# Object-Oriented Analysis and Design

# Learning Objectives

- Key terms
  - Association
  - Class diagram
  - Event
  - Object
  - Object class
  - Operation
  - Sequence diagram
  - State
  - State transition
  - Unified Modeling Language (UML)
  - Use case

## Learning Objectives (continued)

- Discuss the concepts and principles underlying the object-oriented approach.
- Learn to develop requirements models using use-case diagrams.
- Learn to use class diagrams to develop object models of the problem domain.
- Learn to develop requirements models using state and sequence diagrams.

# The Object-Oriented Modeling Approach

- Benefits
  - The ability to tackle more challenging problem domains
  - Improved communication among users, analysts, designers, and programmers
  - Reusability of analysis, design, and programming results
  - Increased consistency among the models developed during object-oriented analysis, design, and programming

## The Object-Oriented Modeling Approach (continued)

- Object-Oriented Systems Development Life Cycle
  - Process of progressively developing representation of a system component (or object) through the phases of analysis, design, and implementation
  - The model is abstract in the early stages
  - As the model evolves, it becomes more and more detailed

## The Object-Oriented Systems Development Life Cycle

- Analysis Phase
  - Model of the real-world application is developed showing its important properties
  - Model specifies the functional behavior of the system independent of implementation details
- Design Phase
  - Analysis model is refined and adapted to the environment
- Implementation Phase
  - Design is implemented using a programming language or database management system

## The Object-Oriented Systems Development Life Cycle (continued)

- Unified Modeling Language (UML)
  - A notation that allows the modeler to specify, visualize and construct the artifacts of software systems, as well as business models
  - Techniques and notations
    - Use cases
    - Class diagrams
    - State diagrams
    - Sequence diagrams

# Use-Case Modeling

- Applied to analyze functional requirements of the system
- Performed during the analysis phase to help developers understand functional requirements of the system without regard for implementation details
- Use Case
  - A complete sequence of related actions initiated by an actor
- Actor
  - An external entity that interacts with the system

## Use-Case Modeling

- Use cases represent complete functionality of the system
- Use cases may participate in relationships with other use cases
- Use cases may also use other use cases

A.9

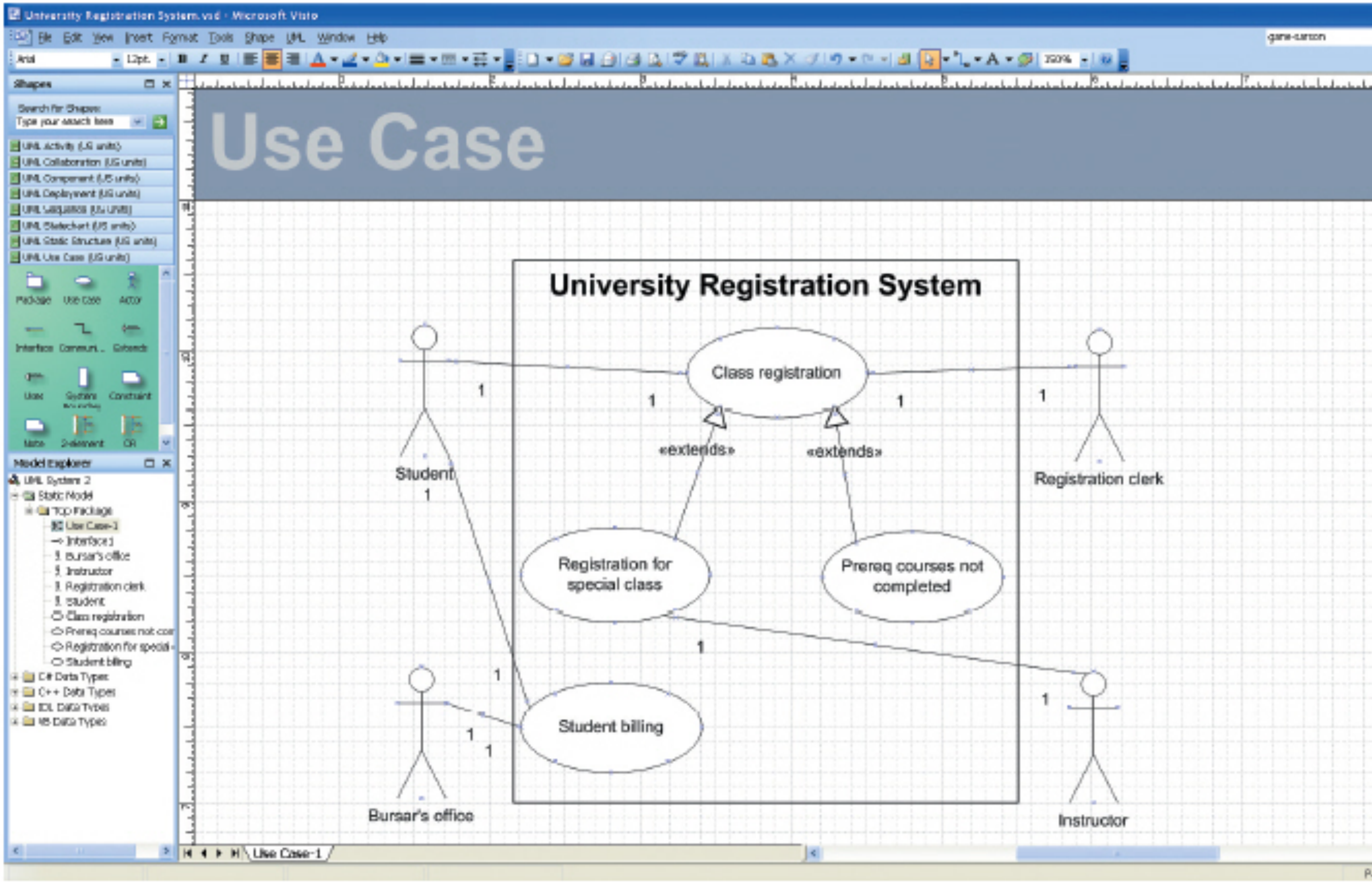


**FIGURE A-1**
**Use-Case Diagram for a University Registration System Drawn Using Microsoft Visio**

## Object Modeling: Class Diagrams

- Object
  - An entity that has a well-defined role in the application domain, and has state, behavior, and identity
- State
  - A condition that encompasses an object’ s properties and the values those properties have
- Behavior
  - A manner that represents how an object acts and reacts
- Object Class
  - A set of objects that share a common structure and a common behavior

## Object Modeling: Class Diagrams (continued)

- Class Diagram
  - Class is represented as a rectangle with three compartments
  - Objects can participate in relationships with objects of the same class

## Object Modeling: Object Diagrams

- Object Diagram
  - A graph of instances that are compatible with a given class diagram; also called an instance diagram
  - Object is represented as a rectangle with two compartments
- Operation
  - A function or service that is provided by all the instances of a class
- Encapsulation
  - The technique of hiding the internal implementation details of an object from its external view

**Figure A.3a** UML Class and Object Diagrams — Class Diagram Showing Two Classes

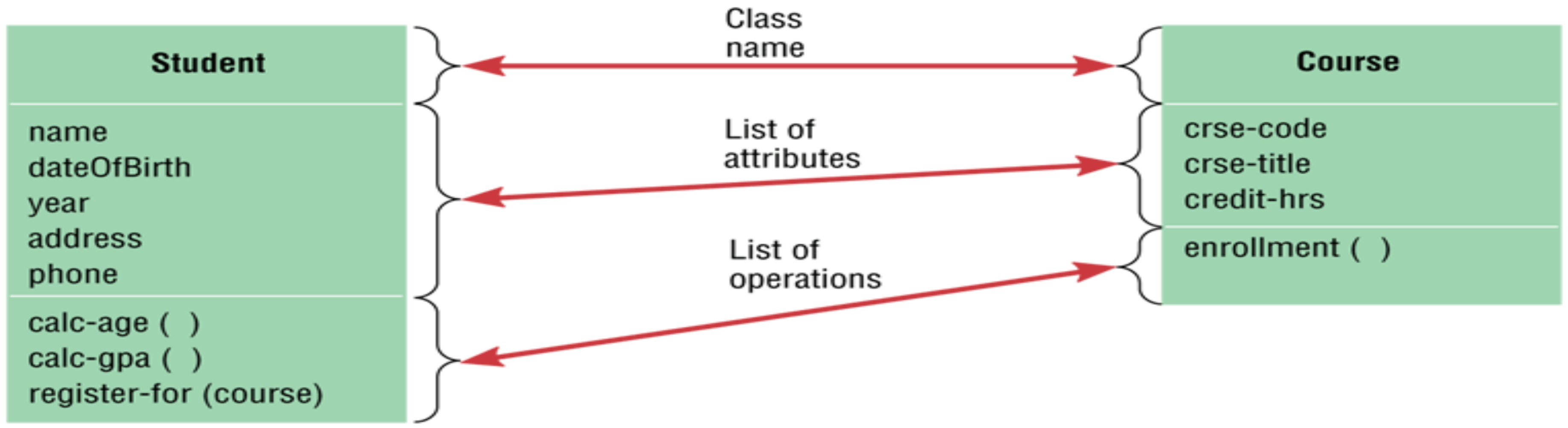
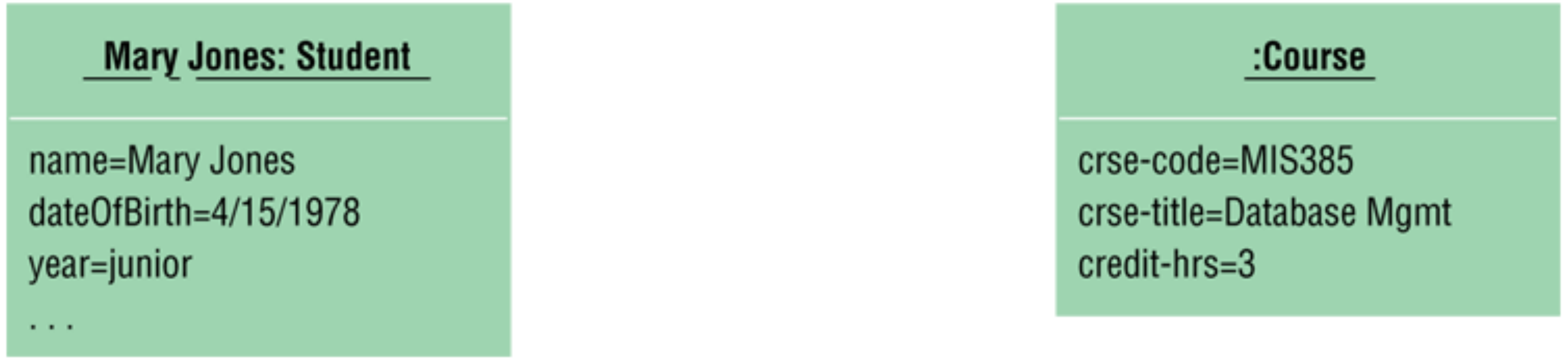


**Figure A.3b** UML Class and Object Diagrams — Object Diagram with Two Instances



## Representing Associations

- Association
  - A relationship between object classes
  - Degree may be unary, binary, ternary or higher
  - Depicted as a solid line between participating classes
- Association Role
  - The end of an association where it connects to a class
  - Each role has multiplicity, which indicates how many objects participate in a given association relationship

**Figure A.4a** Examples of Association Relationships of Different Degrees — Unary

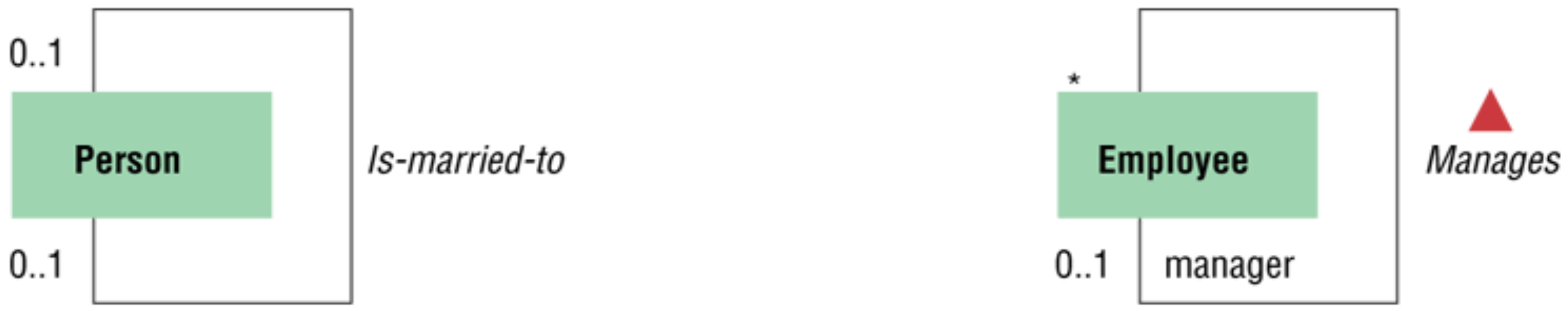


**Figure A.4b** Examples of Association Relationships of Different Degrees — Binary

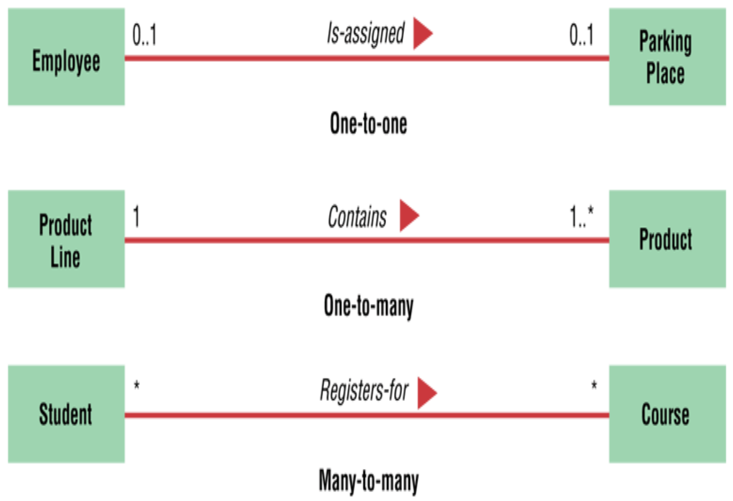
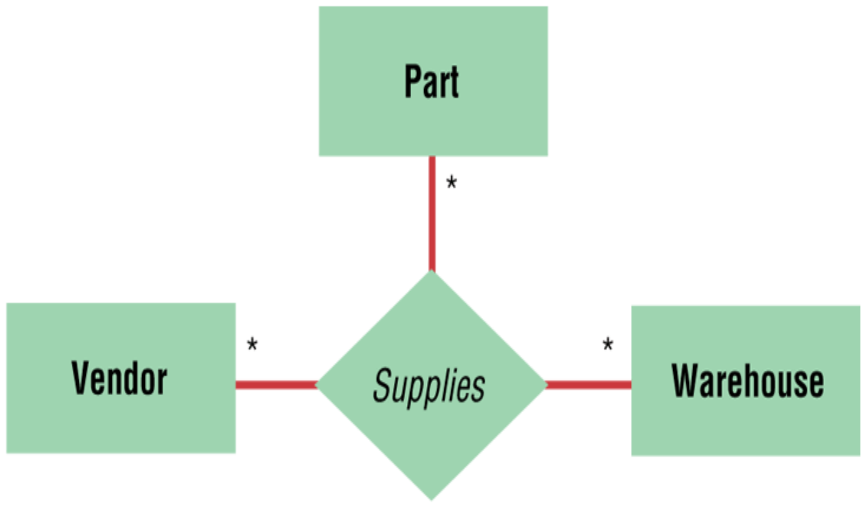


**Figure A.4c** Examples of Association Relationships of Different Degrees — Ternary



## Representing Generalization

- Generalization
  - Abstraction of common features among multiple classes, as well as their relationships, into a more general class
- Subclass
  - A class that has been generalized
- Superclass
  - A class that is composed of several generalized subclasses

## Representing Generalization (continued)

- Discriminator
  - Shows which property of an object class is being abstracted by a generalization relationship
- Inheritance
  - A property that a subclass inherits the features from its superclass
- Abstract Class
  - A class that has no direct instances but whose descendents may have direct instances
- Concrete Class
  - A class that can have direct instances

**Figure A.6a** Examples of Generalization, Inheritance, and Constraints — Employee Superclass with Three Subclasses

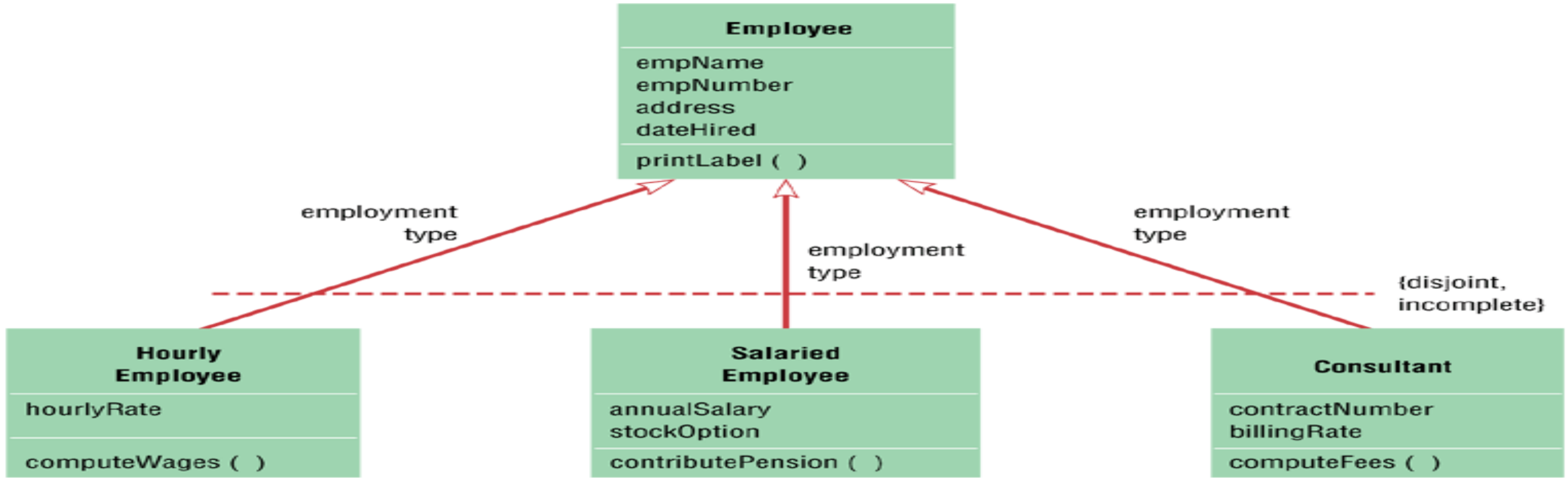
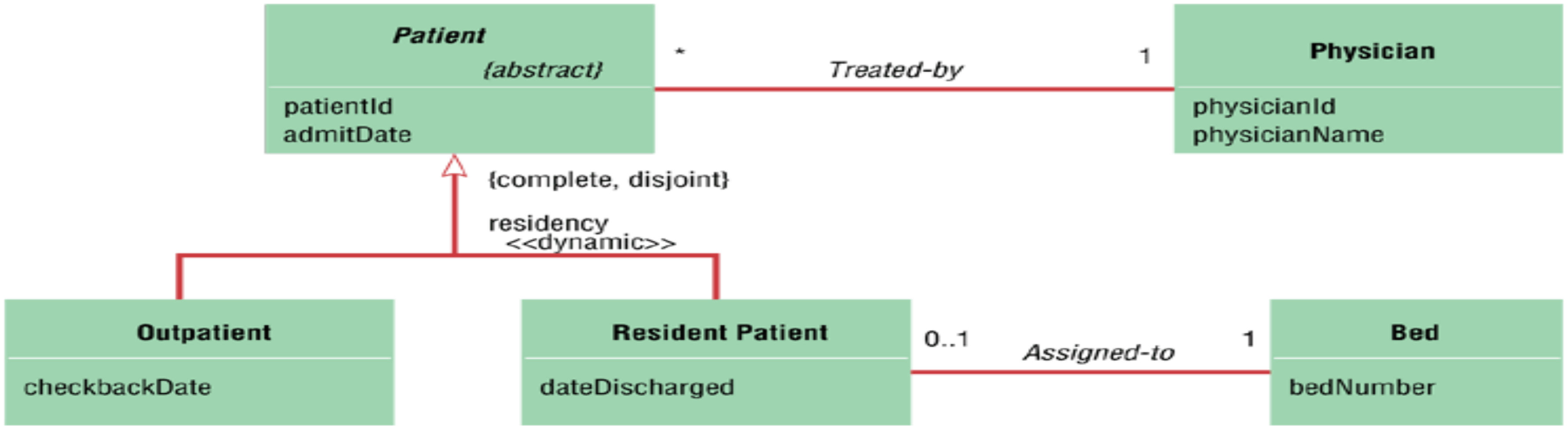


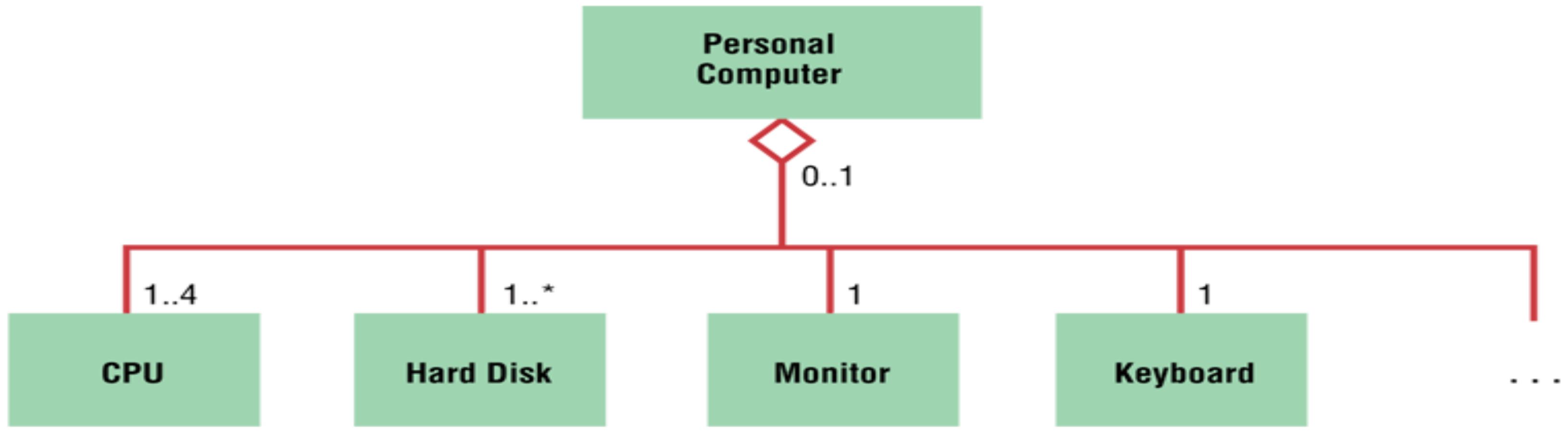
**Figure A.6b** Examples of Generalization, Inheritance, and Constraints — Abstract Patient Class with Two Concrete Subclasses



# Representing Aggregation

- Aggregation
  - A part-of relationship between a component object and an aggregate object
  - Example: Personal computer
    - Composed of CPU, Monitor, Keyboard, etc

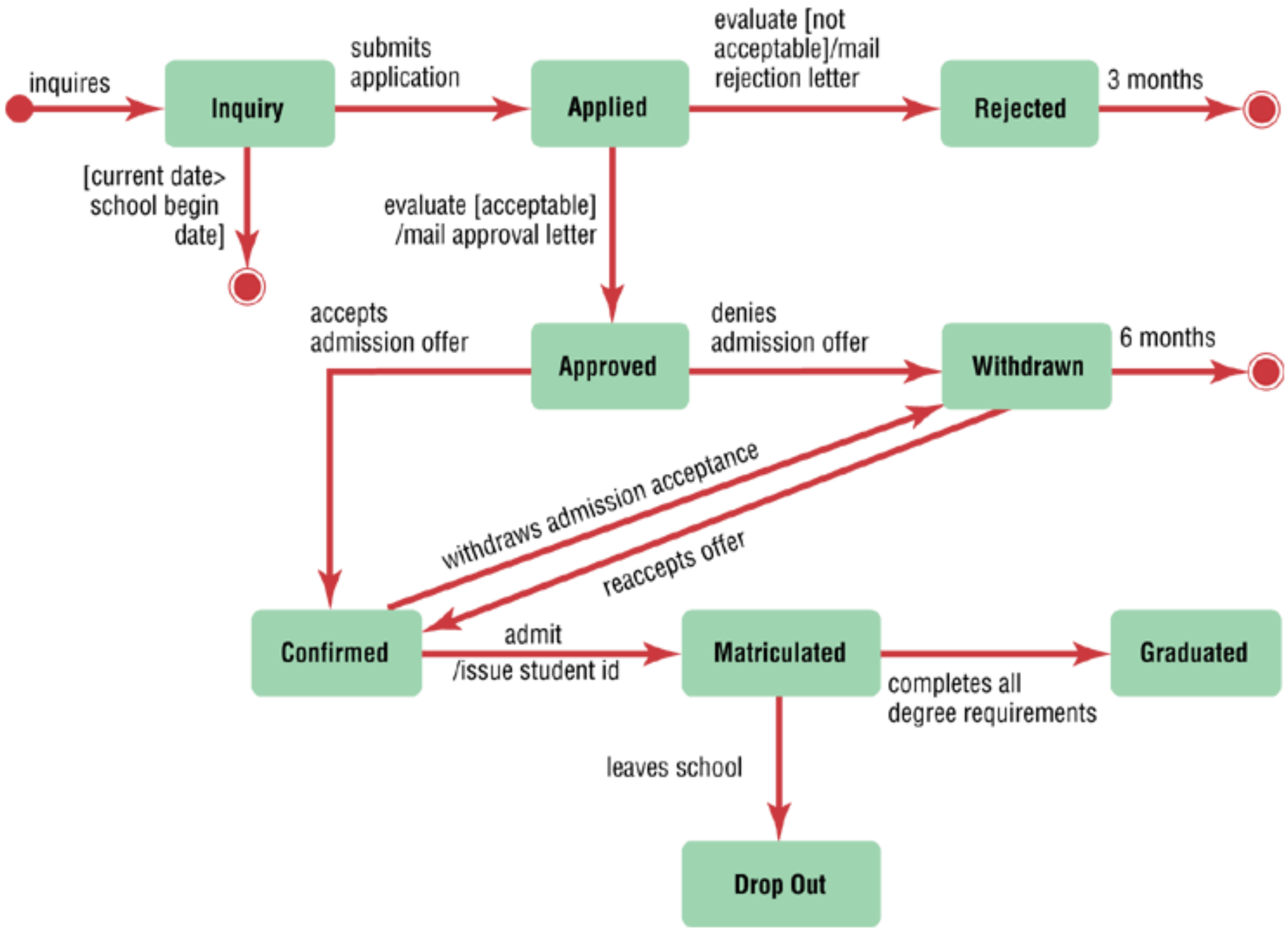
**Figure A.7** Example of Aggregation



## Dynamic Modeling: State Diagrams

- State
  - A condition during the life of an object during which it satisfies some conditions, performs some actions or waits for some events
  - Shown as a rectangle with rounded corners
- State Transition
  - The changes in the attributes of an object or in the links an object has with other objects
  - Shown as a solid arrow
  - Diagrammed with a guard condition and action
- Event
  - Something that takes place at a certain point in time

**Figure A.8** State Diagram for the Student Object



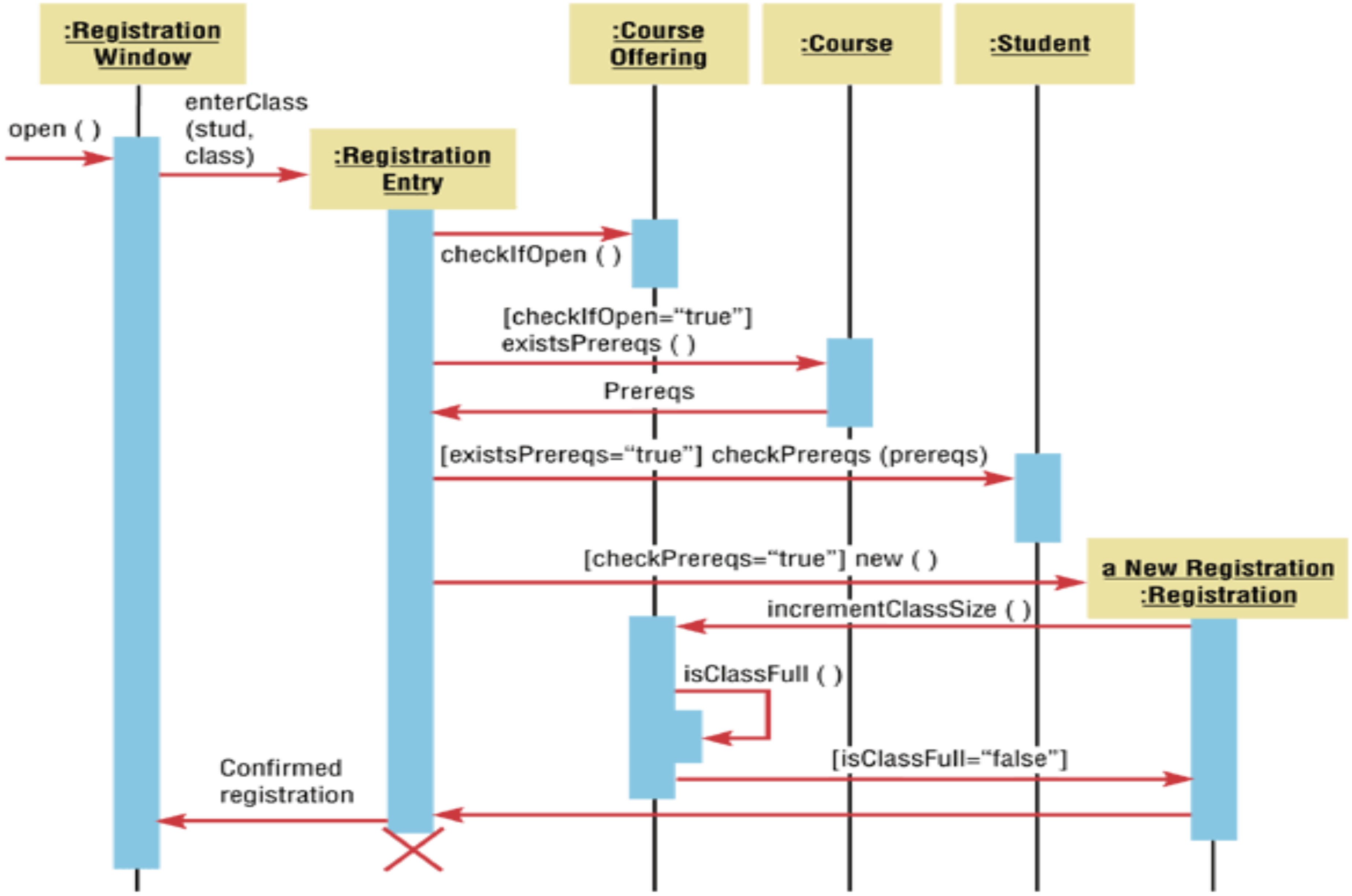
## Dynamic Modeling: Sequence Diagrams

- Sequence Diagram
  - A depiction of the interaction among objects during certain periods of time
- Activation
  - The time period during which an object performs an operation
- Messages
  - Means by which objects communicate with each other

## Dynamic Modeling: Sequence Diagrams (continued)

- Synchronous Message
  - A type of message in which the caller has to wait for the receiving object to finish executing the called operation before it can resume execution itself
- Simple Message
  - A message that transfers control from the sender to the recipient without describing the details of the communication

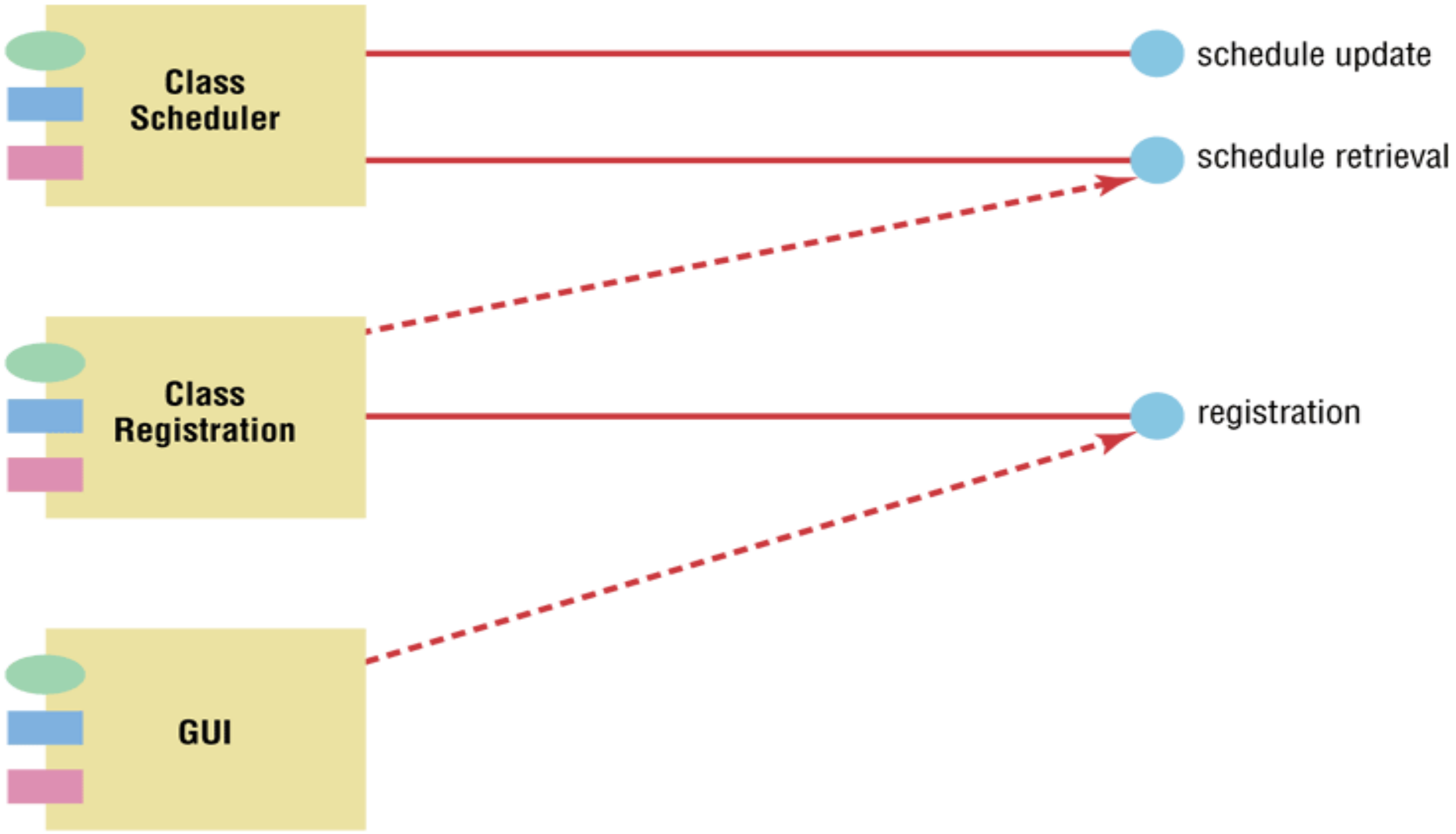
**Figure A.9** Sequence Diagram for a Class Registration Scenario with Prerequisites



## Moving to Design

- Start with existing set of analysis model
- Progressively add technical details
- Design model must be more detailed than analysis model
- Component Diagram
  - A diagram that shows the software components or modules and their dependencies
- Deployment Diagram
  - A diagram that shows how the software components, processes and objects are deployed into the physical architecture of the system

**Figure A.11** A Component Diagram for Class Registration



## Summary

- Object-Oriented Modeling Approach
  - Benefits
  - Unified Modeling Language
    - Use cases
    - Class diagrams
    - State diagrams
    - Sequence diagrams
- Use Case Modeling

## Summary (continued)

- Object Modeling: Class Diagrams
  - Associations
  - Generalizations
  - Aggregation
- Dynamic Modeling: State Diagrams
- Dynamic Modeling: Sequence Diagrams
- Moving to Design