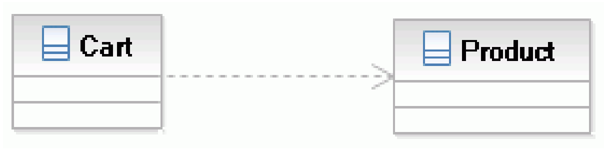
Slide 15: Class Diagram

Made by Md. Mehedi Hasan Rafy

Dependency Relationship

ln UML a dependency relationship is a relationship in which one element (client) uses or depends on another element (supplier). A change to the supplier might require a change to the client. It is represented by a ***dotted arrow*** drawn from dependent class to the independent class.



In this example, Product is the supplier and Cart is the client because Cart class depends on the Product class on any e-commerce application.

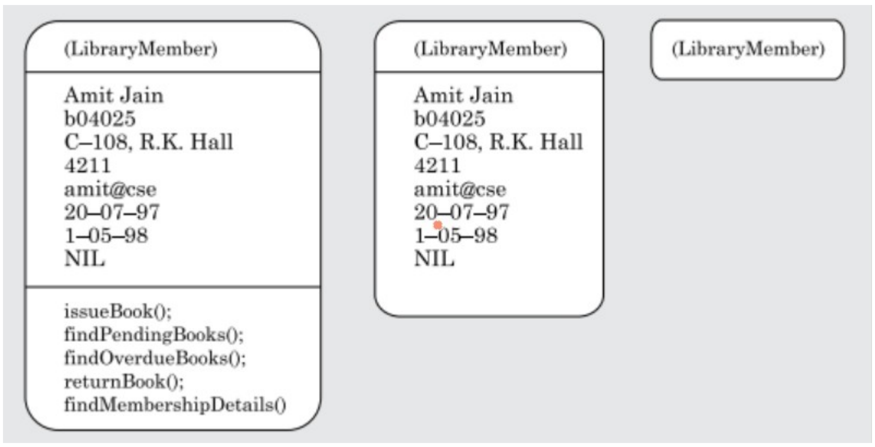
Constraints

A constraints describes either a condition that needs to be satisfied or an integrity rule for some entity.

- It describes aspects such as permissible set of values of an attribute, specification of the preconditions and post conditions for operations, and definition of certain ordering items.
- Constraints must be ***enclosed within braces***.

For example, Books in a library are maintained sorted on ISBN number, we can annotate the book class with the constraint {sorted}.

Object Diagram

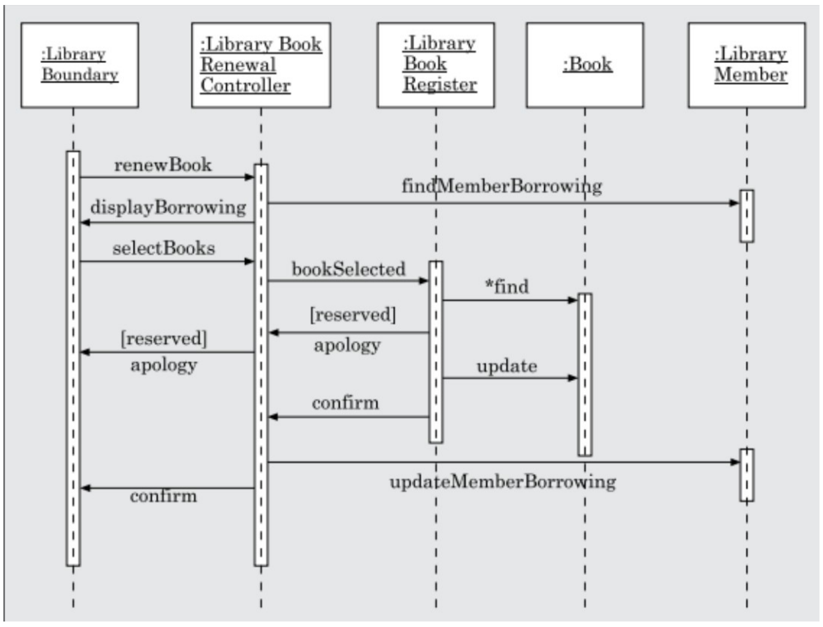


- Shows a snapshot of objects at a point in time.
- Focuses on instances of classes so it's also called an *instance diagram*.
- Objects are drawn as ***rounded rectangles.***
- Can show just the name of the object or include more details.

Interaction Diagram

- Shows how objects interact via message passing to realize a use case.
- Complex use cases may need multiple diagrams.
- Two types:
  - ***Sequence diagram*** – emphasizes *time/order* of messages.
  - ***Collaboration diagram*** – emphasizes *structural links* and message flow.
- They're equivalent: one can be derived from the other.

Sequence Diagram



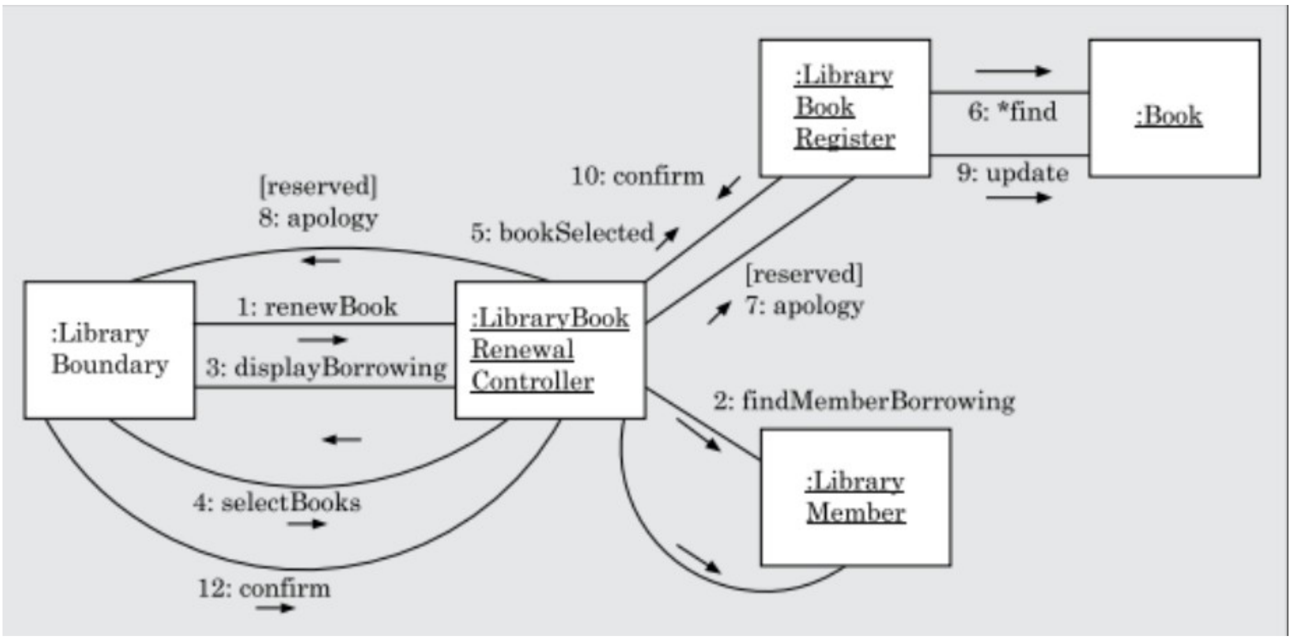
- A sequence diagram is a ***2D chart, read top → bottom*** showing interactions among objects.
- Objects appear as ***boxes*** at the top, with a ***dashed vertical line***.
- ***<u>Object</u> : <u>Class</u>*** notation, underlined (no name = arbitrary instance).
- Object at top existed before use case execution; objects can also be created during execution.
- The vertical dashed line attached to an object is called its lifeline. Lifeline ends (crossed) when object is destroyed.
- If a object receives message an ***activation symbol*** (rectangle on lifeline) to indicate that the object is active.
- Messages are arrows between lifelines, labeled with names, optionally some controlled information (in order, top → bottom).

Collaboration Diagram

- Shows both structural (objects + links ans association between classes) and behavioral (messages exchanged between collaborators) aspects.
- Objects = collaborators, linked with solid lines (associations).

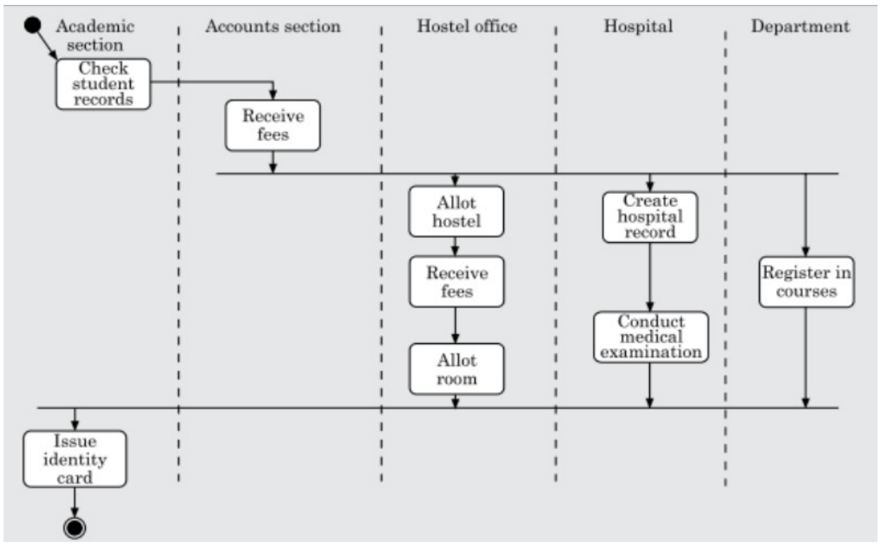
- Messages are labeled arrows on links between objects, numbered to show order (from sequence diagram).
- Highlights which classes are associated, so structural info is clearer than in a sequence diagram.

For example, the use-case sequence of the 'renew-book' is shown in the figure.



Activity Diagram

- Represents activities (processing steps) and their sequence of execution.
- Activity = state with internal action + outgoing transitions.
- If multiple transitions → conditions specify which path executes.
- Similar to flowcharts, but also supports description of parallel activities + synchronization.
- Uses swim lanes to group activities by the responsible component/actor.
- Makes it clear who performs which activity in a use case.



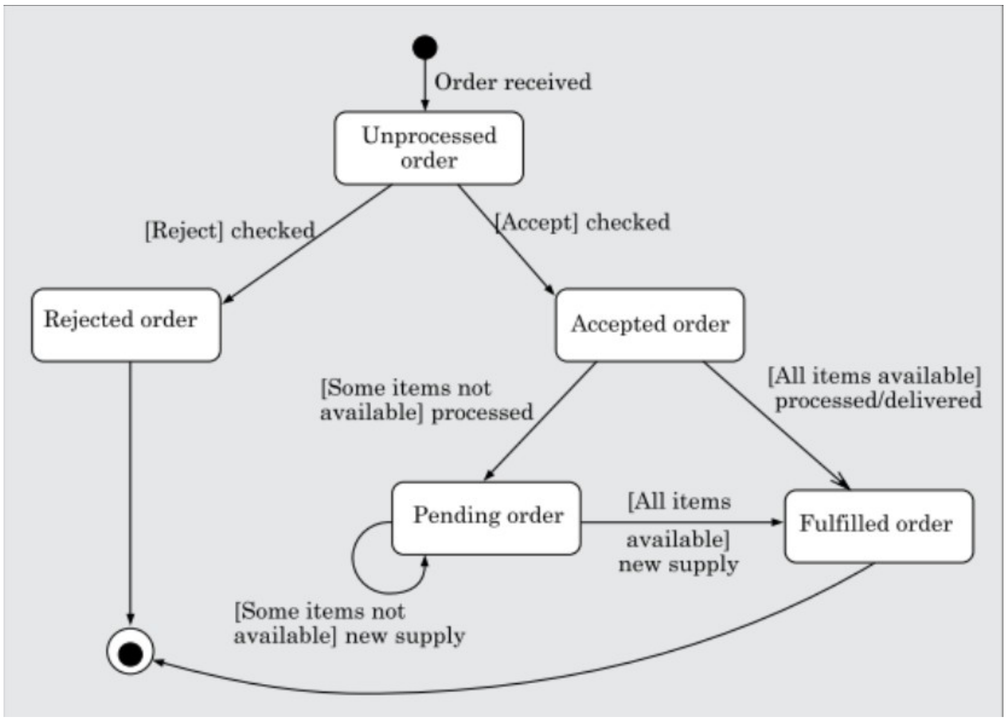
For example, the swim lane corresponding to the academic section, the activities that are carried out by the academic section (check student record and issue identity card) and specific situations in which these are carried out are shown in the figure above.

State Chart Diagram

- Models how an object's state changes over its lifetime.
- Based on finite state machine (FSM) formalism which consists of a finite number of states corresponding to those that object being modeled can take.
- Captures how behavior changes across multiple use cases.

Basic elements of a state chart

- ***Initial state****:* filled circle.
- ***Final state****:* filled circle inside larger circle.
- ***State****:* rounded rectangle.
- ***Transition****:* arrow between states, labeled as `[guard]event/action`.
    - ***Guard*** = Boolean condition, must be true for transition.
    - ***Pseudo transition*** = no event/guard.


Order received
Unprocessed order
[Reject] checked
[Accept] checked
Rejected order
Accepted order
[Some items not available] processed
[All items available] processed/delivered
Pending order
[All items available] new supply
Fulfilled order
[Some items not available] new supply

For example, a Trade House Automation Software. Here, from the rejected order state there is an automatic transition to the end state but it does not have any guard or event annotated with it. Hence its a pseudo transition.