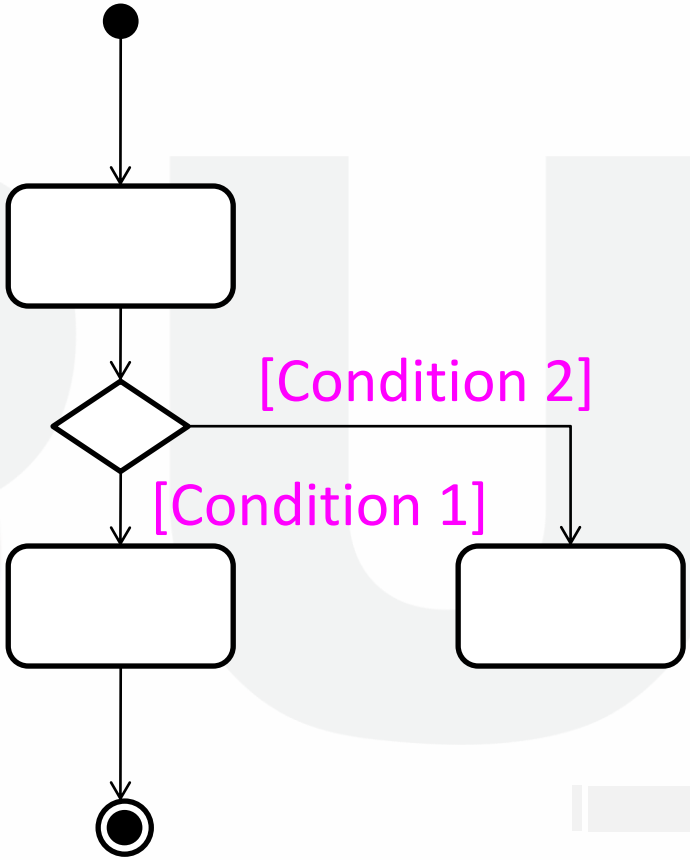



So was it X or Y?

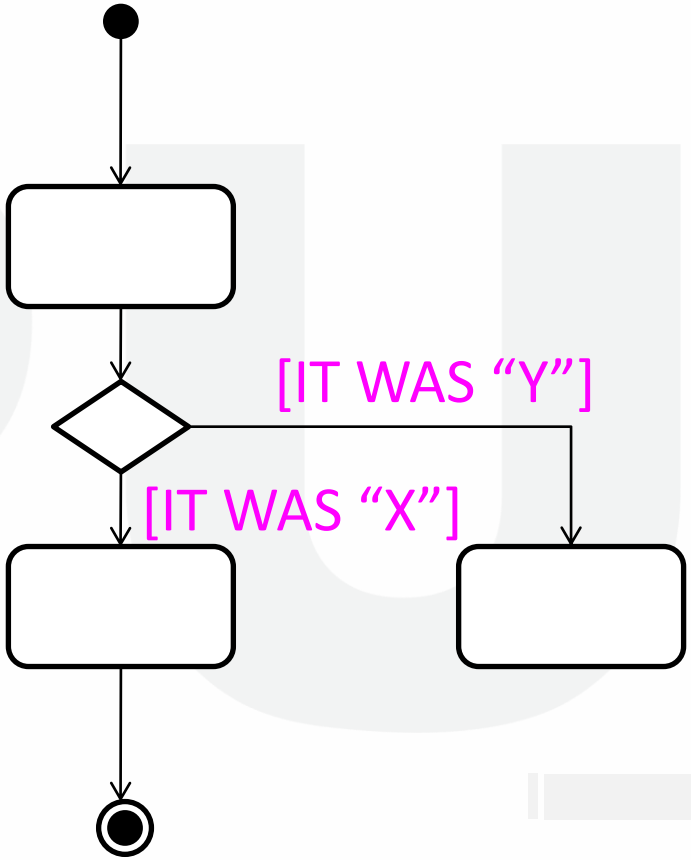


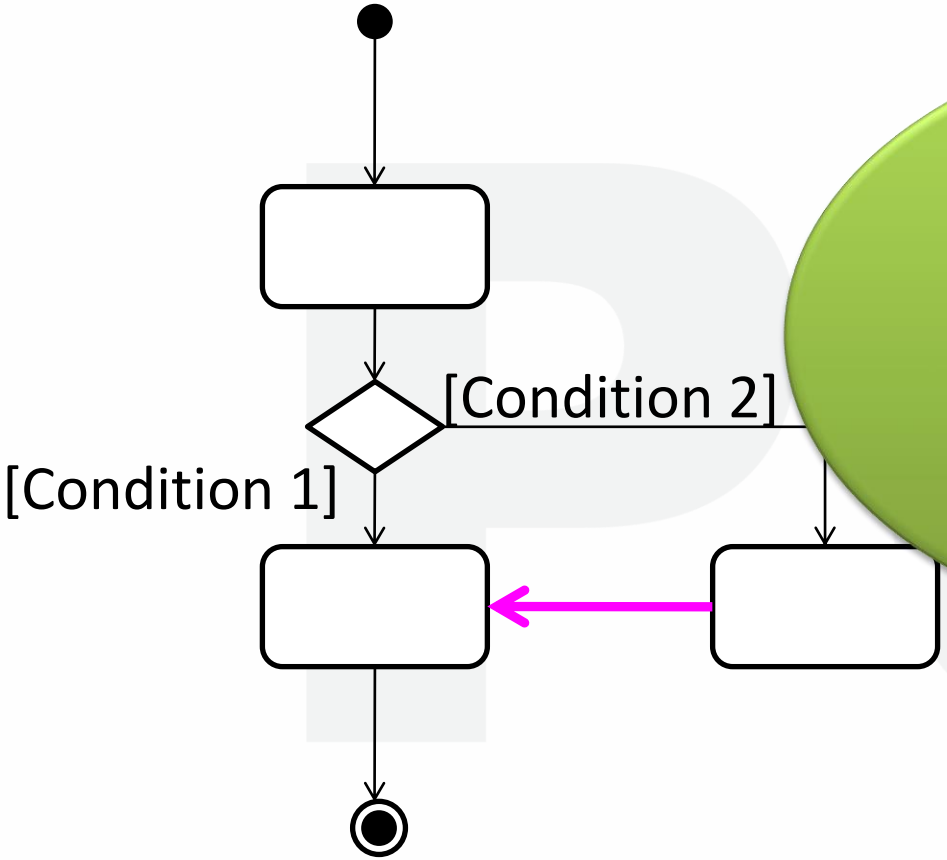
Decision points allow the flow to branch away from the Primary Path.

Transitions coming out of Decision Points must have a GUARD.



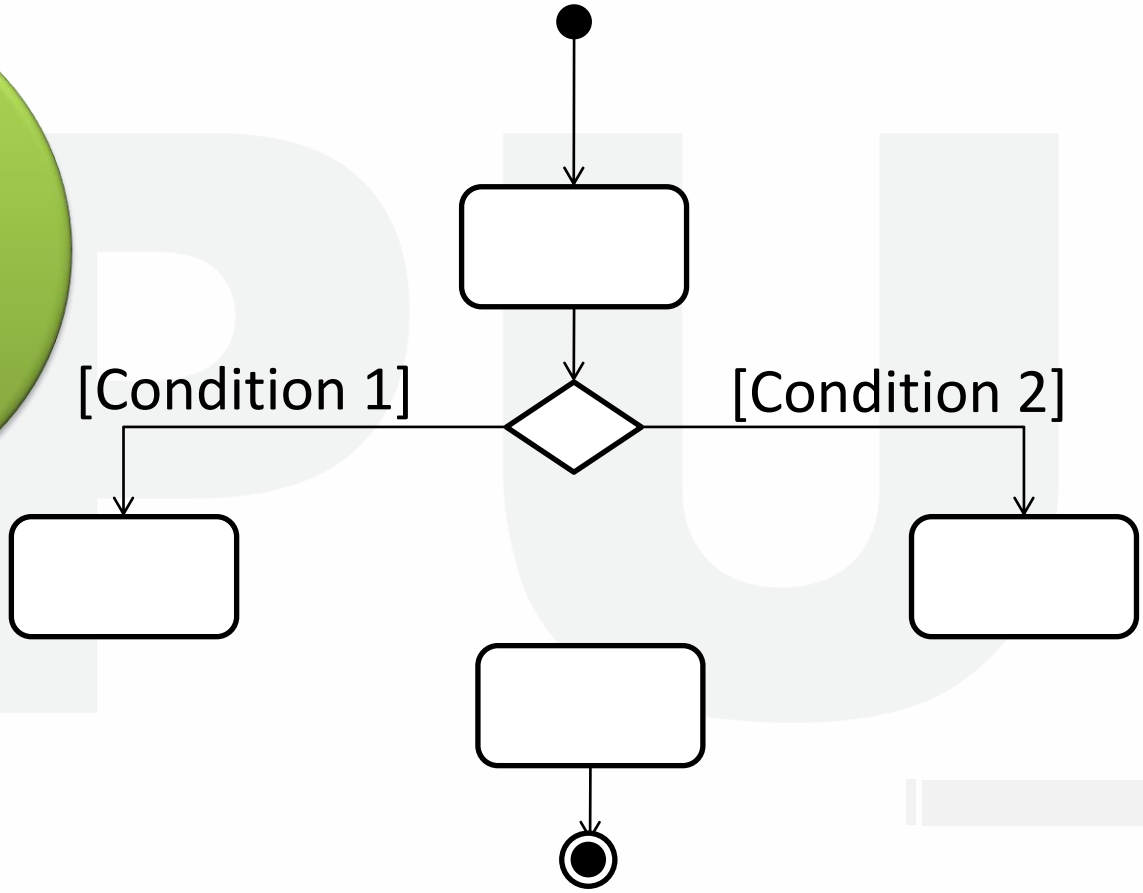
A Guard needs to explicitly describe a condition which must be true in order to proceed down that path.



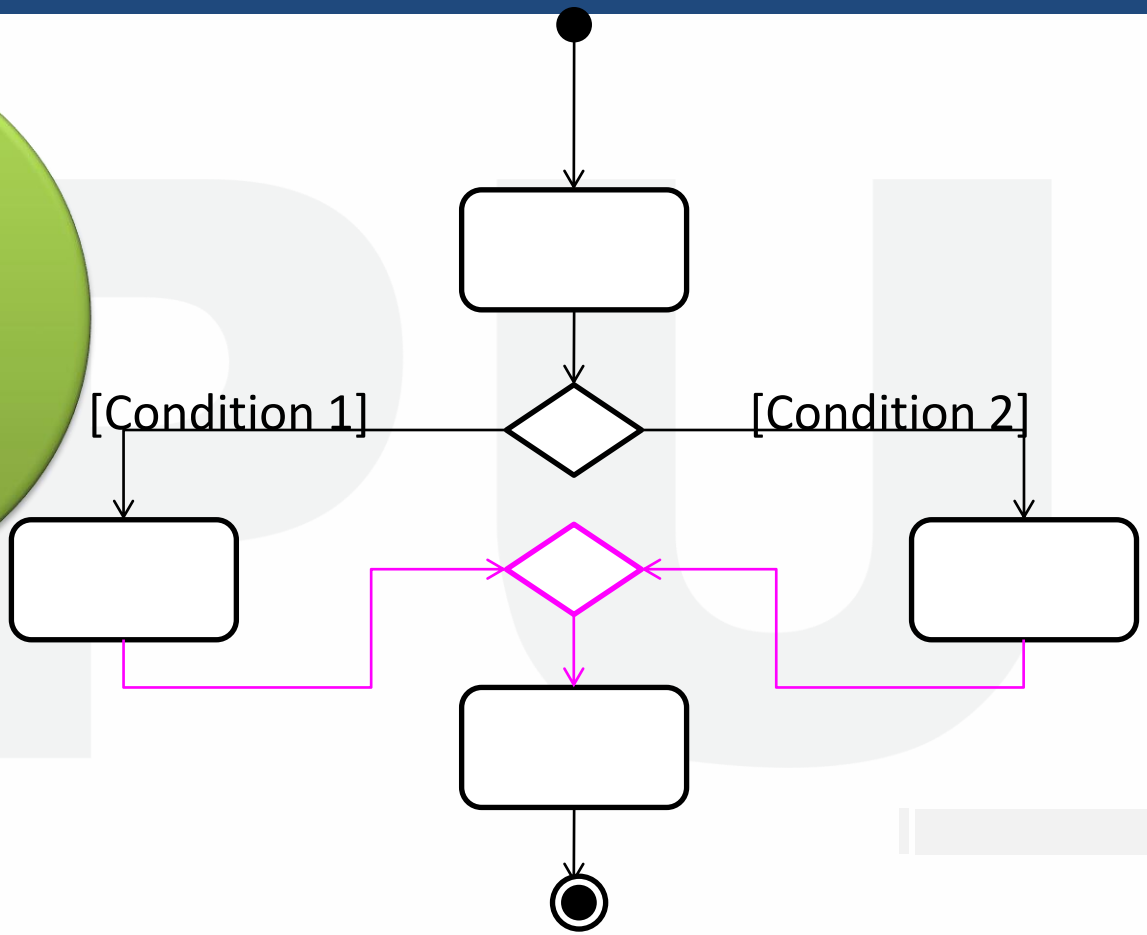


If the flow rejoins the Primary Path, it is known as an Alternate Path.

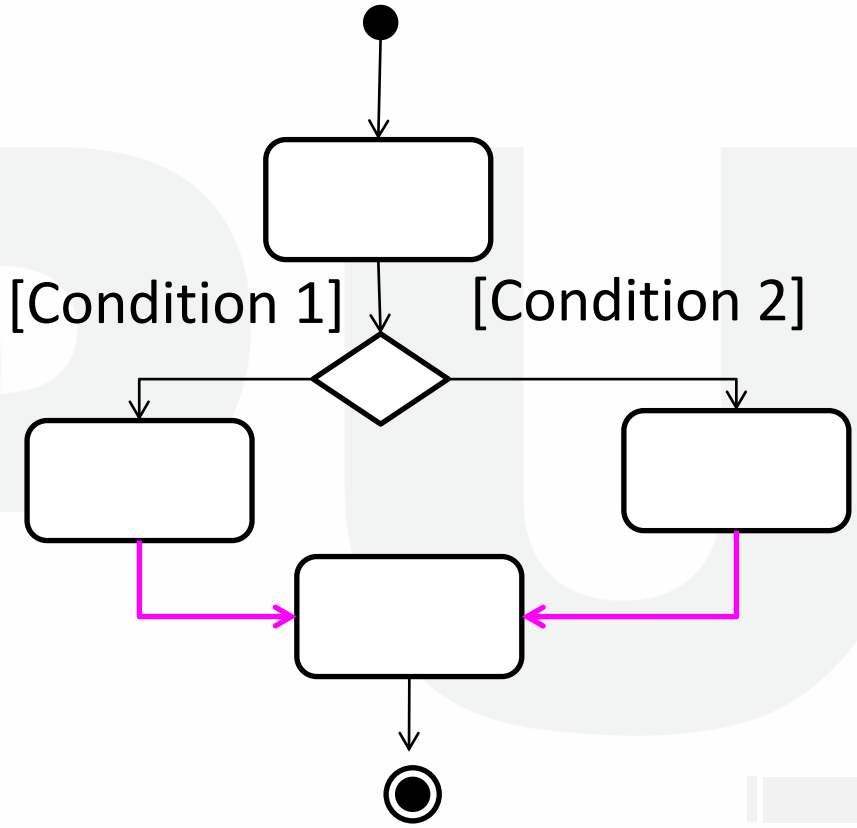
You can show how paths rejoin by using a MERGE POINT.



You can show how paths rejoin by using a MERGE POINT.



I prefer to model merging paths like this.



DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in