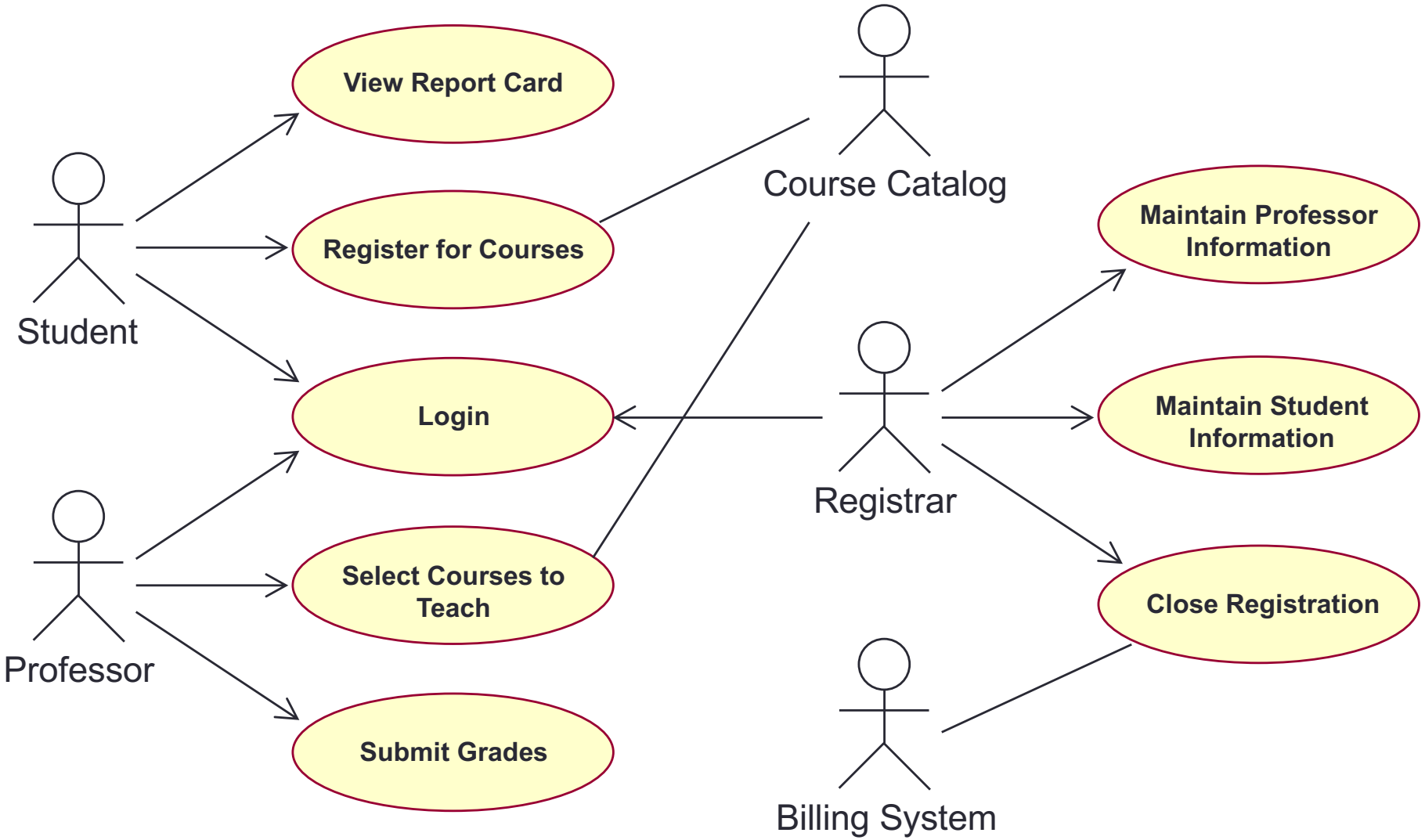
INTRODUCTION TO SOFTWARE ENGINEERING

6. UNIFIED MODELING LANGUAGE

Outline

- **Use-case diagrams**
- Activity diagrams
- Sequence diagrams
- Class diagrams

How Would You Read This Diagram?



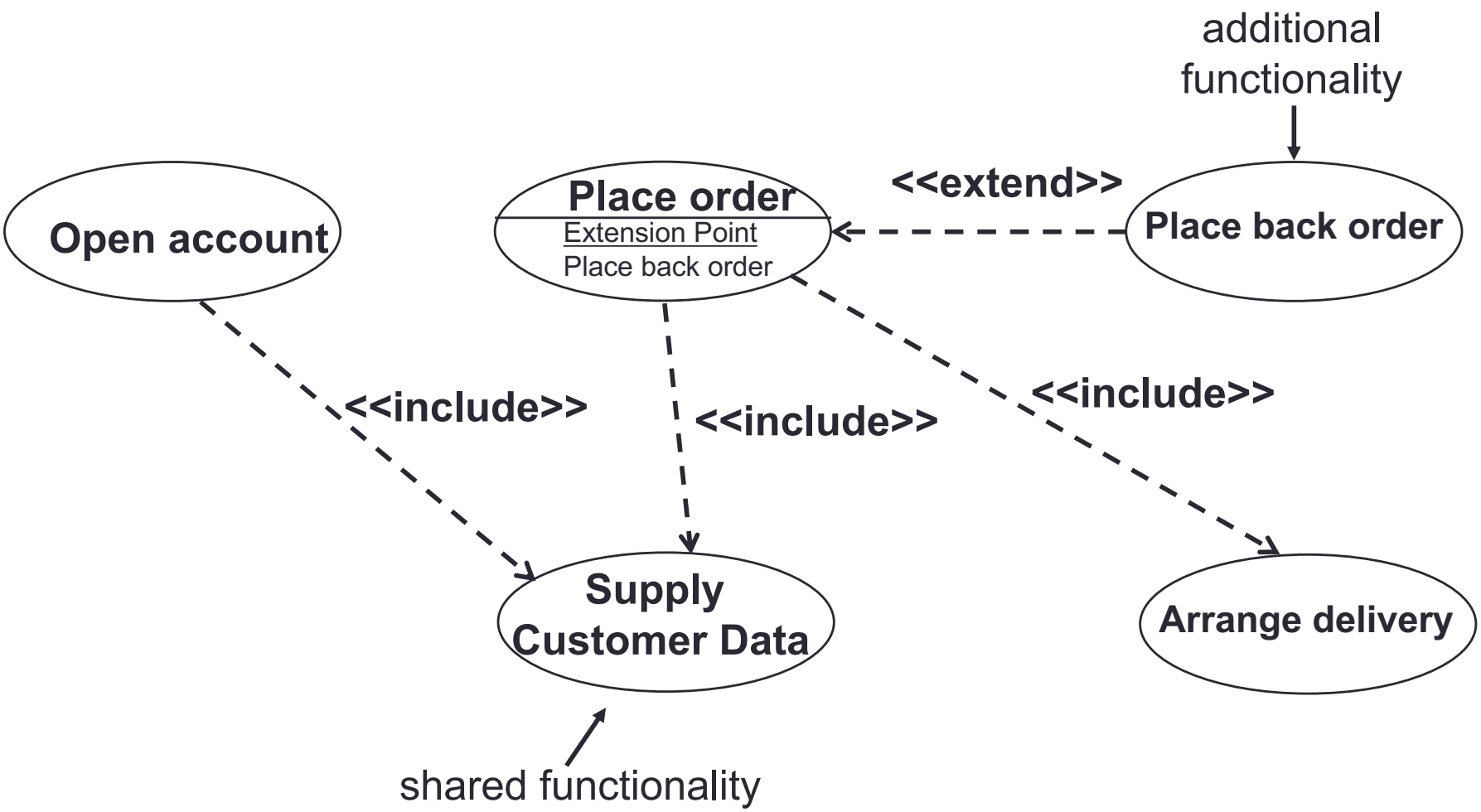
Relationships in use cases

- Between actor and use case
  - Actor uses
- Generalisation of actors
  - Types of users
- Use case stereotypes
  - <<extend>>
    - Optional
  - <<include>>
    - Mandatory
- Stereotype is a UML extension mechanism to indicate a type of behaviour

Use case variants: include and extend

- *include* relationship occurs when you have a chunk of behavior that is similar across more than one Use Case
  - use in two or more separate Use Cases to avoid repetition
  - a significant part of a use case
  - <<*include*>>
- *extend* relationship where you have one Use Case which adds functionality to another Use Case
  - any Use Case can have more than one extend
  - use when describing a variation on or in addition to normal behavior
  - OPTIONAL BEHAVIOUR
    - Otherwise part of use case or
    - <<include>>
  - <<extend>>

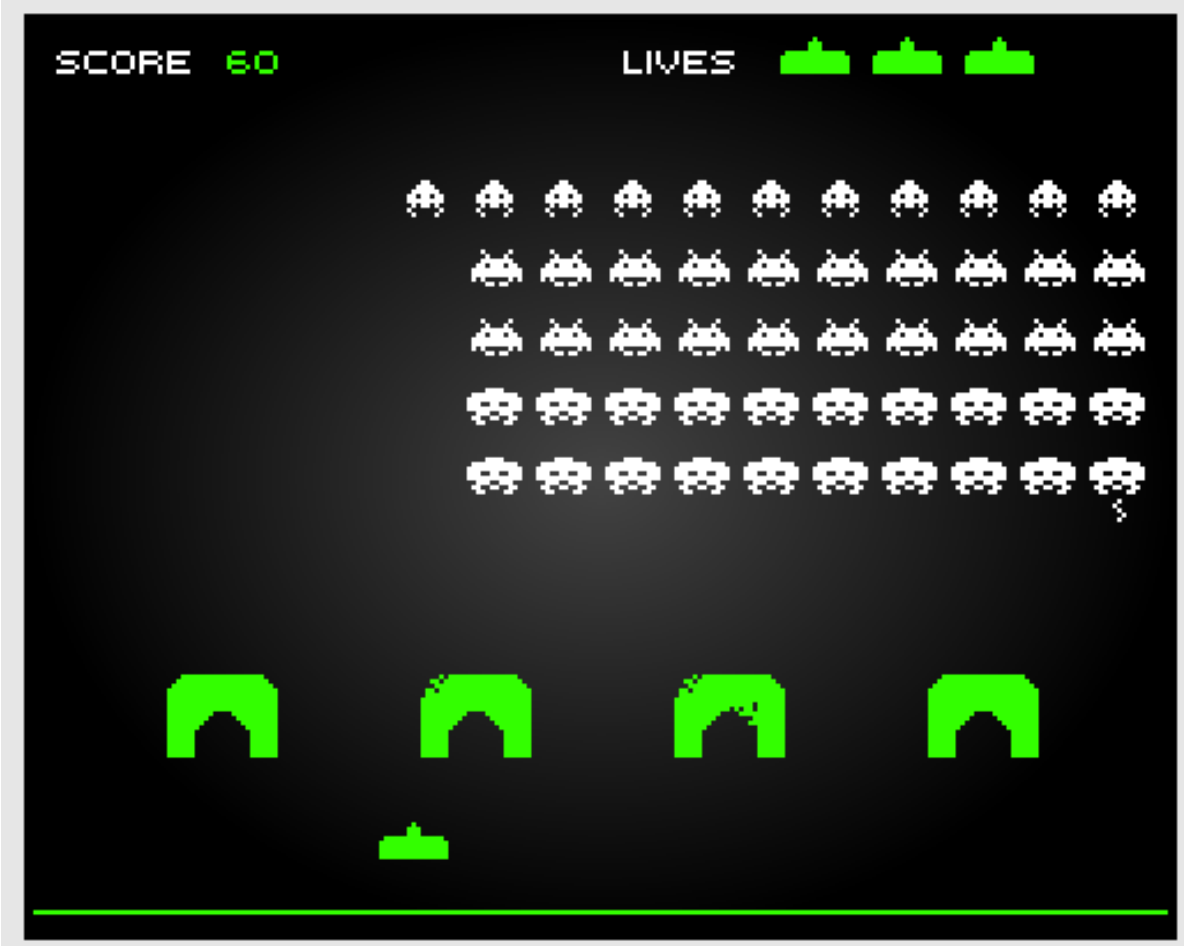
Example of Use Case Variants



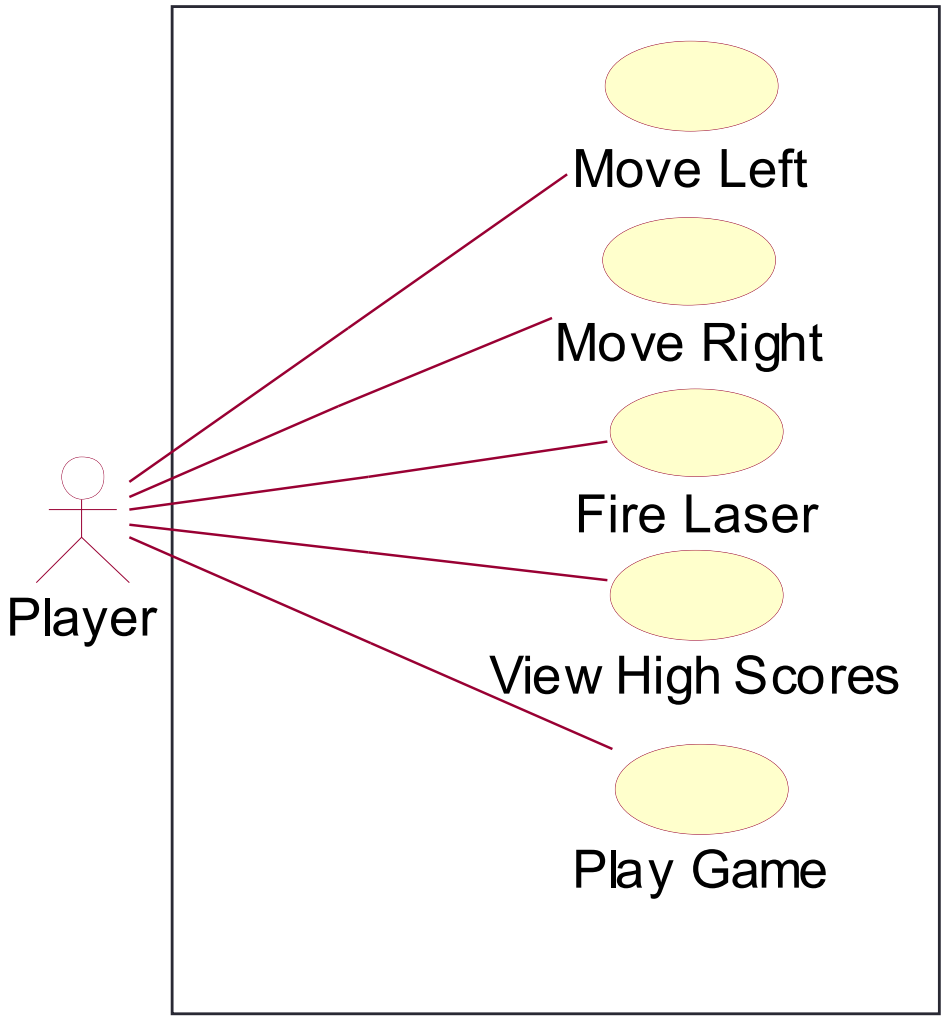
Exercise 1 – Use Case Diagram

- Draw a use case diagram for the space invader game below:

http://www.neave.com/games/invaders/



Exercise 1 Solution



Exercise 2

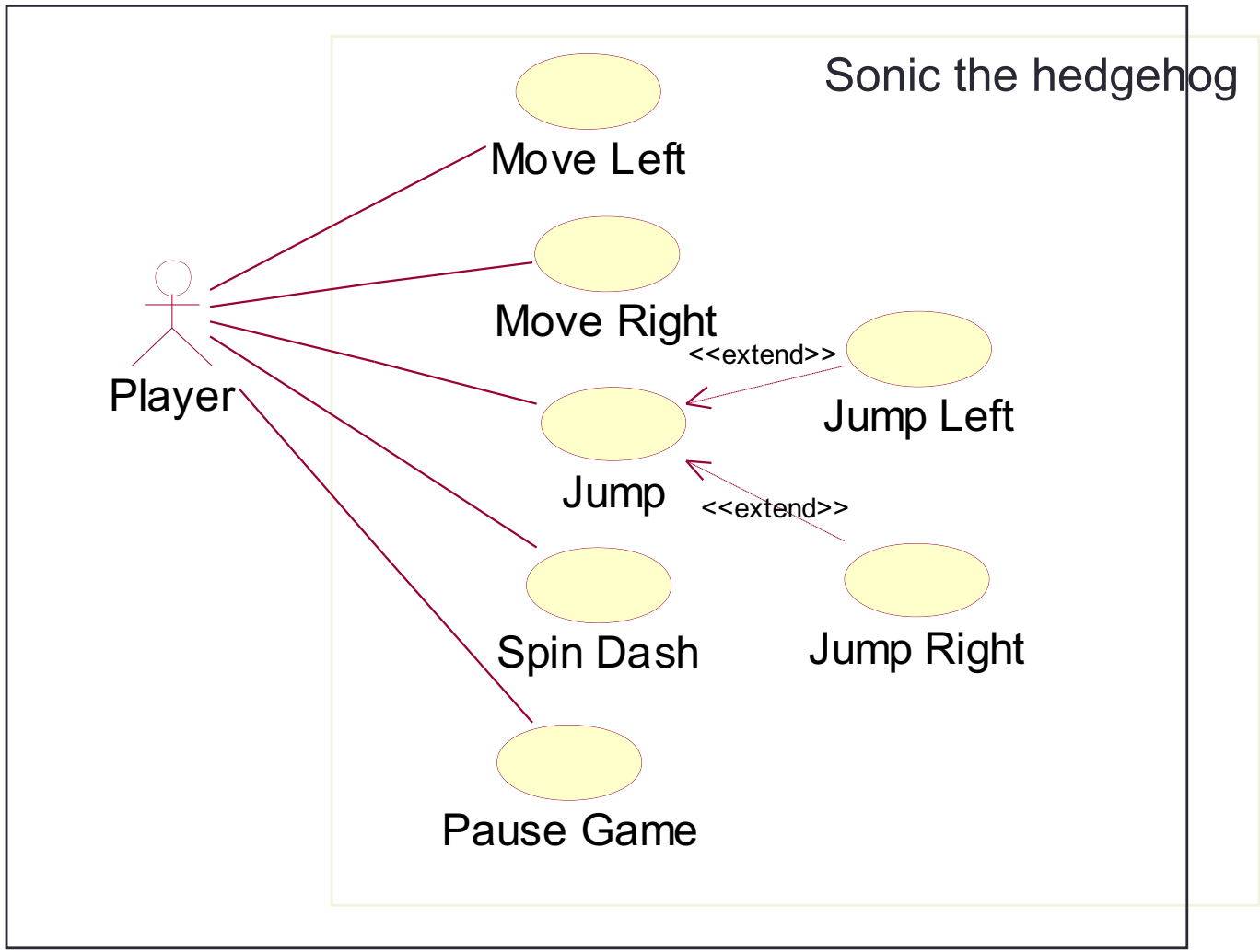
Consider Sonic the hedgehog.

1. What can sonic do?
2. What are the use cases?
3. Are there any relationships
4. Draw the use case diagram

1. Move left
2. Move right
3. Jump
4. Jump left
5. Jump right
6. Spin Dash
7. Pause

http://www.ebaumsworld.com/games/play/1108/

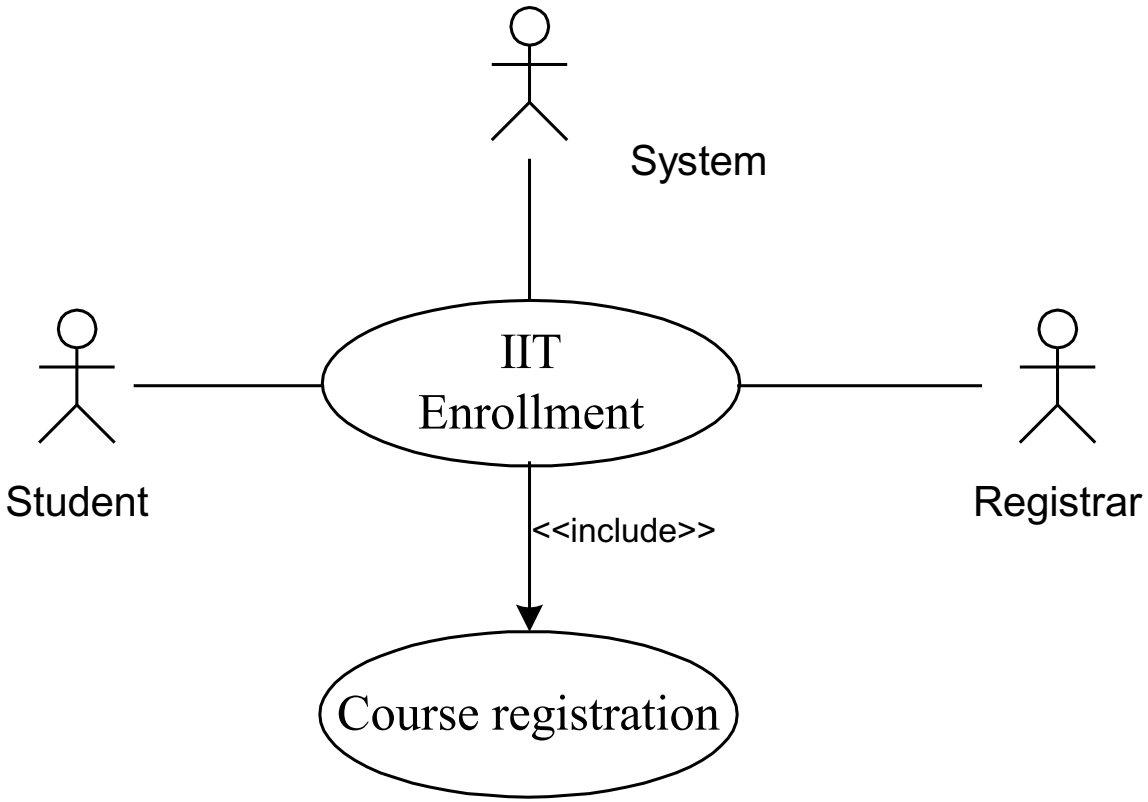
Exercise 2 – Possible Solution



Outline

- Use-case diagrams
- **Activity diagrams**
- Sequence diagrams
- Class diagrams

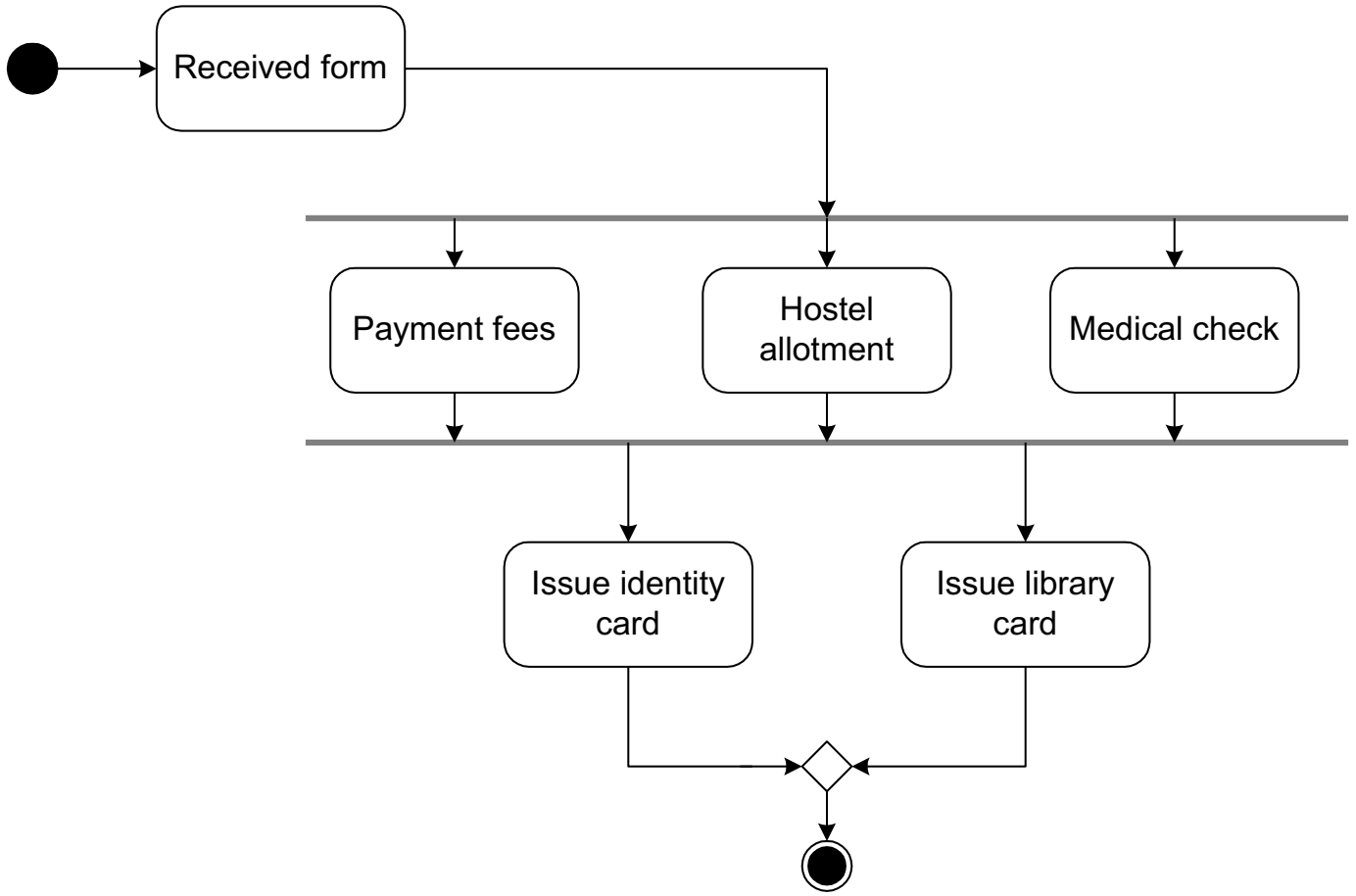
Student Enrollment



SEIT System

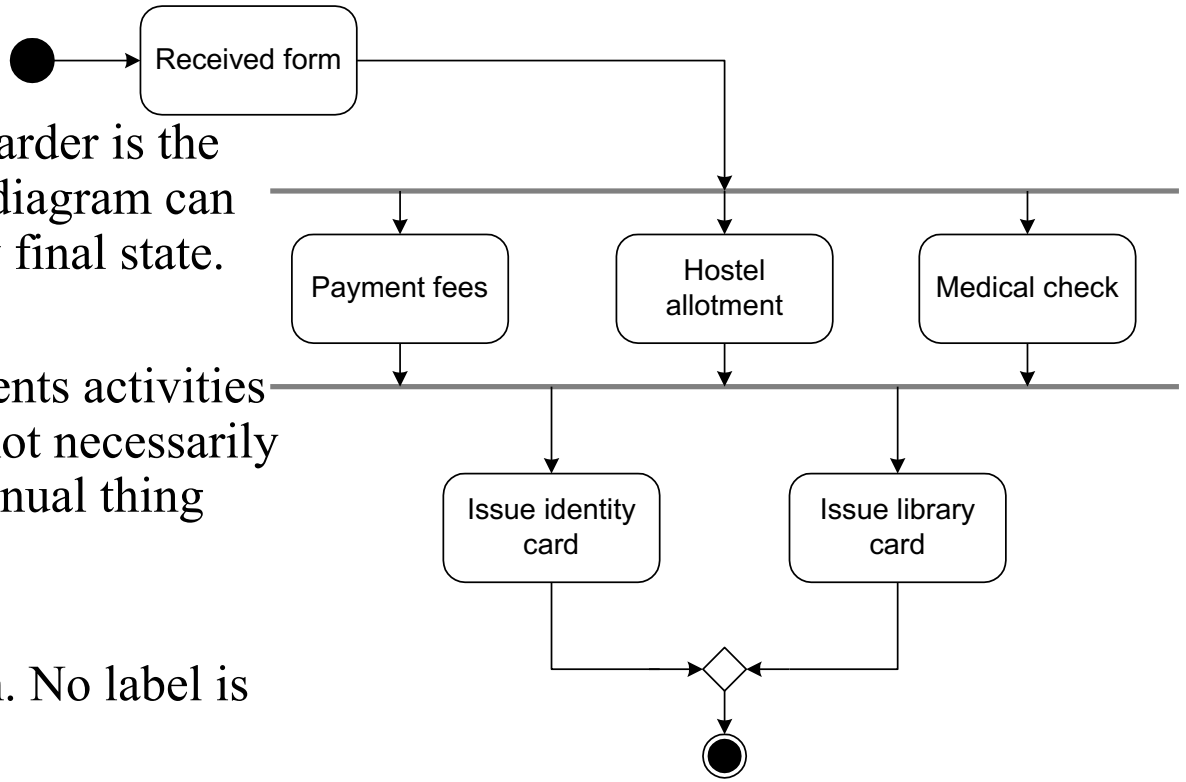
- Here different activities are:
  - Received enrollment form filled by the student
    - Registrar checks the form
    - Input data to the system
    - System authenticate the environment
  - Pay fees by the student
    - Registrar checks the amount to be remitted and prepare a bill
    - System acknowledge fee receipts and print receipt
  - Hostel allotment
    - Allot hostel
    - Receive hostel charge
    - Allot room
  - Medical check up
    - Create hostel record
    - Conduct medical bill
    - Enter record
  - Issue library card
  - Issue identity card

Activity Diagram for the Use Case



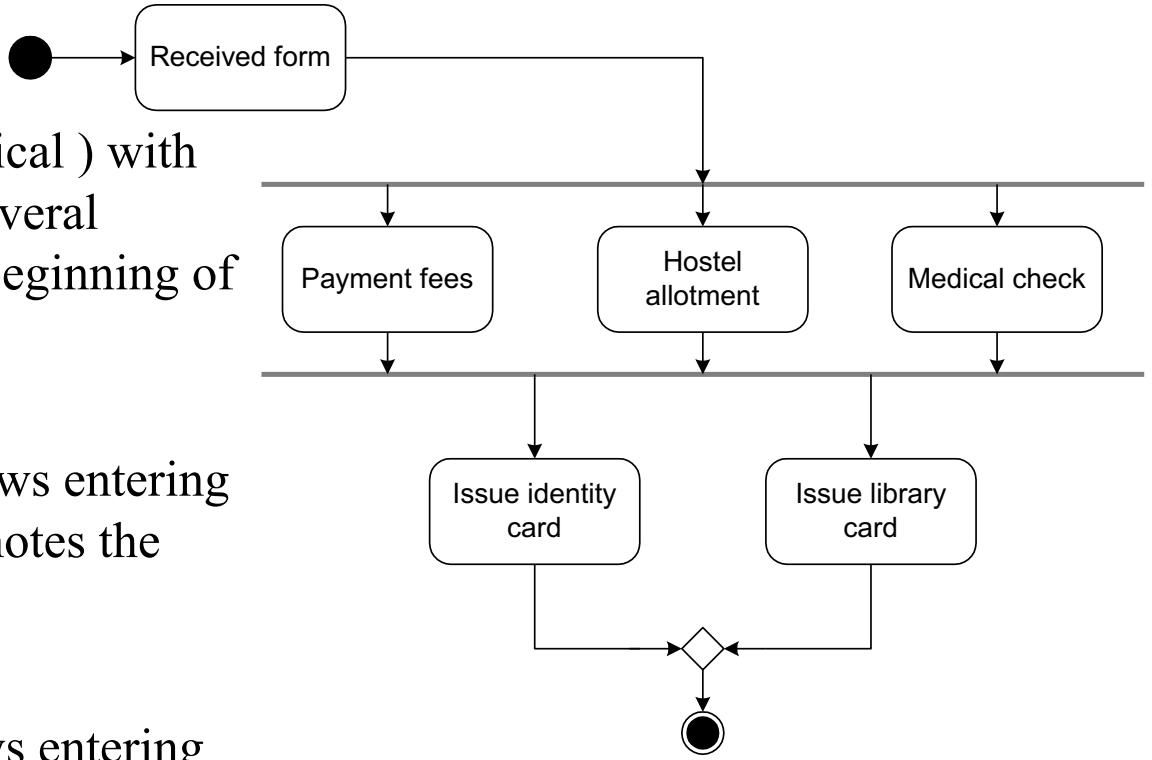
Basic Components in an Activity Diagram

- **Initial node**
  - The filled circle is the starting point of the diagram
- **Final node**
  - The filled circle with a boarder is the ending point. An activity diagram can have zero or more activity final state.
- **Activity**
  - The rounded circle represents activities that occur. An activity is not necessarily a program, it may be a manual thing also
- **Flow/ edge**
  - The arrows in the diagram. No label is necessary



Basic Components in an Activity Diagram

- **Fork**
  - A black bar (horizontal/vertical) with one flow going into it and several leaving it. This denotes the beginning of parallel activities
- **Join**
  - A block bar with several flows entering it and one leaving it. this denotes the end of parallel activities
- **Merge**
  - A diamond with several flows entering and one leaving. The implication is that all incoming flow to reach this point until processing continues

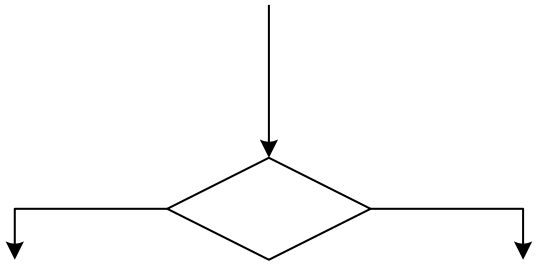


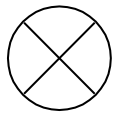
Basic Components in an Activity Diagram

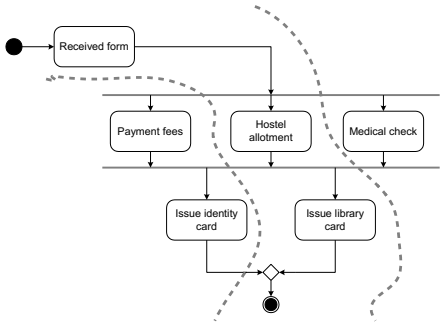
- Difference between Join and Merge
  - A join is different from a merge in that the join synchronizes two inflows and produces a single outflow. The outflow from a join cannot execute until all inflows have been received
  - A merge passes any control flows straight through it. If two or more inflows are received by a merge symbol, the action pointed to by its outflow is executed two or more times

Basic Components in an Activity Diagram

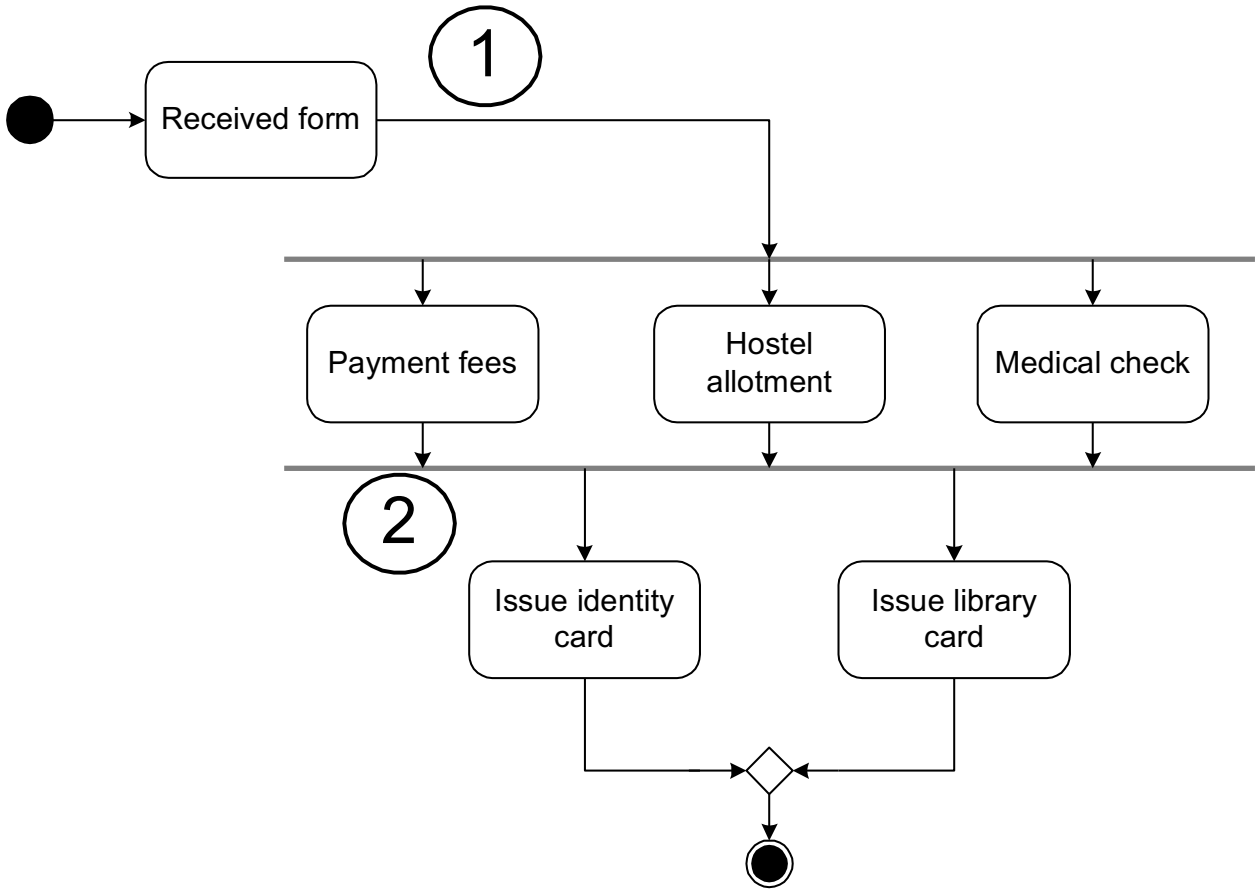
- **Decision**
  - A diamond with one flow entering and several leaving. The flow leaving includes conditions as yes/ no state
- **Flow final**
  - The circle with X though it. This indicates that Process stop at this point
- **Swim lane**
  - A partition in activity diagram by means of dashed line, called swim lane. This swim lane may be horizontal or vertical



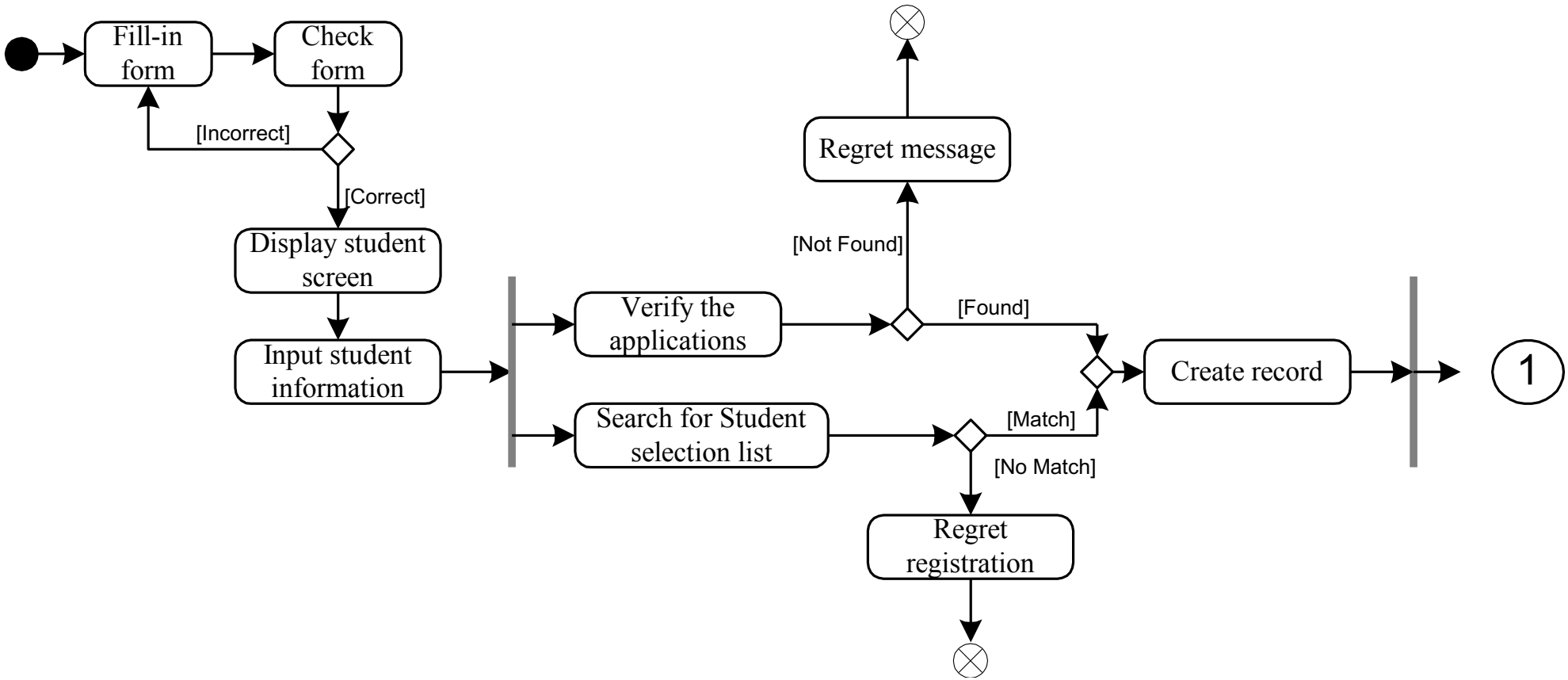




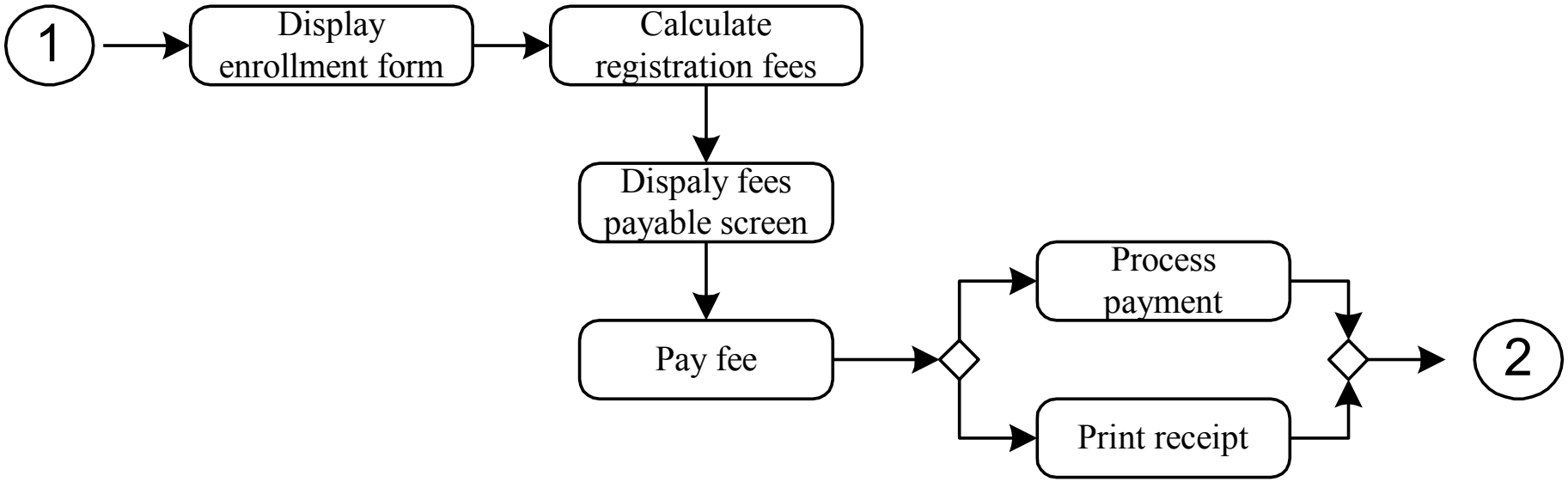
Detailed Activity Diagram of SEIIT



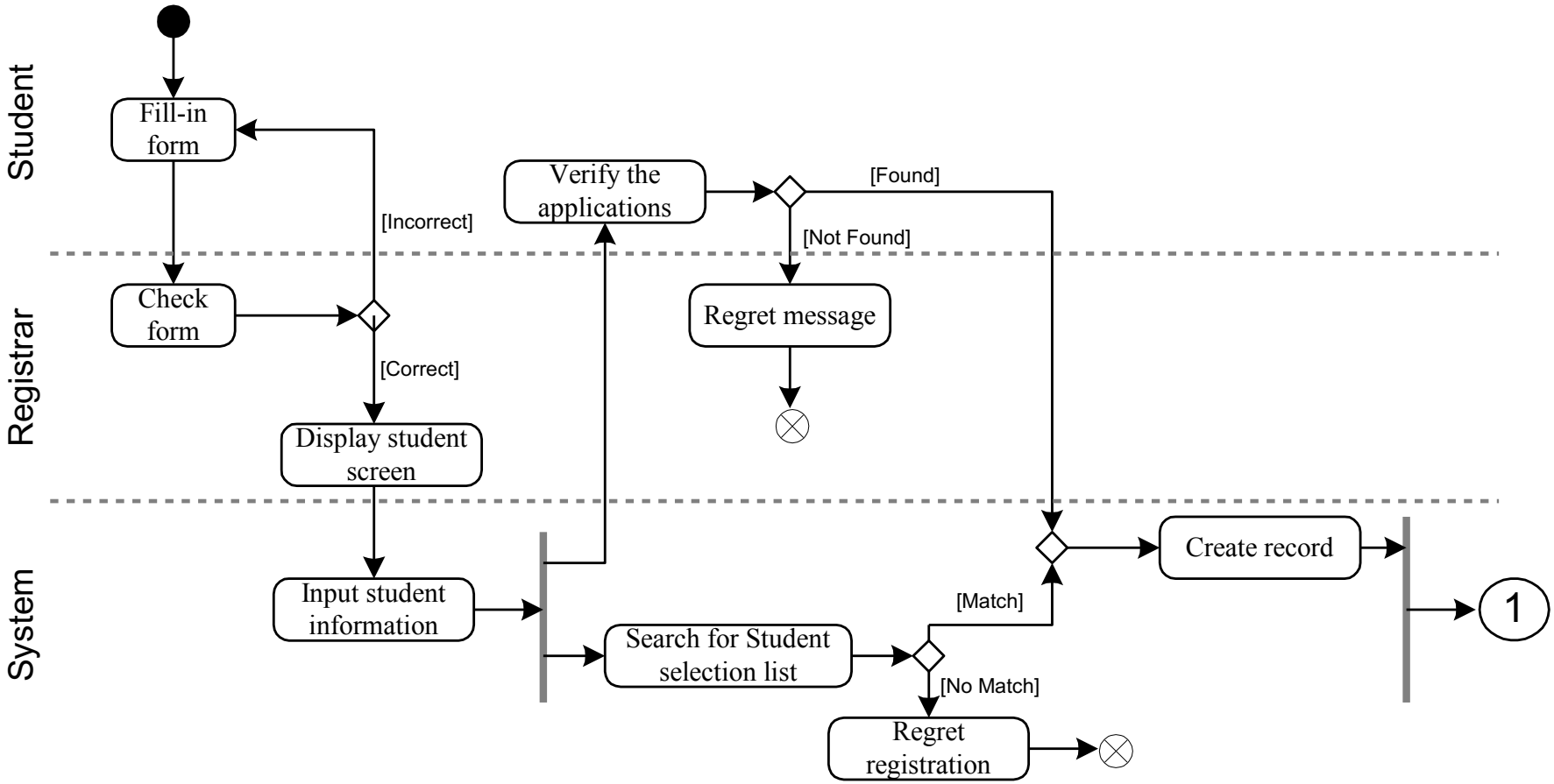
Detailed Activity Diagram of SEIIT



Detailed Activity Diagram of SEIIT

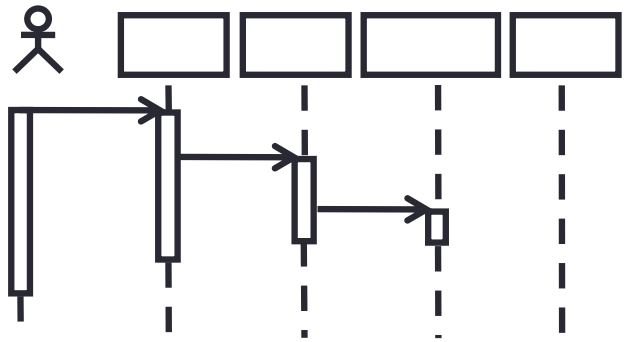


Activity Diagram of SEIIT with Swim Lane



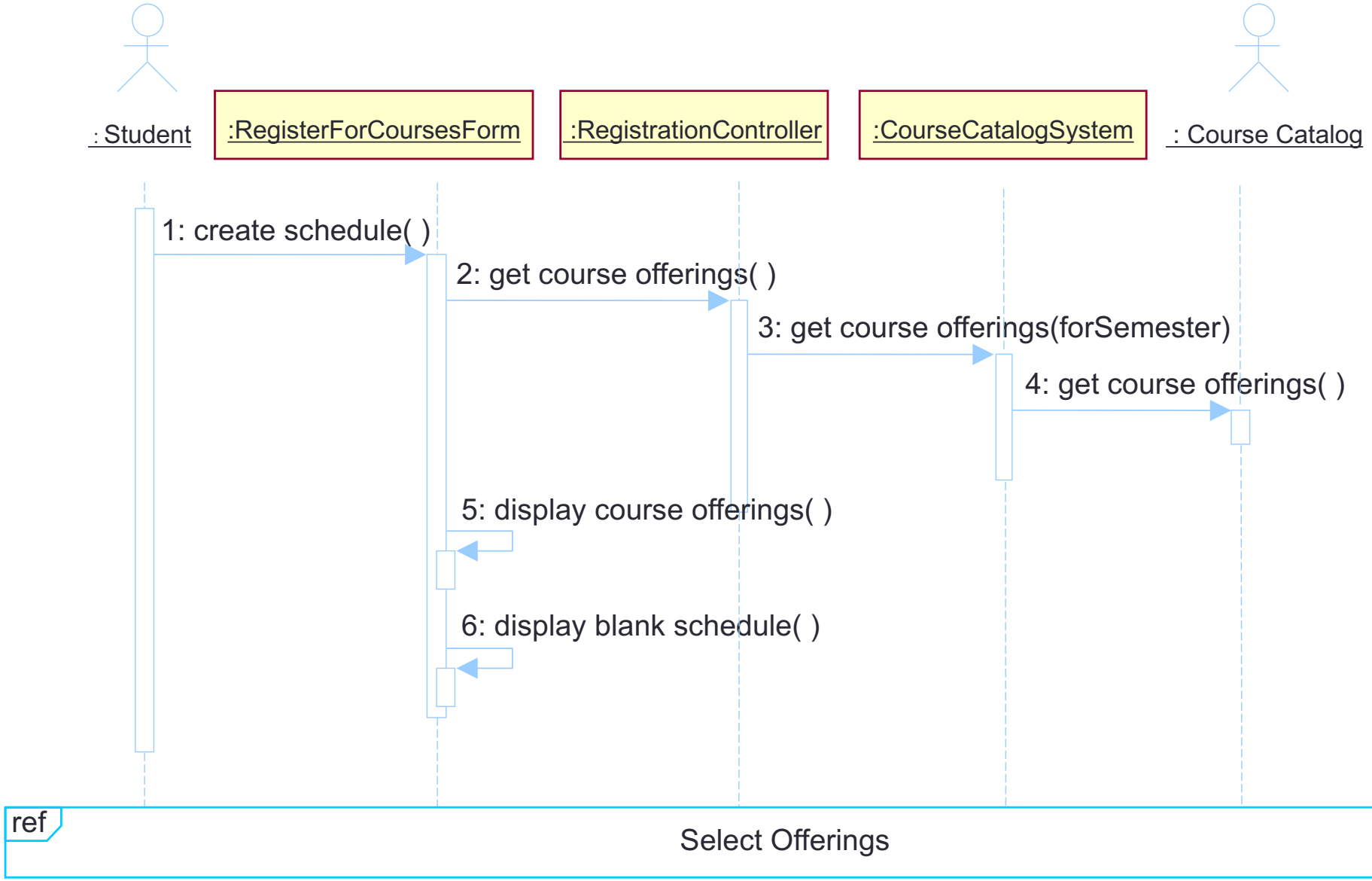
What Is a Sequence Diagram?

- A sequence diagram is an interaction diagram that emphasizes the time ordering of messages.
- The diagram shows:
  - The objects participating in the interaction.
  - The sequence of messages exchanged.

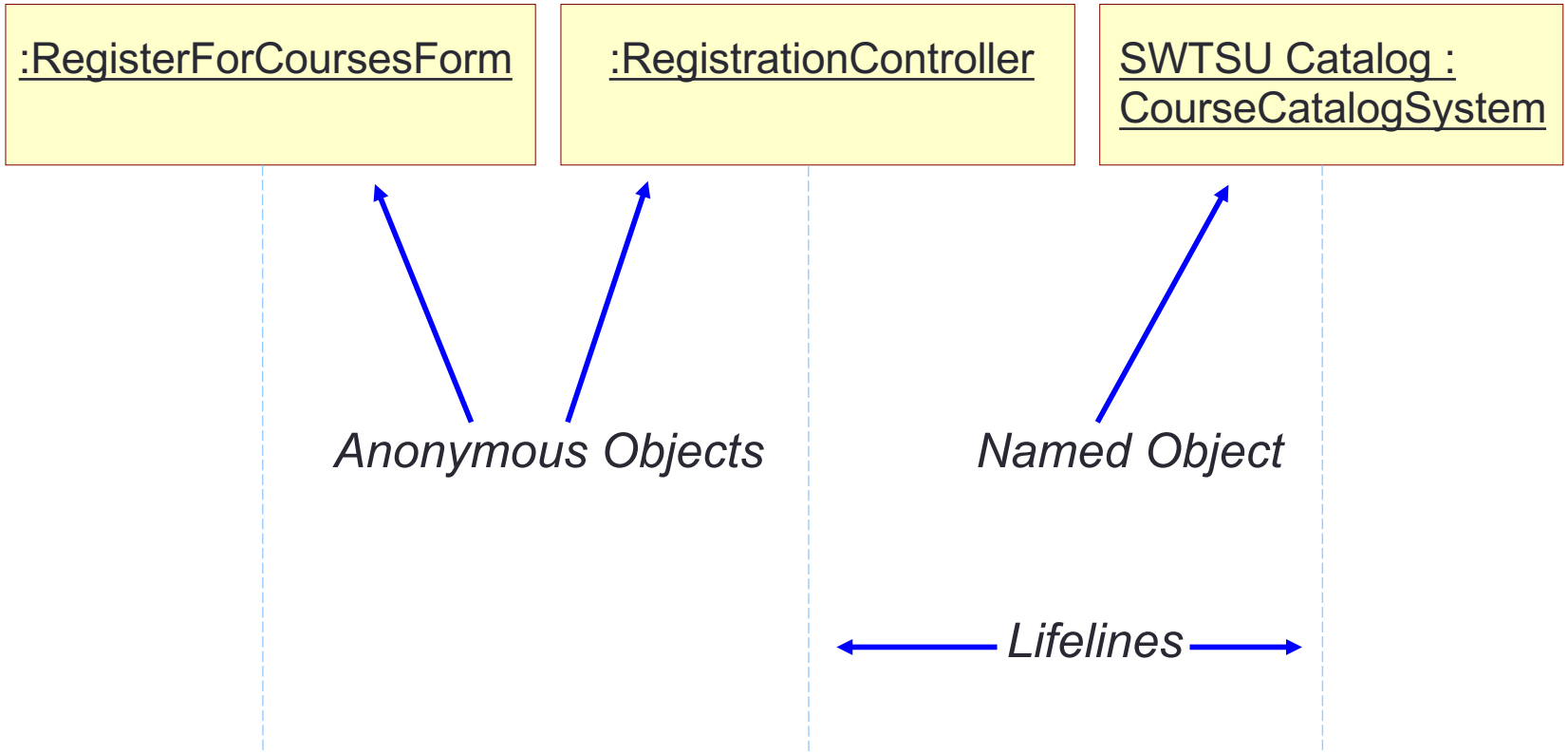


Sequence Diagram

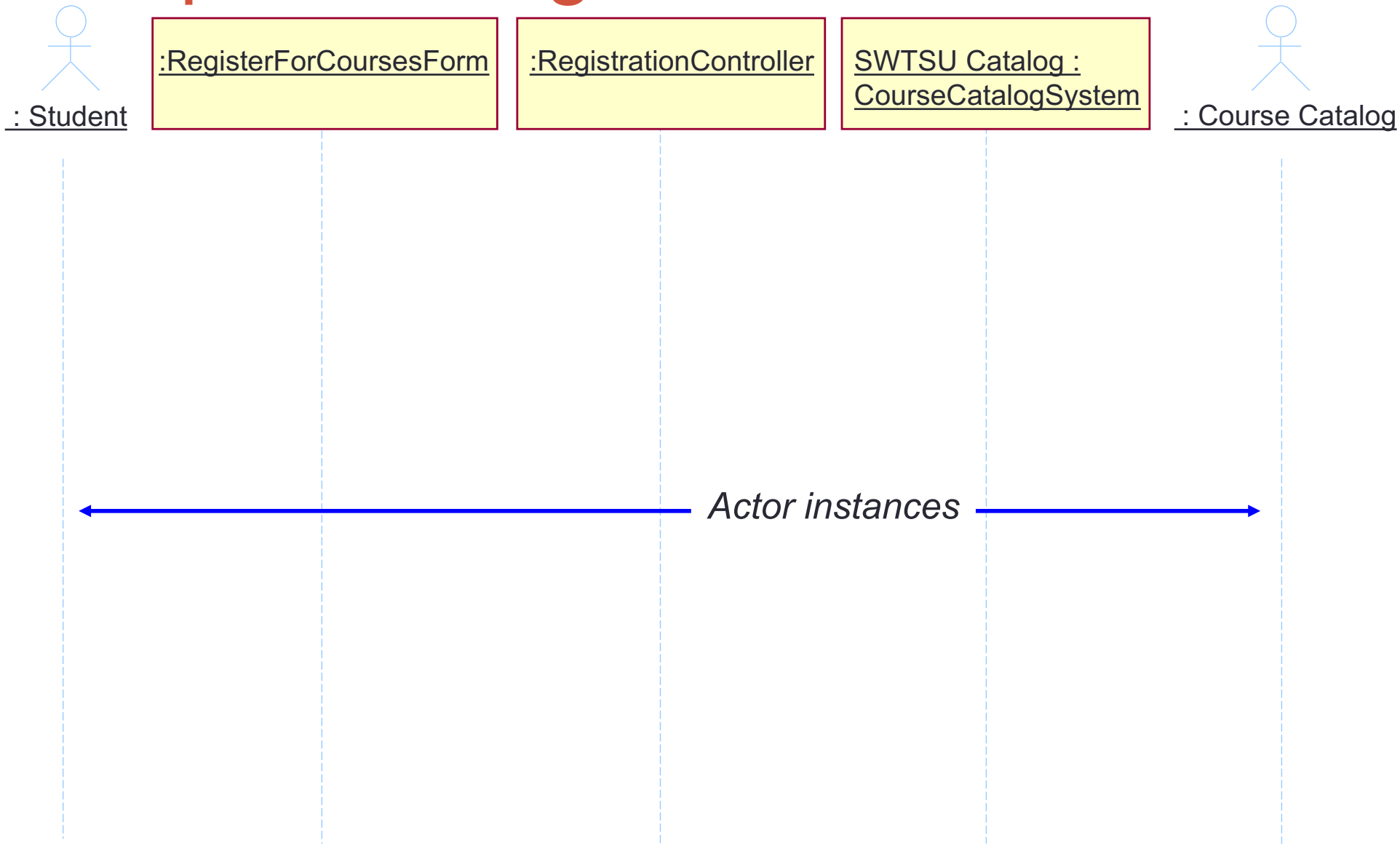
Example: Sequence Diagram



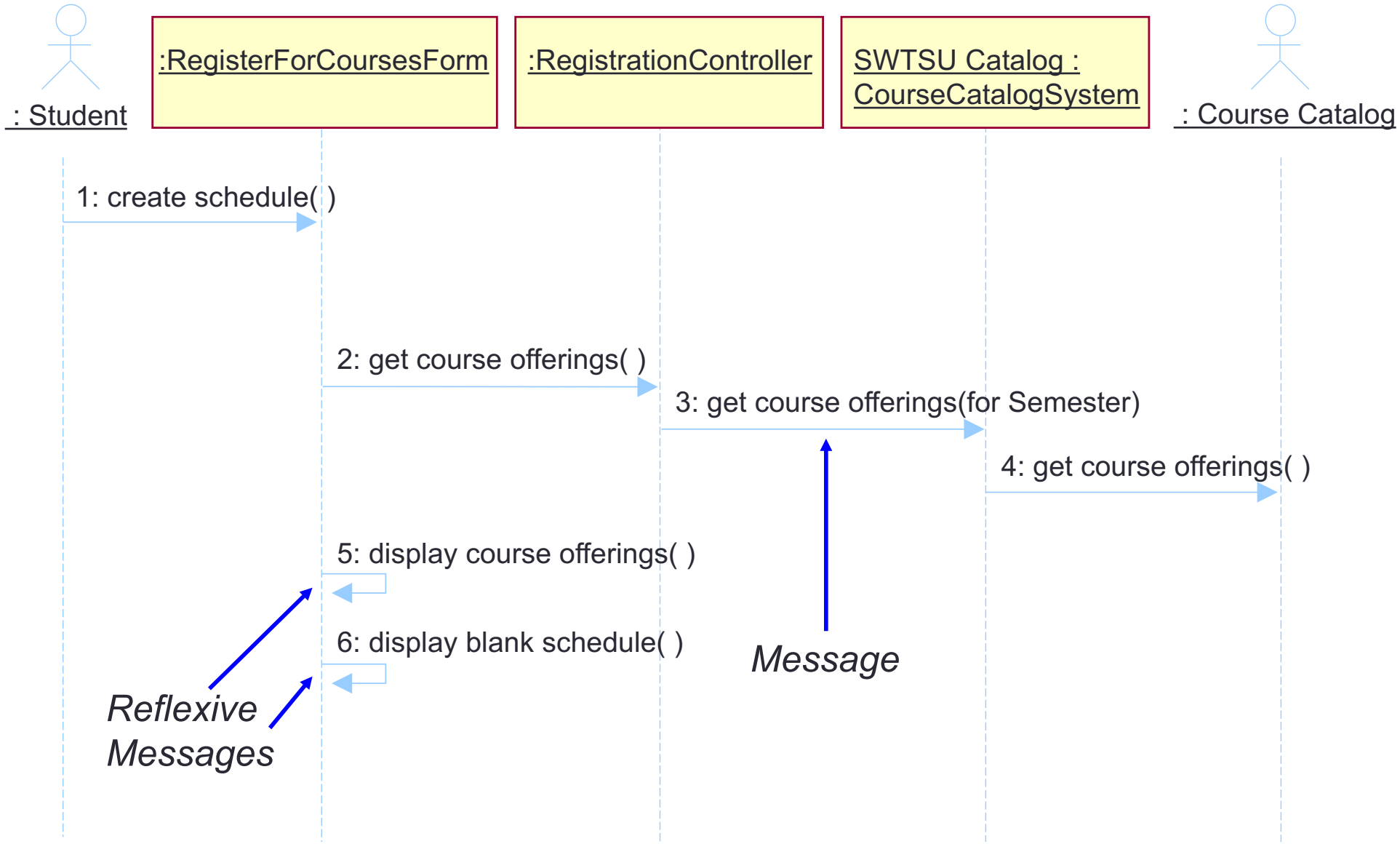
Sequence Diagram Contents: Objects



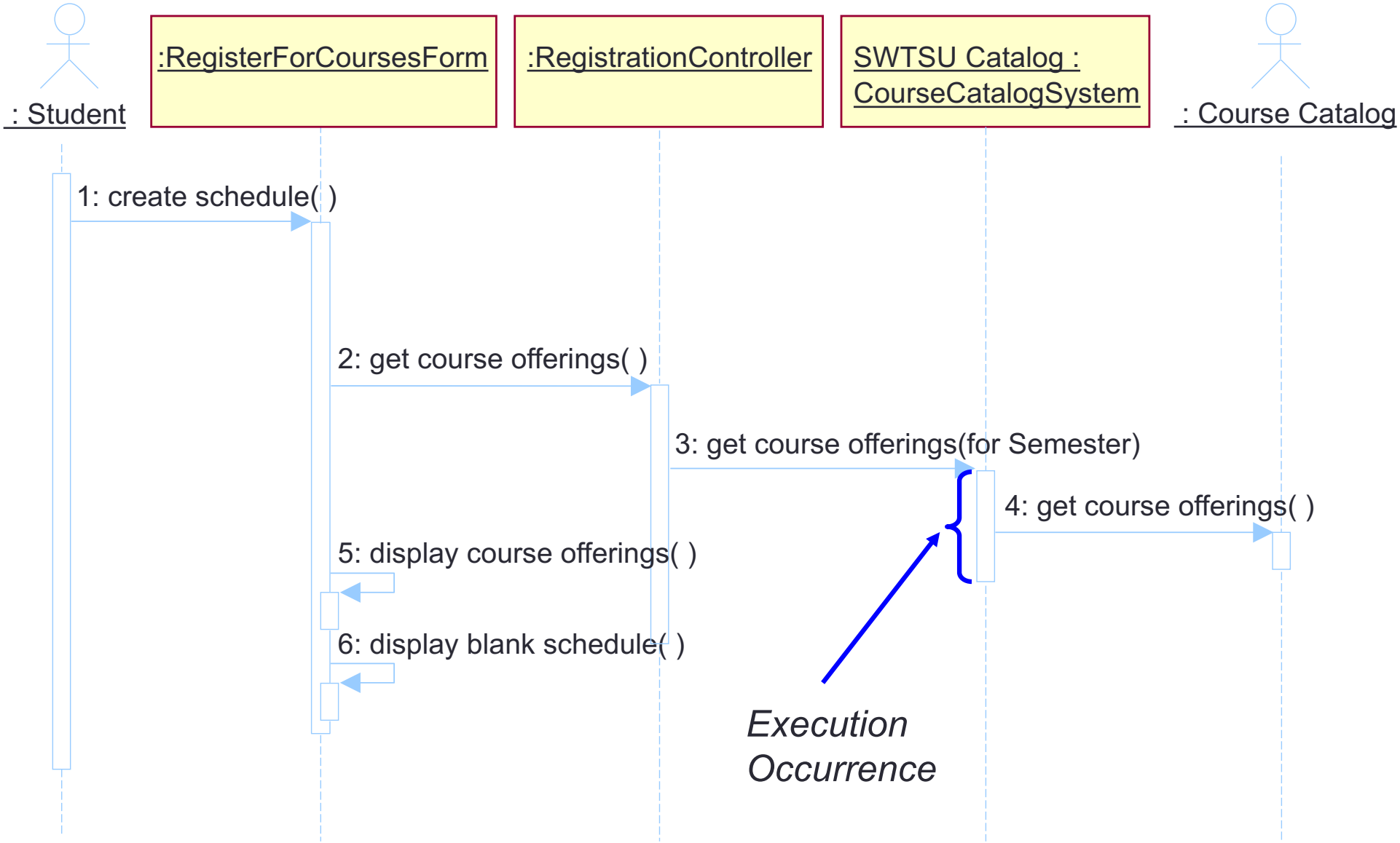
Sequence Diagram Contents: Actor



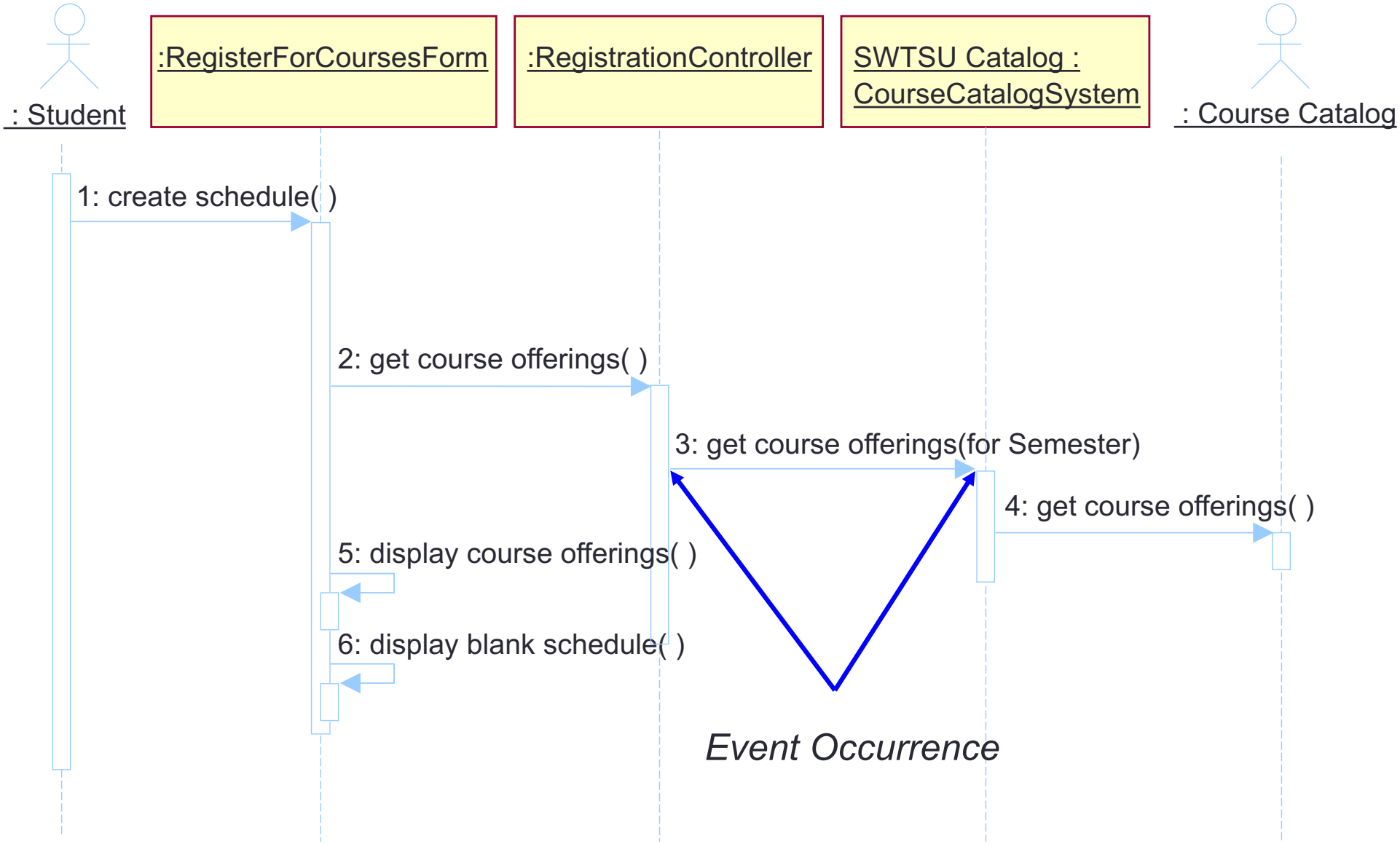
Sequence Diagram Contents: Messages



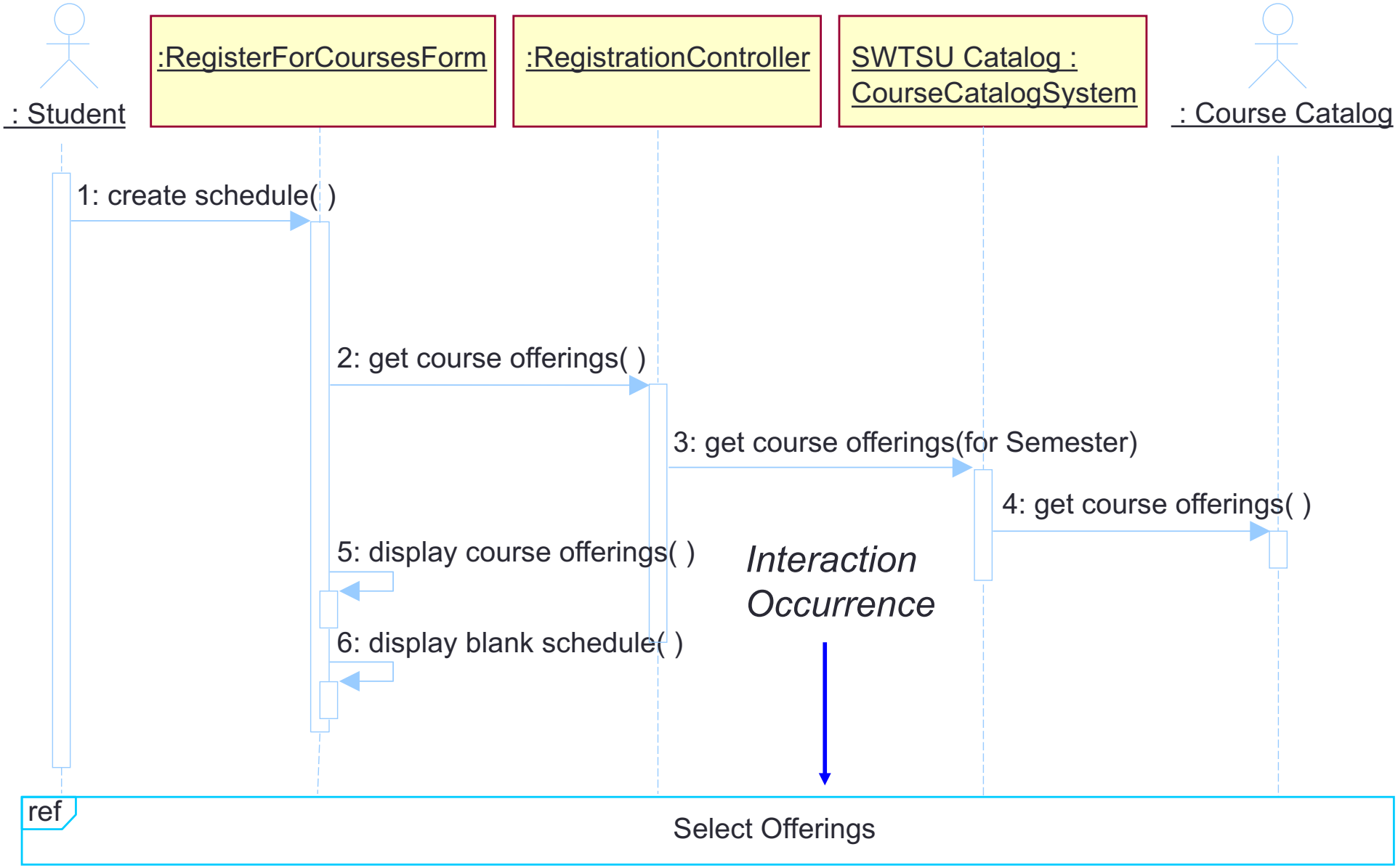
Sequence Diagram Contents: Execution Occurrence



Sequence Diagram Contents: Event Occurrence



Sequence Diagram Contents: Interaction Occurrence

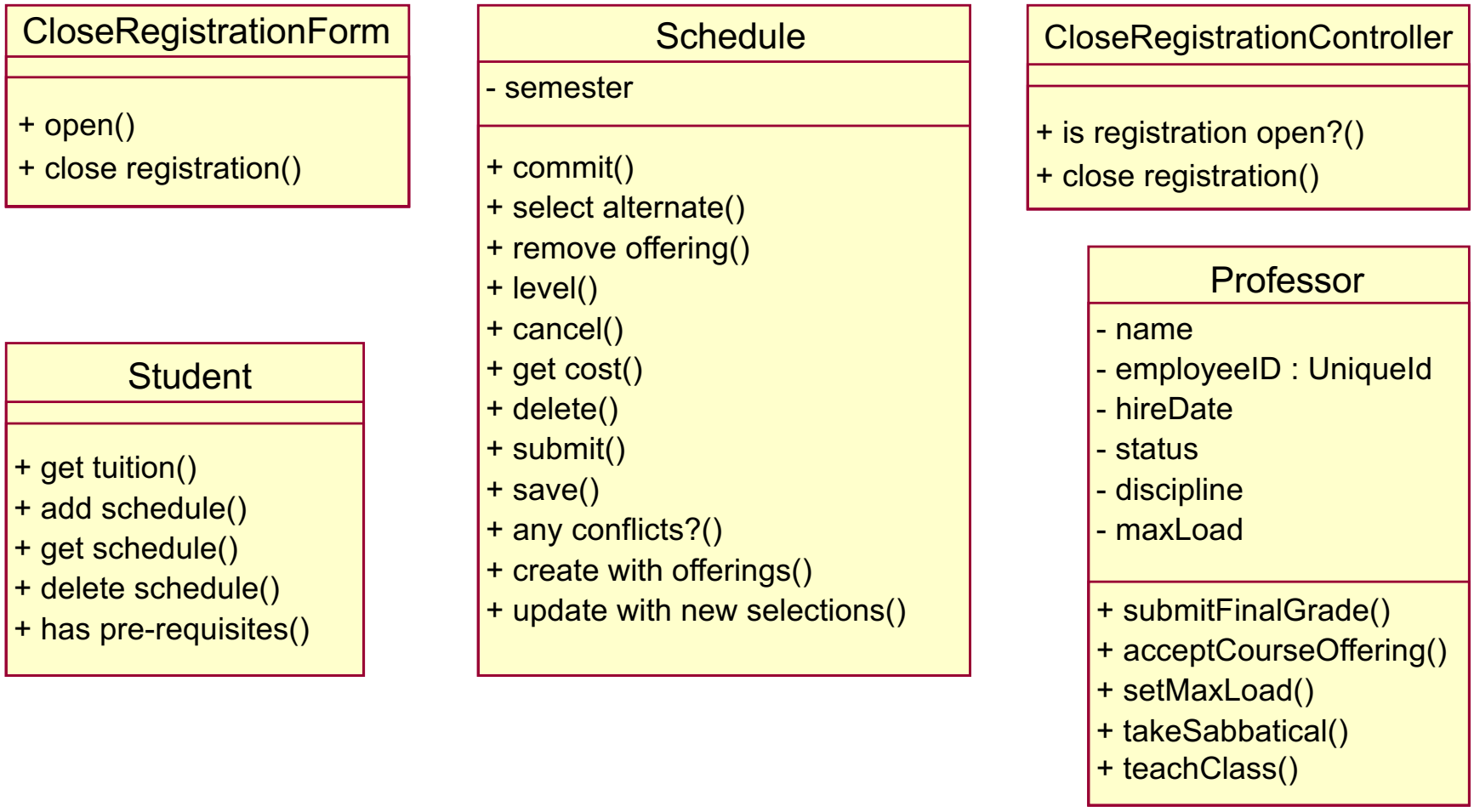


Outline

- Use-case diagrams
- Activity diagrams
- Sequence diagrams
- **Class diagrams**

What Is a Class Diagram?

- Static view of a system

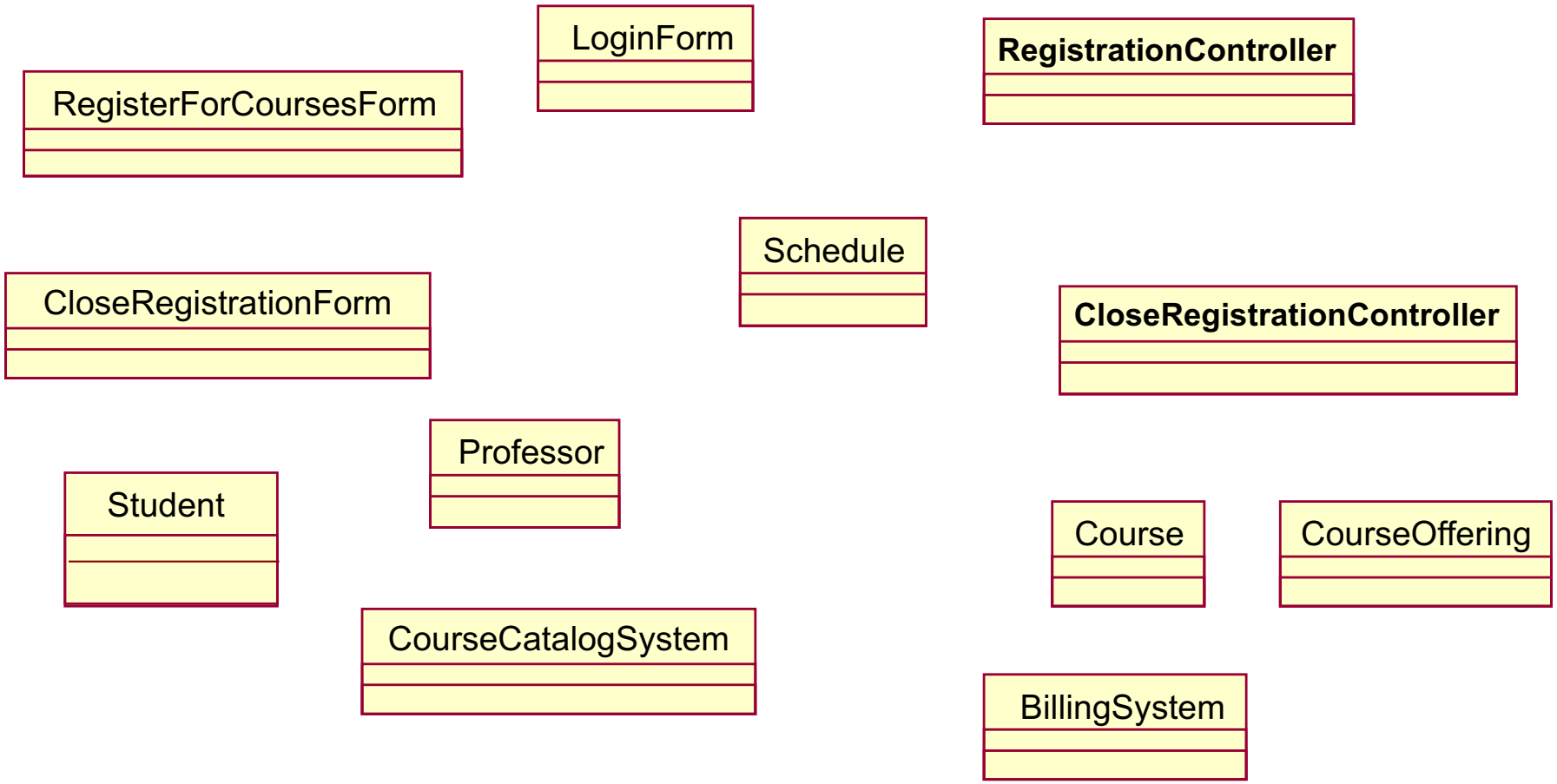


Class Diagram Usage

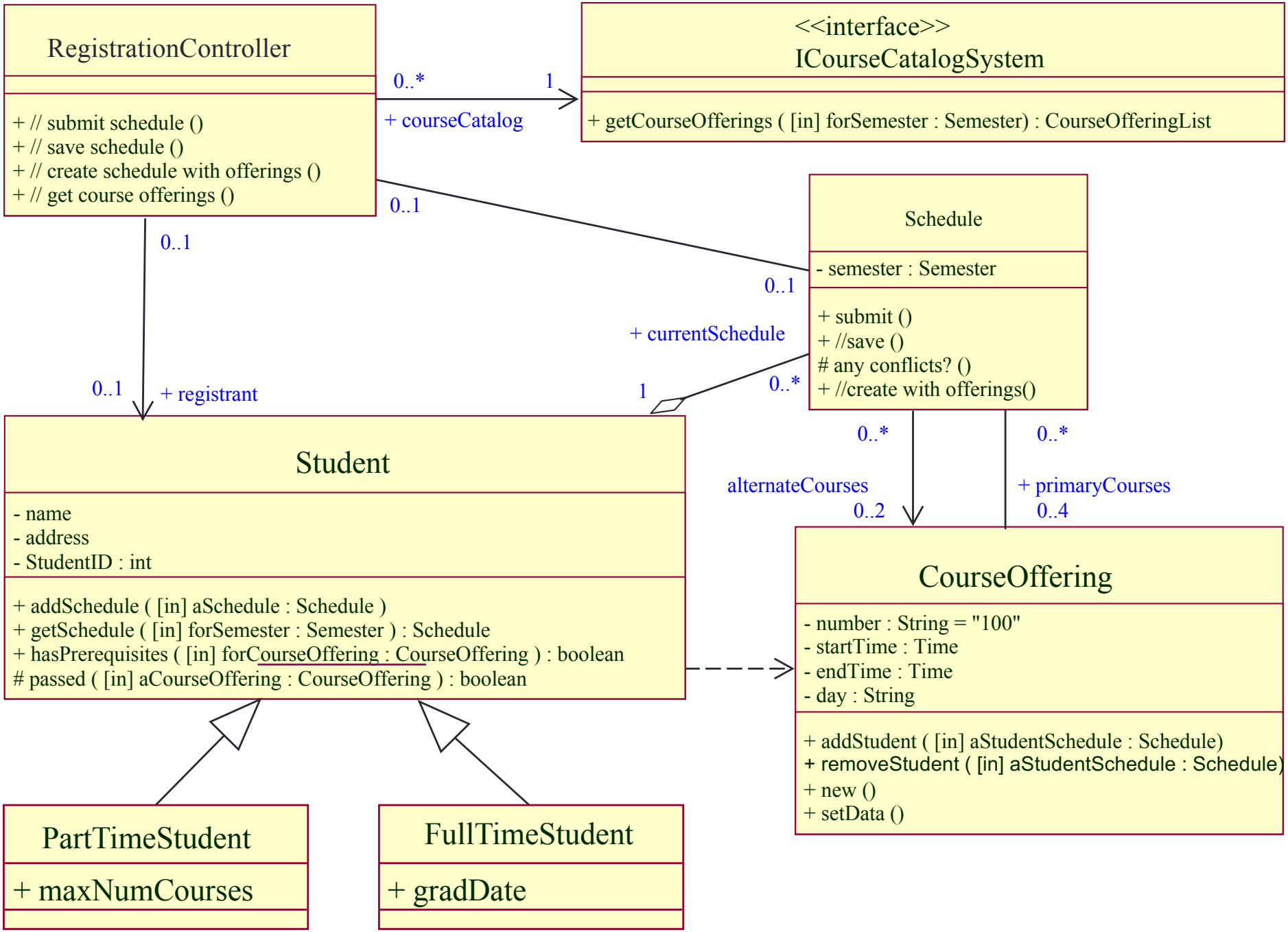
- When modeling the static view of a system, class diagrams are typically used in one of three ways, to model:
  - The vocabulary of a system
  - Collaborations
  - A logical database schema

Example: Class Diagram

- Is there a better way to organize class diagrams?

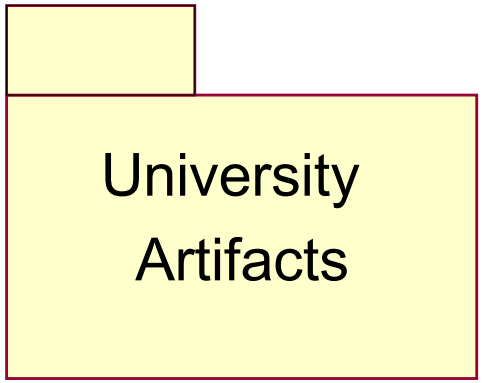


E.g. Class diagram for UC "Register for course"

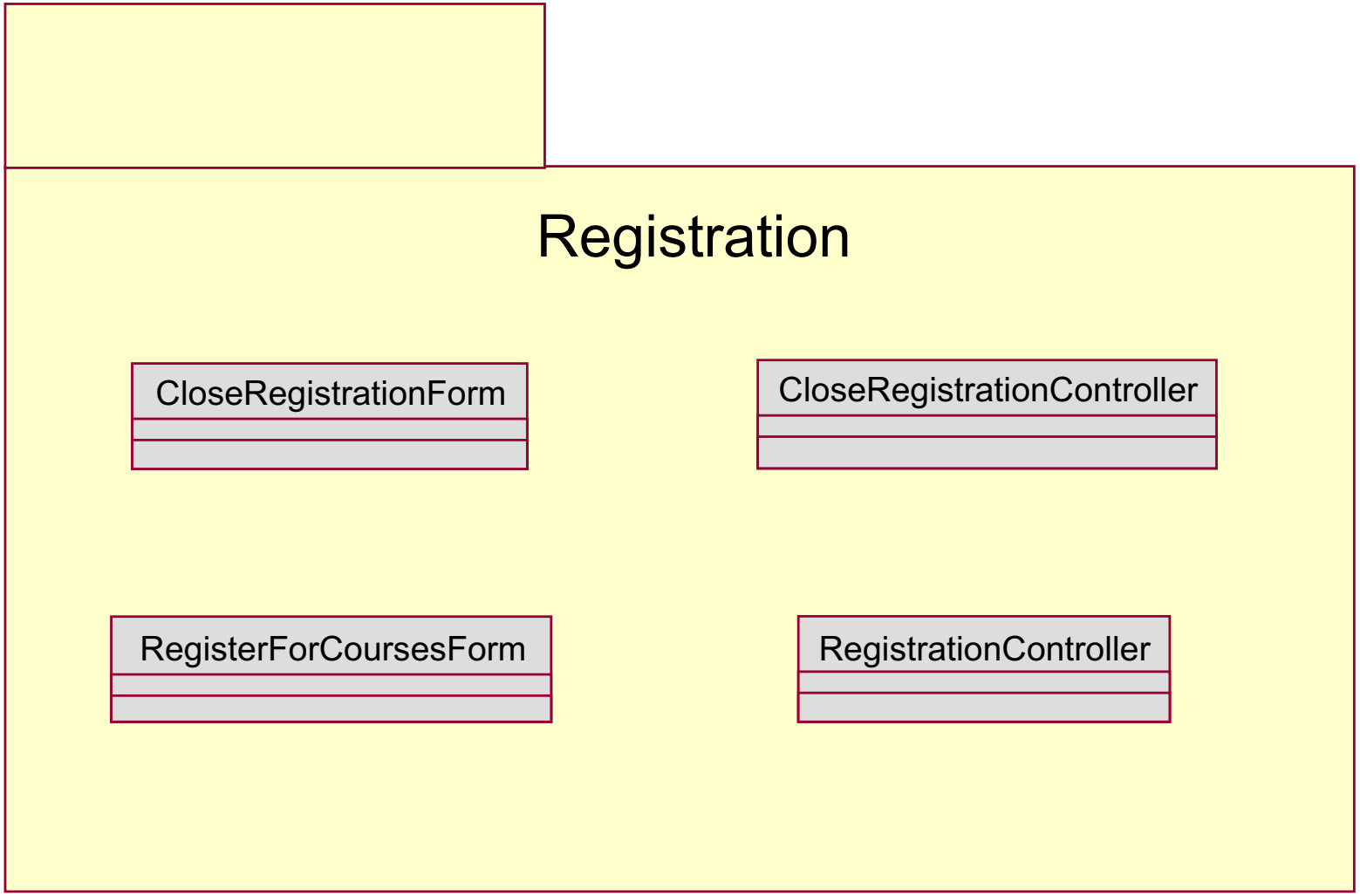


Review: What Is a Package?

- A general purpose mechanism for organizing elements into groups.
- A model element that can contain other model elements.
- A package can be used:
  - To organize the model under development
  - As a unit of configuration management

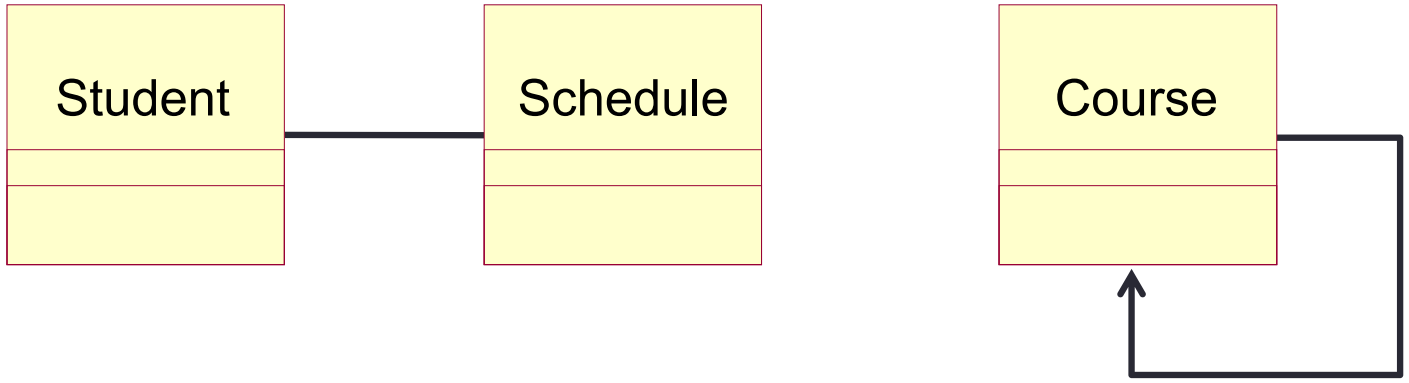


Example: Registration Package



What Is an Association?

- The semantic relationship between two or more classifiers that specifies connections among their instances.
- A structural relationship specifying that objects of one thing are connected to objects of another thing.



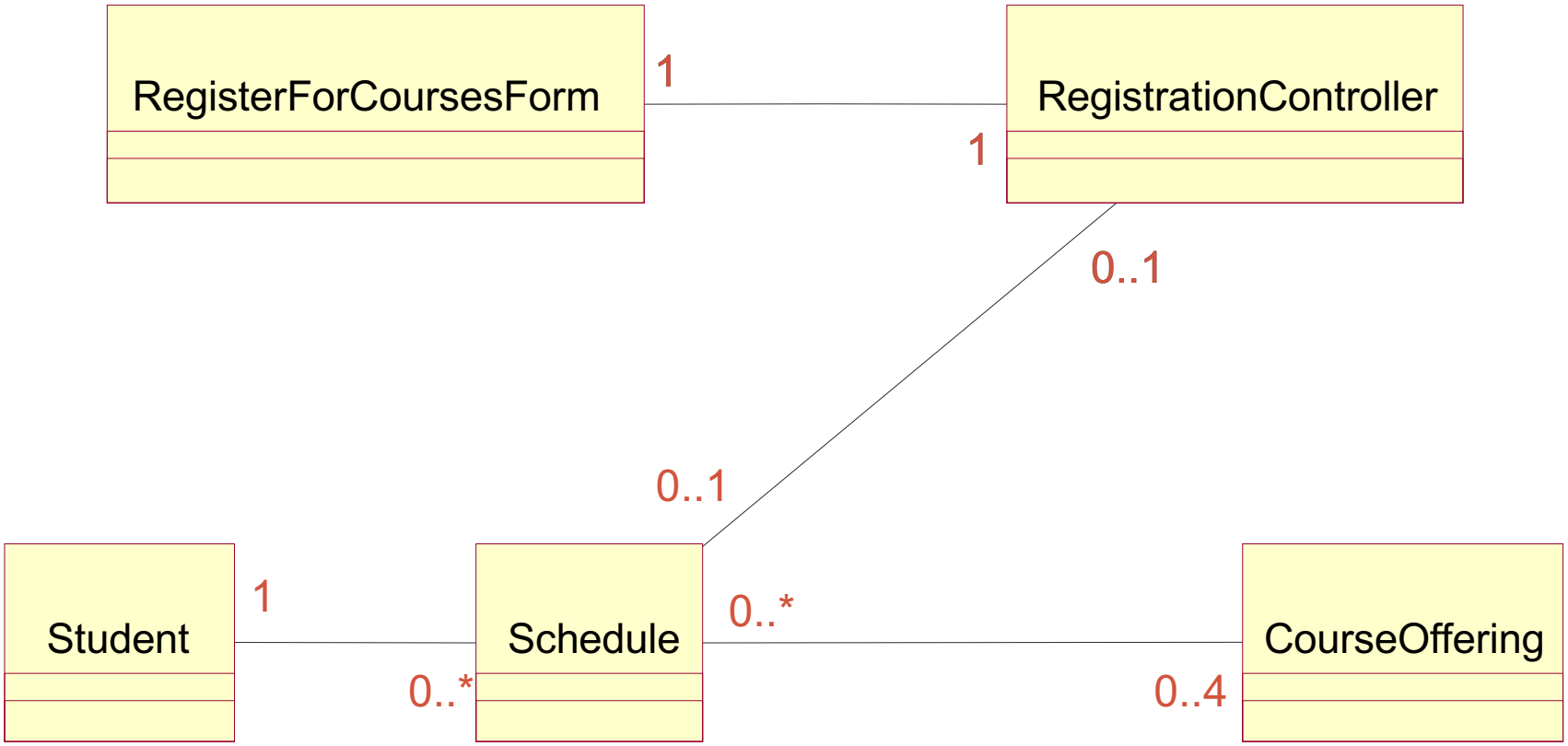
What Is Multiplicity?

- Multiplicity is the number of instances one class relates to ONE instance of another class.
- For each association, there are two multiplicity decisions to make, one for each end of the association.
  - For each instance of Professor, many Course Offerings may be taught.
  - For each instance of Course Offering, there may be either one or zero Professor as the instructor.

Multiplicity Indicators

Unspecified	
Exactly One	1
Zero or More	0..*
Zero or More	*
One or More	1..*
Zero or One (optional value)	0..1
Specified Range	2..4
Multiple, Disjoint Ranges	2, 4..6

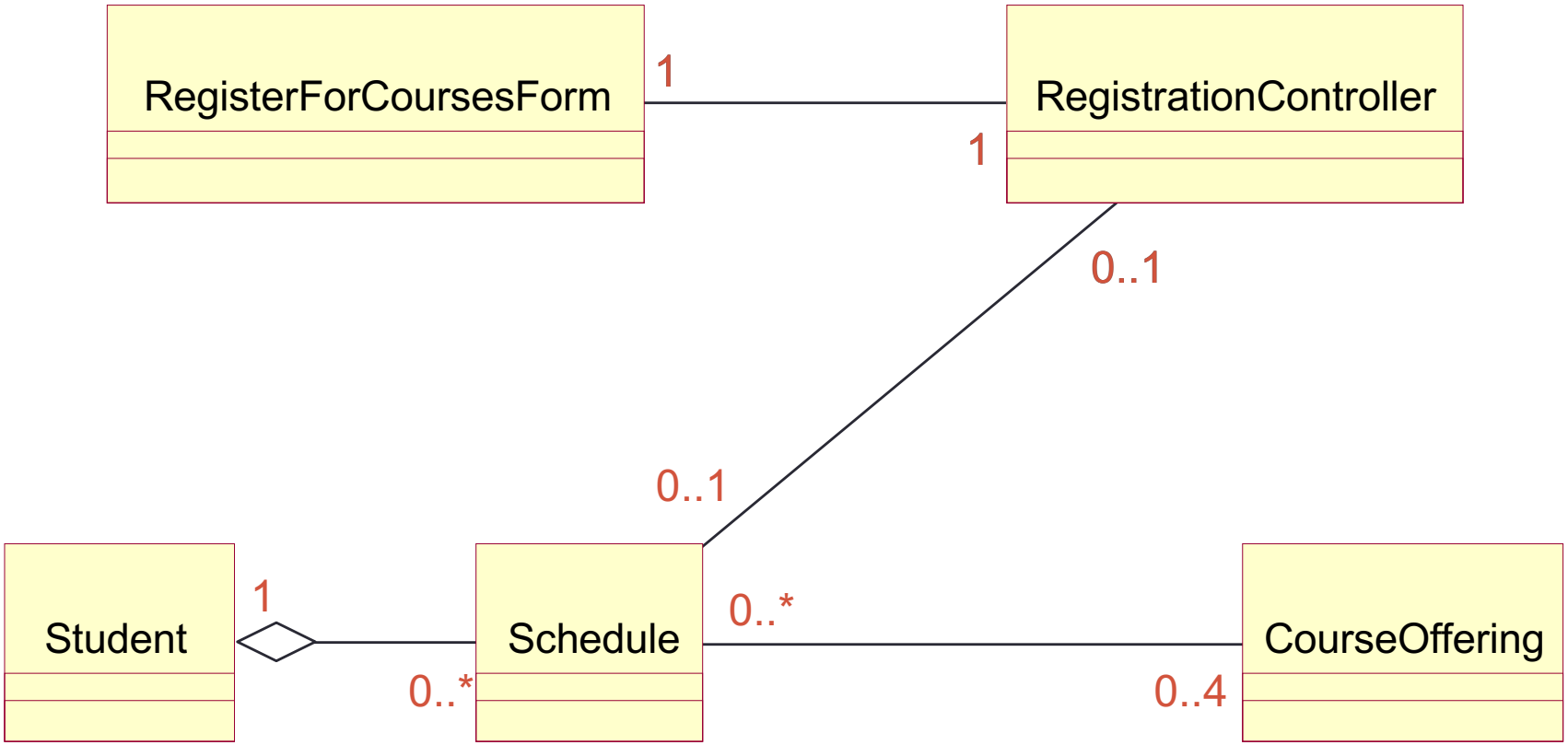
Example: Multiplicity



What Is an Aggregation?

- A special form of association that models a whole-part relationship between the aggregate (the whole) and its parts.
  - An aggregation is an “is a part-of” relationship.
- Multiplicity is represented like other associations.

Example: Aggregation

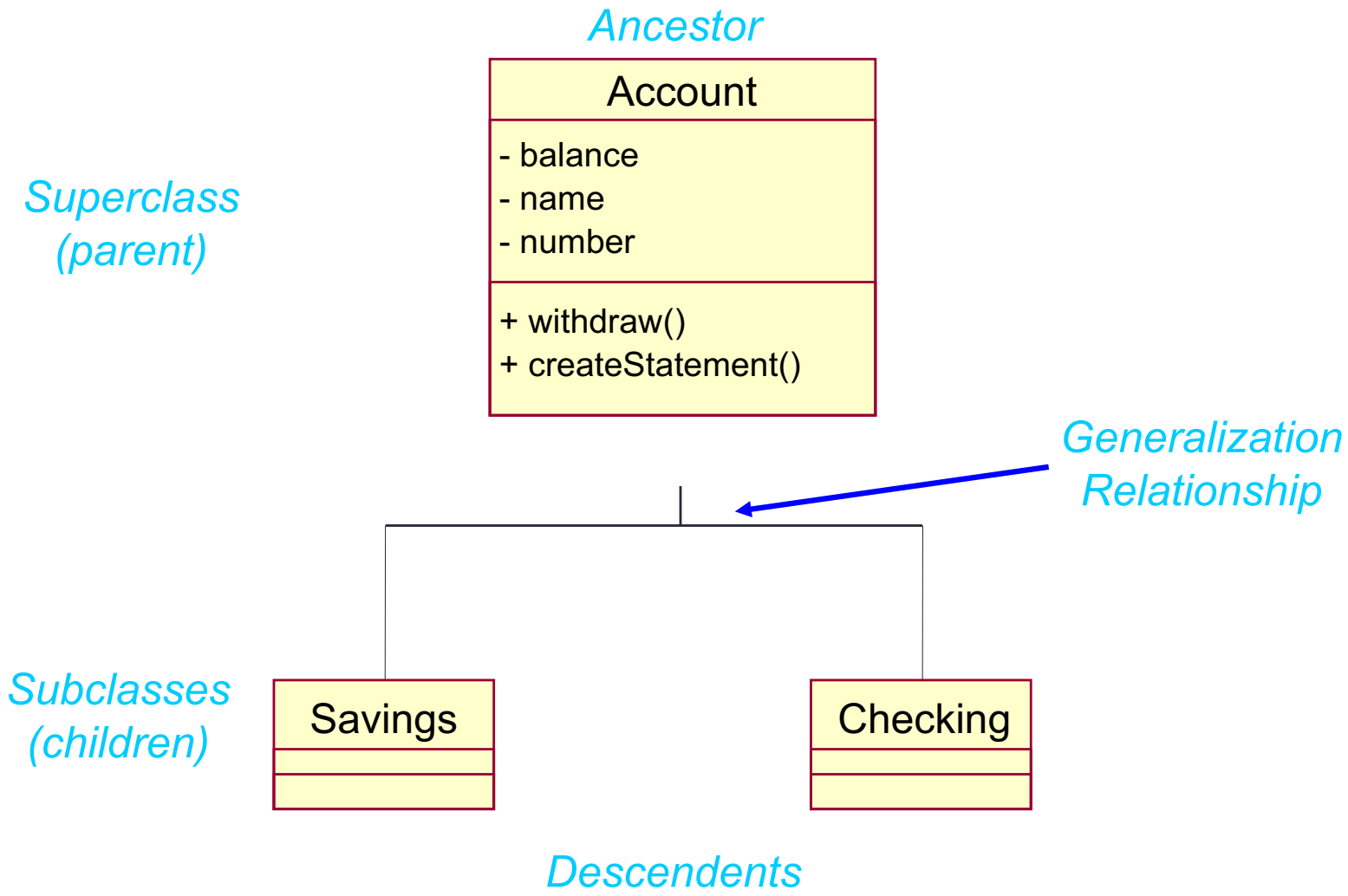


Review: What Is Generalization?

- A relationship among classes where one class shares the structure and/or behavior of one or more classes.
- Defines a hierarchy of abstractions where a subclass inherits from one or more superclasses.
  - Single inheritance
  - Multiple inheritance
- Is an “is a kind of” relationship.

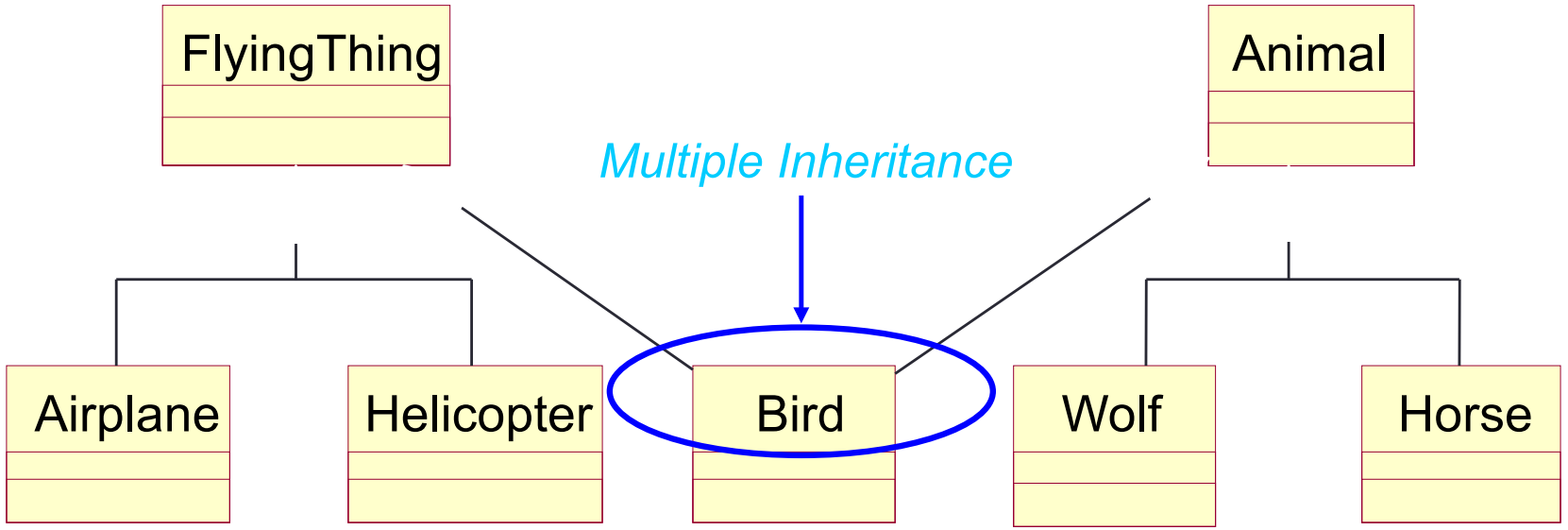
Example: Single Inheritance

- One class inherits from another.



Example: Multiple Inheritance

- A class can inherit from several other classes.



Use multiple inheritance only when needed and always with caution!