LECTURE-11

Unified Modeling Language (UML) and Modeling

- UML is a graphical notation useful for OO analysis and design
- Allows representing various aspects of the system
- Various notations are used to build different models for the system
- OOAD methodologies use UML to represent the models they create

Modeling

- Modeling is used in many disciplines – architecture, aircraft building, …
- A model is a simplification of reality
- “All models are wrong, some are useful”
- A good model includes those elements that have broad effect and omits minor elements
- A model of a system is not the system!

Why build models?

- Models help us visualize a system
- Help specify the system structure
- Gives us a template that can guide the construction
- Document the decisions taken and their rationale

Modeling

- Every complex system requires multiple models, representing different aspects
- These models are related but can be studied in isolation
- Eg. Architecture view, electrical view, plumbing view of a building
- Model can be structural, or behavioral

Views in an UML

- Different views
  - A use case view
  - A design view
  - A process view
  - Implementation view
  - Deployment view
- We will focus primarily on models for design
  - class diagram, interaction diagram, etc.

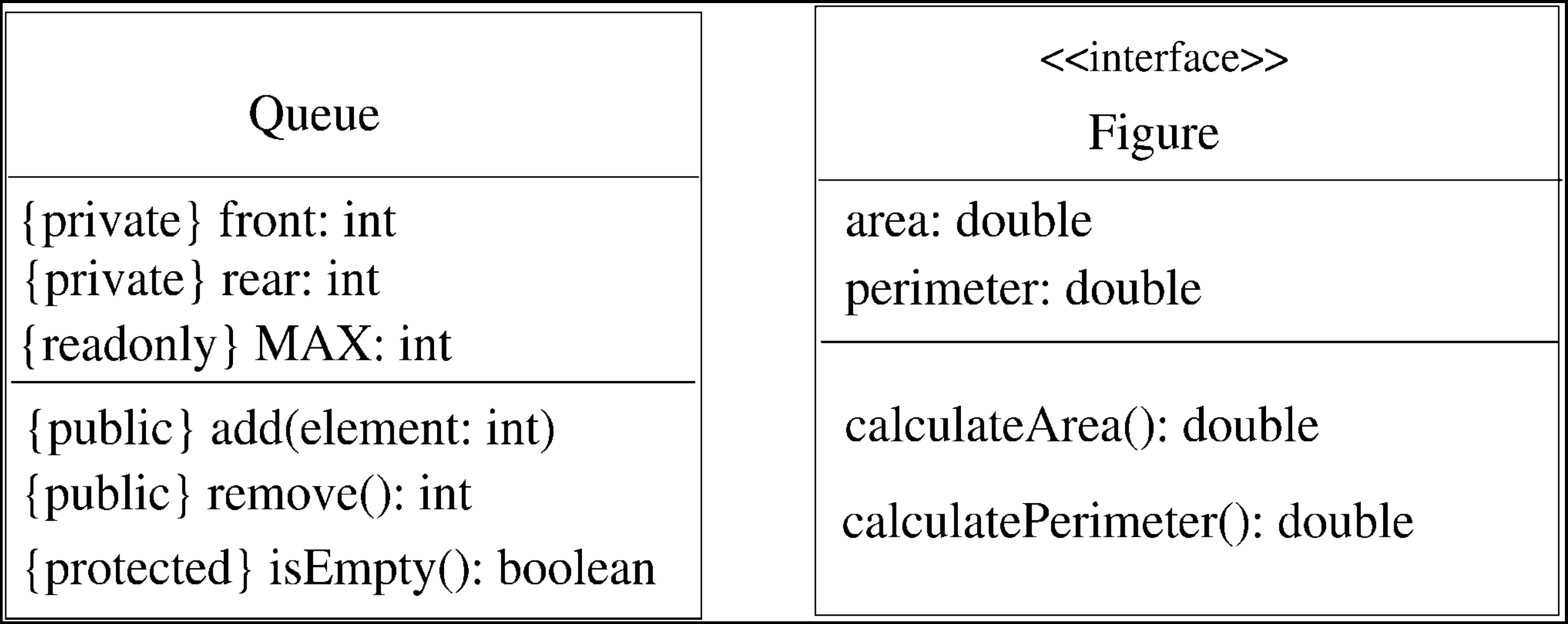
Class Diagrams

- Classes are the basic building blocks of an OO system as classes are the implementation units also
- Class diagram is the central piece in an OO design. It specifies
  - Classes in the system
  - Association between classes
  - Subtype, supertype relationship

Class Diagram…

- Class itself represented as a box with name, attributes, and methods
- There are conventions for naming
- If a class is an interface, this can be specified by <<interface>> stereotype
- Properties of attributes/methods can be specified by tags between { }

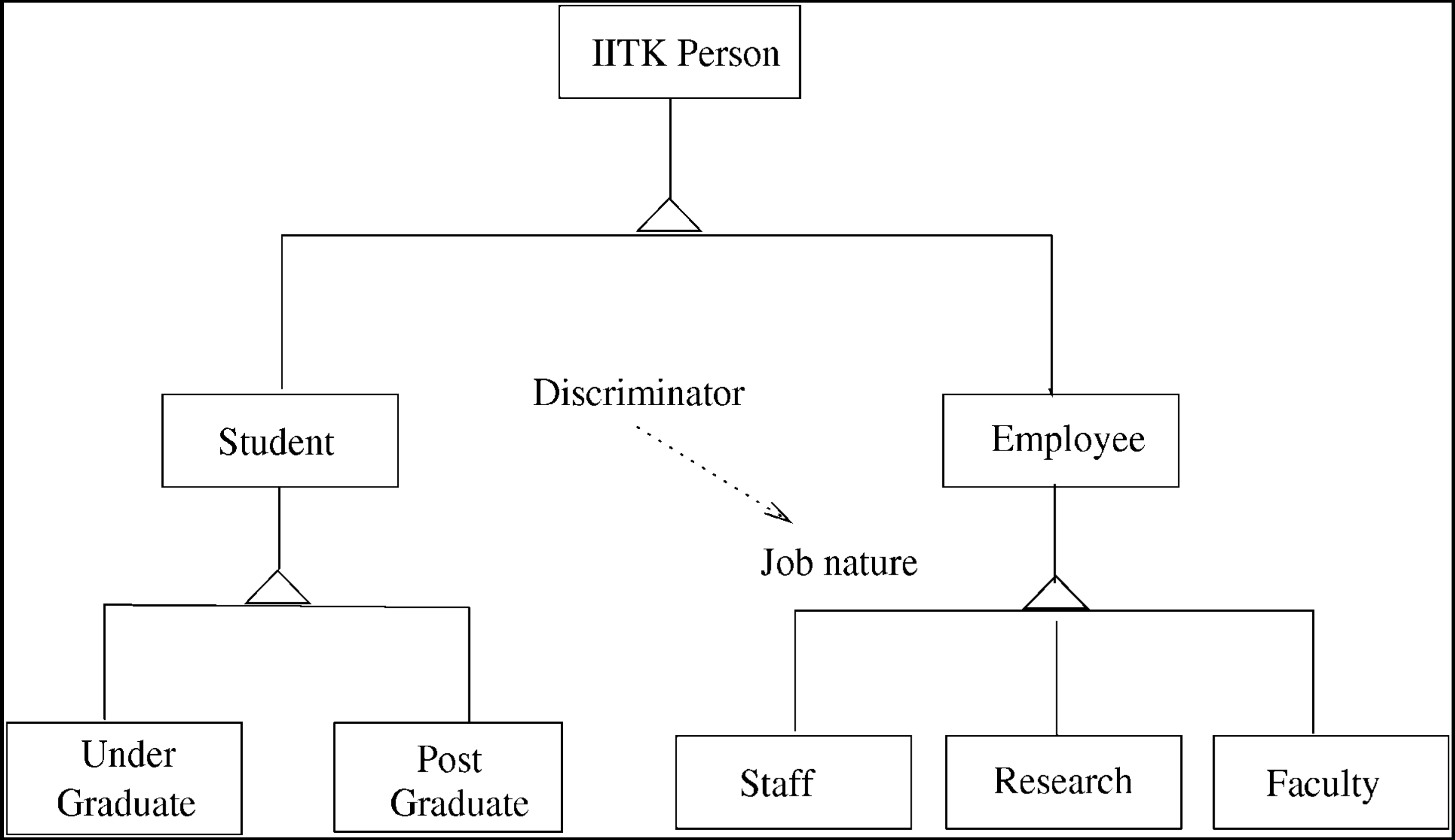
Class – example



Generalization-Specialization

- This relationship leads to class hierarchy
- Can be captured in a class diagram
  - Arrows coming from the subclass to the superclass with head touching super
  - Allows multiple subclasses
  - If specialization is done on the basis of some discriminator, arrow can be labeled

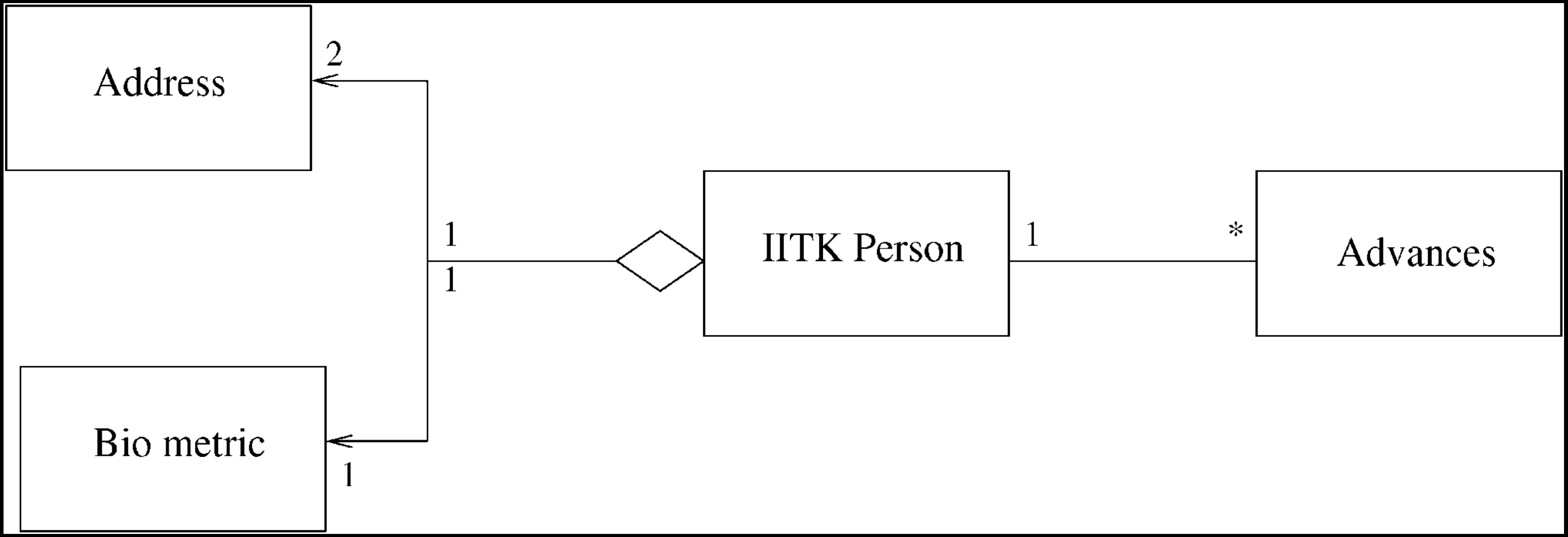
Example – class hierarchy



Association/aggregation

- Classes have other relationships
- Association: when objects of a class need services from other objects
  - Shown by a line joining classes
  - Multiplicity can be represented
- Aggregation: when an object is composed of other objects
  - Captures part-whole relationship
  - Shown with a diamond connecting classes

Example – association/aggregation



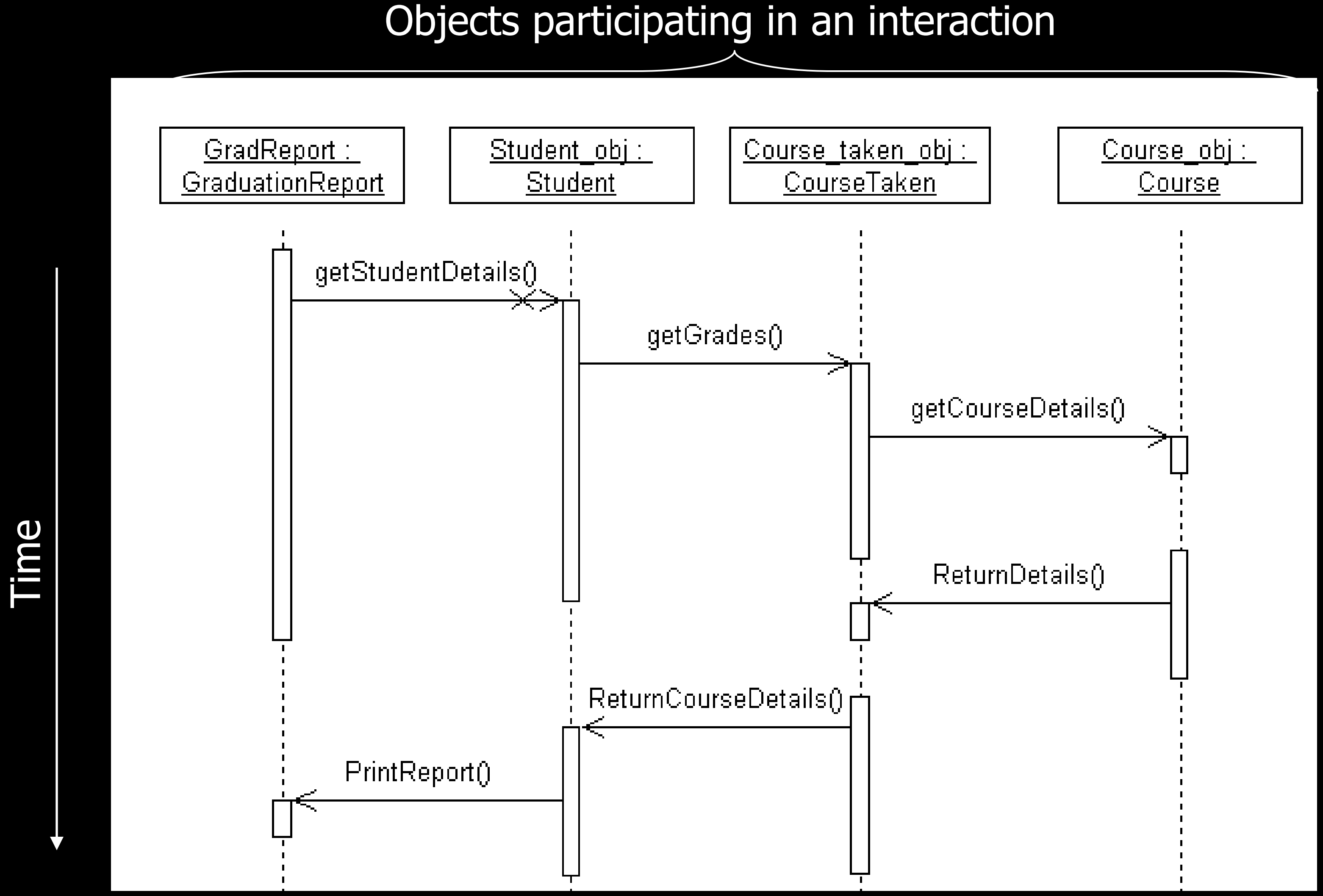
Interaction Diagrams

- Class diagrams represent static structure of the system (classes and their relationships)
- Do not model the behavior of system
- Behavioral view
  - shows how objects interact for performing actions (typically a use case)
- Interaction is between objects, not classes
- Interaction diagram in two styles
  - Collaboration diagram
  - Sequence diagram
- Two are equivalent in power

Sequence Diagram

- Objects participating in an interaction are shown at the top
- For each object a vertical bar represents its lifeline
- Message from an object to another, represented as a labeled arrow
- If message sent under some condition, it can be specified in bracket
- Time increases downwards, ordering of events is captured

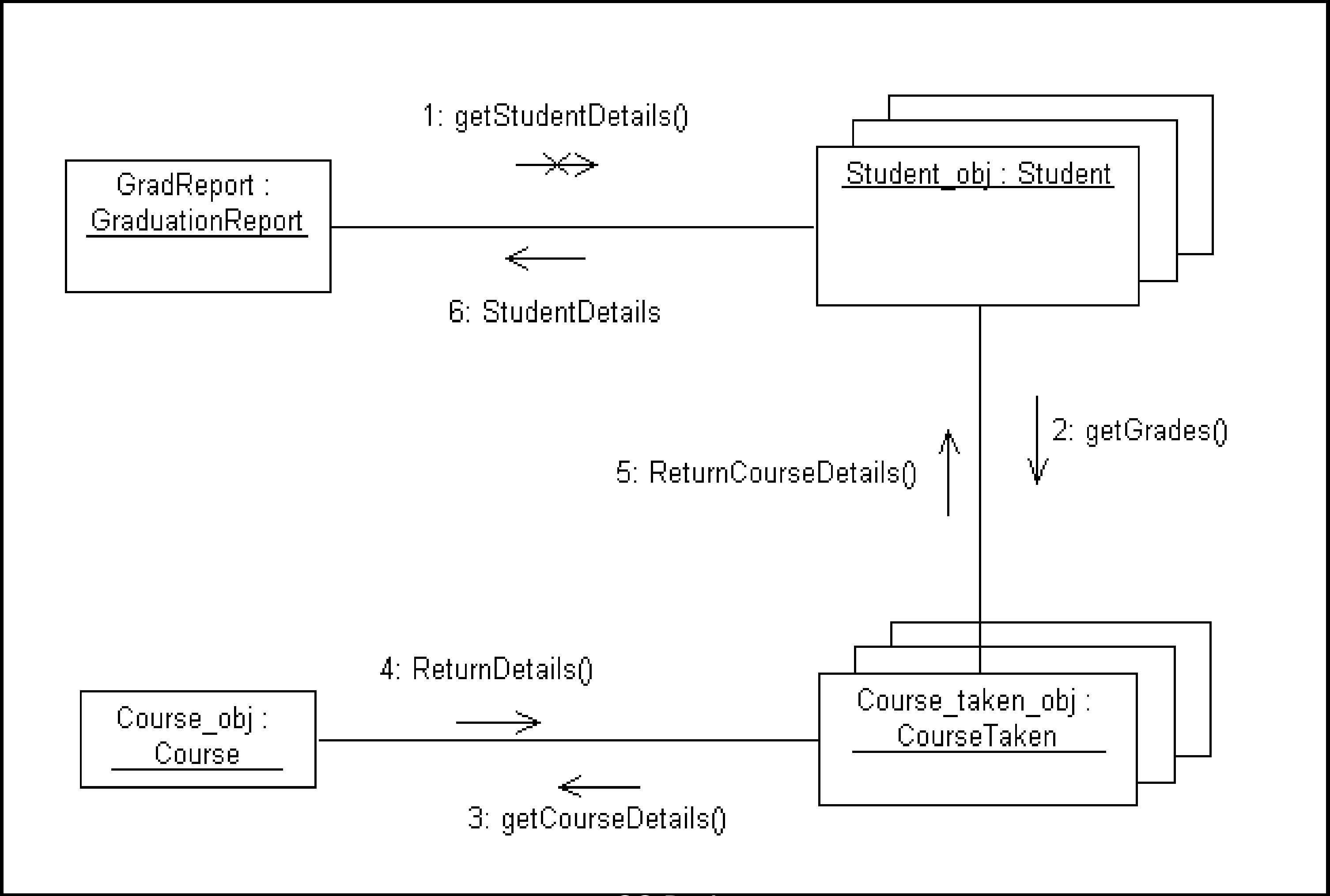
Example – sequence diagram



Collaboration diagram

- Also shows how objects interact
- Instead of timeline, this diagram looks more like a state diagram
- Ordering of messages captured by numbering them
- Is equivalent to sequence diagram in modeling power

Example – collaboration diag


1: getStudentDetails()
GradReport : GraduationReport
Student_obj : Student
6: StudentDetails
2: getGrades()
5: ReturnCourseDetails()
4: ReturnDetails()
Course_obj : Course
Course_taken_obj : CourseTaken
3: getCourseDetails()