
IS 0020

Program Design and Software Tools

Unified Modeling Language
Lecture 13

April 13, 2005

What is UML?

- The **Unified Modelling Language** is a standard notation to model [object oriented] systems.
 - Syntax and semantics
 - Model systems
 - Visual documentations
 - Collections of best practices
 - Used to produce a set of artifacts that can be delivered
 - Wide support

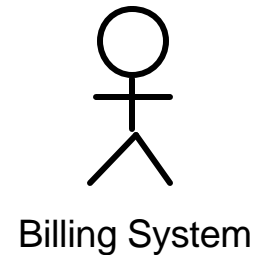
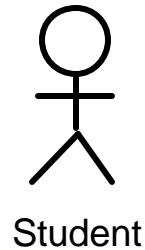
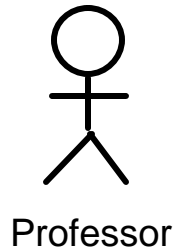
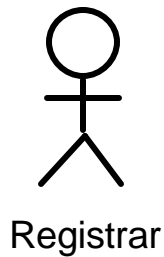
 - Independent of programming languages
 - Support high level concepts

UML Diagram Types

- Use Case diagrams
- Static structure diagrams
 - Class diagrams
 - Object diagrams
- Interaction diagrams
 - Sequence diagrams
 - Collaboration diagrams
- Statechart diagrams
- Activity diagrams
- Component diagrams
- Deployment diagram

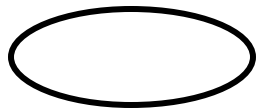
Actors

- An actor is someone or some thing that must interact with the system under development

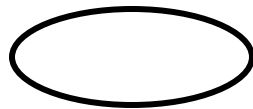


Use Cases

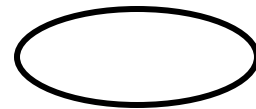
- A use case is a pattern of behavior the system exhibits
 - Each use case is a sequence of related transactions performed by an actor and the system in a dialogue
- Actors are examined to determine their needs
 - Registrar -- maintain the curriculum
 - Professor -- request roster
 - Student -- maintain schedule



Maintain Curriculum



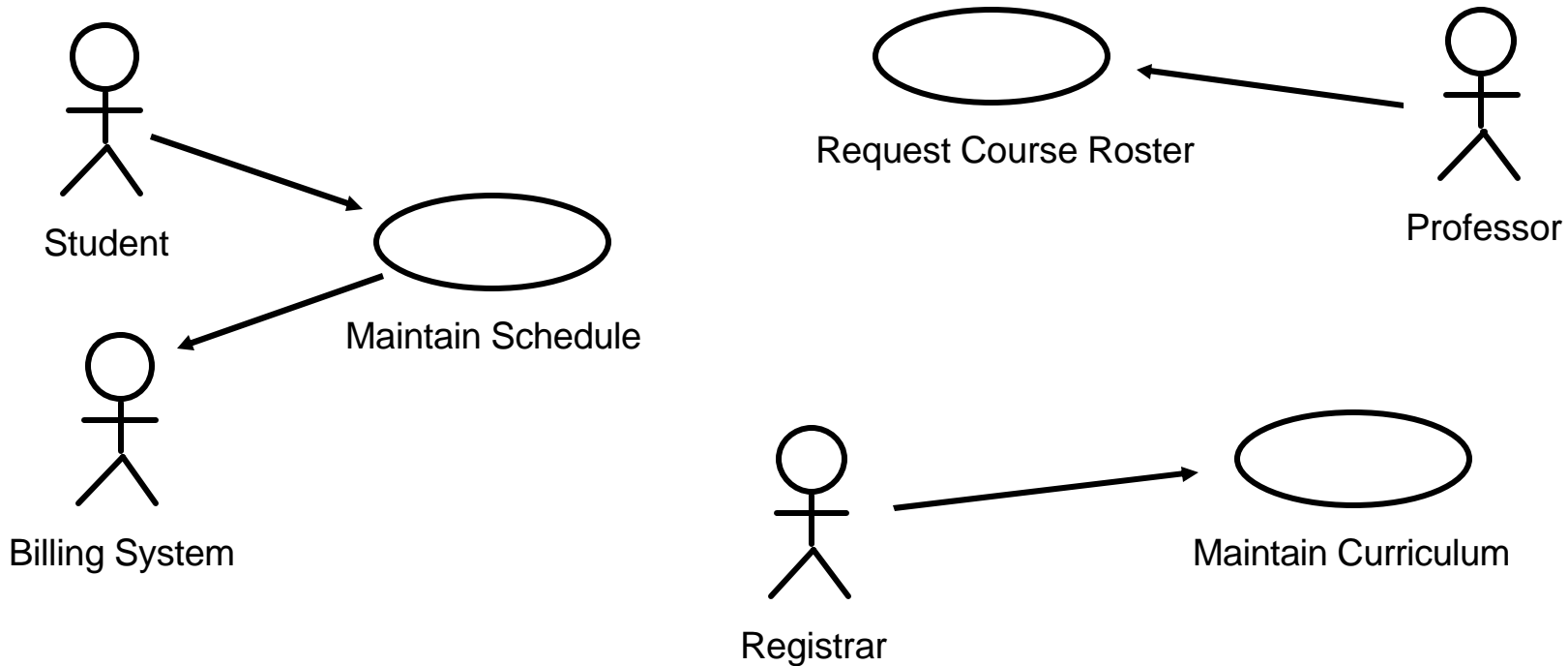
Request Course Roster



Maintain Schedule

Use Case Diagram

- Use case diagrams are created to visualize the relationships between actors and use cases



Use Case Realizations

- The use case diagram presents an outside view of the system
- Interaction diagrams describe how use cases are realized as interactions among societies of objects
- Two types of interaction diagrams
 - Sequence diagrams
 - Collaboration diagrams

Class Diagram

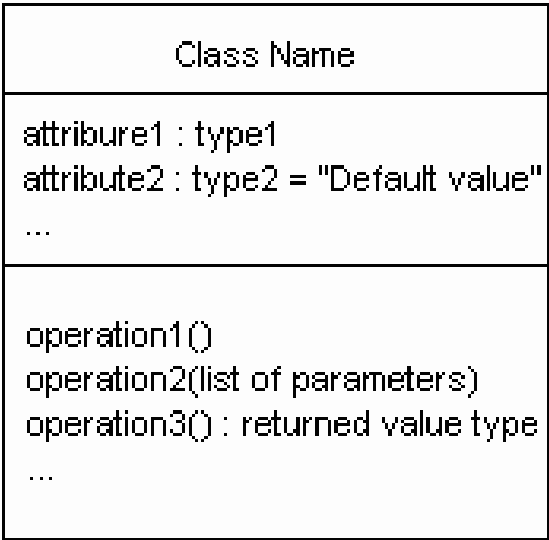
- Class diagrams are the most commonly used diagrams in UML.
- Class diagrams are for visualizing, specifying and documenting the system from a static perspective.
- Class diagrams indicate which classes know about other classes and, if they do, what type of relationship exists.
- Class diagrams will have different levels of detail (abstraction) depending on where we are in the software development process.

Class Diagrams

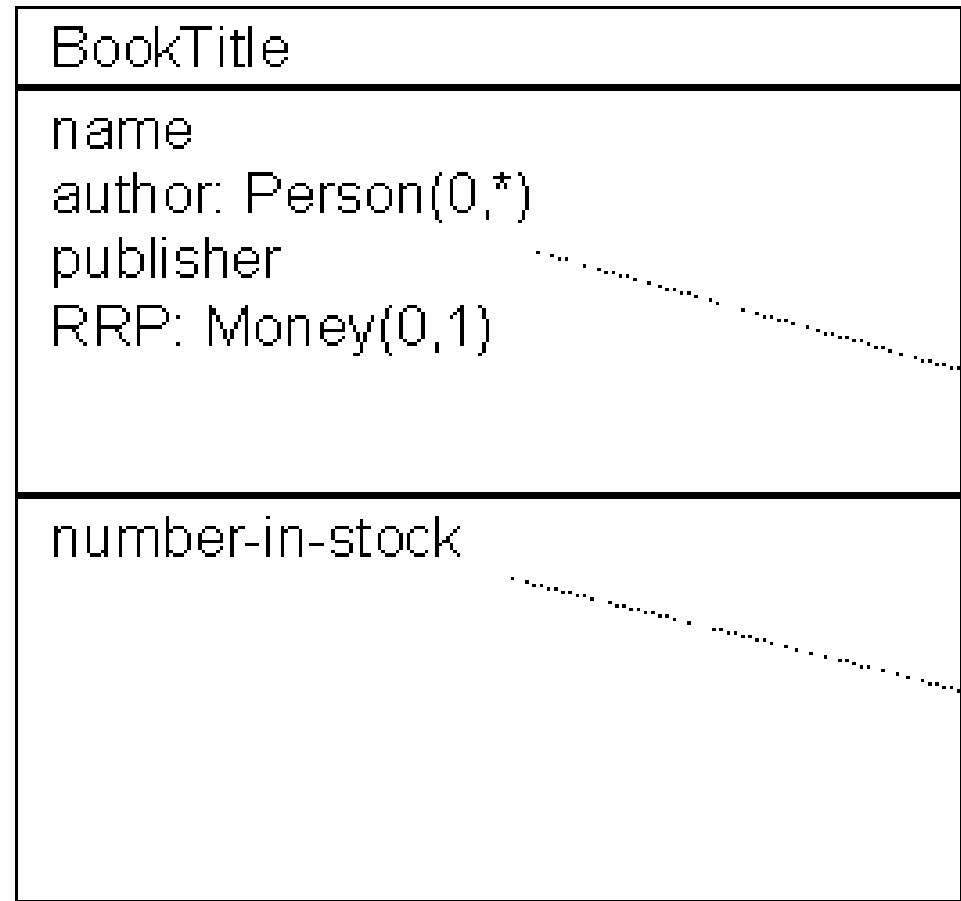
- Class diagrams describe types of objects in a system and their relationships.
- There are three types of relationships between classes:
 - Dependency (“uses”)
 - Aggregation (“has”)
 - Inheritance (“is”)
- Class diagrams also show the attributes and operations of a class.

Class Diagrams

- Class diagrams consist of several classes connected with relationships
- Representation of a class:



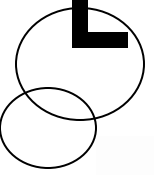
Class Diagram



This is a type.

Attributes & associations

operations

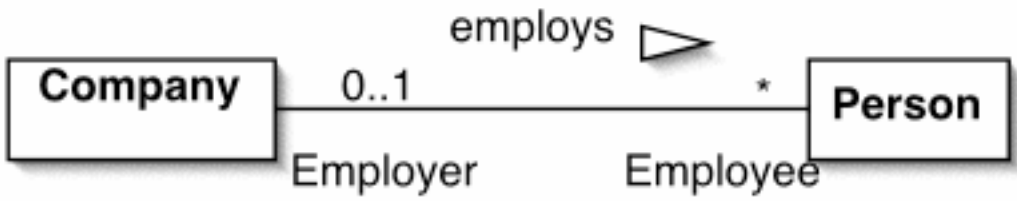


Convention

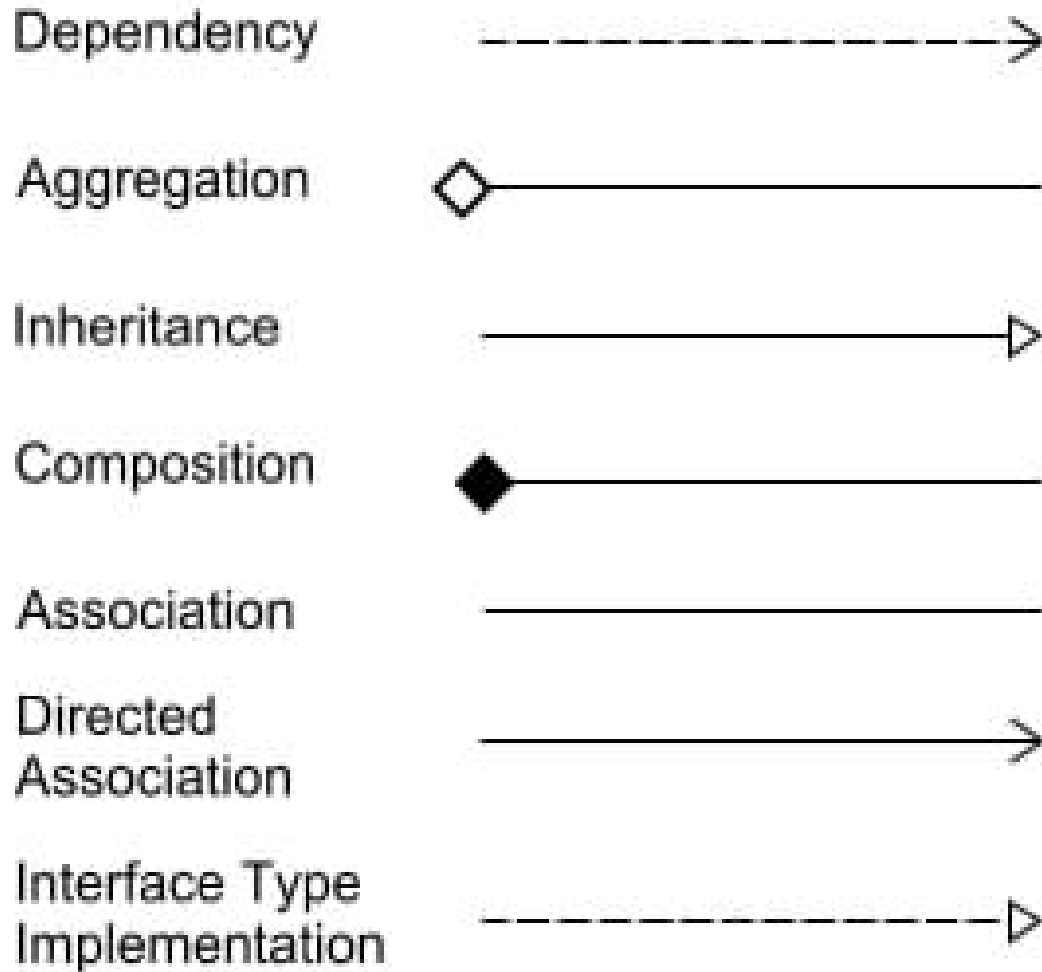
- Name of class is a word with initial upper case letter
- Capitalize first letter of every word in name
- Name of attributes is lower case letter
- Capitalize first letter of every word other than first
- Operations have same naming schemes as attributes
- Brackets include parameter operation works on

Associations

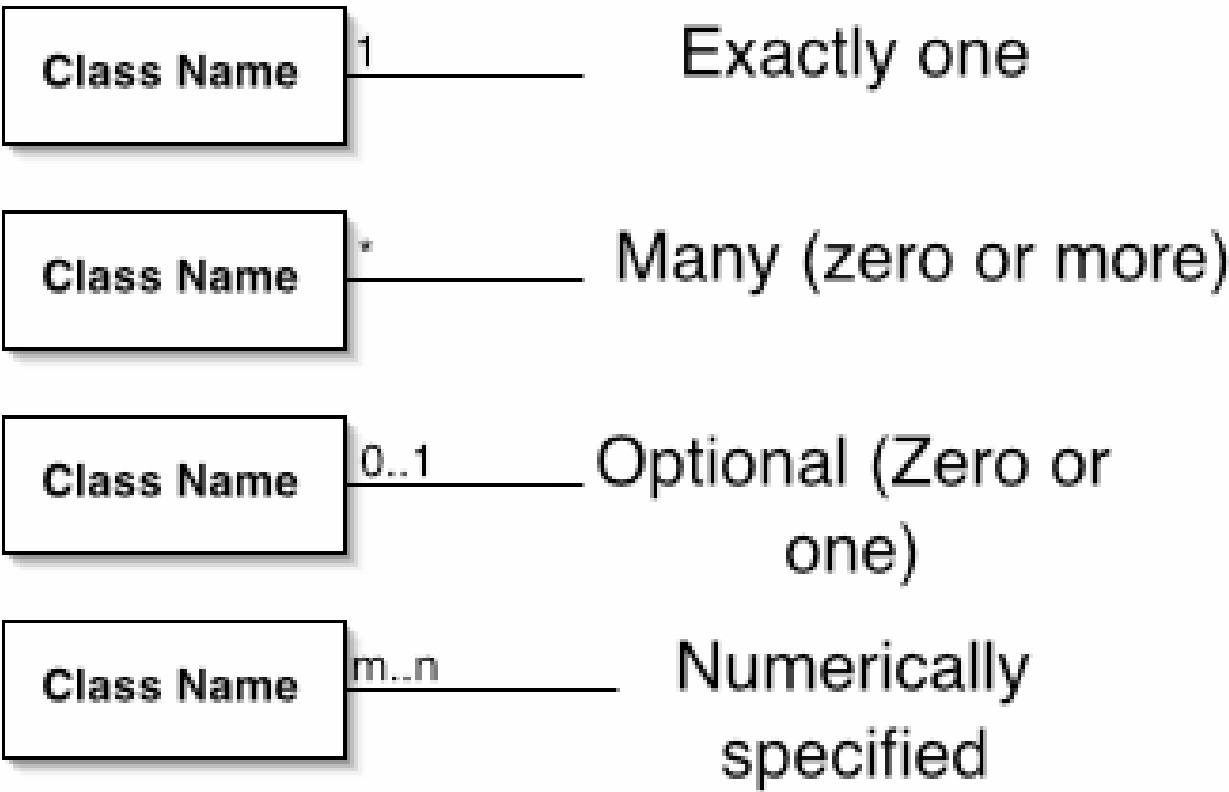
- “relationship between different objects of one or more classes”



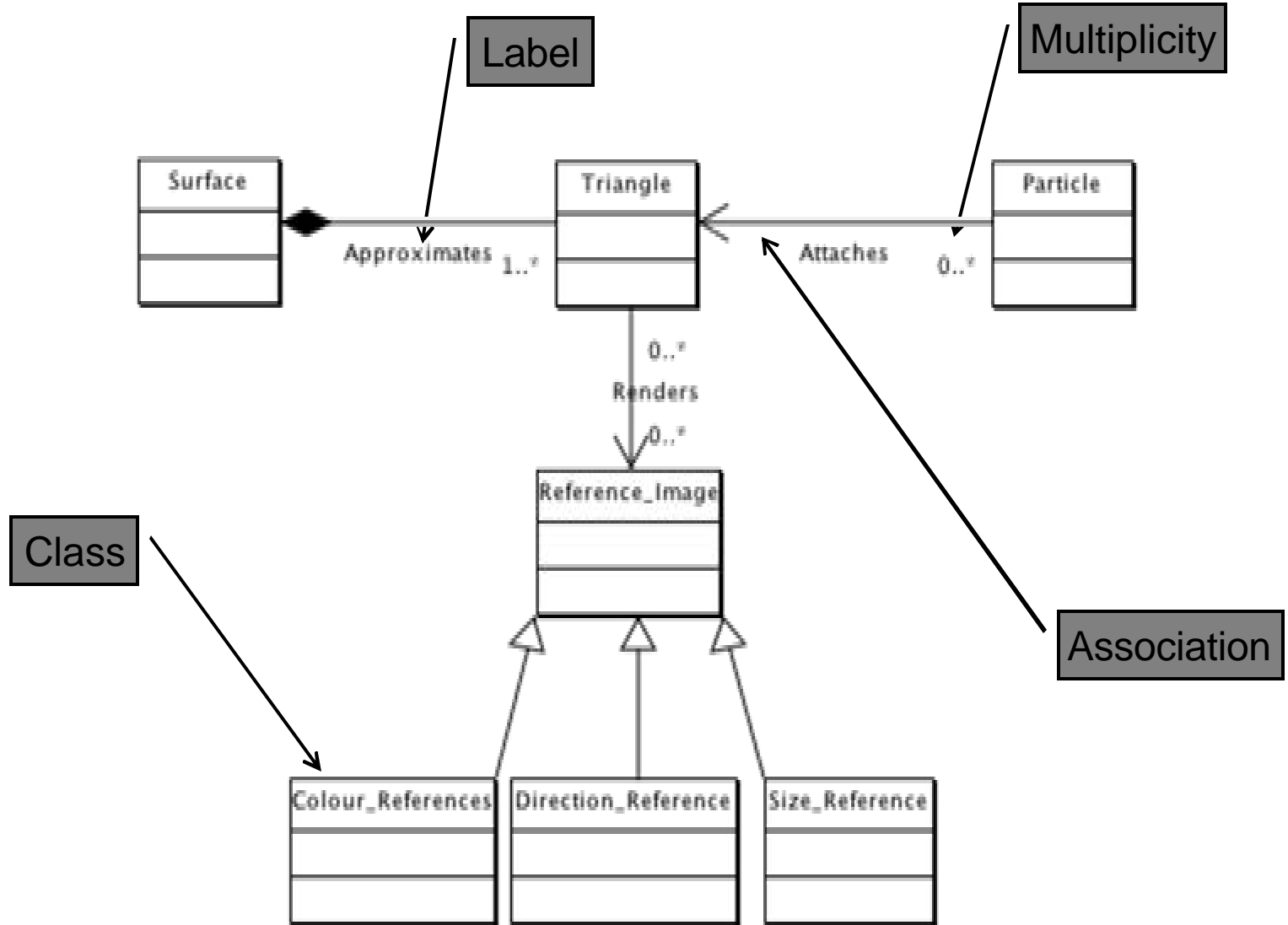
Connectors



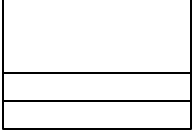
Multiplicities



UML Class Diagram



Class

Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	

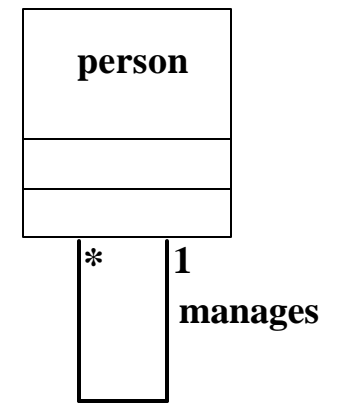
- Derived from the CRC Cards
- Sections: name, attributes, operations

Name
Attributes
Methods

Association

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances.	_____

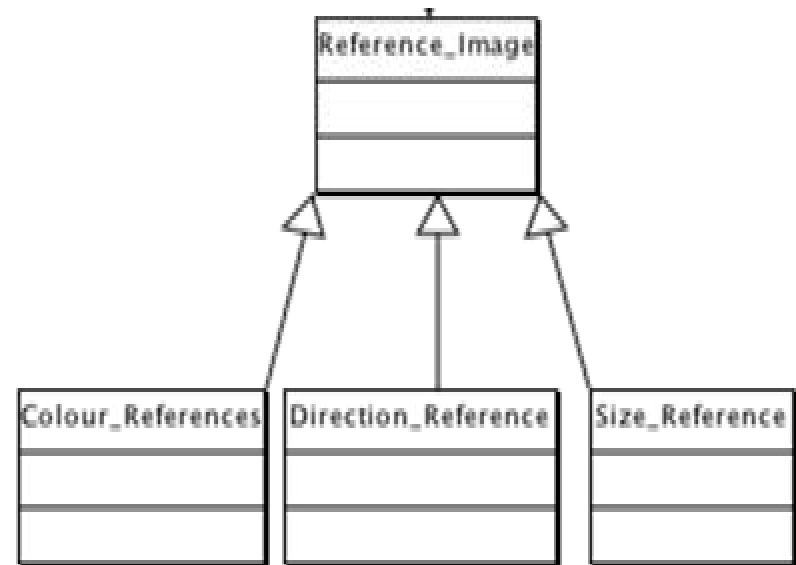
- It is more important to identify classes than to identify associations
- Too many associations can be confusing
- Classes may have association with themselves




Generalization

Construct	Description	Syntax
generalization	a taxonomic relationship between a more general and a more specific element.	→

- Generalization: identifying commonality among classes



Aggregation

Construct	Description	Syntax
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	

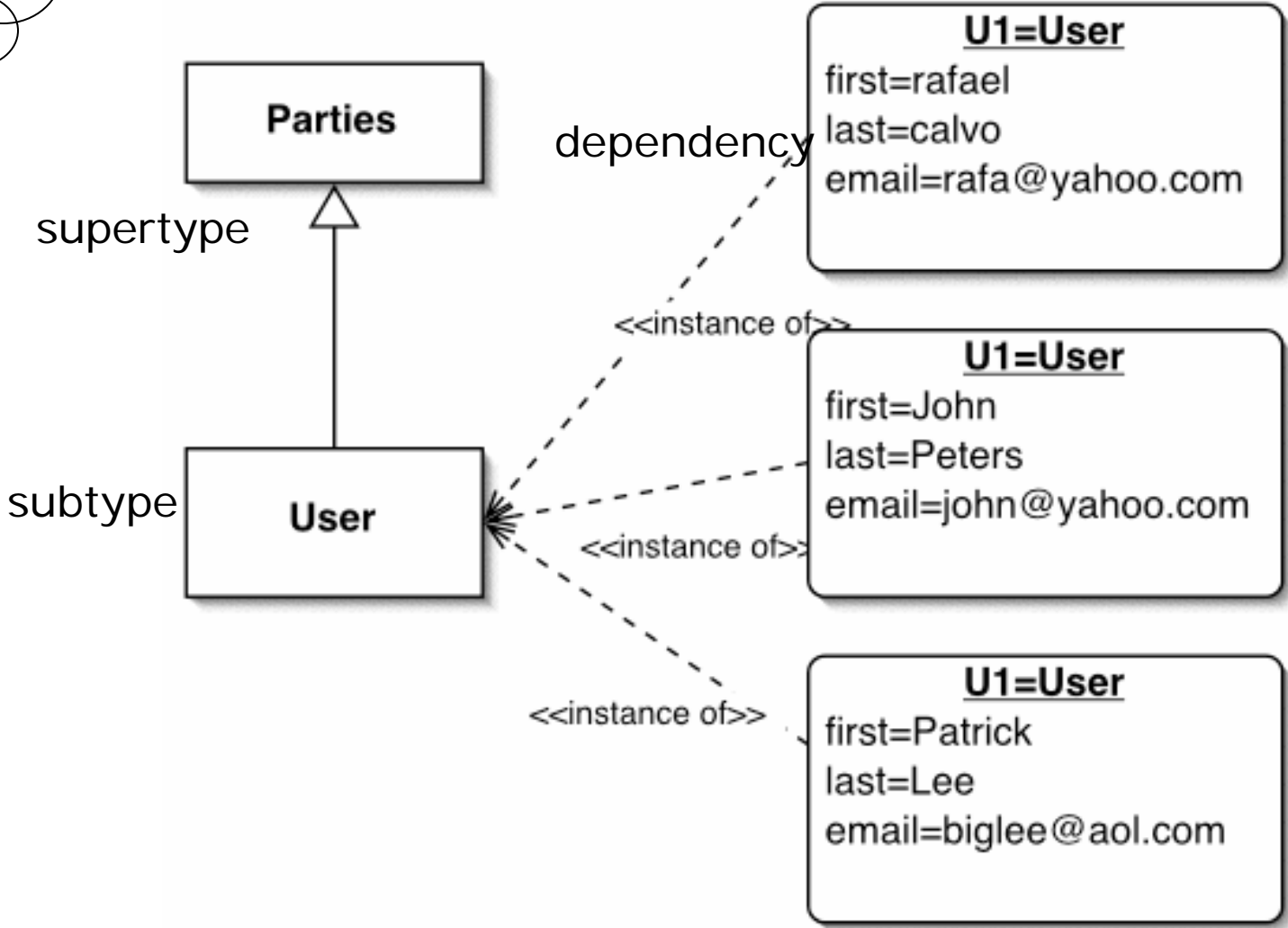
- A composite aggregation (filled diamond) means that the multiplicity at the composite end may be at most one.
- Shared aggregation (hollow diamond) means that the multiplicity at the composite end may be more than one.

Aggregation Guidelines

- There is an obvious physical or logical assembly
- The part and composite have the same lifetime
- Operations applied to the composite propagate to the parts e.g. destruction, movement, etc.



Objects and Inheritance



Classifier

- A *classifier* is a mechanism that describes structural and behavioral features.
- Types of classifiers are ...
 - classes, interfaces, datatypes, signals, components, nodes, use cases and subsystems.
- Classes are the most important kind of classifier.
 - Classes have a number of features beyond attributes and behaviors that allow you to model some of the more subtle/advanced features of a system.

Advanced Class Features

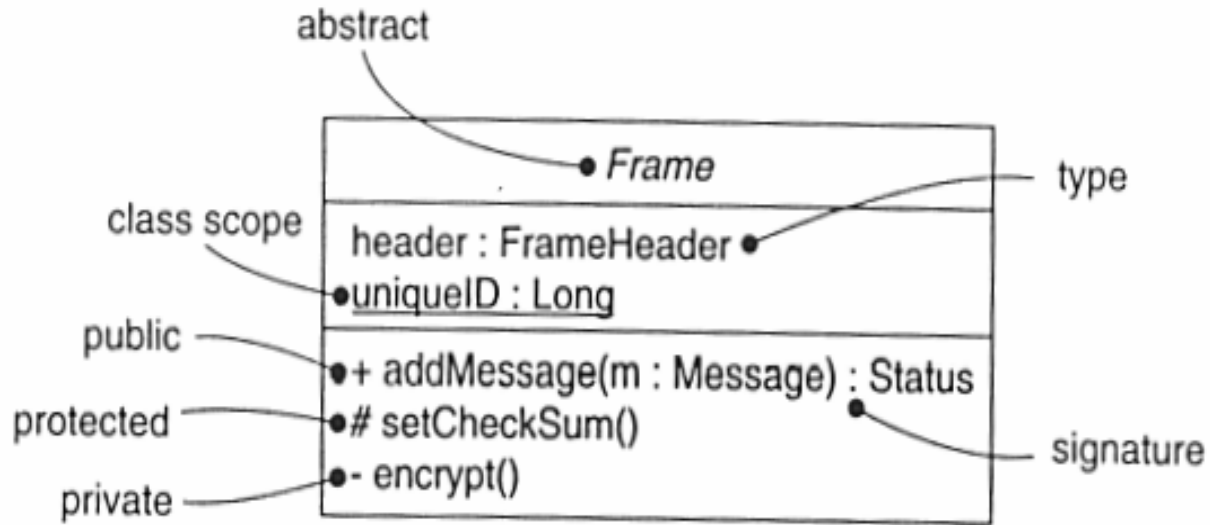


Figure 9-1: Advanced Classes

Abstract and Concrete Classes and Operations

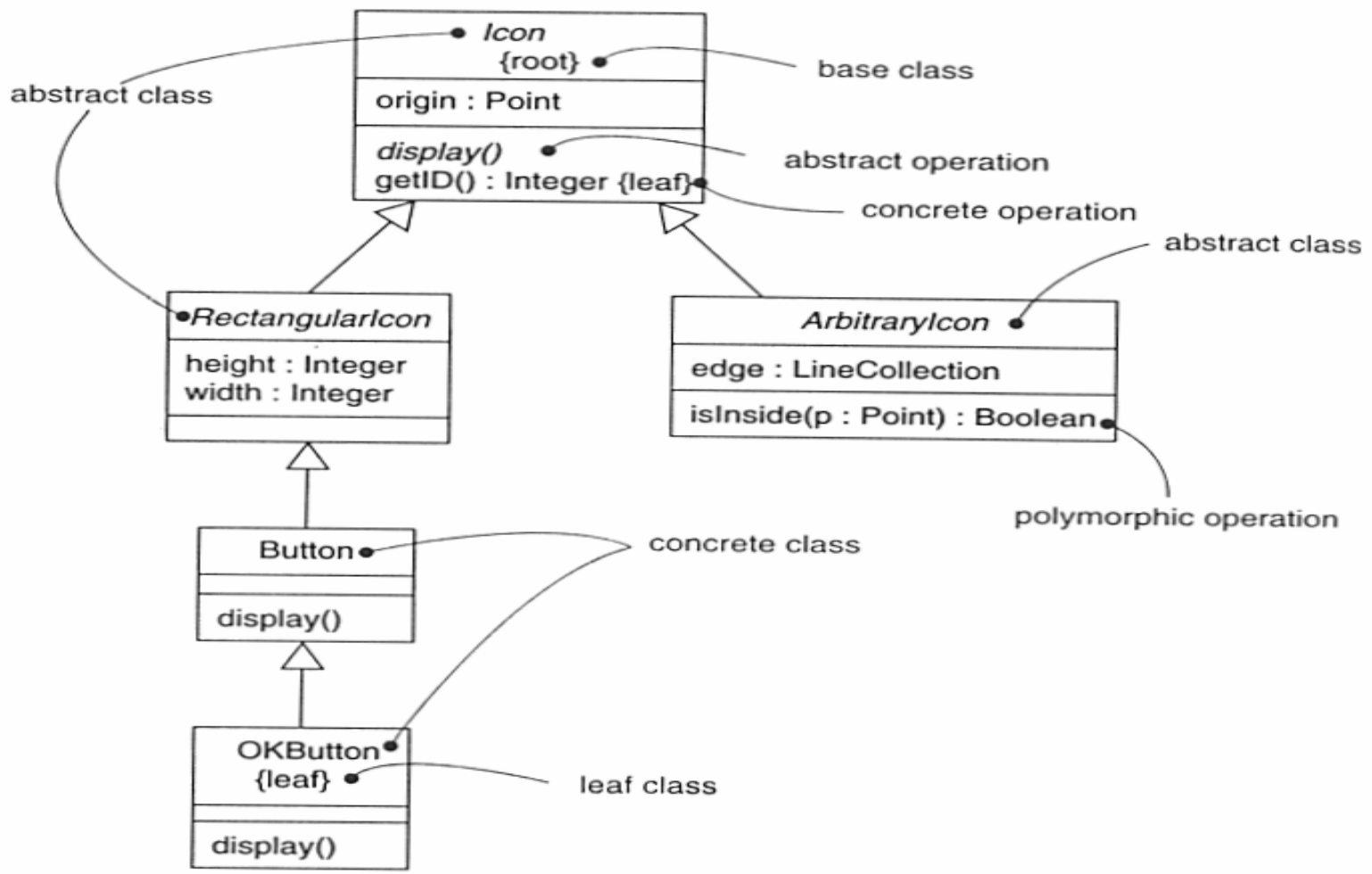


Figure 9-5: Abstract and Concrete Classes and Operations

Template Icon

- A parameterized element.
- Represented in UML as a dashed box in the upper right-hand corner of the class icon, which lists the template parameters.

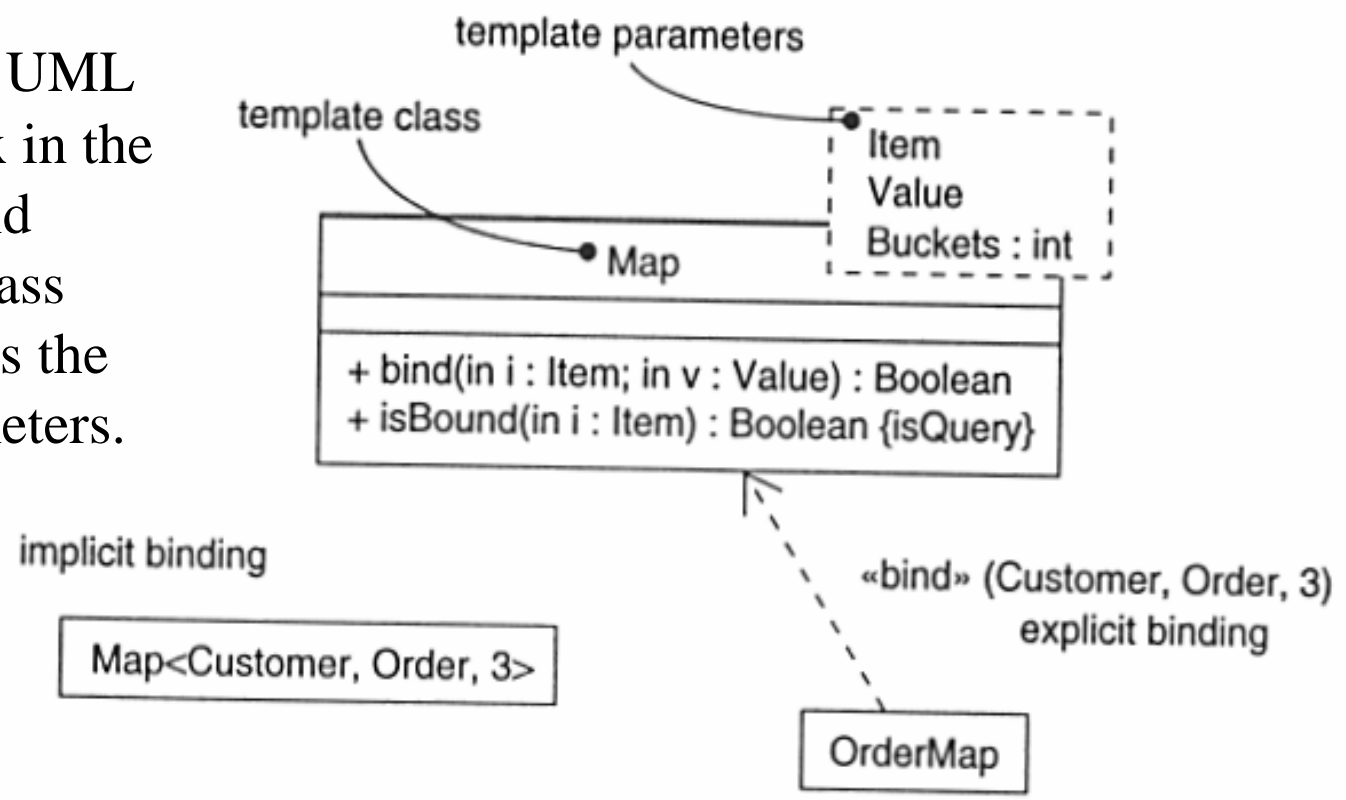
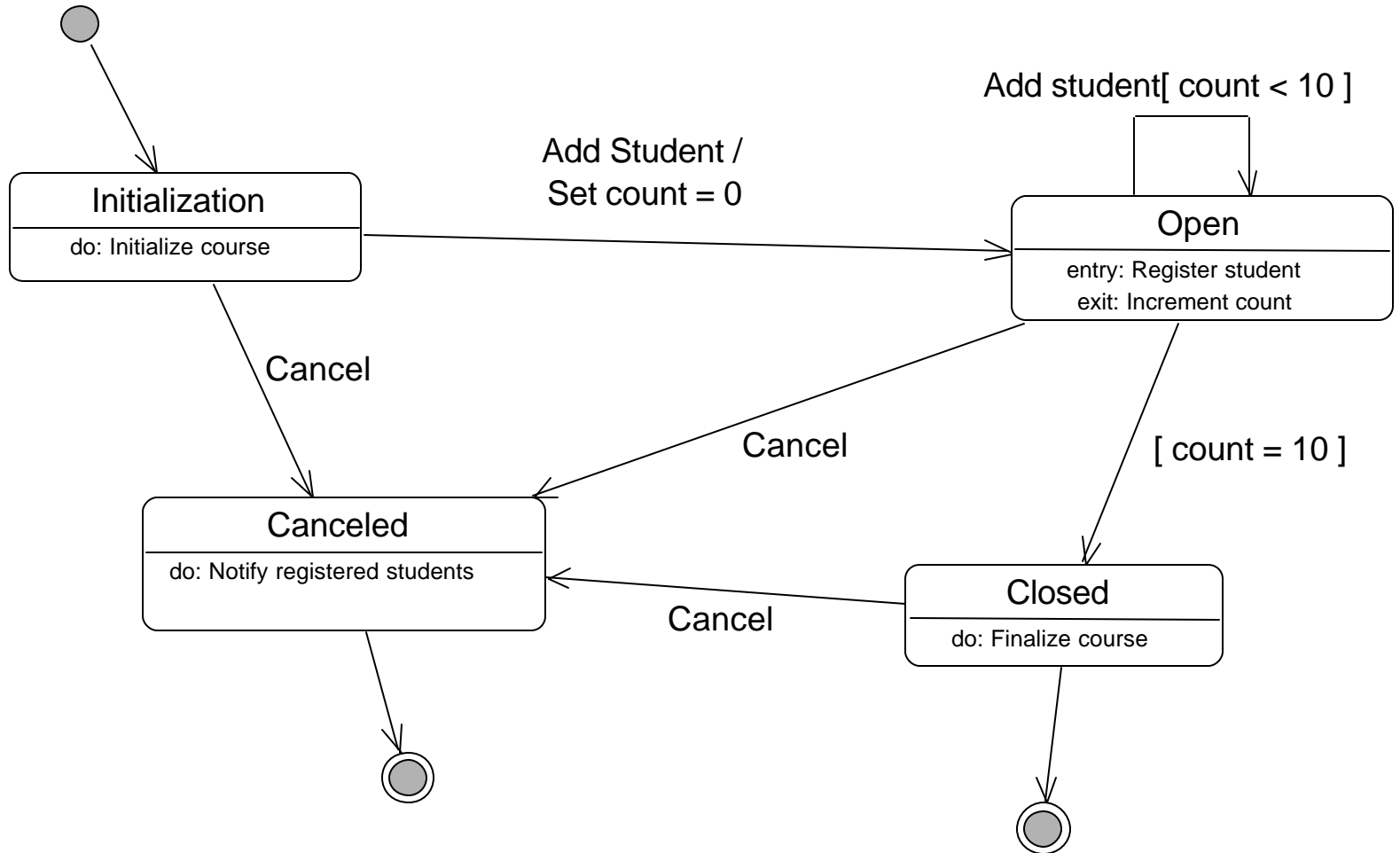


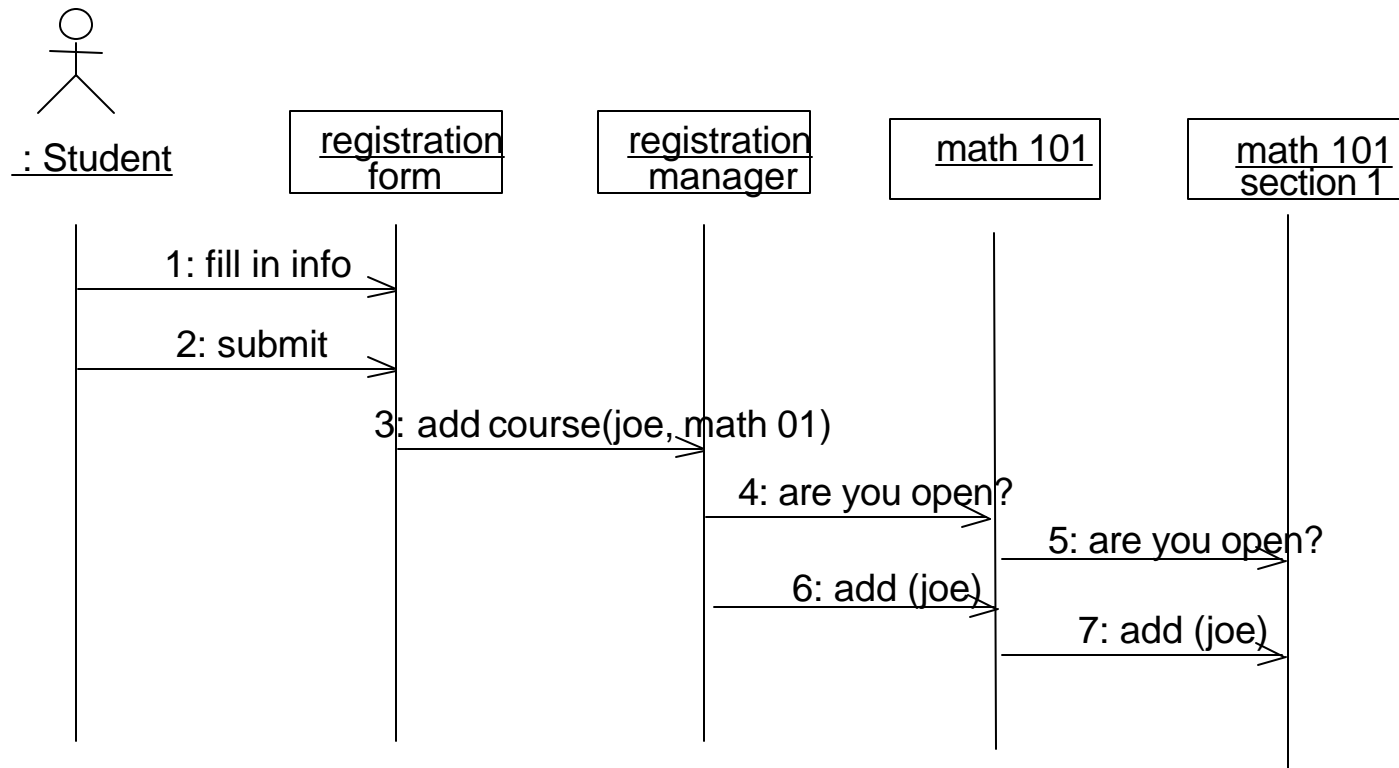
Figure 9-7: Template Classes

State Transition Diagram



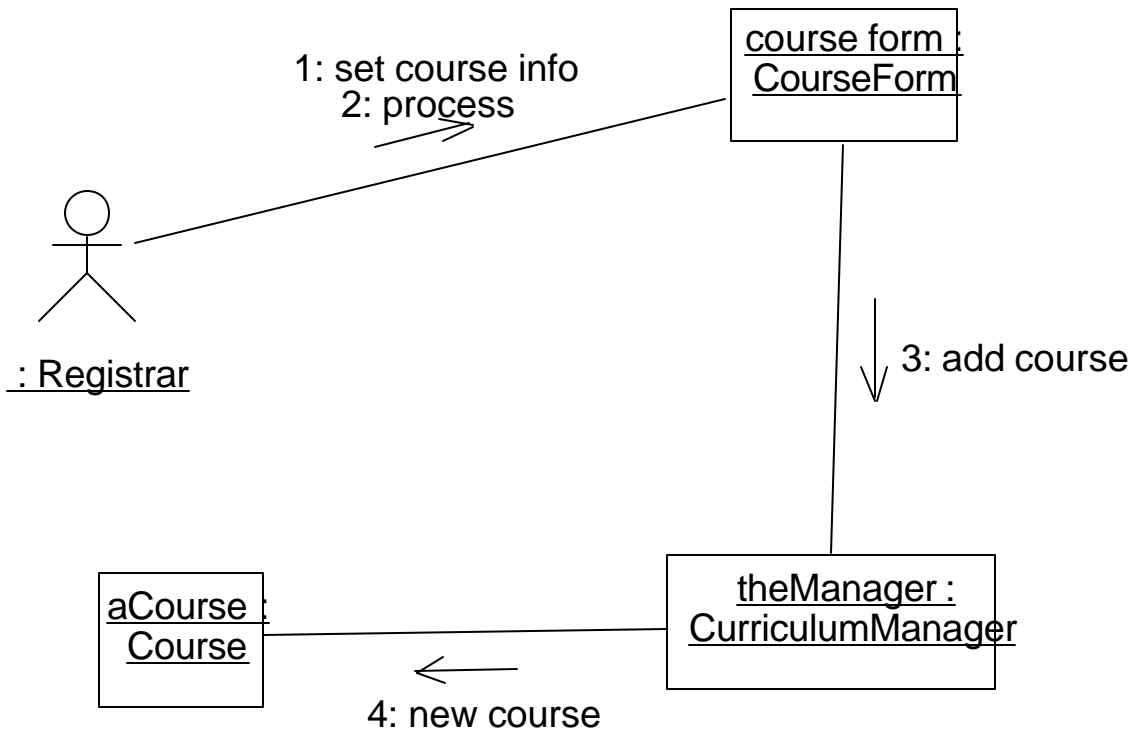
Sequence Diagram

- A sequence diagram displays object interactions arranged in a time sequence



Collaboration Diagram

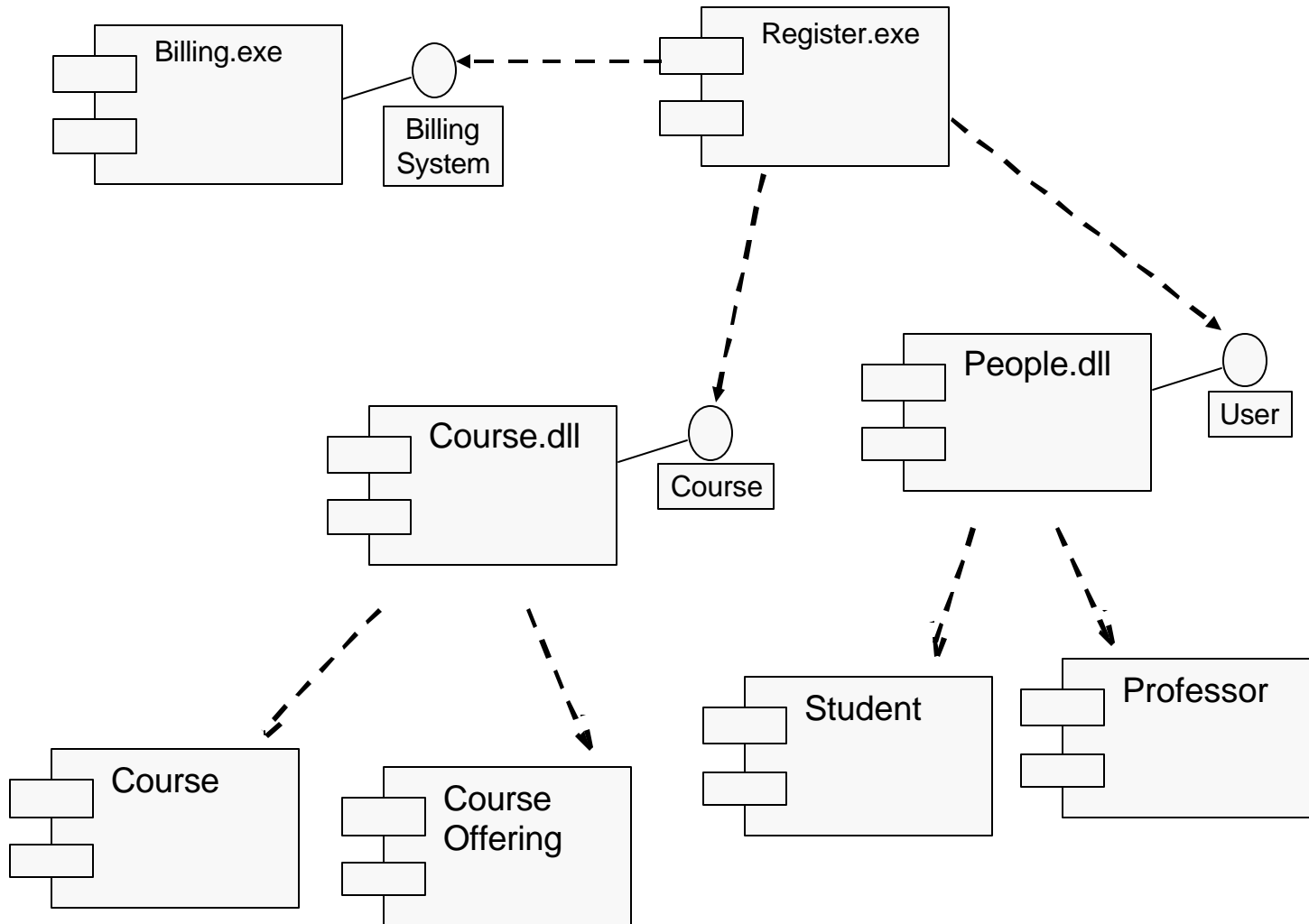
- A collaboration diagram displays object interactions organized around objects and their links to one another



The Physical World

- Component diagrams illustrate the organizations and dependencies among software components
- A component may be
 - A source code component
 - A run time components or
 - An executable component

Component Diagram



Deploying the System

- The deployment diagram shows the configuration of run-time processing elements and the software processes living on them
- The deployment diagram visualizes the distribution of components across the enterprise.

Deployment Diagram

