Unified Modeling Language (UML) Part 3

Dr. Krishnendu Guha

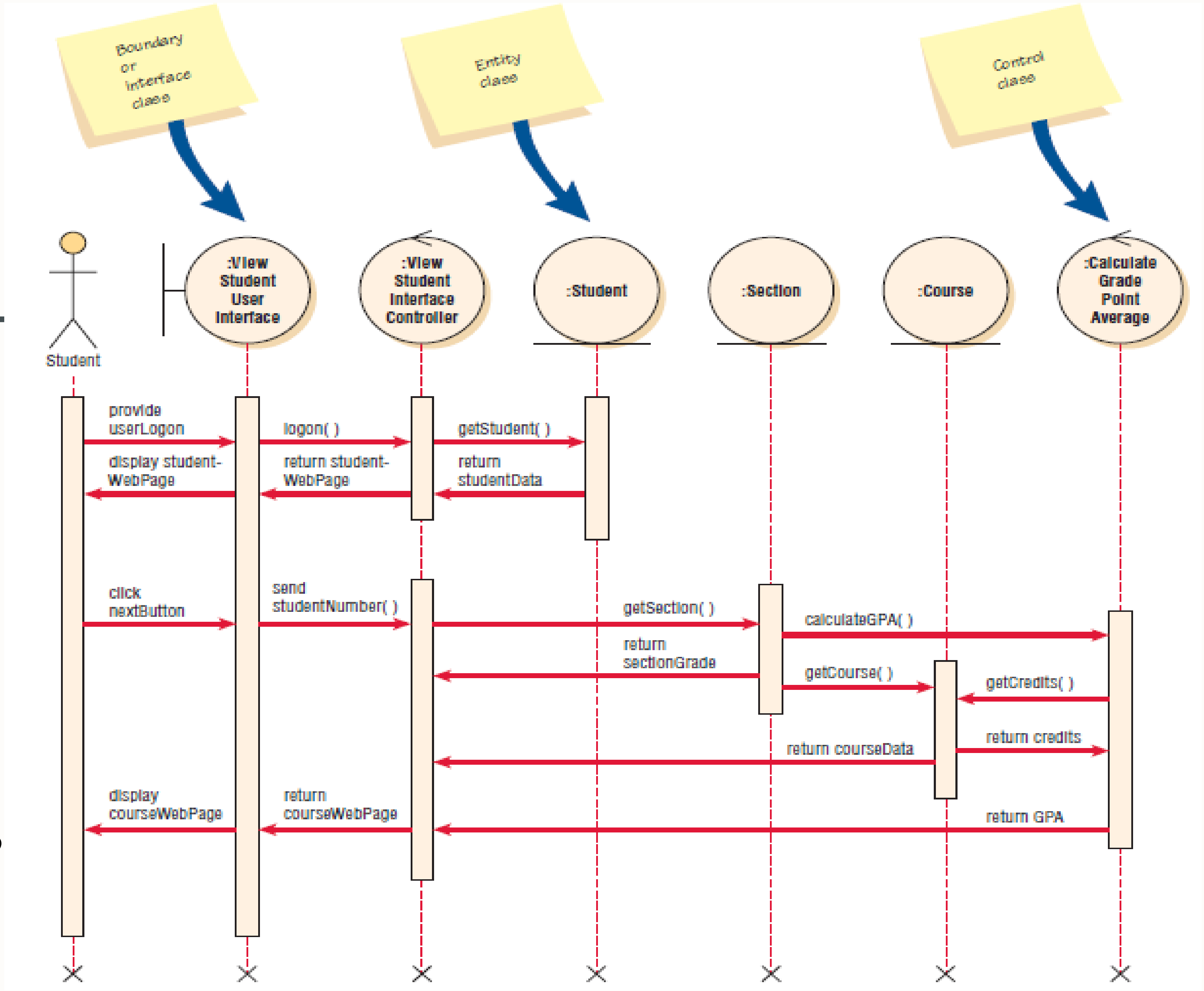
Assistant Professor
(On Contract)

National Institute of Technology (NIT), Jamshedpur

Email: krishnendu.ca@nitjsr.ac.in

Enhancing sequence diagrams

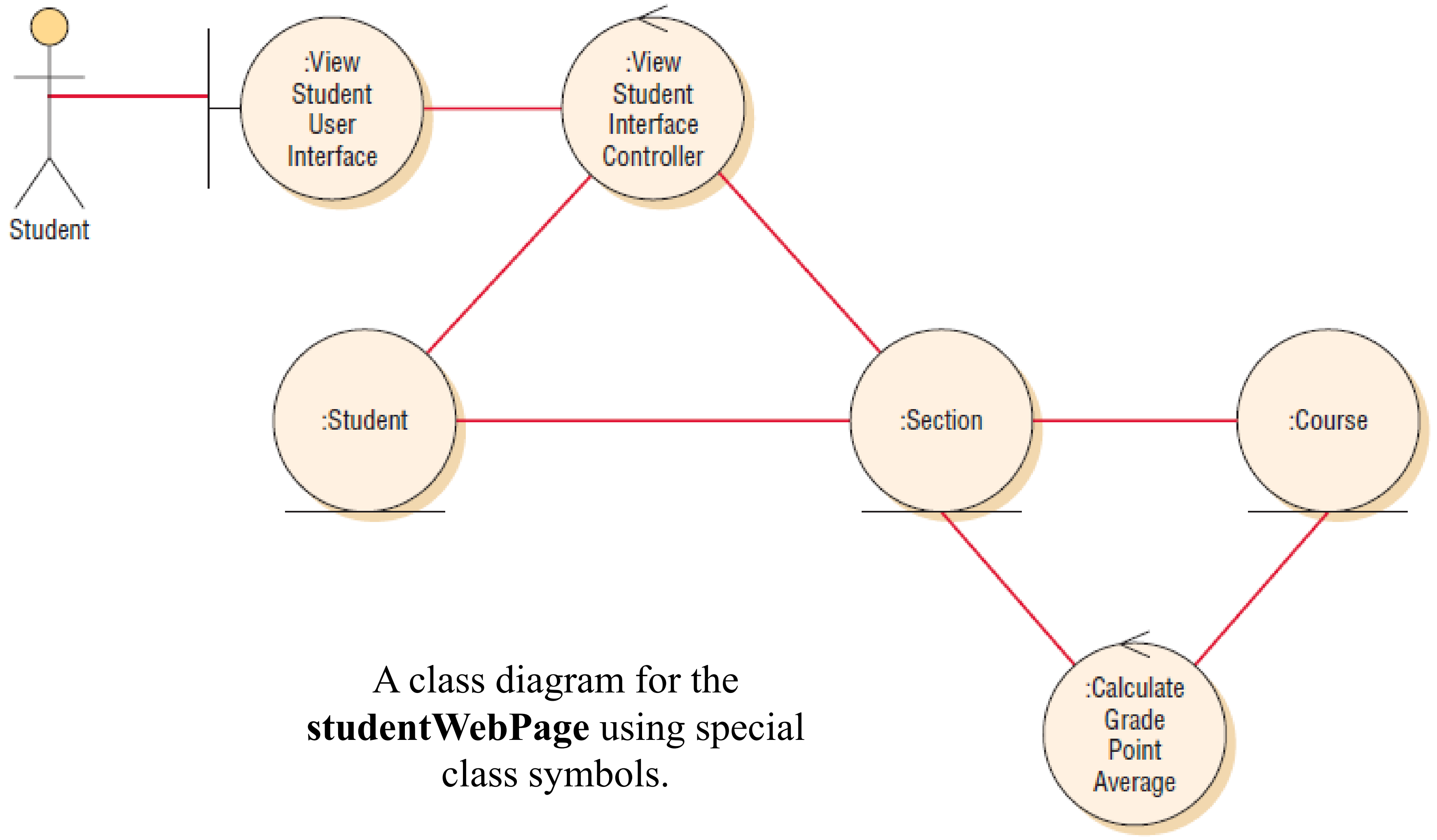
- Once the class diagram is drawn, it may be desirable to go back to the sequence diagram and include special symbols for each of the different types of classes introduced
- The following steps are a useful approach to enhancing a sequence diagram:
- **1.** Include the *actor* from the use case diagram in the enhanced sequence diagram.
- **2.** Define one or more *interface classes* for each actor.
- **3.** Create prototype Web pages for all human interfaces.
- **4.** Ensure each use case has one *control class*, although more may be created during the detailed design.
- **5.** Examine the use case to see what *entity classes* are present. Include these on the diagram.
- **6.** Realize that the sequence diagram may be modified again when doing detailed design
- **7.** To obtain a greater degree of reuse, consider moving methods from a control class to an entity class.



A sequence diagram for using two Web pages: one for student information, one for course information.

Enhancing class diagrams

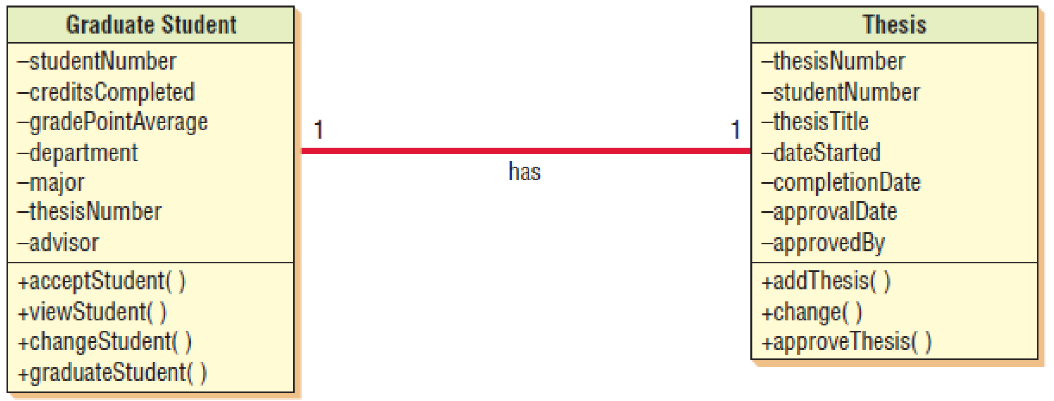
- The class symbols also may be used on class and communication diagrams.
- If the class is a user interface type of class, the attributes are the controls (or fields) on the screen or form.
- The methods would be those that work with the screen, such as submit or reset.
- They might also be JavaScript for a Web page, because the code works directly with the Web page.
- If the class is a control class, the attributes would be those needed to implement the class, such as variables used just in the control class.
- The methods would be those used to perform calculations, make decisions, and send messages to other classes.
- If the class is an entity class, the attributes represent those stored for the entity and the methods working directly with the entity, such as creating a new instance, modifying, deleting, obtaining, or printing.

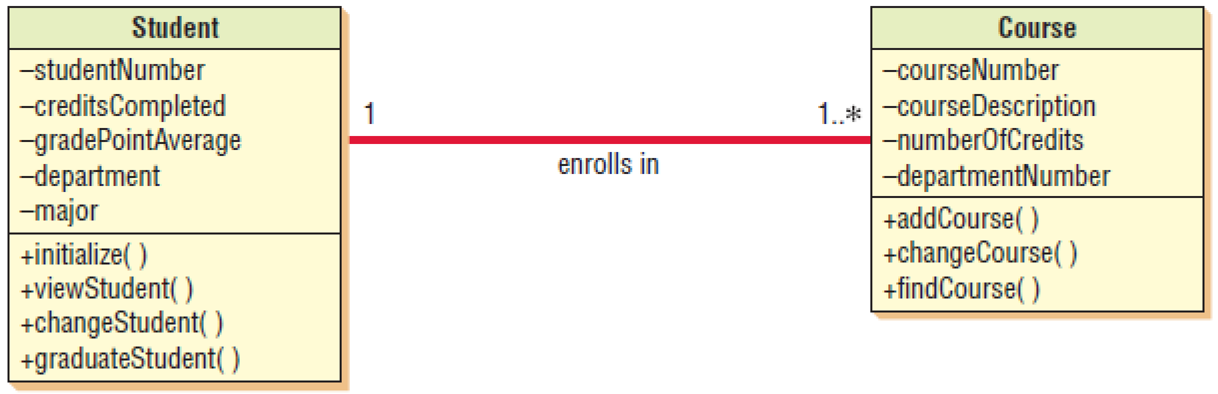

Student
:View Student User Interface
:View Student Interface Controller
:Student
:Section
:Course
:Calculate Grade Point Average

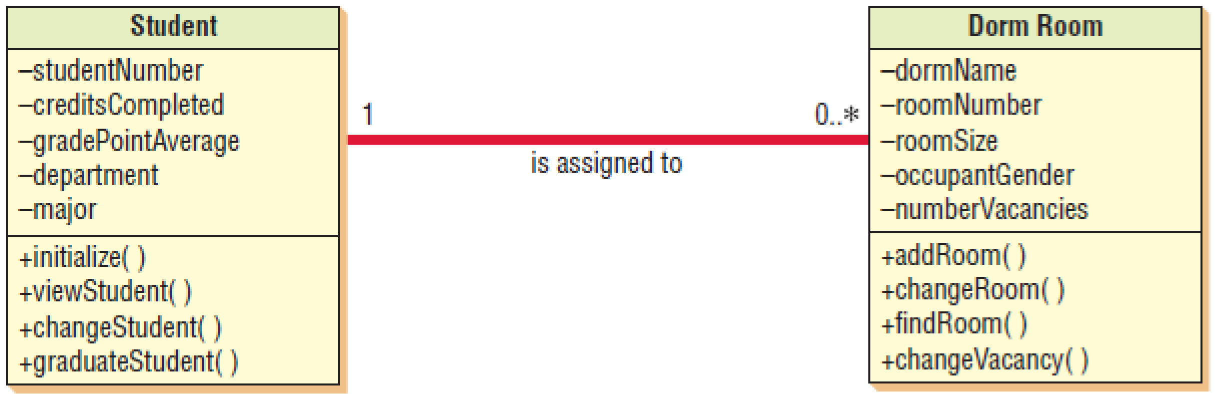
A class diagram for the **studentWebPage** using special class symbols.

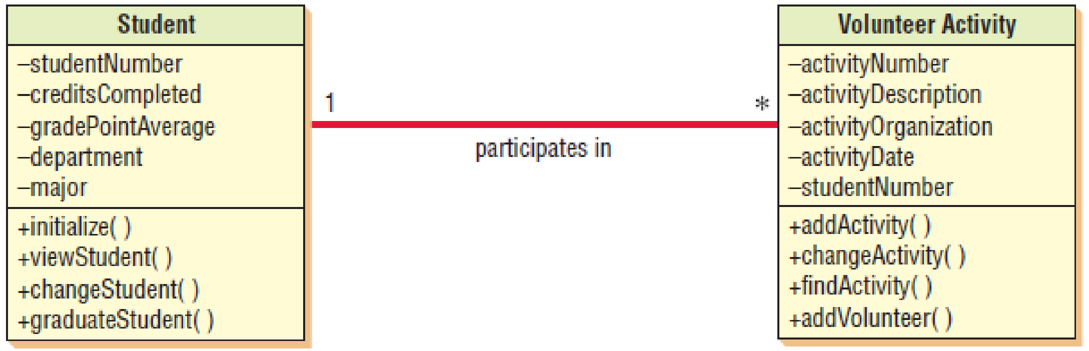
Relationships

- Another way to enhance class diagrams is to show relationships.
- Relationships are connections between classes, similar to those found on an entity-relationship diagram. These are shown as lines connecting classes on a class diagram.
- There are two categories of relationships: associations and whole/p
- **ASSOCIATIONS.** The simplest type of relationship is an association, or a structural connection between classes or objects. Associations are shown as a simple line on a class diagram.
- The end points of the line are labeled with a symbol indicating the multiplicity, which is the same as cardinality on an entity-relationship diagram.
- A zero represents none, a one represents one and only one, and an asterisk represents many.
- The notation 0..1 represents from zero to one, and the notation 1..* represents from one to many.art relationships.



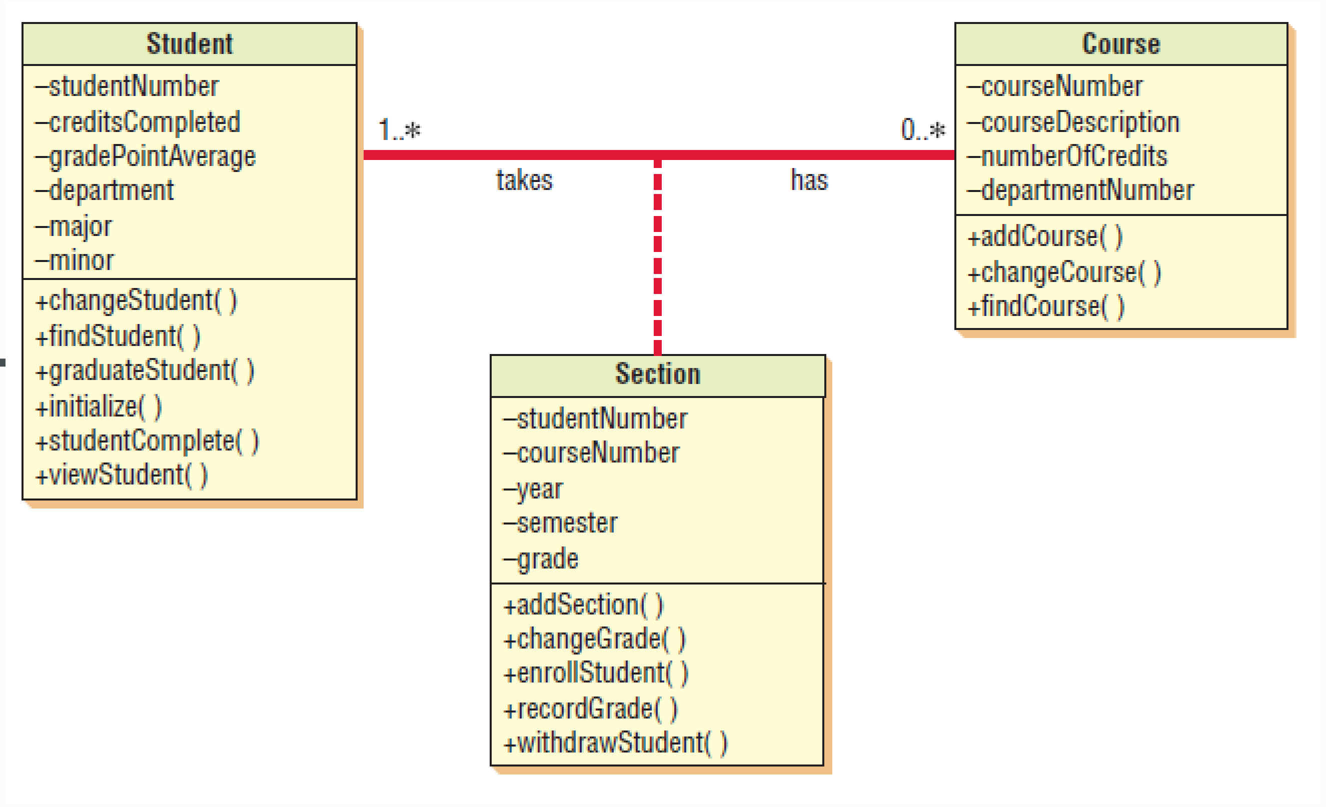






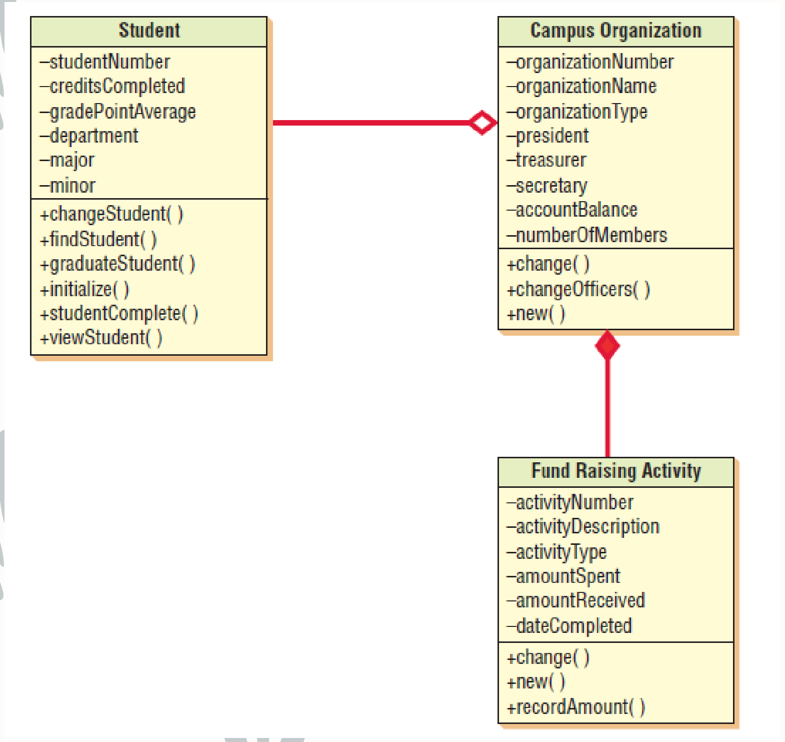
Types of associations that may occur in class diagrams

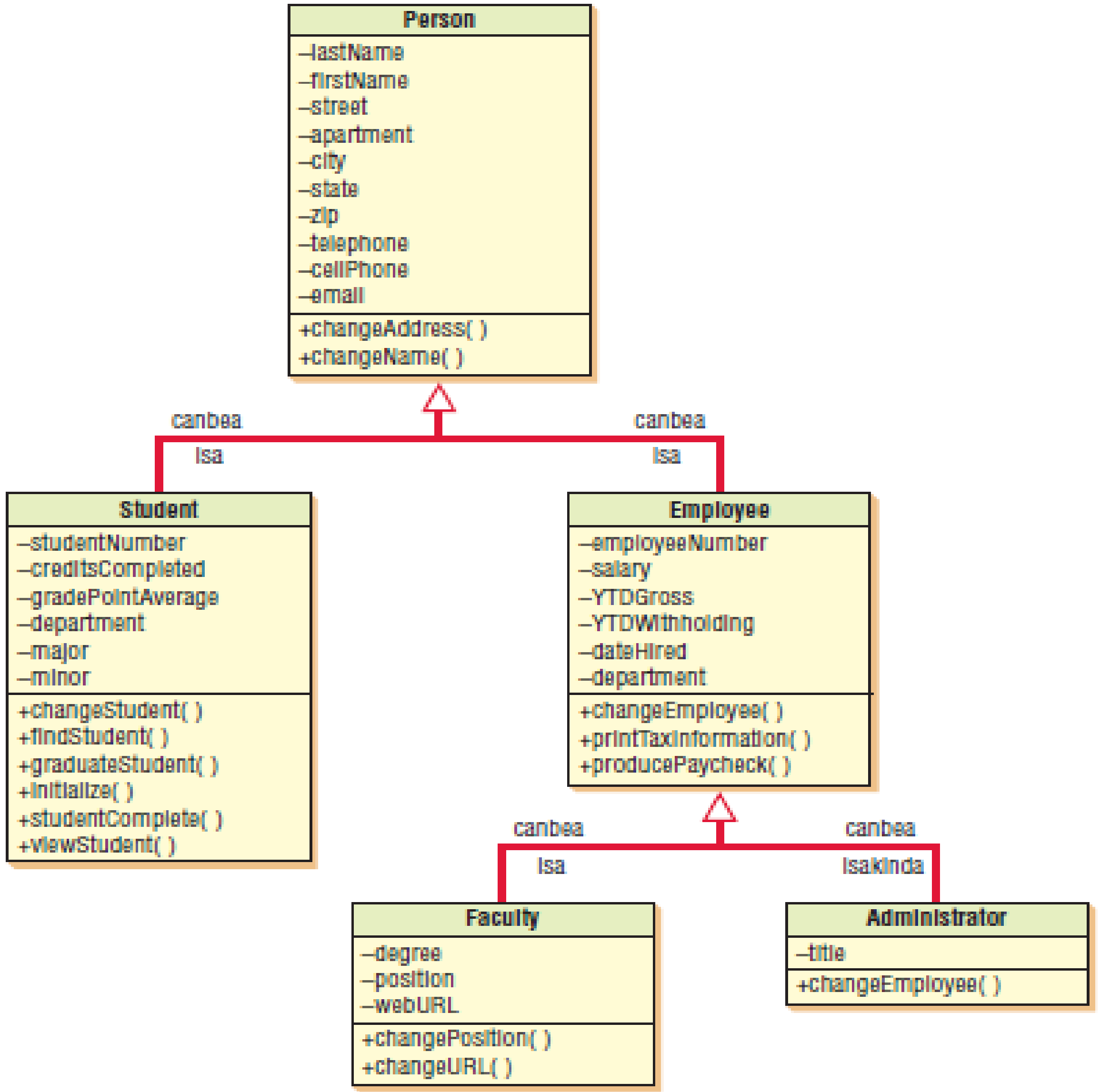
- Association classes are those that are used to break up a many-to-many association between classes.
- These are similar to associative entities on an entity-relationship diagram. **Student** and **Course** have a many-to-many relationship, which is resolved by adding an association class called **Section** between the classes of **Student** and **Course.**
- An object in a class may have a relationship to other objects in the same class, called a reflexive association



An example of an associative class in which a particular section defines the relationship between a student and a course

- **WHOLE/PART RELATIONSHIPS.** Whole/part relationships are when one class represents the whole object and other classes represent parts.
- The whole acts as a container for the parts.
- These relationships are shown on a class diagram by a line with a diamond on one end.
- The diamond is connected to the object that is the whole.





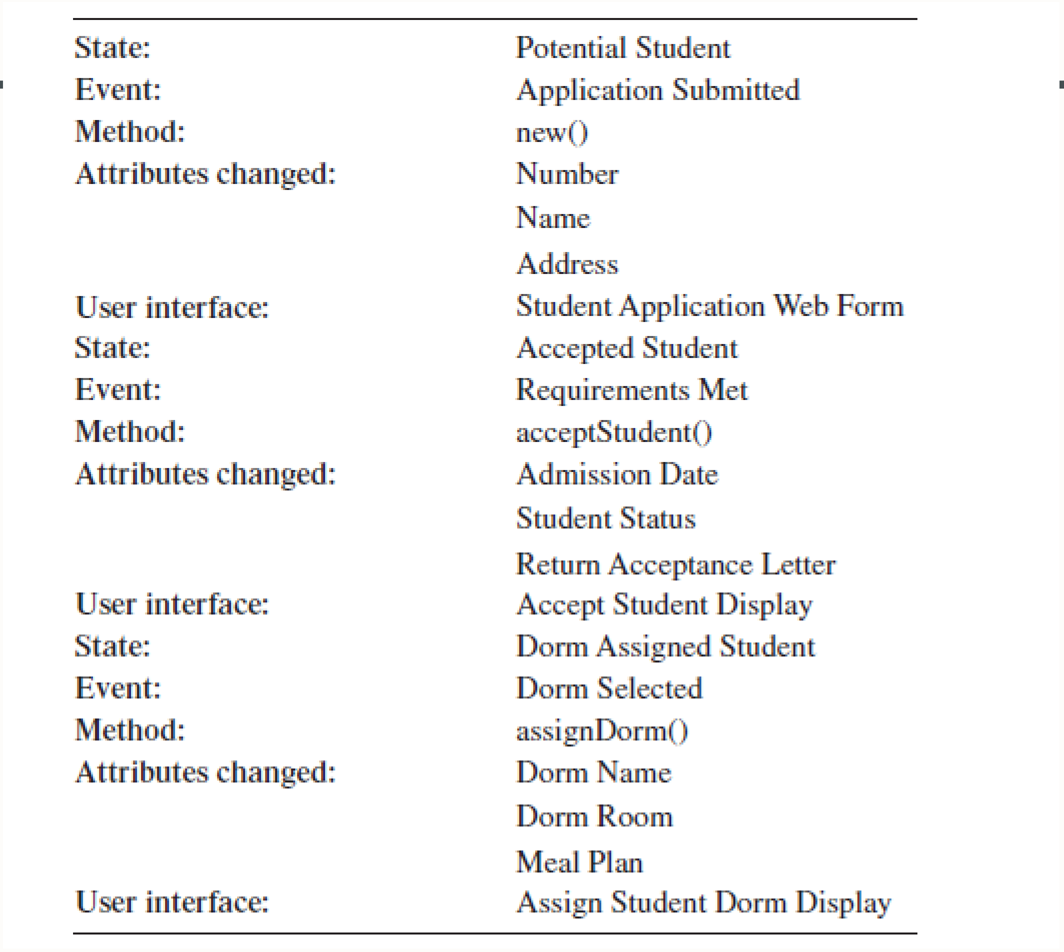
A gen/spec diagram is a refined form of a class diagram.

Statechart Diagrams

- The statechart, or state transition, diagram is another way to determine class methods.
- It is used to examine the different states that an object may have.
- A statechart diagram is created for a single class.
- Typically objects are created, go through changes, and are deleted or removed.
- Objects exist in these various states, which are the conditions of an object at a specific time.
- An object’s attribute values define the state that the object is in, and sometimes there is an attribute, such as Order Status (pending, picking, packaged, shipped, received, and so on) that indicates the state.
- A state has a name with each word capitalized.
- A state also has entry and exit actions, the things the object must do every time it enters or leaves a given state.

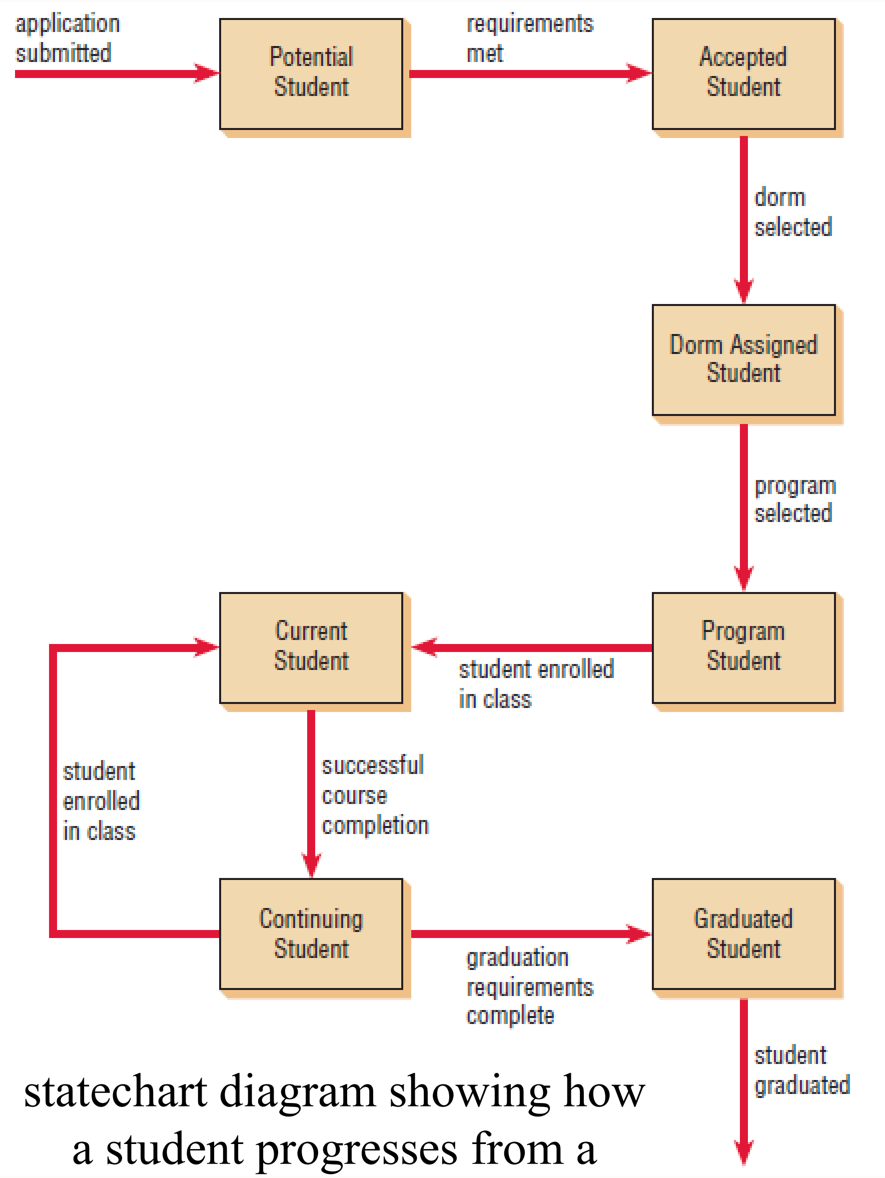
- An event is something that happens at a specific time and place.
- Events cause a change of the object state, and it is said that a transition “fires.”
- States separate events, such as an order that is waiting to be filled, and events separate states, such as an Order Received event or an Order Complete event.
- An event causes the transition, and happens when a guard condition has been met.
- A guard condition is something that evaluates to either true or false, and may be as simple as “Click to confirm order.” It also may be a condition that occurs in a method, such as an item that is out of stock.
- Guard conditions are shown in square brackets next to the event label.
- There are also deferred events, or events that are held until an object changes to a state that can accept them. A user keying something in when a word processor is performing a timed backup is an example of a deferred event. After the timed backup has completed, the text appears in the document.

- **A State Transition Example**
- Consider a student enrolling at a university and the various states that he or she would go through.
- Three of the states are



State:	Potential Student
Event:	Application Submitted
Method:	new()
Attributes changed:	Number
	Name
	Address
User interface:	Student Application Web Form
State:	Accepted Student
Event:	Requirements Met
Method:	acceptStudent()
Attributes changed:	Admission Date
	Student Status
	Return Acceptance Letter
User interface:	Accept Student Display
State:	Dorm Assigned Student
Event:	Dorm Selected
Method:	assignDorm()
Attributes changed:	Dorm Name
	Dorm Room
	Meal Plan
User interface:	Assign Student Dorm Display

- States are represented by rectangles, and events or activities are the arrows that link the states and cause one state to change to another state.
- Transition events are named in the past tense, because they have already occurred to create the transition.
- Statechart diagrams are not created for all classes. They are created when:
- **1.** A class has a complex life cycle.
- **2.** An instance of a class may update its attributes in a number of ways through the life cycle.
- **3.** A class has an operational life cycle.
- **4.** Two classes depend on each other.
- **5.** The object's current behavior depends on what happened previously.
- Each state should have at least one transition in and out of it.
- Some statechart diagrams use the same start and terminator symbols that an activity diagram uses: a filled-in circle to represent the start, and concentric circles with the center filled in to signify the end of the diagram.

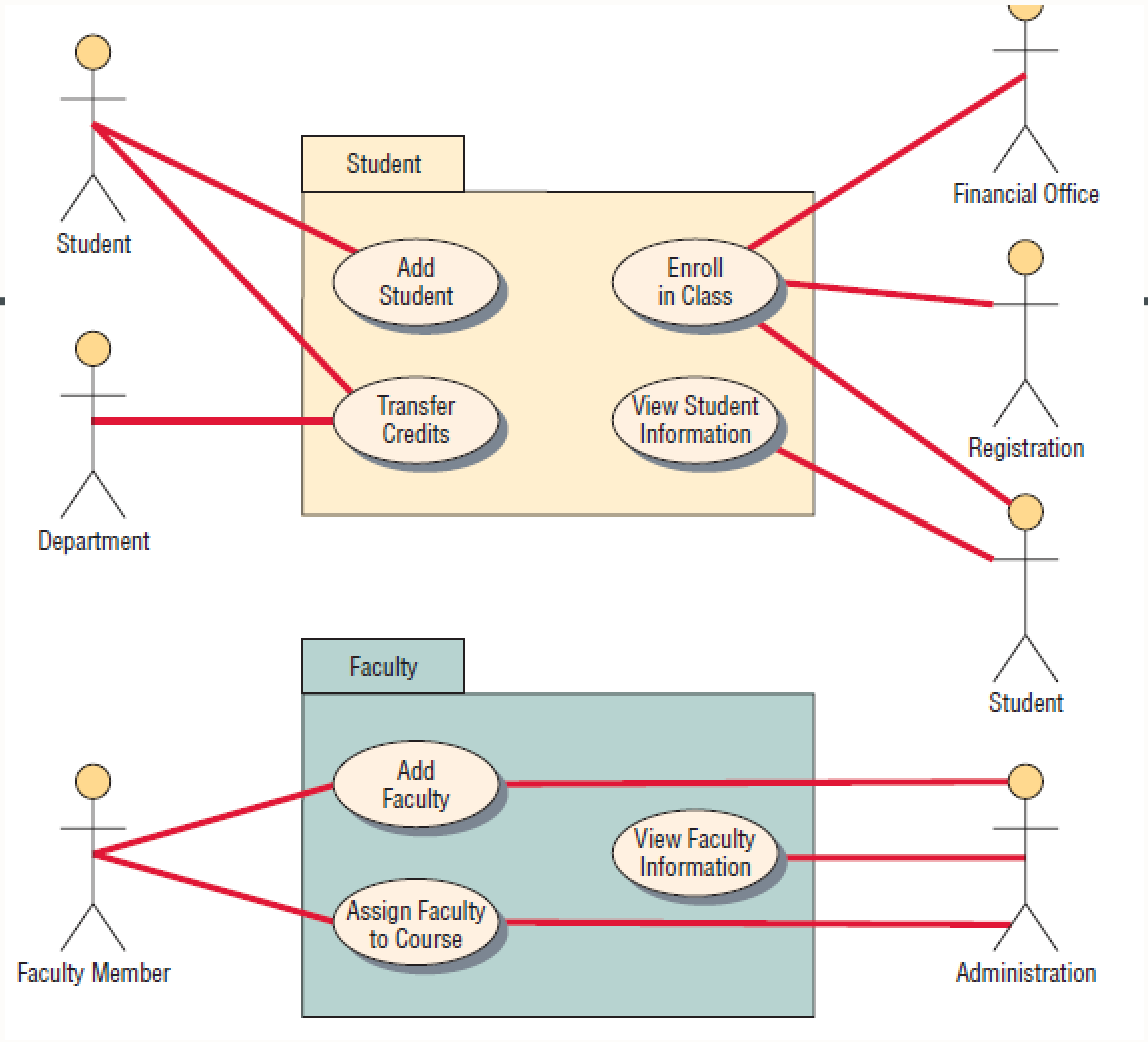


statechart diagram showing how a student progresses from a potential student to a graduated student.

Packages and other UML Artifacts

- Packages are containers for other UML things, such as use cases or classes.
- Packages can show system partitioning, indicating which classes or use cases are grouped into a subsystem, called logical packages.
- They may also be component packages, which contain physical system components, or use case packages, containing a group of use cases.
- Packages use a folder symbol with the package name either in the folder tab or centered in the folder.
- Packaging can occur during systems analysis, or later when the system is being designed.
- Packages may also have relationships, similar to class diagrams, which may include associations and inheritance.

- It shows that four use cases, **Add Student, Enroll in Class, Transfer Credits,** and **View Student Information,** are part of the **Student** package.
- There are three use cases, **Add Faculty, View Faculty Information,** and **Assign Faculty to Course,** that are part of the **Faculty** package



Thanks!
m©