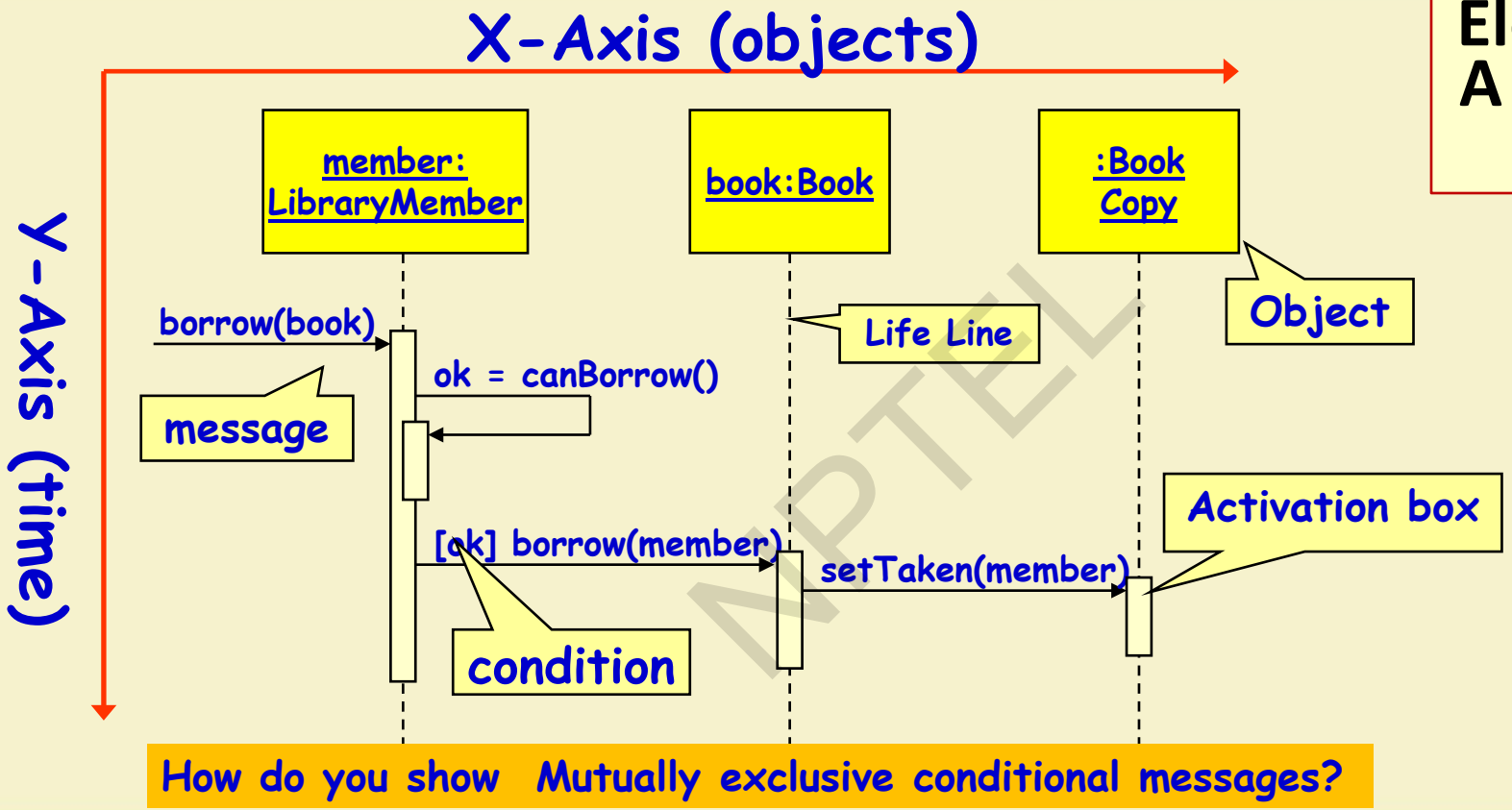
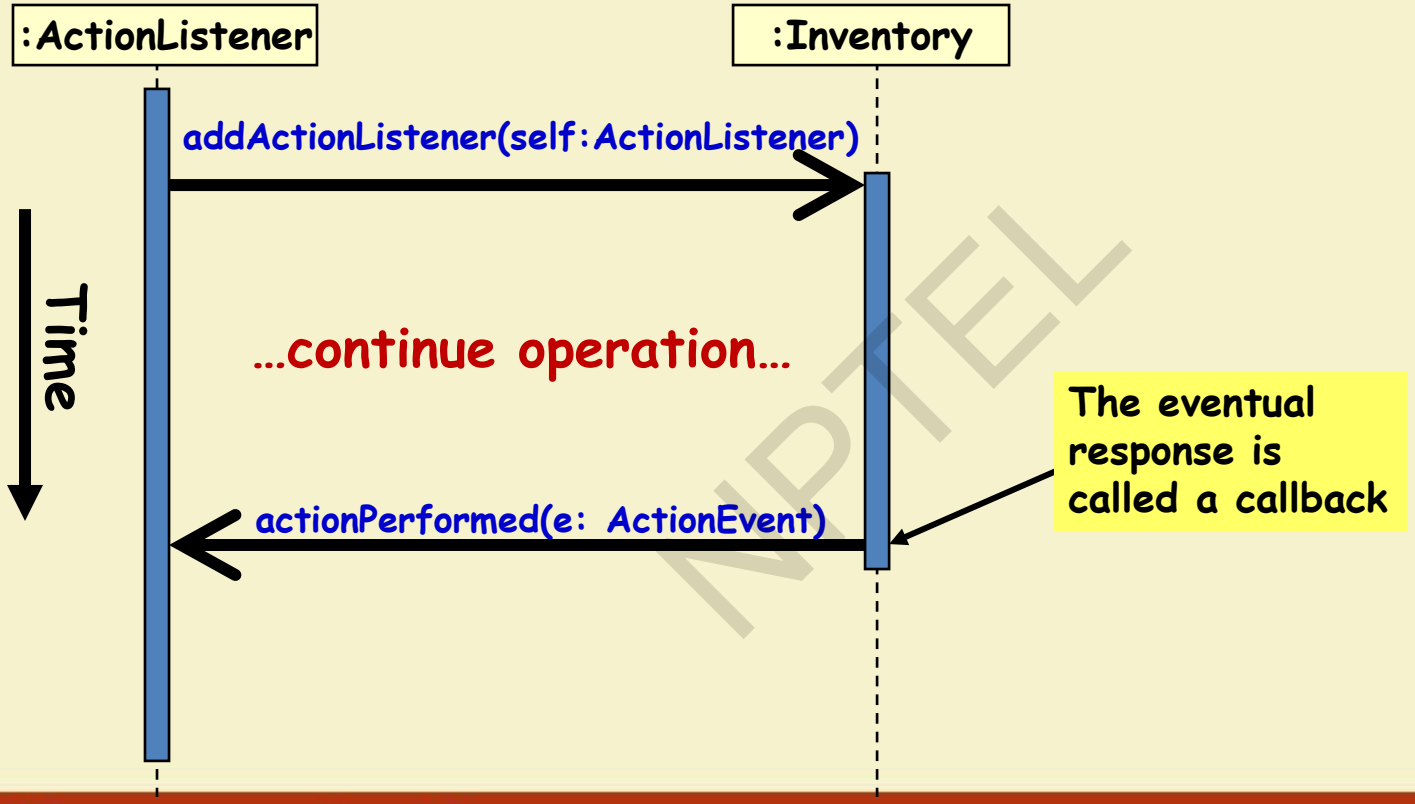
Elements of A Sequence Diagram



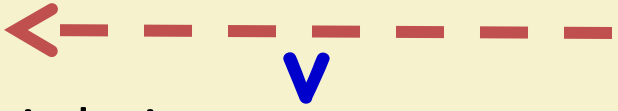
How do you show Mutually exclusive conditional messages?

Asynchronous Messages



Return Values

- Optionally indicated using a dashed arrow:
  - Label indicates the return value.
  - Don't need when it is obvious what is being returned, e.g. getTotal()

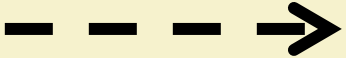


- **Model a return value only when you need to refer to it elsewhere:**
  - **Example:** A parameter passed to another message.

Summary of Kinds of Arrows

Method call (The sender losses control until the receiver finishes handling the message)

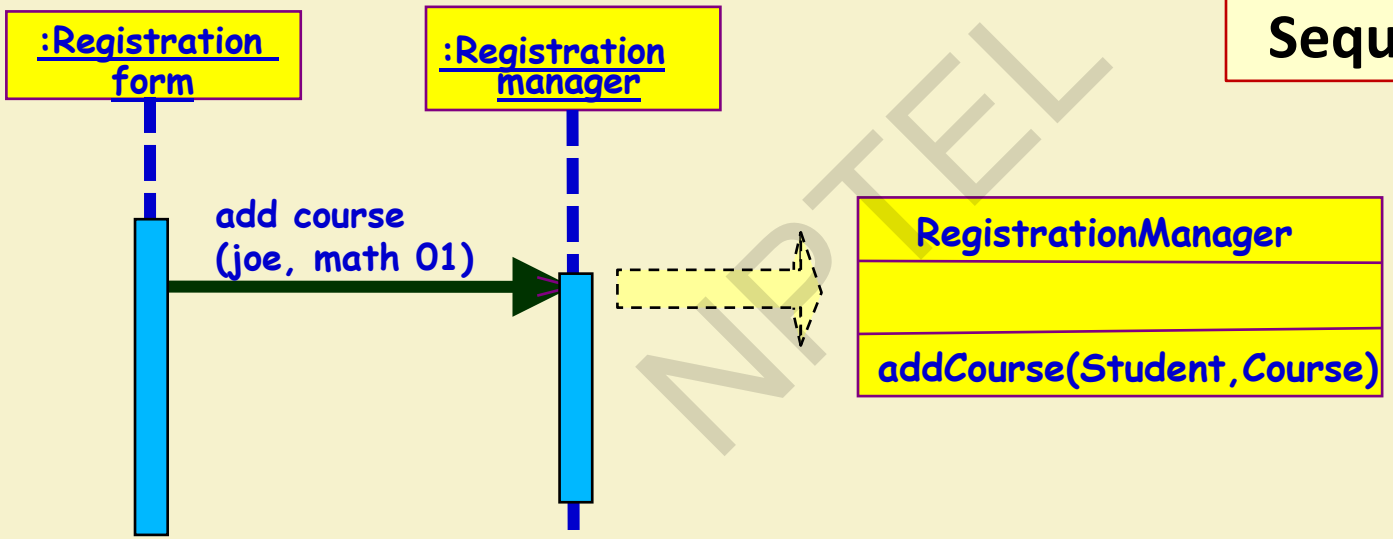
Flat flow of control(The sender does not expect a reply)



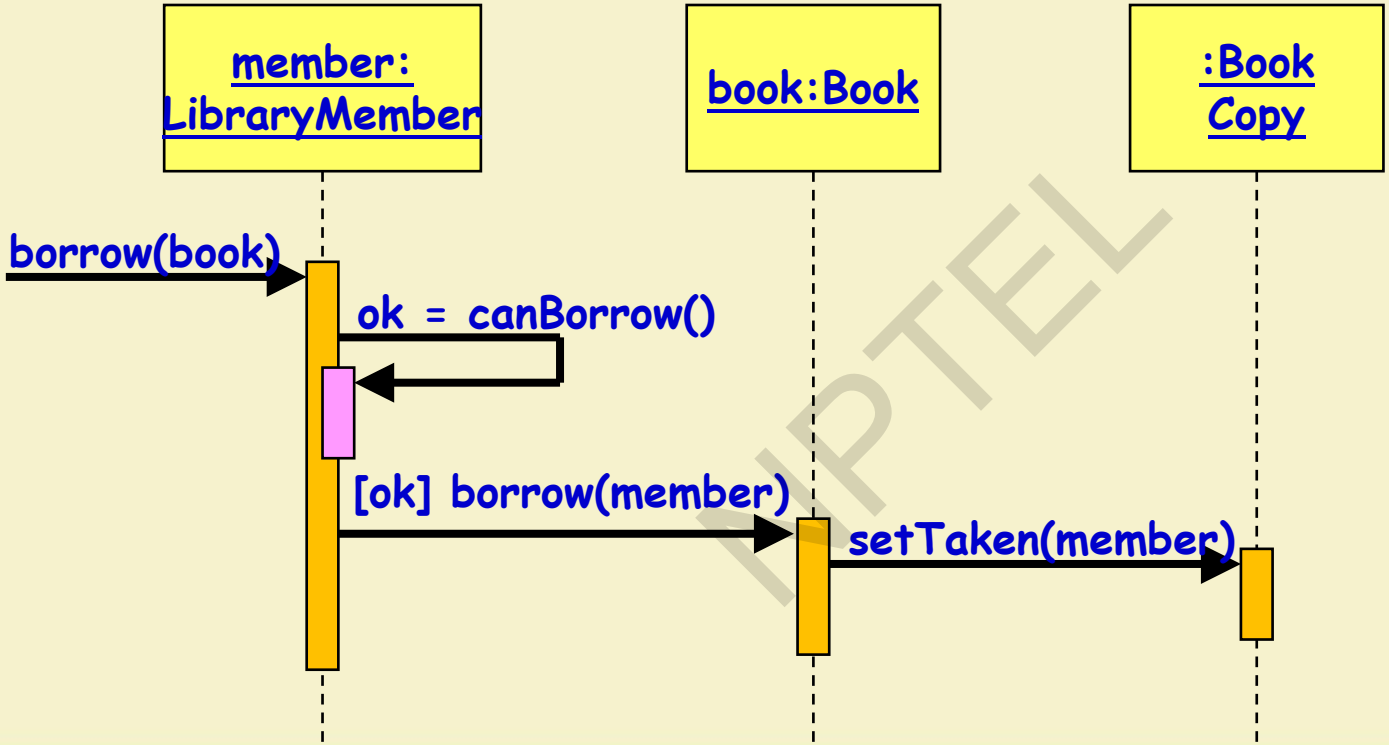
Return
(Unblocks a synchronous send)

- Methods of a class are determined from the interaction diagrams…

Method Population in Classes from Sequence Diagrams

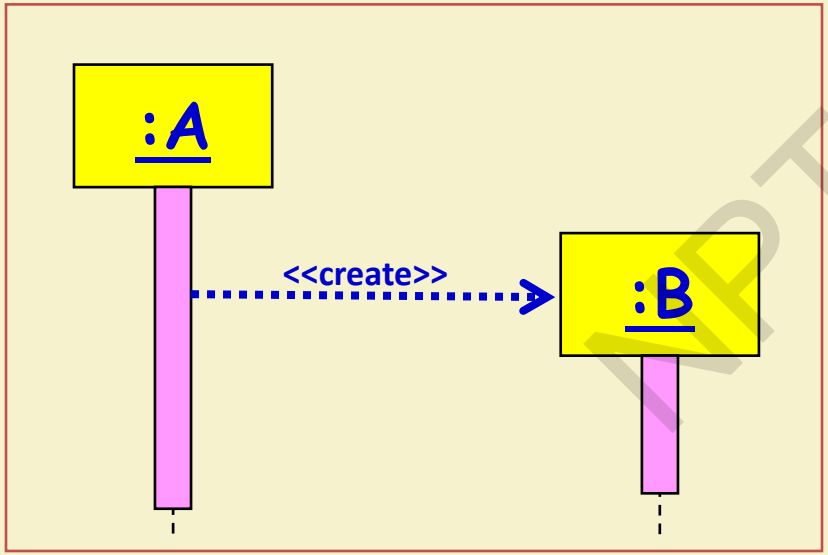


Example Sequence Diagram: Borrow Book Use Case



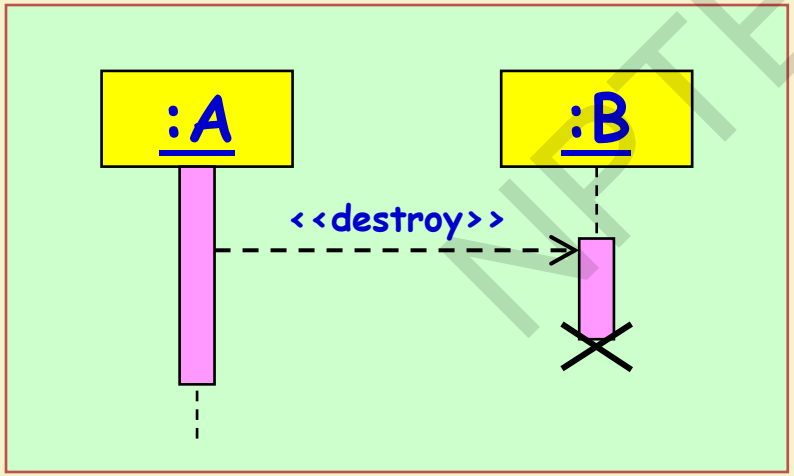
Object Creation

- An object may create another object ---represented using a **<<create>>** message.



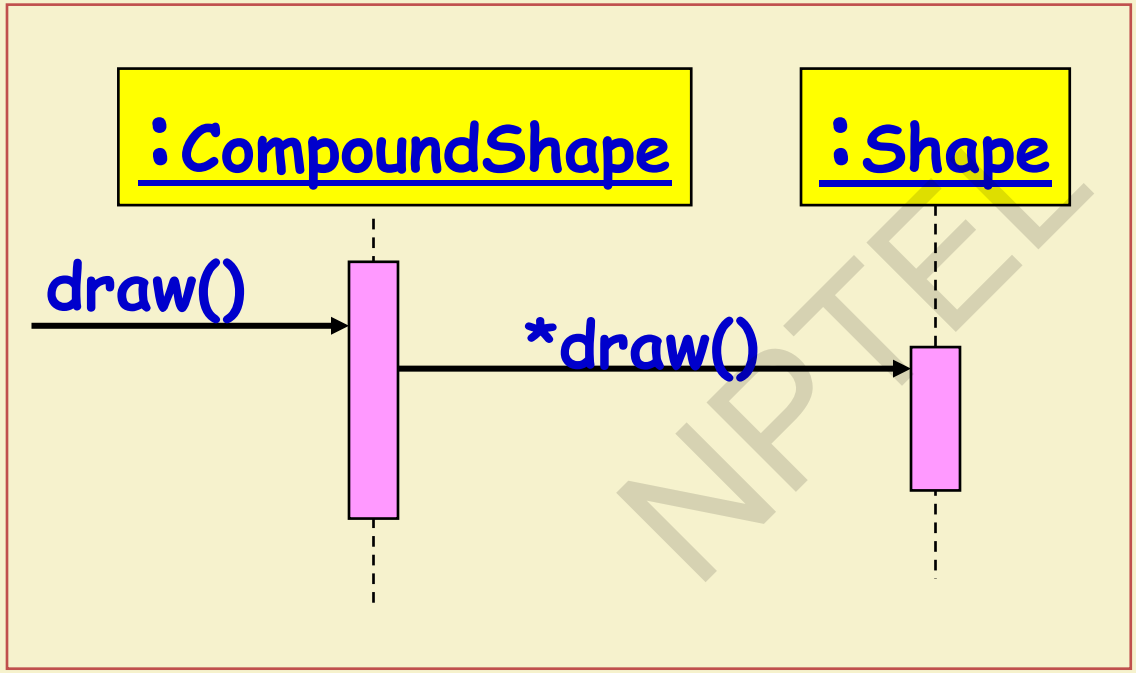
Object Destruction

- An object may destroy another object via a <<destroy>> message.
  - An object may also destroy itself.
- **But, how do you destroy an object in Java?**
  - **Avoid modeling object destruction unless memory management is critical.**

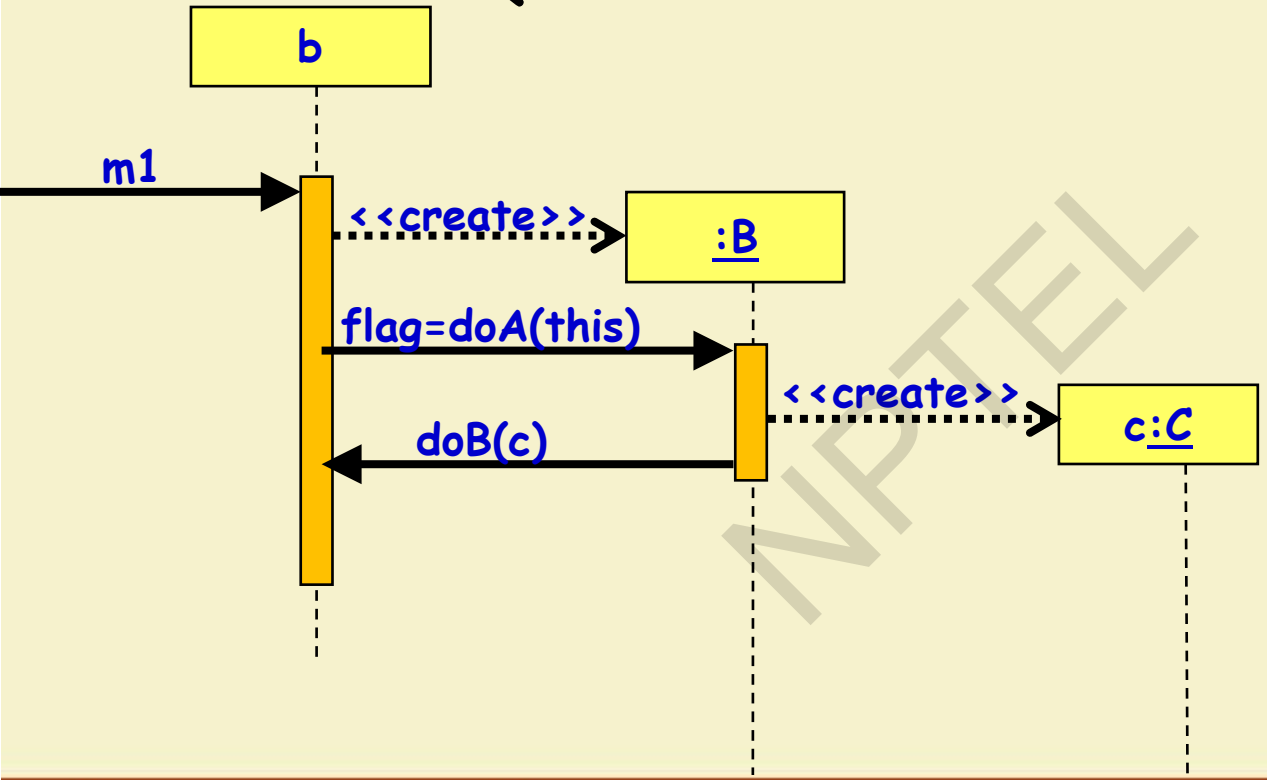


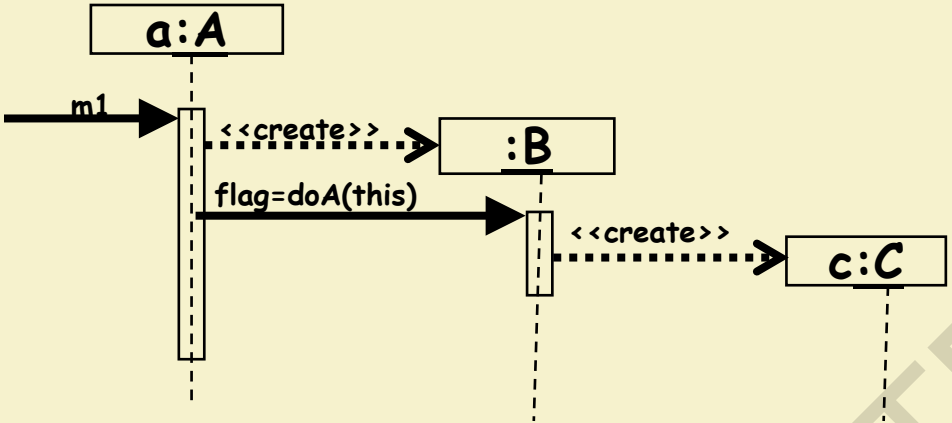
Control Information

Iteration example UML 1.x:



Quiz: Write Code for class B

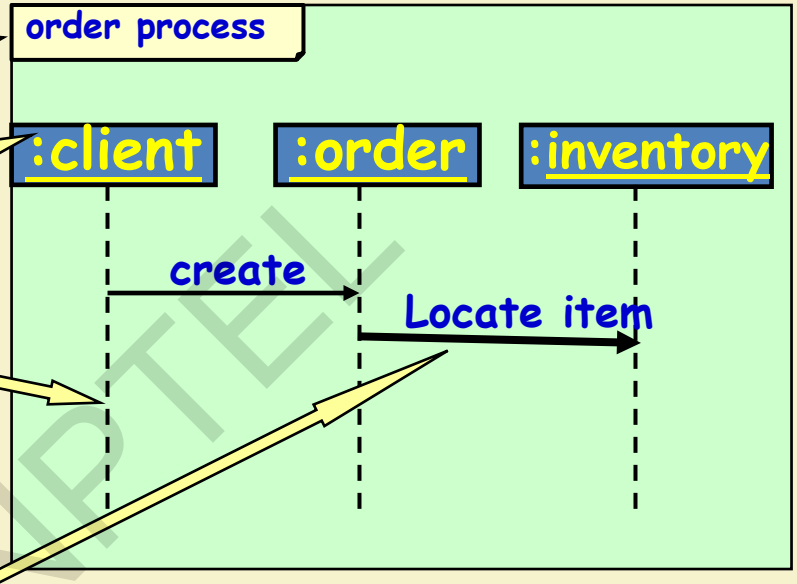




Quiz: Ans

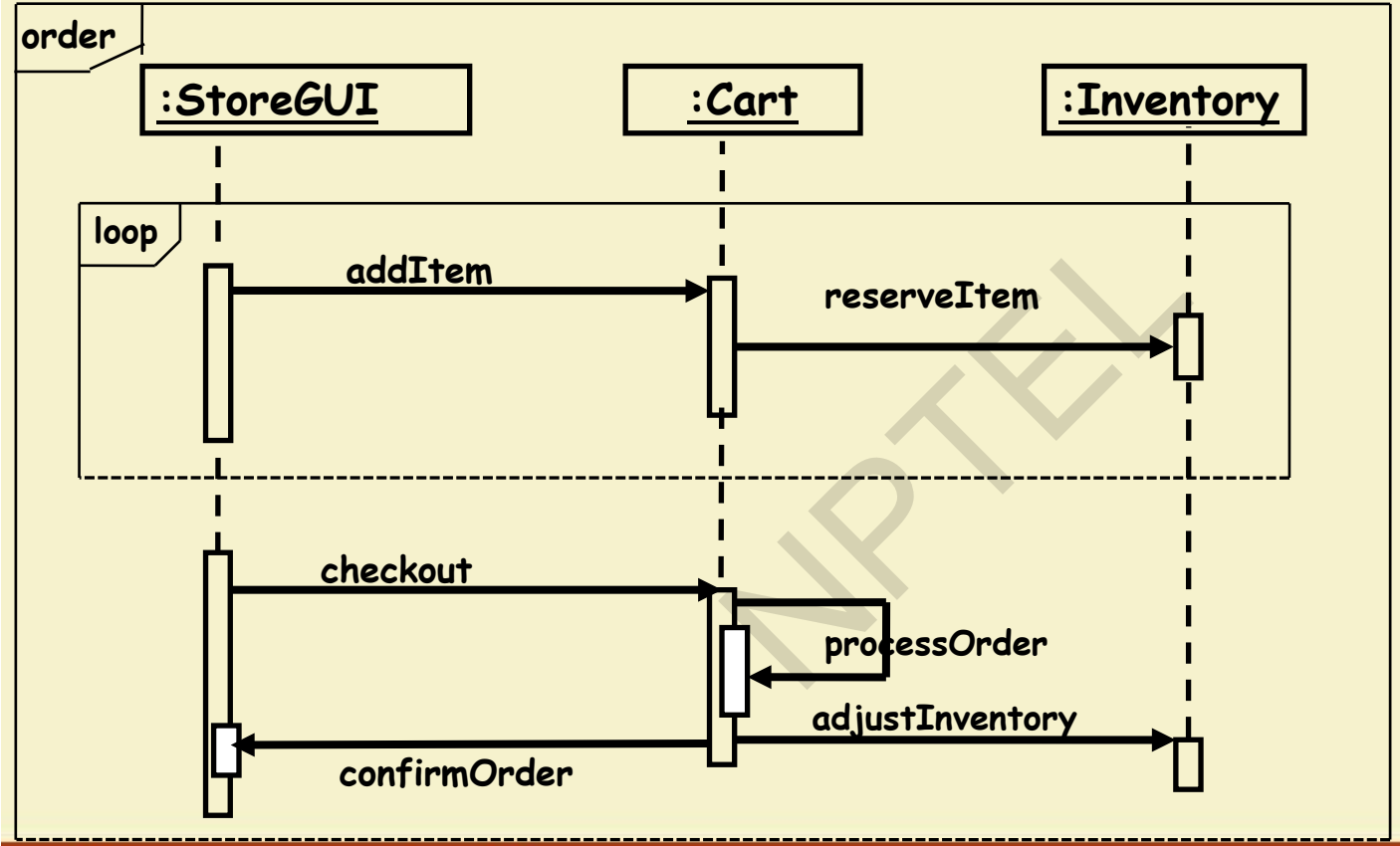
```
public class B {
…
int doA(A a) {
        int flag;
       C c = new C();
       a.doB(c); …
       return flag; }
}
```

- A **frame** consists of:
  1) Diagram identifier
  2) Participating objects:
     - A dotted line that extends for the time period of the interaction
  3) Messages to communicate among the participants

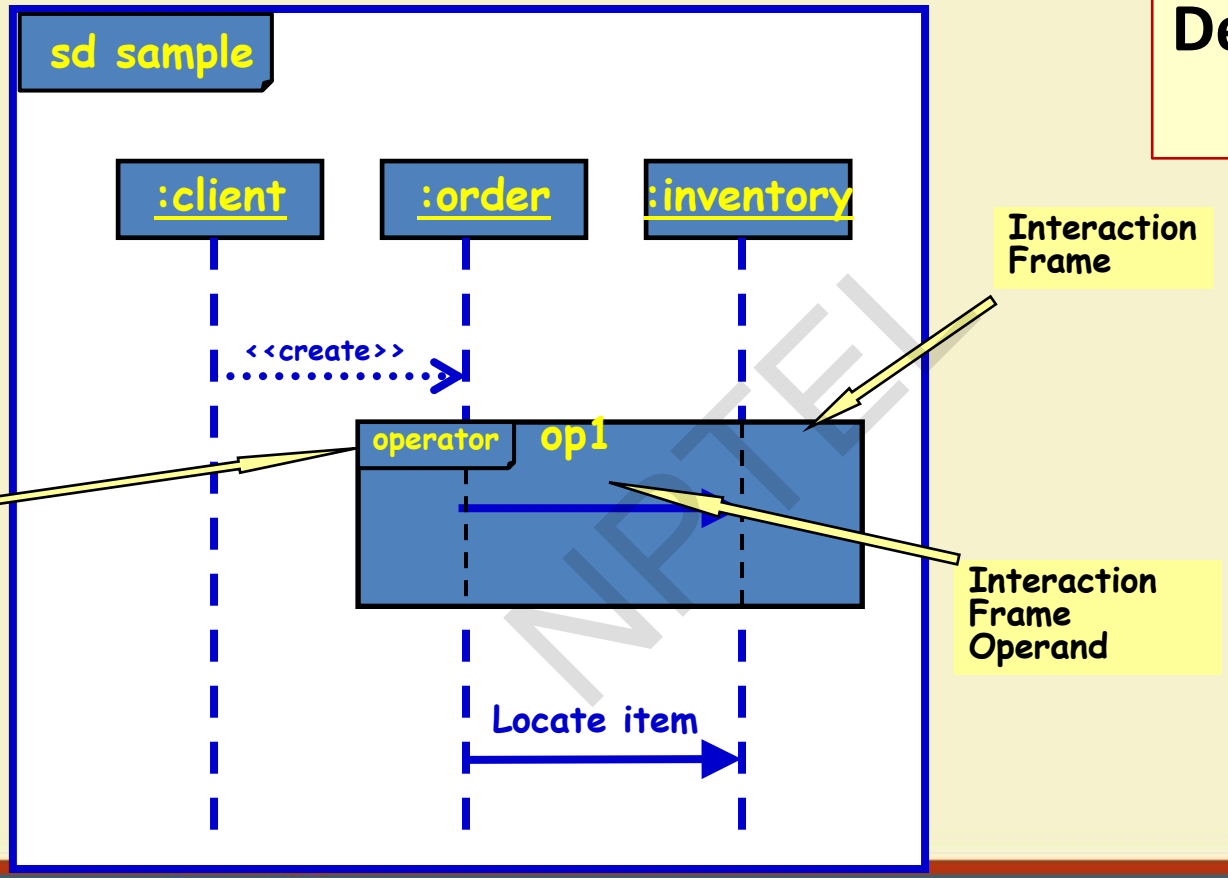


Frames in a Sequence Diagram

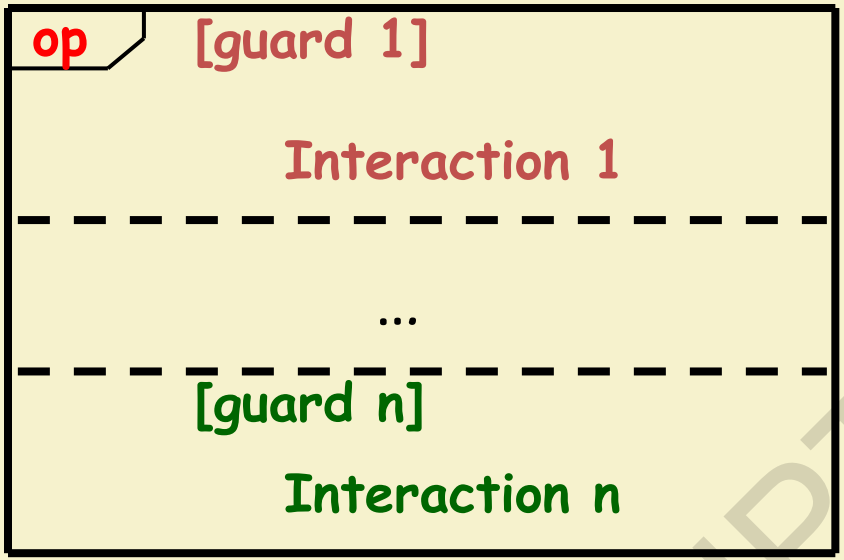
Example Frames in Sequence Diagrams



Depicting a Frame Graphically



Interaction Frame Syntax

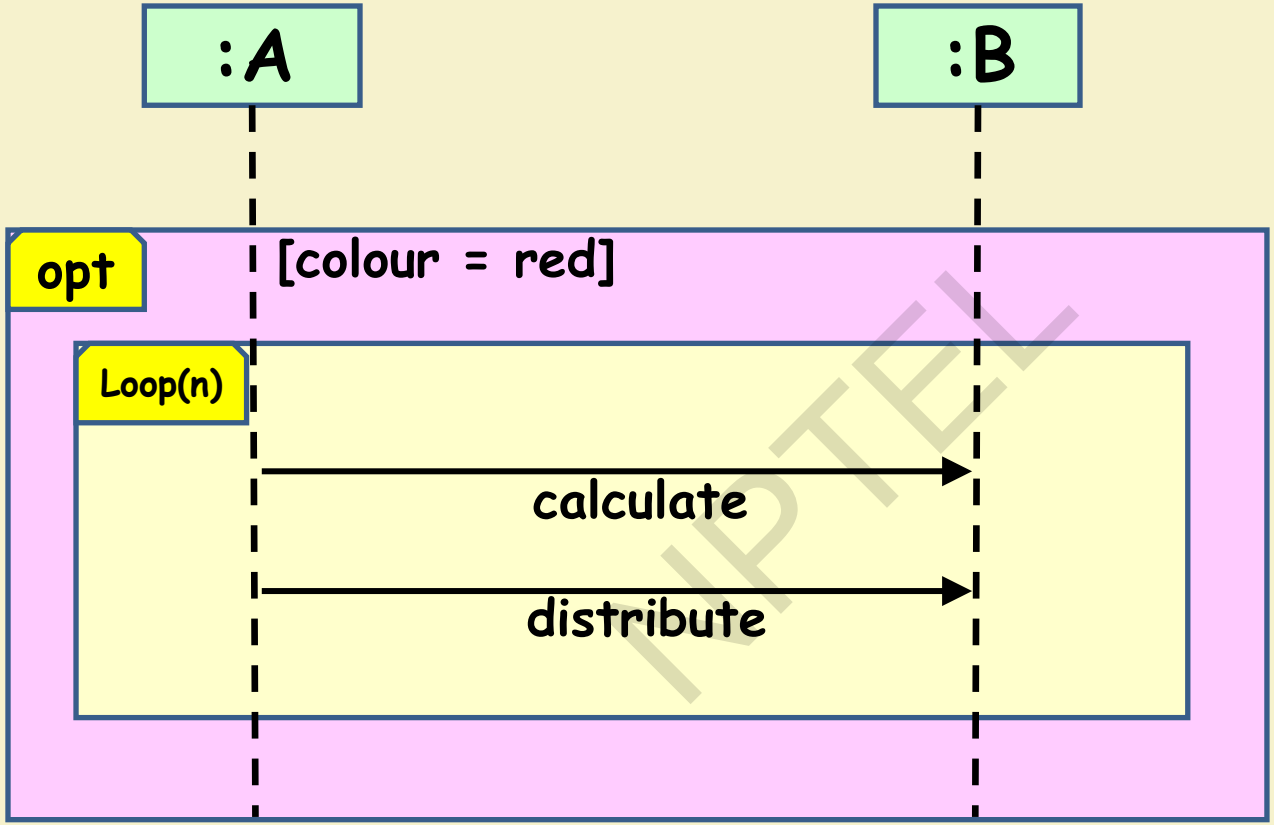


- Divided into a set of interactions by dotted lines
- **Interaction i** is executed if **guard i** is true

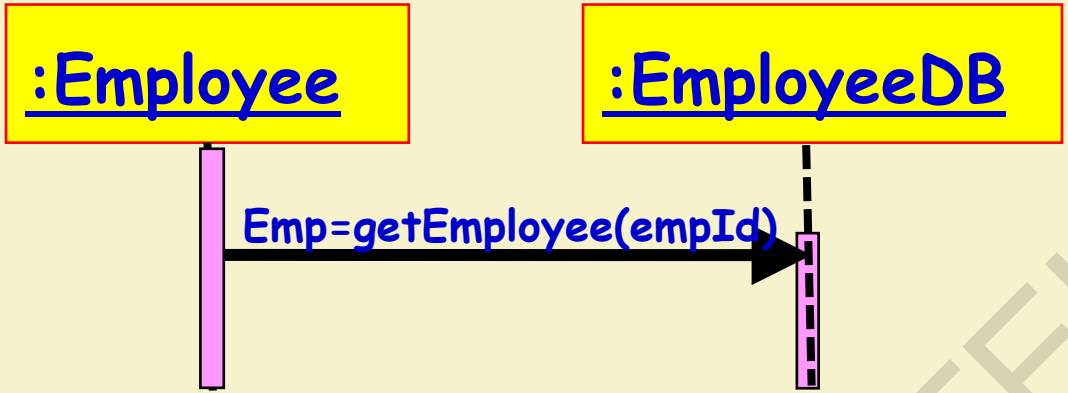
Frame Operators

Frame Operator	Meaning
Alt	Alternative fragment for conditional logic expressed in the guards
Loop	Loop fragment while guard is true. Can also write loop(n) to indicate looping n times.
Opt	Optional fragment that executes if guard is true
Par	Parallel fragments that execute in parallel
Region	Critical region within which only one thread can run

Nesting of Frames

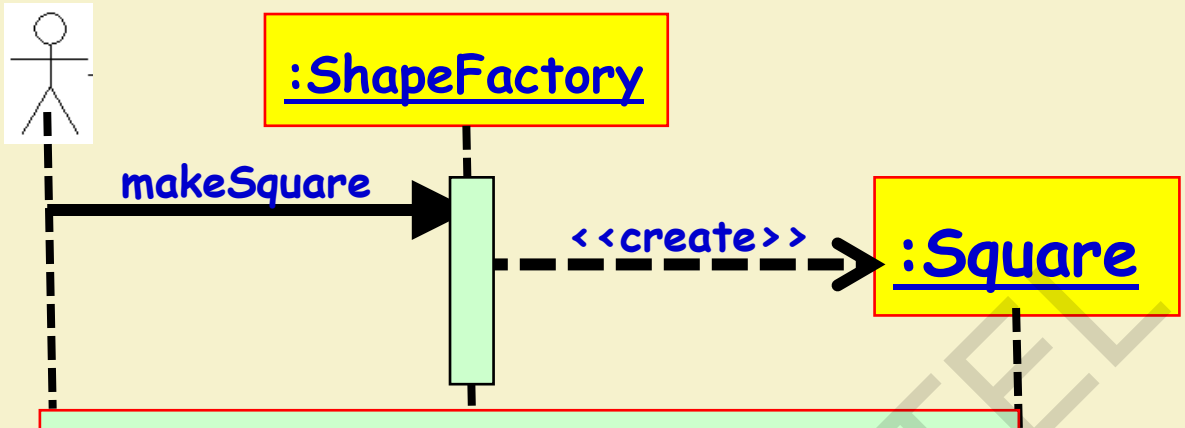


Code Generation: Example 1

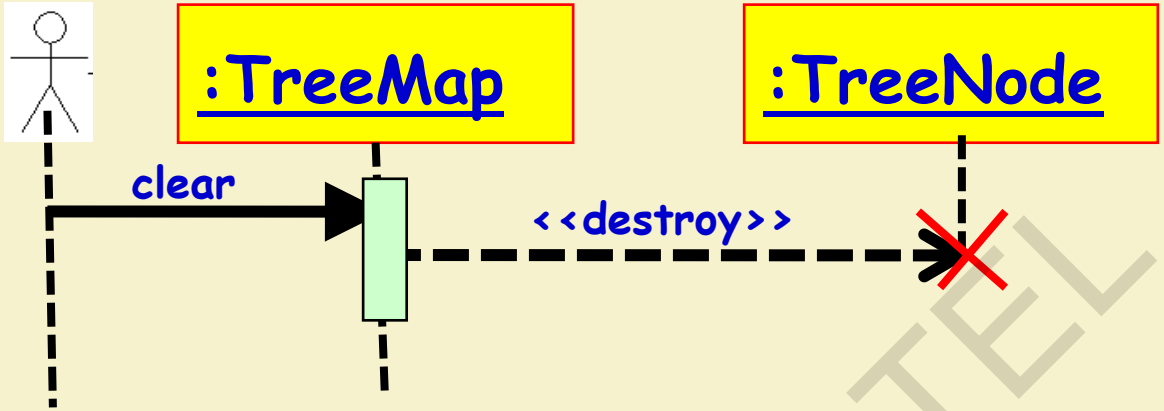



```
Public class EmployeeDB{
    public Employee getEmployee(String empId){
        …
    }
}
```

Code Generation: Example 2




```
Public class ShapeFactory{
    public Shape makeSquare(){
        return new Square(); }
}
```



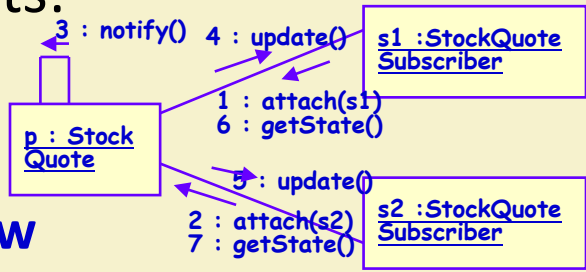
Exercise

Write Java Code

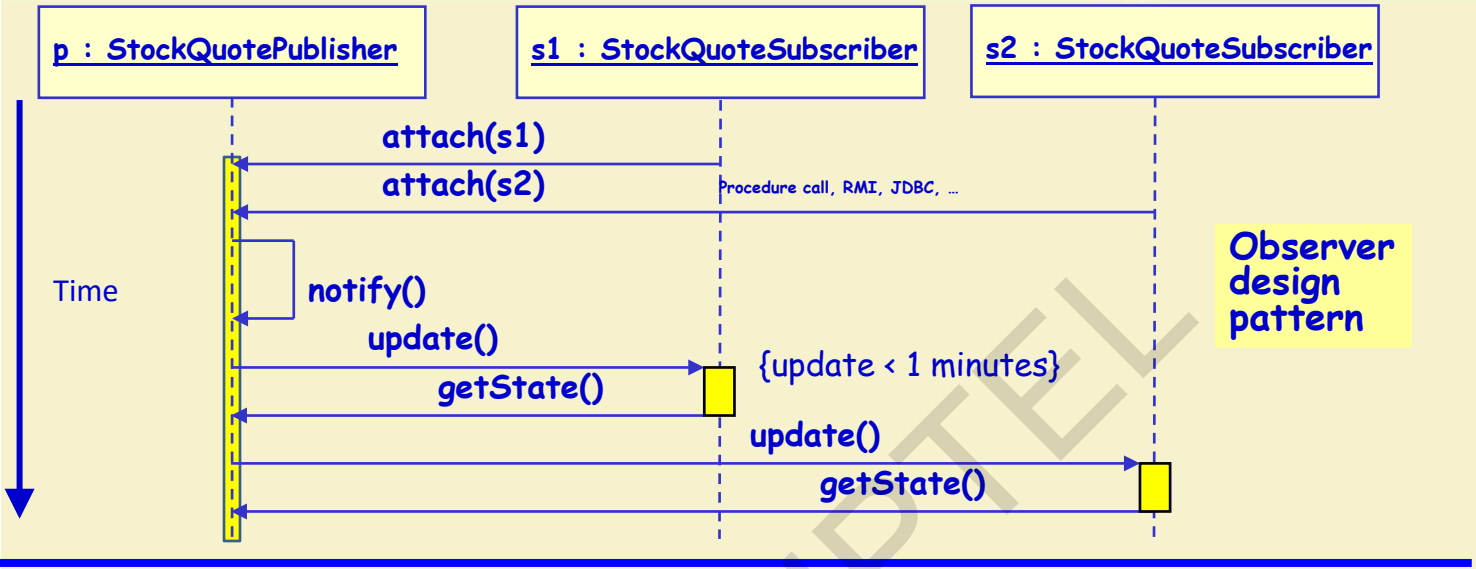
```
Public class TreeMap{
    private TreeNode t;
    public void clear(){
        t=null;
    }
}
```

Collaboration Diagram

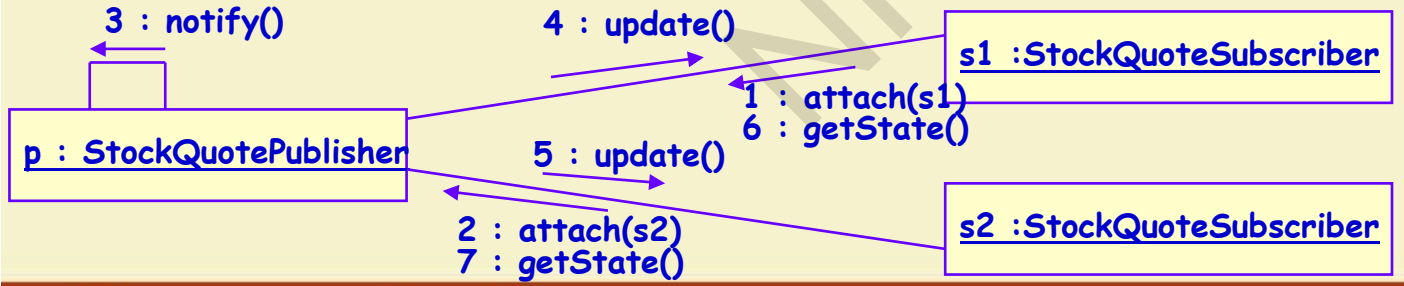
- **Known as Communication Diagram in UML 2**
- Shows both **structural** and **behavioural** aspects:
  - Objects are **collaborators**, shown as boxes
  - Messages between objects shown as **labelled arrow placed near links**
- Messages are prefixed with **sequence numbers** to show relative sequencing



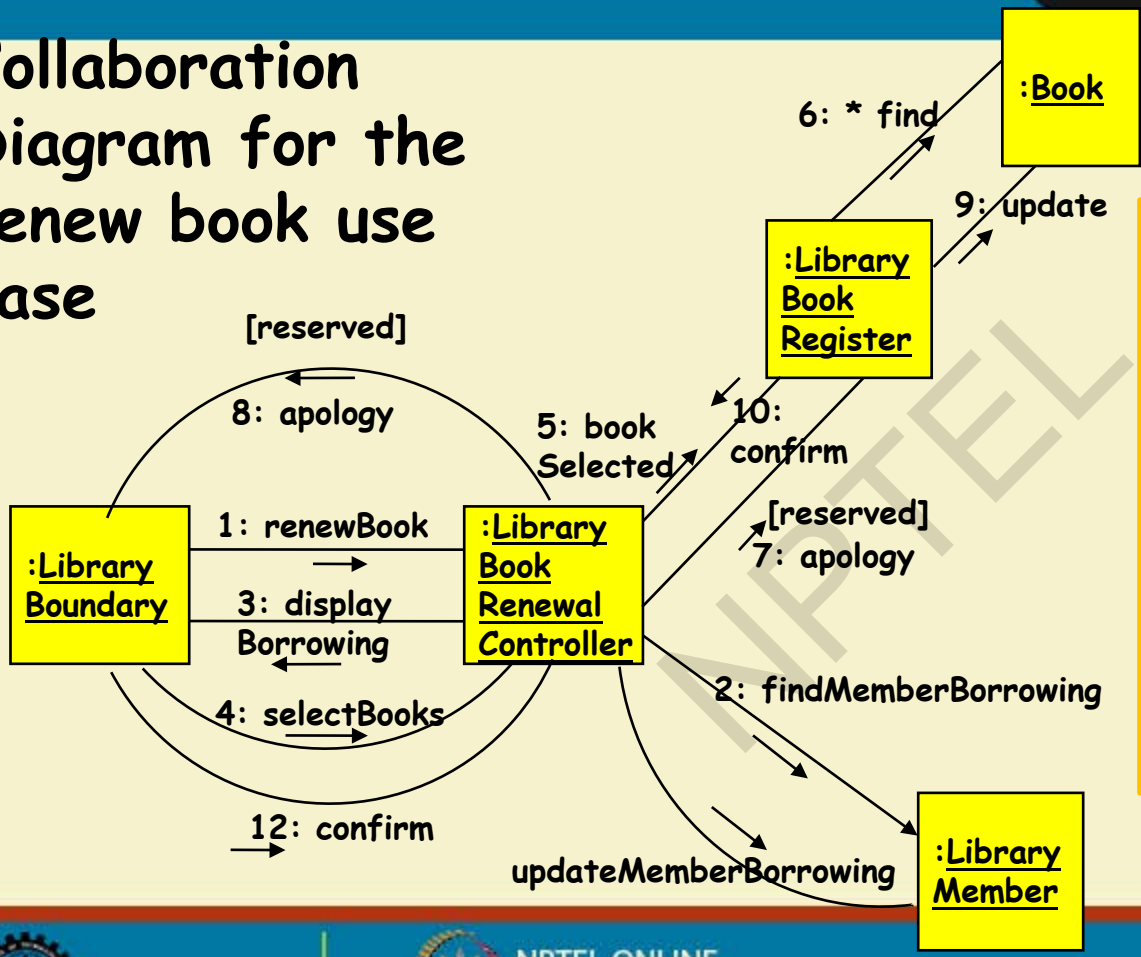
Sequence vs communication Diagrams

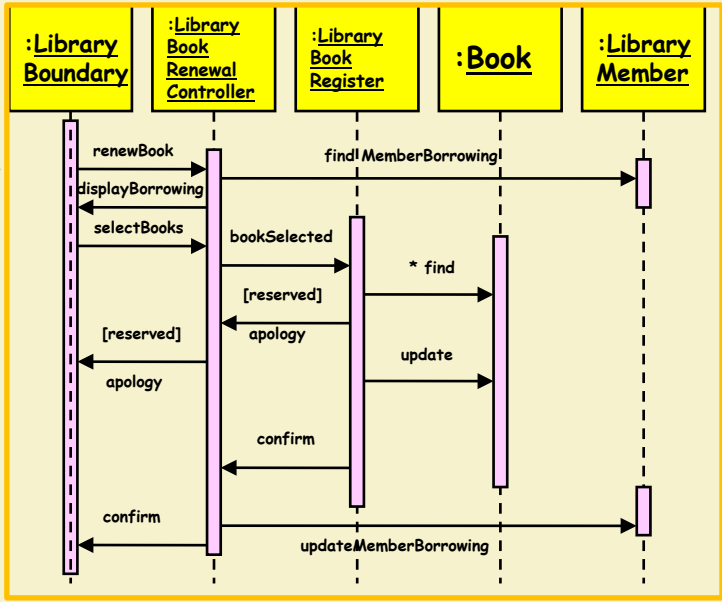


Observer design pattern



Collaboration Diagram for the renew book use case





Activity Diagram

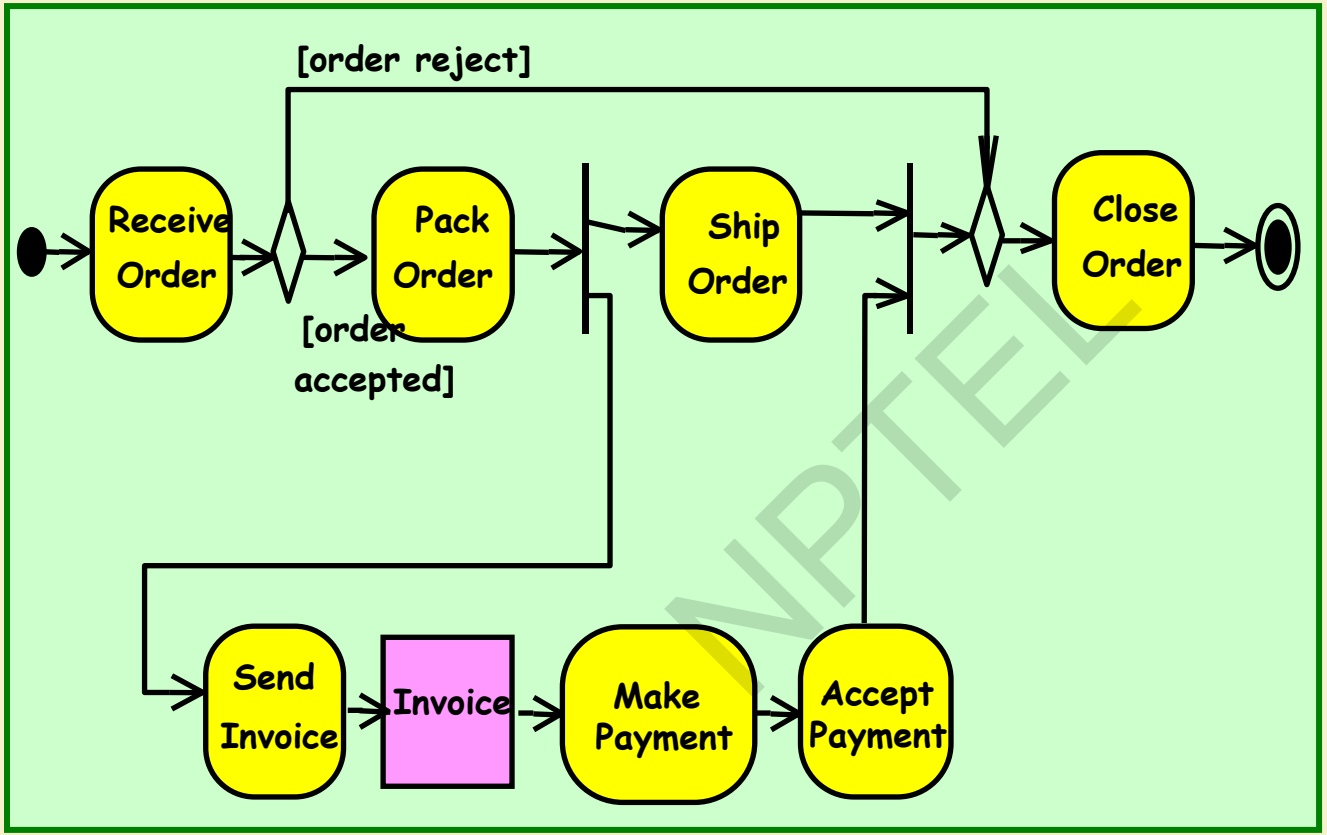
Activity Diagram

- Not present in earlier modelling languages:
  - Possibly based on event diagram of **Odell** [1992]
- **Often used to represent processing steps in a use case or a group of use cases (workflow) :**
  - **Basic activities may or may not correspond to methods**
- **An activity is a state with an internal action and has one or more outgoing transitions, e.g. fillOrder.**
- Vaguely similar to a flowchart...

Uses of Activity Diagram

- Normally employed in business process modelling.
  - **Spans one or more use cases**
- Carried out during requirements analysis and specification stage.
- Useful in developing use cases and test cases.

Activity diagram First Example



Activity Diagram Elements

- Initial node
- Activity final node
- Action
- Action Flow/Object flow/edge
- Fork
- Join
- Decision
- Merge
- Synch
- Object

Actions

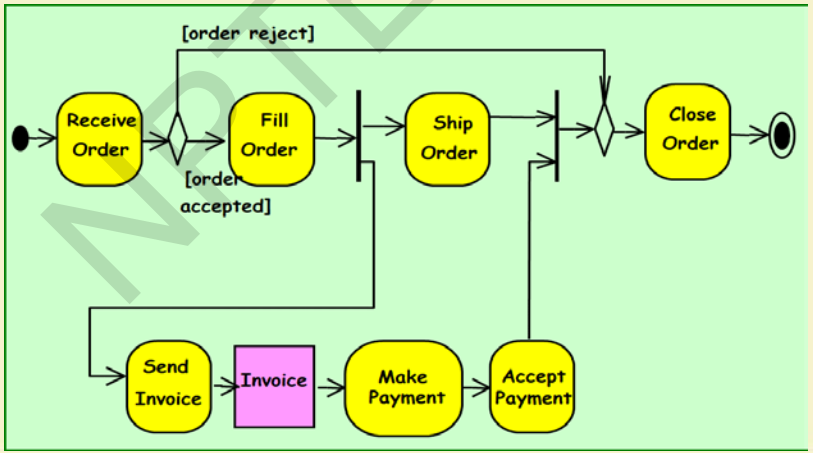
- **The fundamental unit of execution and behavior modelling.**

  –Represents some processing steps in the modeled system…

  –**Examples:** Receive order, Check order, Ship order, Ship invoice, etc.

Flows

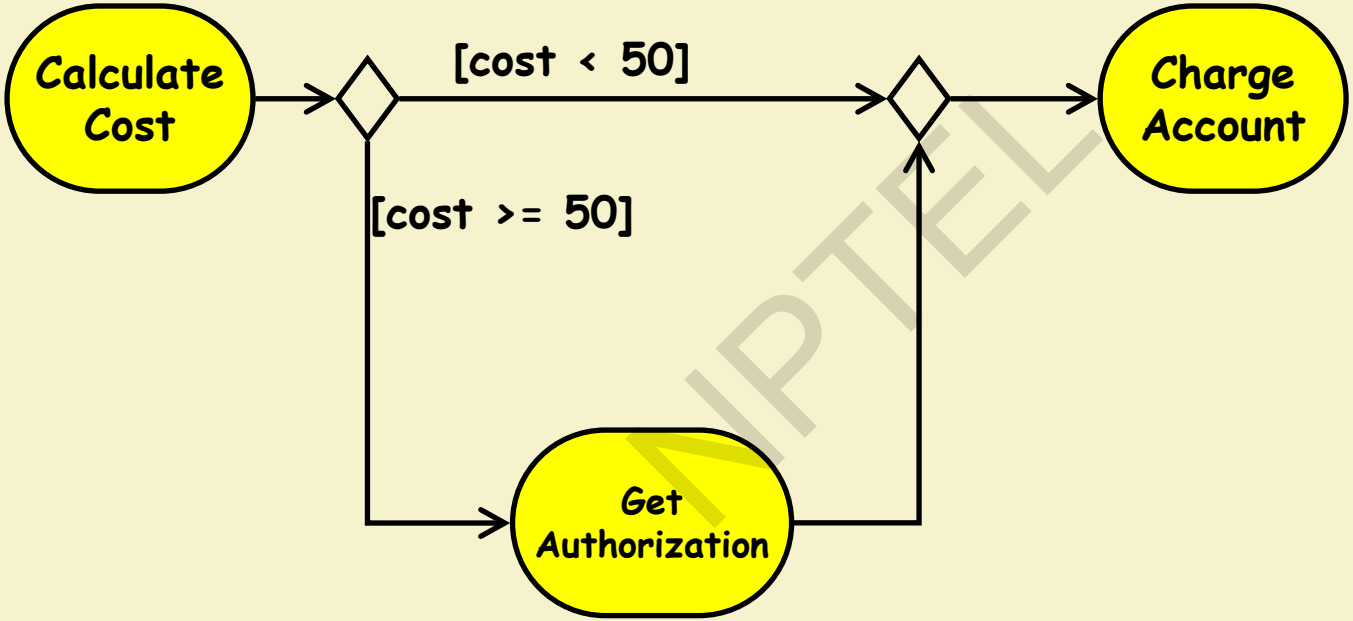
- Flows in an activity diagram do not have labels.
- Two types:
  - **Control-flow** transitions indicate completion of an action and possibly start of another
  - **Object-flow** transitions indicate that an action inputs or outputs an object



Controlling Steps

- Similar to state machines
- Initial state
- Final state
- Fork and join

- Decision point and merge ( ): ◇



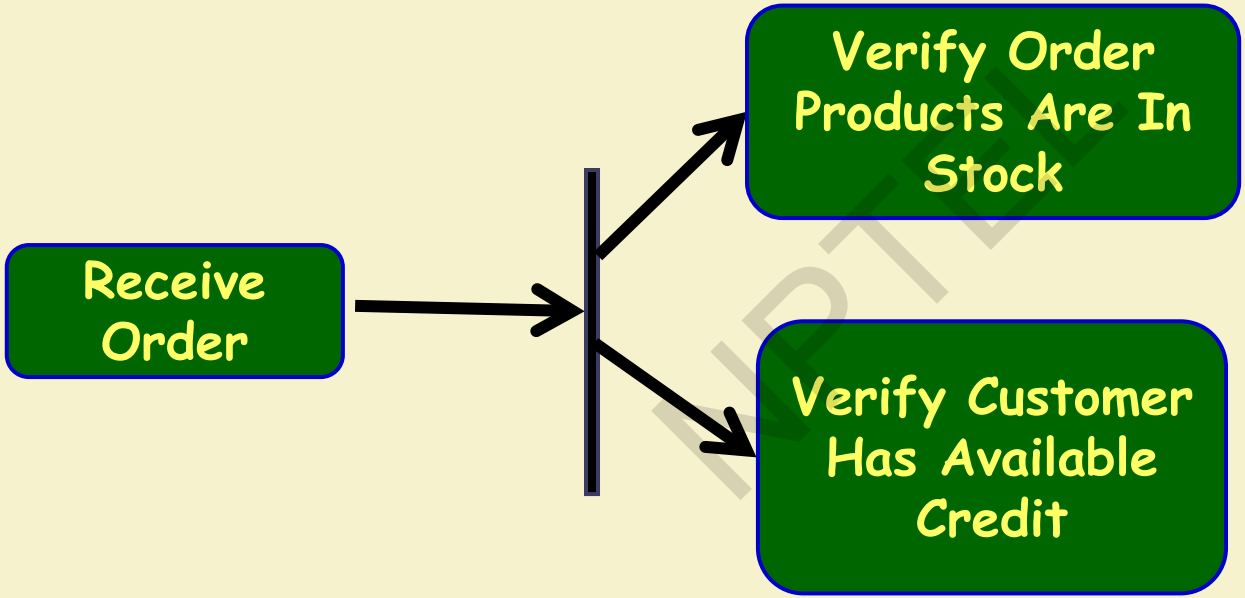
Initial and Final Nodes

- Initial node: filled circle.
- Final node: Circle around filled circle

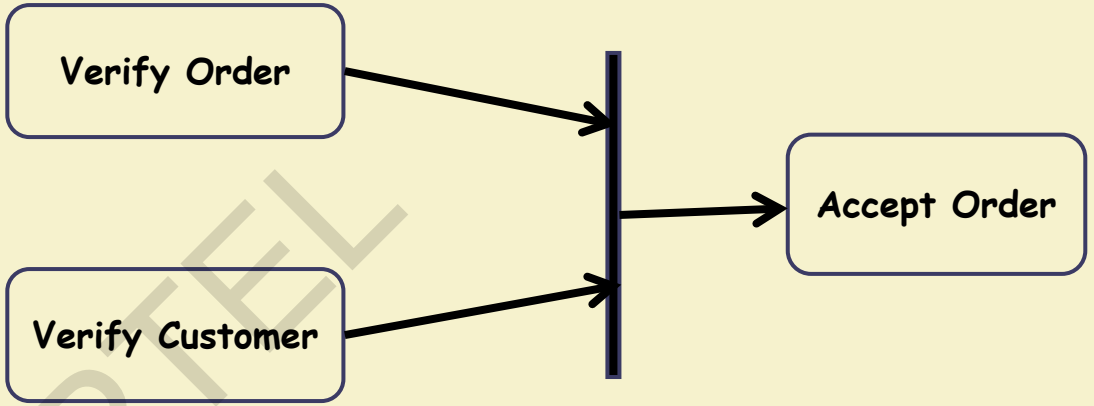
An activity diagram can have zero or more final nodes

Fork

- Denotes the beginning of parallel actions.

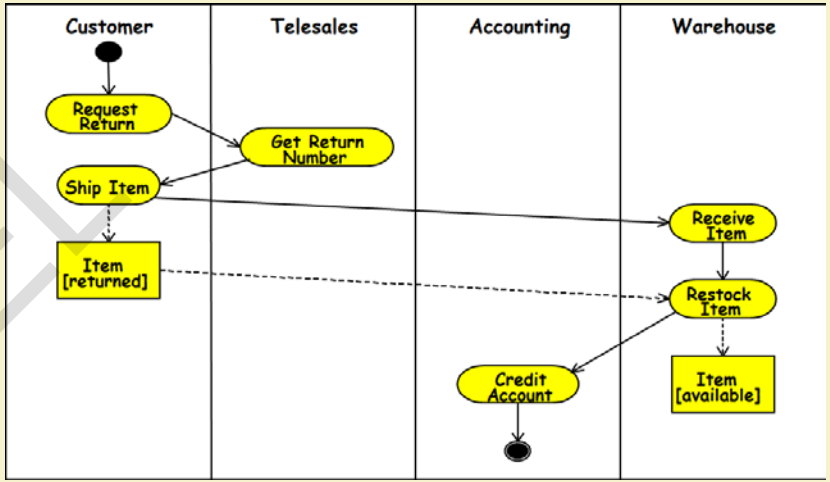


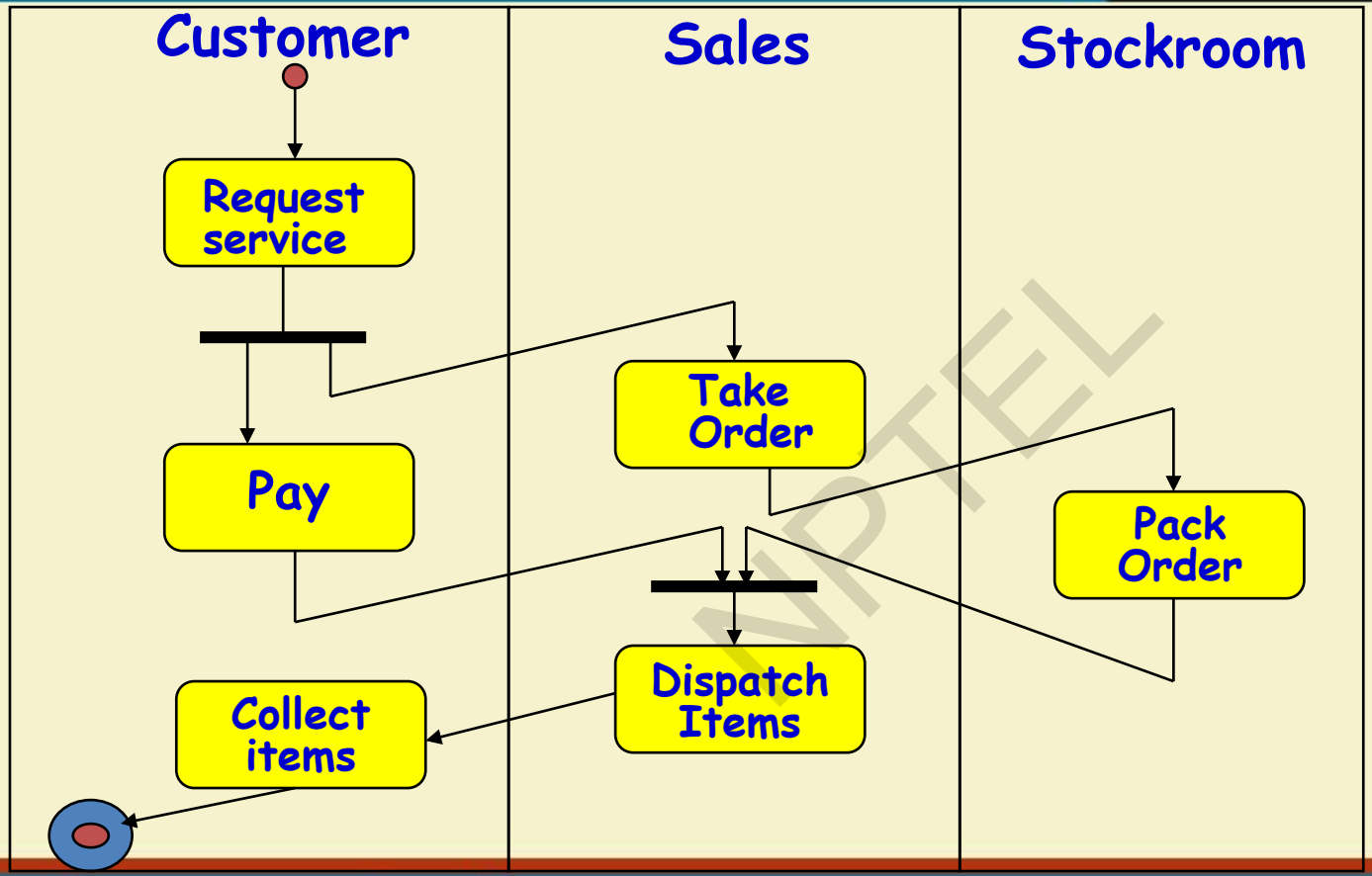
Join



Swim Lane

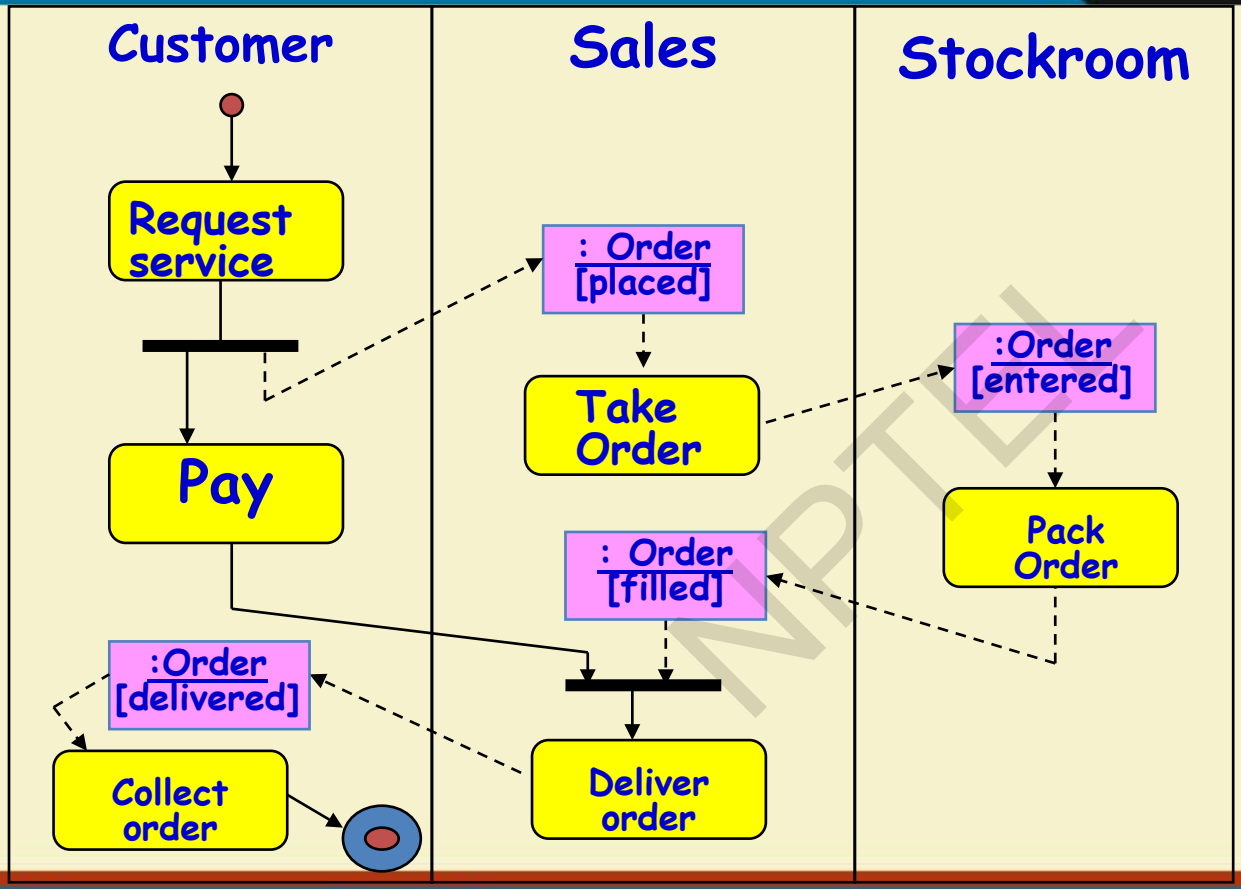
- Swim lanes are used to represent who performs which activities.
- Each action is assigned to one swim lane.
- Activity flows can cross lanes.
- Relative ordering of swim lanes has no semantic significance.




Customer
Sales
Stockroom
Request service
Pay
Collect items
Take Order
Dispatch Items
Pack Order

Object Flow

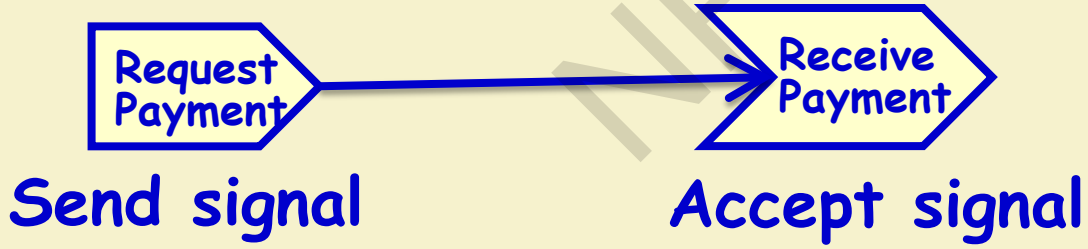
- **Take Order** produces an order object.
- **Pack Order** takes an order object as input.
- **Dashed lines used with object flow have the same semantics as normal transition.**



The transition between an object parameter and an action state is represented with a dashed line, instead of a solid line

Signals

- Signals are essentially messages with external systems/processes.
  - Usually appear in pairs: sent and received signals, as response is often expected for a sent signal.



Received Signals

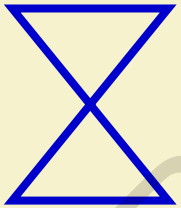
- A received signal:
  - **Indicates that the activity receives an event from an outside process.**
  - **An activity gets reported of the signal, and it is represented in the diagram how the activity reacts.**
- A time signal occurs on passage of time

  - For example, each month end might trigger a signal.

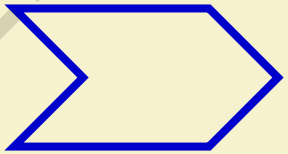
Signals

- Look at signals are flow triggers...

Send signal

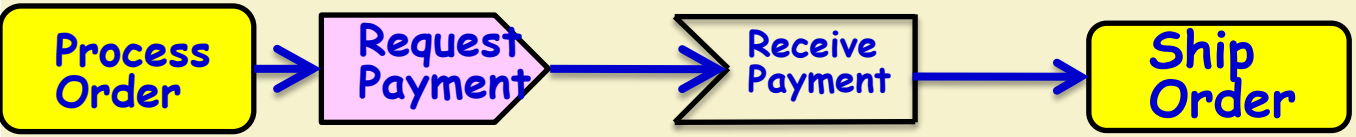


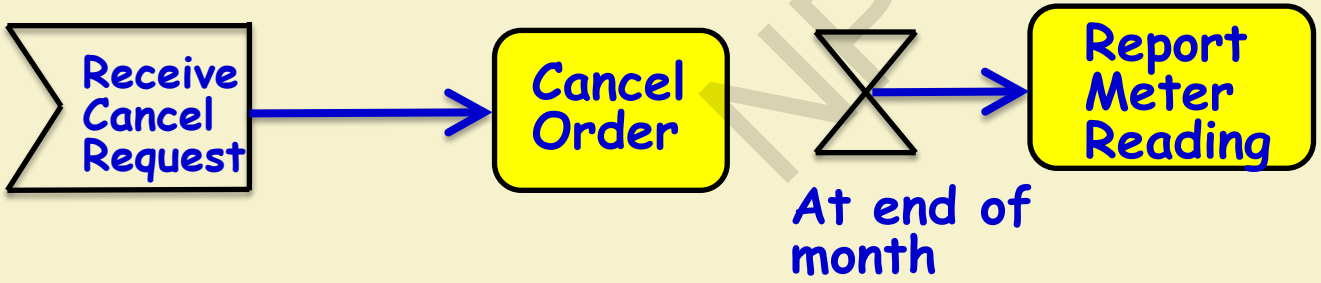
Time signal



Accept signal

Signal: Examples

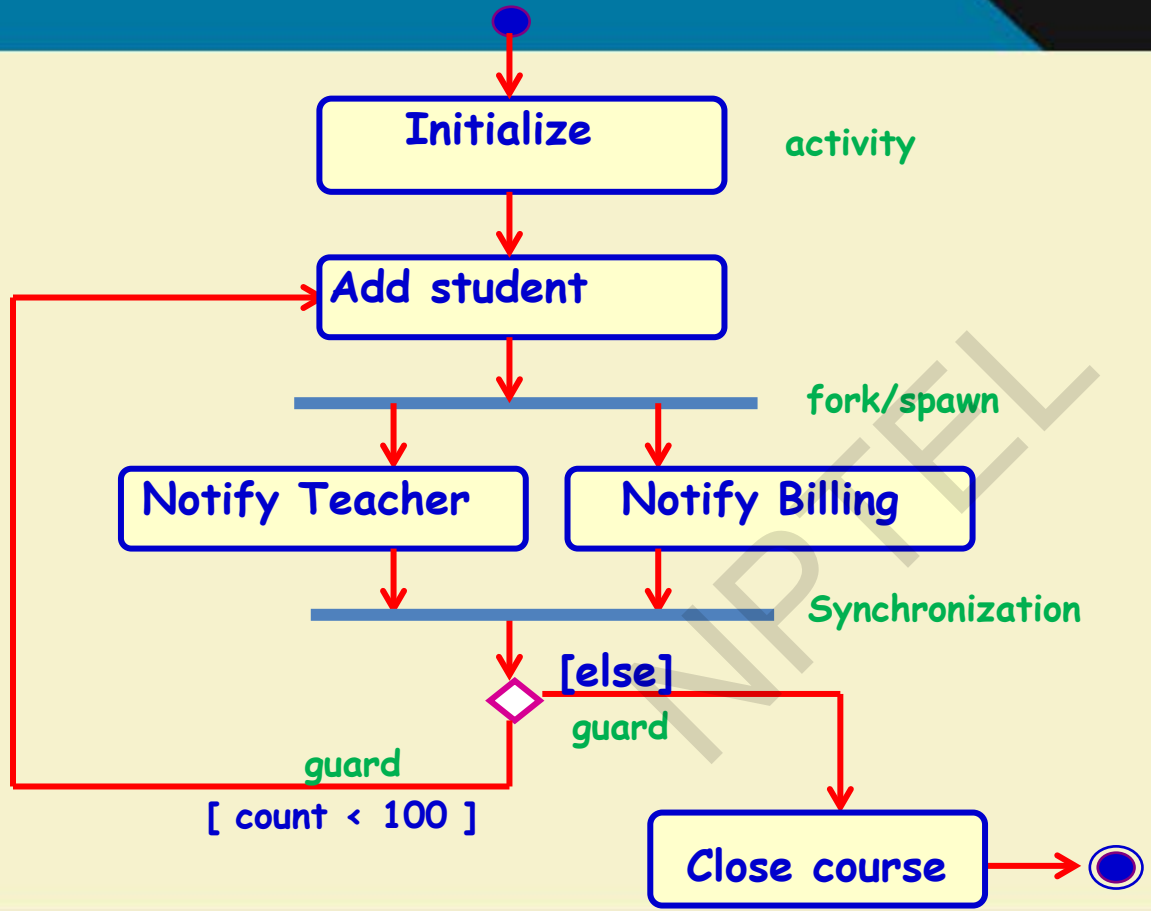


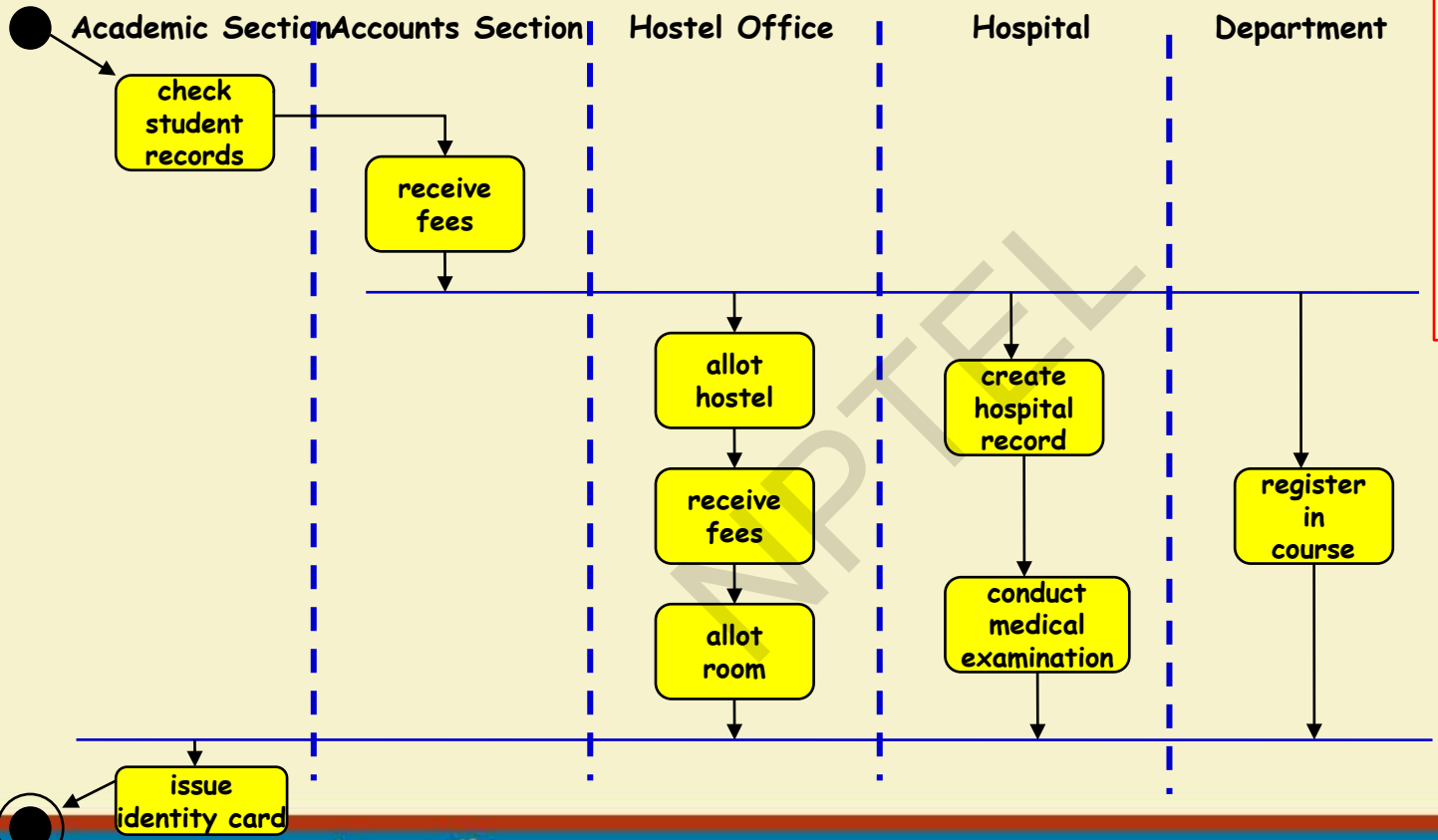


Activity Diagram vs Flow Chart

- Can represent parallel activity and synchronization aspects
- **Swim lanes**:
  - **Allows grouping activities based on who is performing them**
- **Example:** Academic department vs. Hostel

Activity Diagram Example: Student Registration





Another Example Activity Diagram: student admission process at IIT

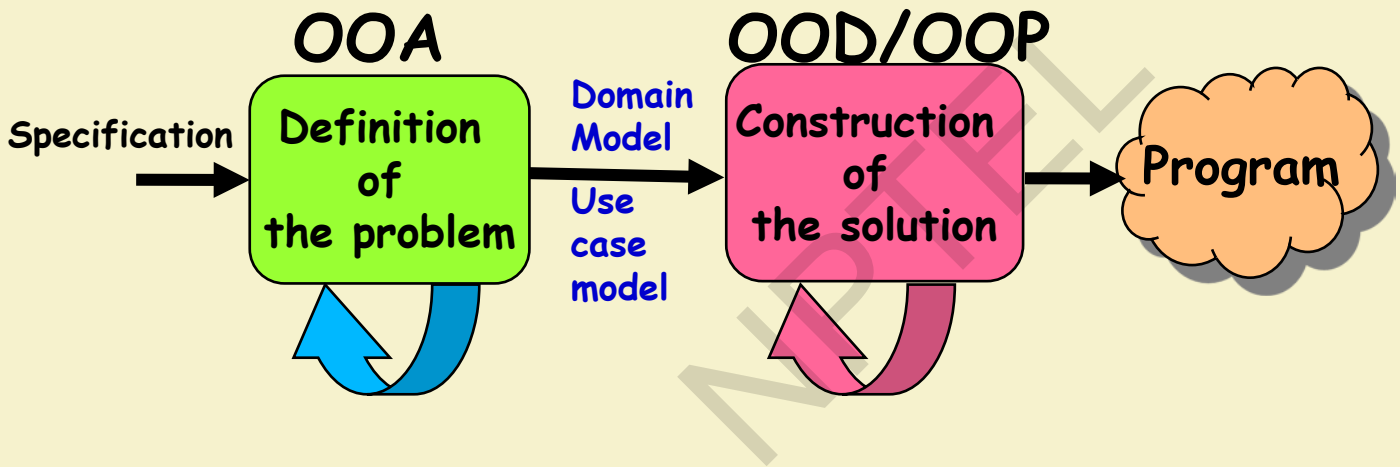
OOAD

A Design Process

- We discuss a design process based to a large extent on Larman's approach:
  - Also synthesizes features from various other methodologies.
- From requirements specification, an initial model is developed (OOA):
  - **Analysis model is iteratively refined into a design model**
- Design model is implemented using an OO language.

Iterative and Incremental

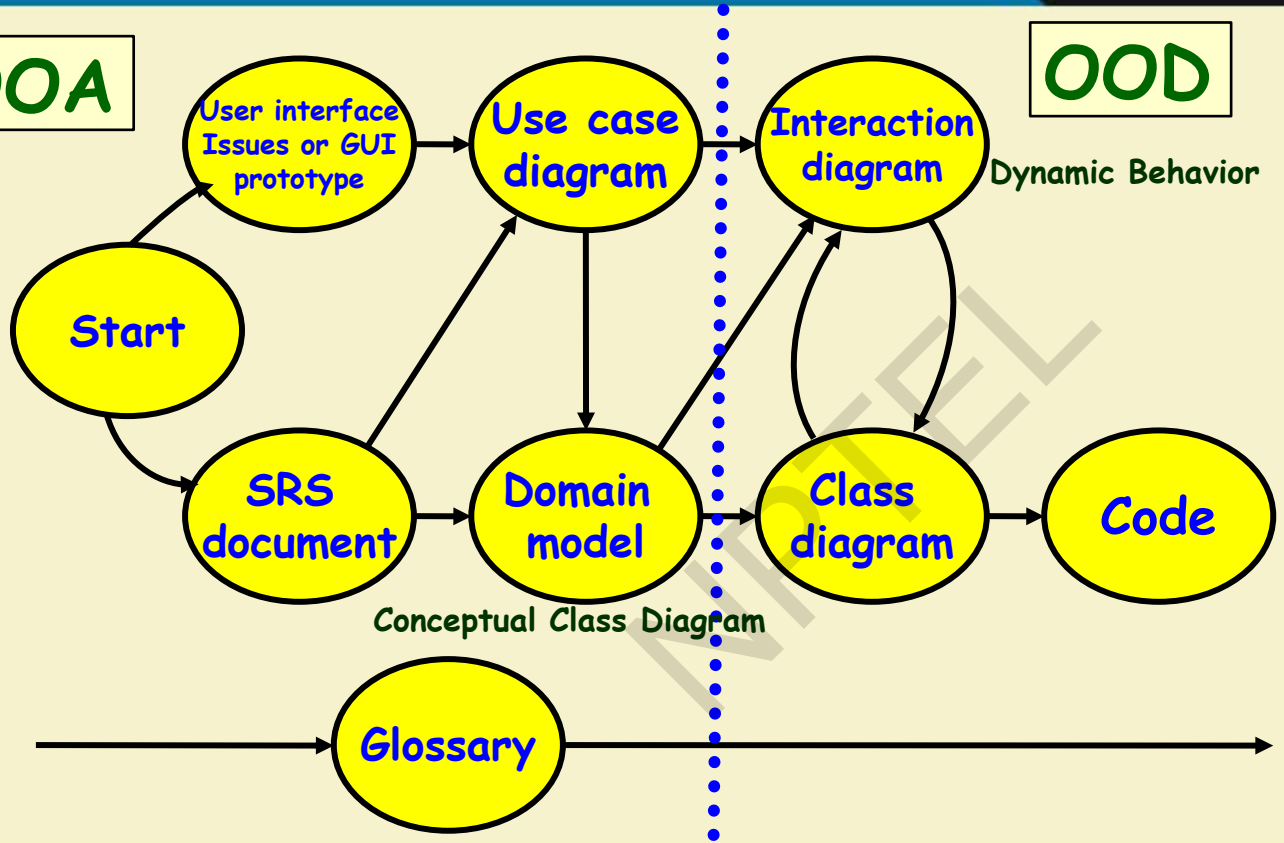
OOAD



OOA versus OOD?

- **Analysis:**
  - **An elaboration of requirements.**
  - Independent of any specific implementation
- **Design:**
  - **A refinement of the analysis model.**
  - Takes implementation constraints into account

Design Process



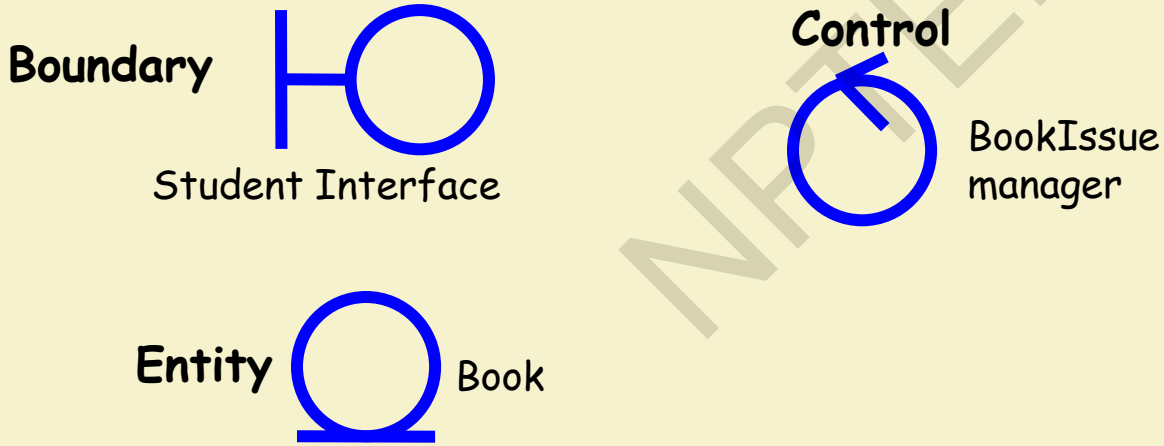
Domain Model

- Represent concepts or objects appearing in the problem domain.
  - Also capture object relationships.
- Three types of objects are identified:
  - **Boundary objects**
  - **Entity objects**
  - **Controller objects**

Class Stereotypes

Three different stereotypes are used to represent classes :

<<boundary>>, <<control>>, <<entity>>.



Boundary Objects

- Handle interaction with actors:
  - **User interface objects**
- Often implemented as screens, menus, forms, dialogs etc.
- Do not perform processing:
  - But may validate input, format output, etc.