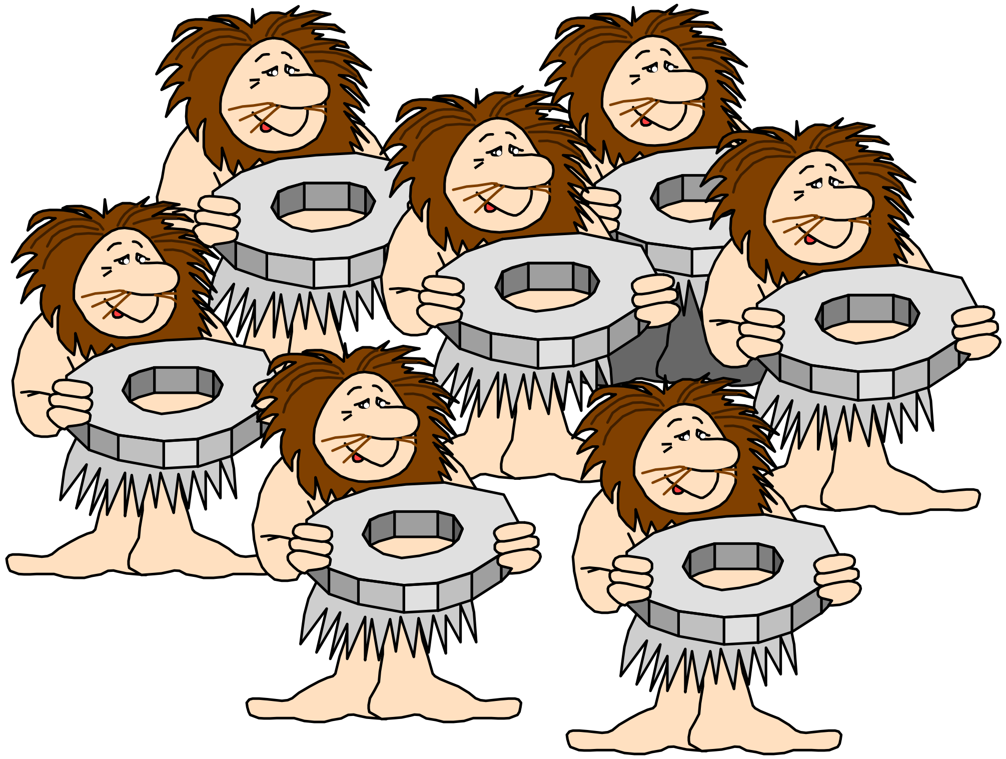
Transitioning from Analysis to Design ~ Collaborations



Objectives: Identifying Collaborations (Classes supporting Use Cases)

☞ *In this lesson we will:*

- ❑ **Understand what it means to *realize* a Use Case in terms of collaborating classes**
- ❑ **Identify a set of objects (classes) that collaborate to realize a Use Case**
- ❑ **Document this information**

  Simple matrix: Use Cases / Classes

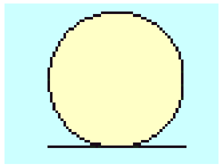
  Graphically: as a Collaboration Diagram
- ❑ **Validate the Class Model**

What is Use Case Realization?

- **Use Case Realization is the process of examining use cases to discover objects and classes for the system being developed**
  - The processing 'behind' the Use Cases is detailed and shown graphically in interaction diagrams
    - ✓ ***Entity* classes supporting a Use Case are identified.**
    - ✓ ***Boundary* and *Controller* classes are created.**
  - Classes are grouped into packages
  - Design classes are defined and refined.

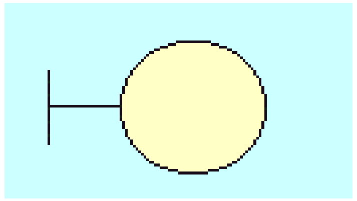
Class Stereotypes are used

- **Entity Classes**



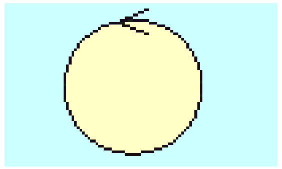
  - model long-lived (often persistent) information
  - usually participate in **many** use cases

- **Boundary Classes**



  - model interaction between the system and its actors

- **Controller Classes**



  - represent coordination, sequencing, and control of other objects (“dispatchers”).
  - encapsulate the control logic of a specific use case.

A Scenario for the “Create a Schedule” Use Case

1. John enters the student ID number 369-52-3449, and the system validates the number. The system asks which semester. John indicates the current semester and chooses ‘create a new schedule.’

2. From a list of available courses, John selects the primary courses English 101, Geology 110, World History 200, and College Algebra 110. He then selects the alternate courses, Music Theory 110 and Introduction to Java Programming 180.

3. The system determines that John has all the necessary prerequisites by examining the the student record and adds him to the course rosters.

4. The system indicates that the activity is complete. The system prints the student schedule and sends billing information for four courses to the billing system for processing.

A Scenario for the “Launch Auction” Use Case

- *In our scenario, EU-Rent has decided to place 22 (of its 172) Chevy Cavalier models on auction.*

<u>Postconditions (at the end of successful processing)</u>:

- ❑ The surplus-count has been recorded, and this amount has also been removed from the active inventory-count.
- ❑ 3 ClubMembers who were “interested enough” in the Mid-Size rental group have been each sent a letter, inviting them to bid, and an initial (zero-amount) bid has been entered into the system to record this.
  - Chris Bronson - “3”
  - Jo Miller - “3”
  - Sandy Thomas - “3”
- ❑ One ClubMember, who was not interested enough in this group, was bypassed.
  - Happy Thymes - “5”

Candidate Entity Objects in the Scenario

New schedule -- list of courses for a semester for a student

List of available courses -- list of all courses being taught in a semester

English 101 -- an offering for a semester

Geology 110 -- an offering for a semester

World History 200 -- an offering for a semester

College Algebra 110 -- an offering for a semester

Music Theory 110 -- an offering for a semester

Introduction to Java Programming 180 -- an offering for a semester

Candidate Entity Objects in the Scenario

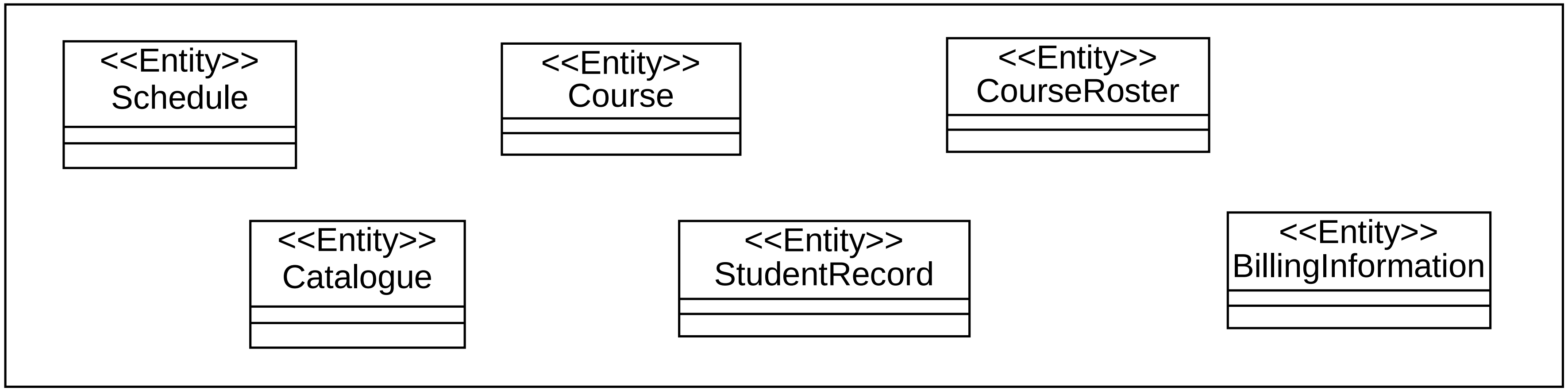
Student record -- a list of courses taken in previous semesters by a student

Course roster -- list of students for a specific course offering

Billing information -- information needed by the billing system actor

Candidate Entity Classes -- “Create a Schedule” Scenario

- Schedule -- list of courses for a semester for a student
- Catalogue -- list of all courses being taught in a semester
- Course -- an offering for a semester
- StudentRecord -- list of previously taken courses
- CourseRoster -- list of students for a specific course offering
- BillingInformation -- information needed by the billing system actor



Use Case / Entity Class Collaboration Matrix

- *What role does each object (class) play in the use case?*
  - **Initially, just “C R U D” level…**

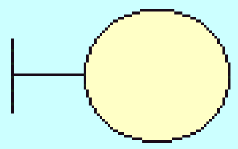
	Entity Class-1	Entity Class-2	Entity Class-3	Entity Class-4	Entity Class-5	Entity Class-6	Entity Class-6
UseCase-1	U			R	U	D	
UseCase-2		C	U				
UseCase-3		U			R		
UseCase-4	R			C		R	
UseCase-5			D			R	
UseCase-6	R		R		U		U
UseCase-7		U		R	R		

Use Case / Entity Class Collaboration Matrix

- *What role does each object (class) play in the use case?*
  - **Then, describing specific responsibilities in each 'cell.'**

	Entity Class-1	Entity Class-2	Entity Class-3	Entity Class-4	Entity Class-5	Entity Class-6	Entity Class-6
UseCase-1	Upjjfdalk jfklsd jklfd			Fjkl jkla jkl jklfjkl	Jjl jlkjlk fj kla Jklfsjl lfjds l ljljsdfa	Fsjkl jkfds kjljls Kljds jl jfdlks klj jlfdsafs	
UseCase-2		Fasfds sa fsajk jlkf; Jfklsdj ljfd sl lfdjsj	Fsa jkl kljsa kl jlfdsjlk Fklsdj lj ls jlk sfalkj				
UseCase-3					Fs jkljkls jklf dsal;		
UseCase-4	Fsdaf jklfdsjkl jklfjsd Hlfsdhljl sl jklfsdR			Fdsa jkdsfj jklfs Fsjkl lj ldfsjlk fsjkl		Rfdsa jkldfs jkld Jfkldsj jkljsdf l Jfklsj fdsjkl jls jlfd	
UseCase-5			Fdsf jklfjds jkf sjkl Jklsfdjlk jklsf jkl			Fdsa jflds jklfd Fjkslj jlksdfj lfslkjR	
UseCase-6	Fsjk j klfsdj sjklj fsdjkl Fjkljsf l jklfsdj ljsfld		Fsdfsdj jklfdsj kl sjkl Jklfdsjlsj fdklsj		Fsdjkl fsjkl sfjkl sf Jfklsjlfsdjkl jsfjlk		Fdsasf jklfdsj kjlfds l
UseCase-7		Fdsf jlkfjkl sjkl fjklds Jfkldsj lsf jkl dsfkjlj		Fdsf jkldfsjs jkldfs jl hklfsdhlhsfdlk	Fsd jkldfsj sjklsf Jklsdjkl fsjkl sfjklsf		

Finding Boundary Classes



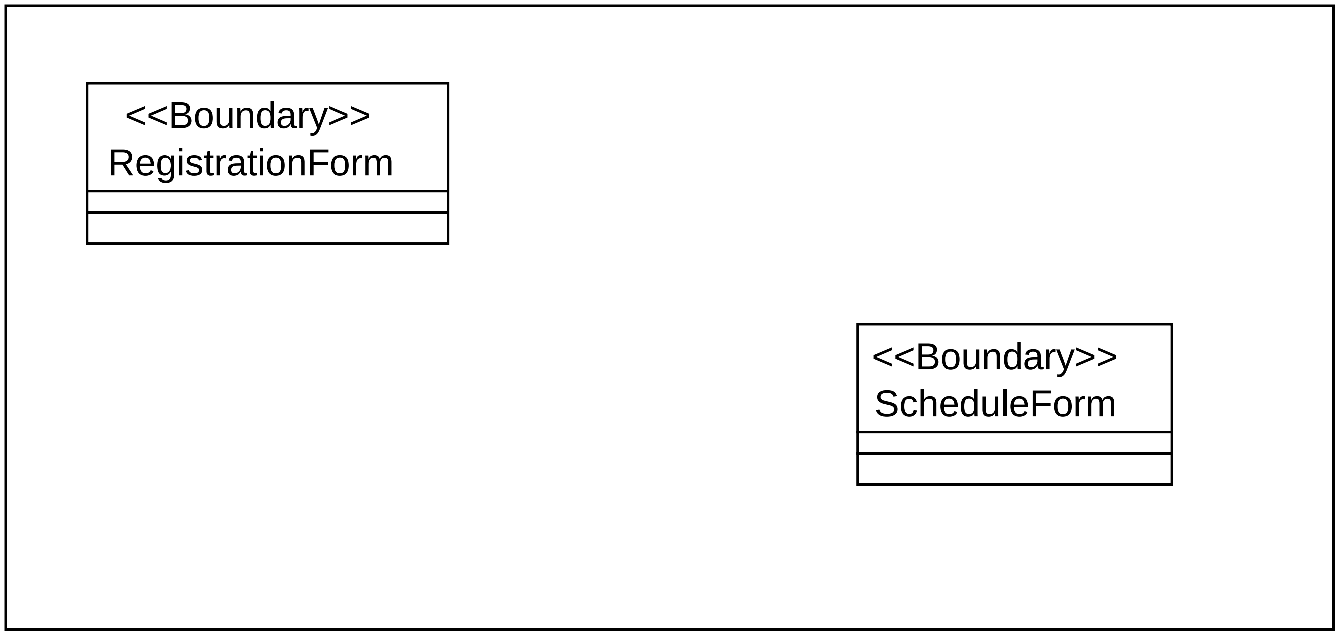
- **For each actor / scenario pair, create an initial boundary class.**
  - During UI design, this “placeholder” UI class will be expanded, and eventually refined based on the chosen user interface mechanisms
  - Example:
  - The student is presented with different options in the “Register for Courses” use case.

    A boundary class called “RegistrationForm” is created to allow the student to select a desired option.
  - The student must input course information to the system in the “Create a Schedule” scenario.

    A boundary class called ScheduleForm is created to hold the information input by the student.

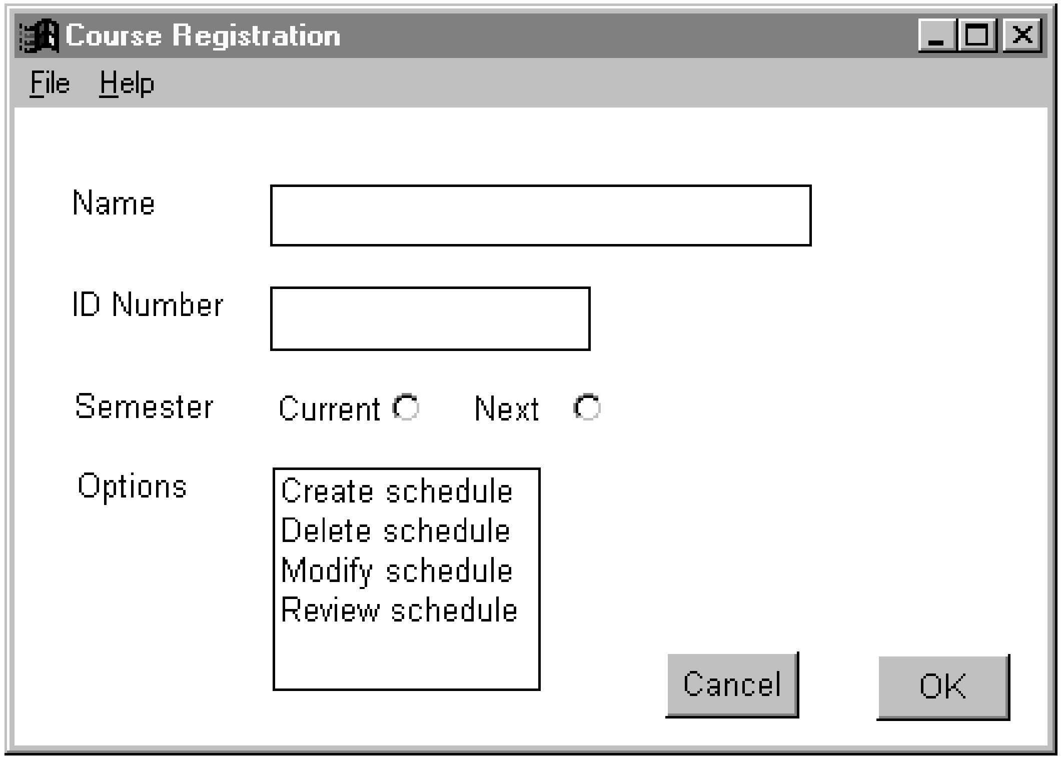
Candidate Boundary Classes

☛ *"Create a Schedule" Scenario*



Window Sketch

- Prototypes and/or window sketches may be created to communicate the look and feel of the boundary class to the user

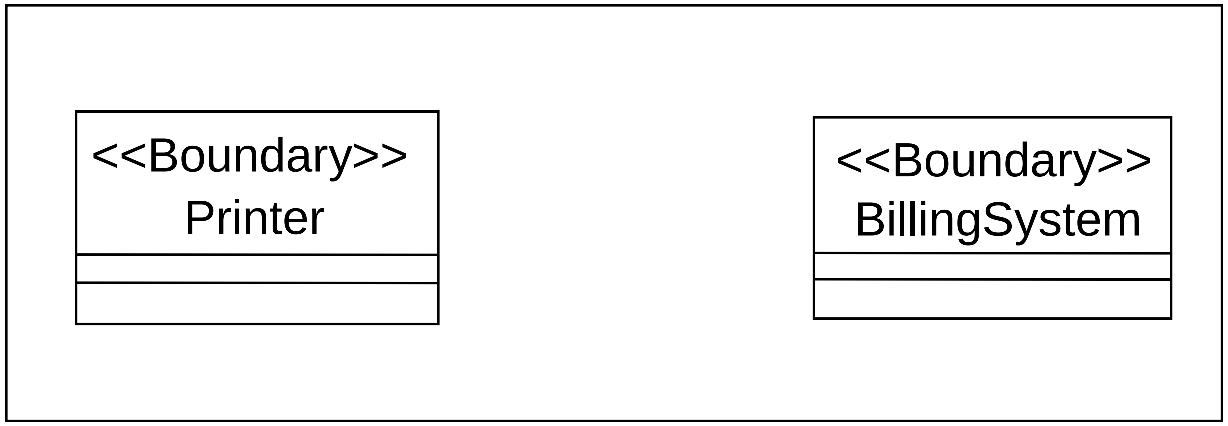


Finding Boundary Classes

- **Boundary classes are also created for communication with system actors**
  - A system may be another software system or a piece of hardware (printers, alarms, etc.)
- **Boundary classes are added to describe the specified system interface, and eventually the communication protocol.**

Candidate Boundary Classes

- *“Create a Schedule” Scenario*
  - **The student schedule is printed in the “Create a Schedule” scenario**
    - A boundary class called Printer is created
  - **Billing information is sent to the Billing System in the “Create a Schedule” scenario**
    - A boundary class called BillingSystem is created



Finding Controller Classes

- **Controller classes typically contain sequencing information.**
  - Caution: controller classes should NOT perform the responsibilities that typically belong to entity and / or boundary classes.
- **At this level of analysis, a controller class is typically added for each use case.**
  - Responsible for handling the flow of events in the use case.
- **This is just an initial cut.**
  - As the design is refined, control classes may be eliminated, split up, or combined.

Controller for the “Register for Courses” Use Case

- **A controller class called RegistrationManager is added.**
  - Receives information from the “ScheduleForm” boundary class.
  - For each selected course:

    Asks the course for its prerequisites.

    Checks to make sure all prerequisite courses have been taken by asking the StudentRecord if a prerequisite course has been successfully completed.

    Knows what to do if a prerequisite has not been taken.

    Asks the course if it is open.

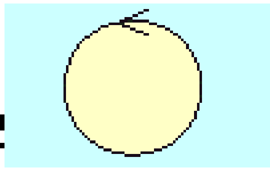
    Asks the course to add the student (if the course is open).

    Knows what to do if 4 courses are not available.

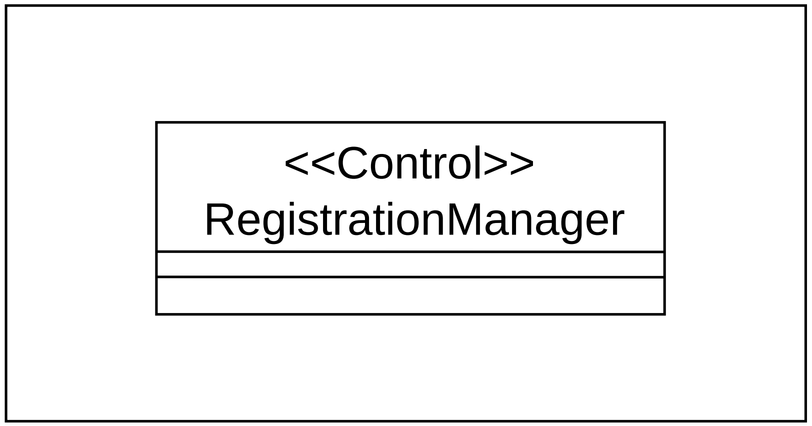
    Creates the StudentSchedule and BillingInformation objects.

    Asks the BillingSystem to send the BillingInformation.

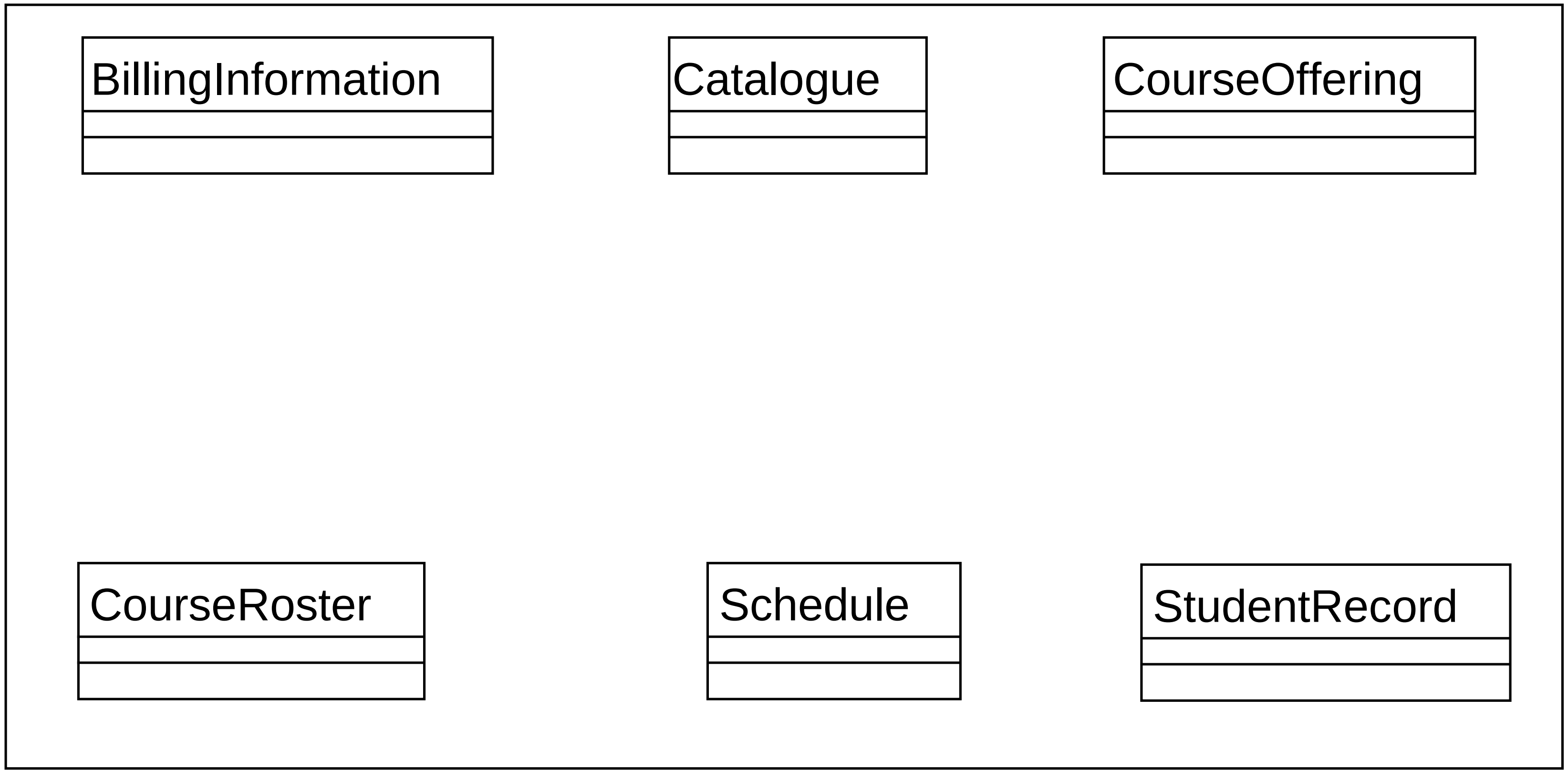
Candidate Controller Class



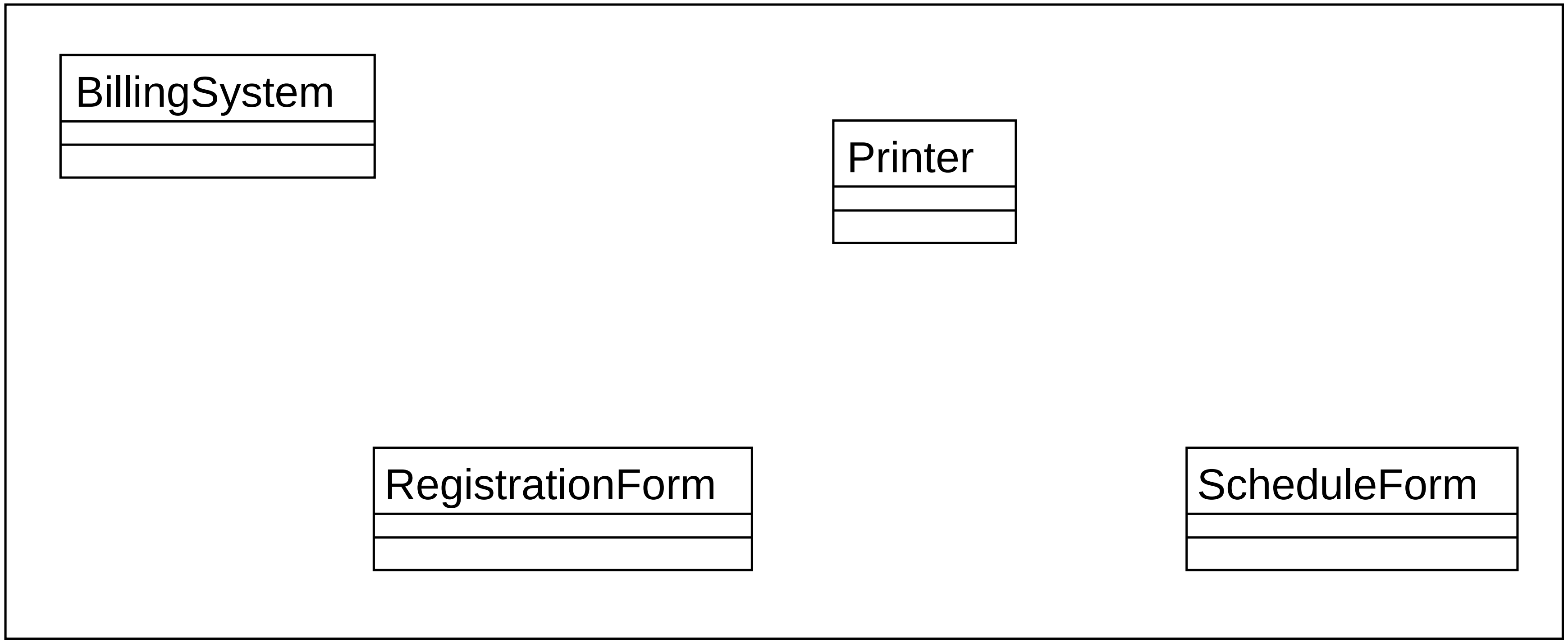
☞ *For the “Register for Courses” Use Case*



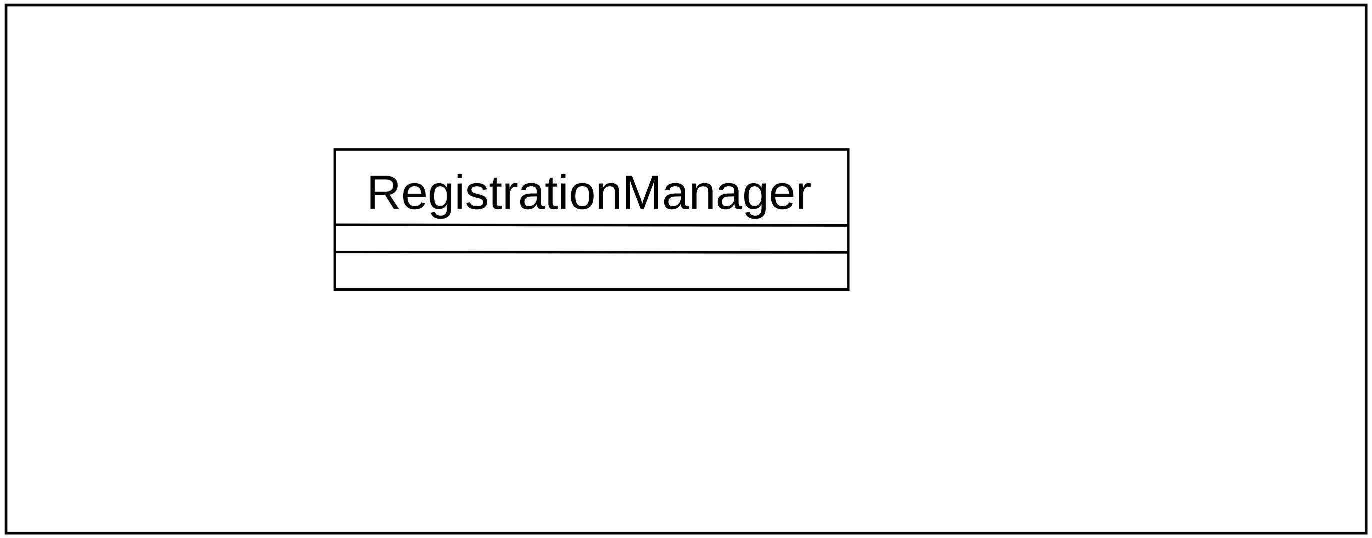
Entity Classes are grouped into one or more Packages



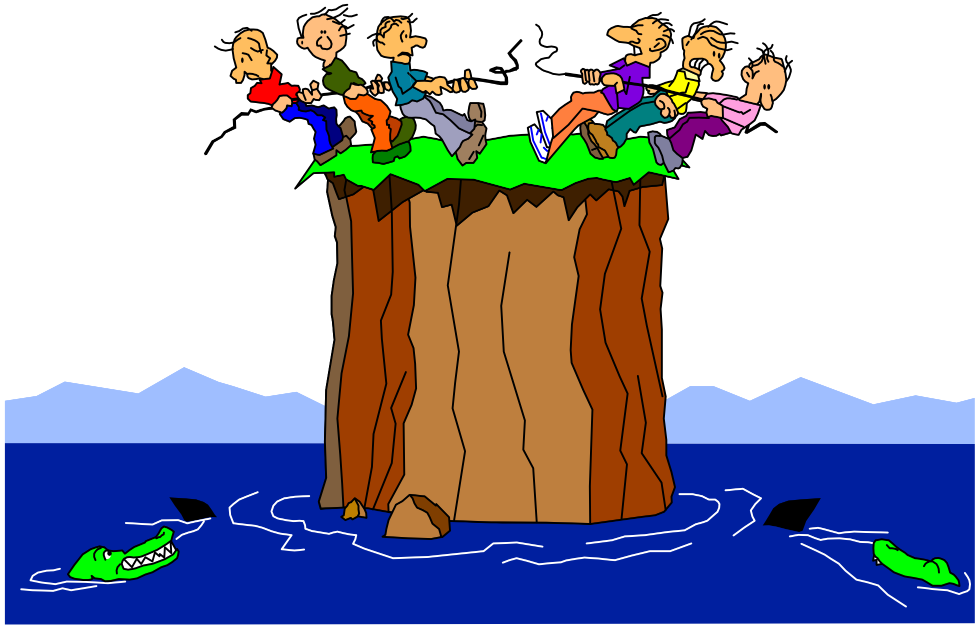
Boundary Classes in an *Interfaces* Package



Controller Classes in a *Controllers* Package



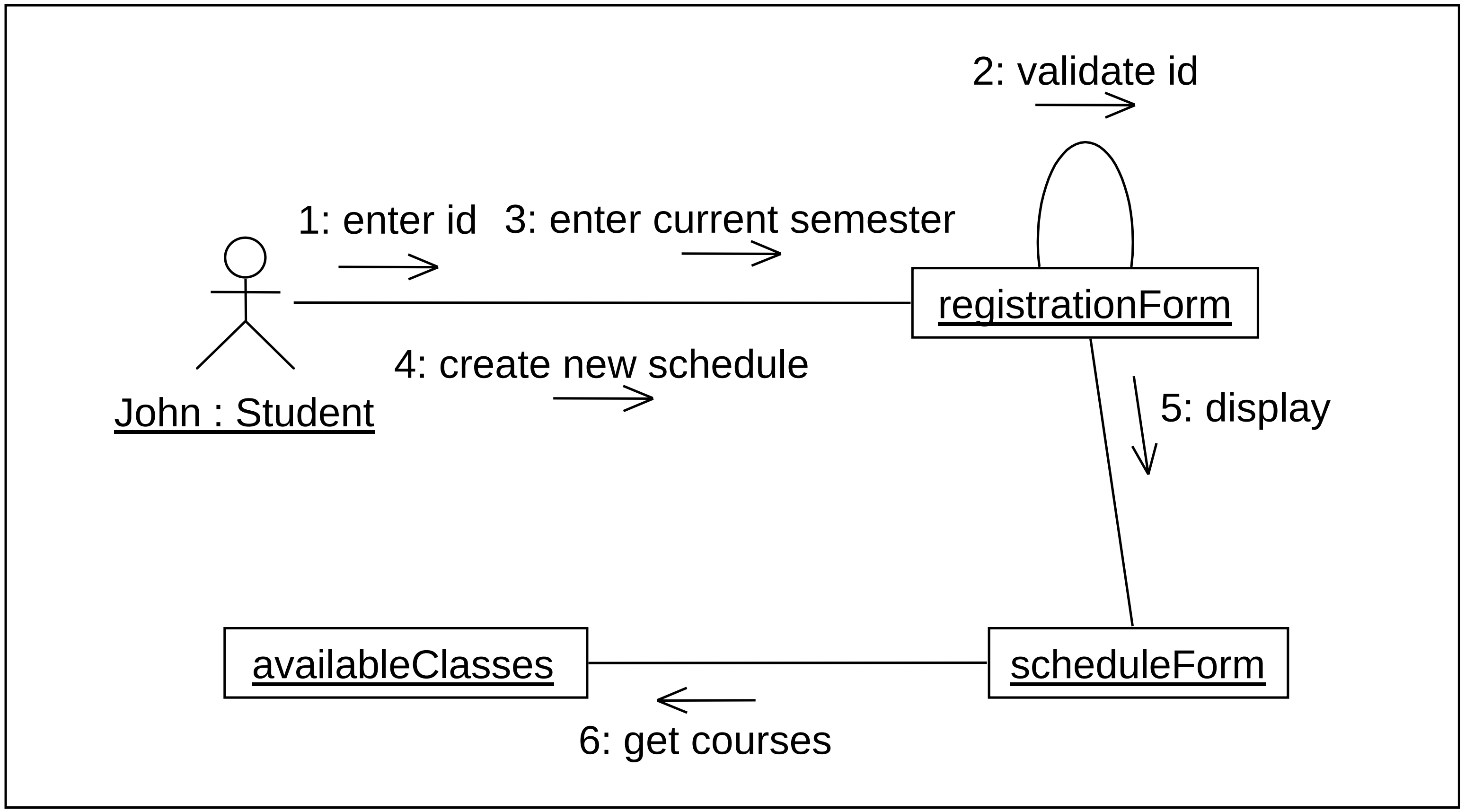
Representing Object Collaborations as Diagrams



Collaboration Diagrams

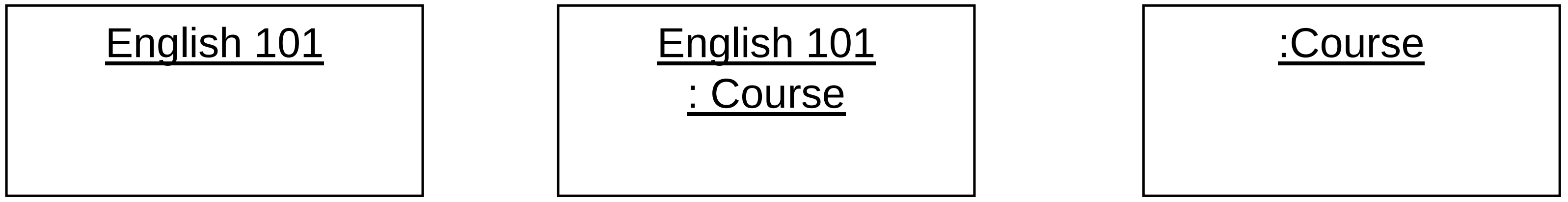
- **A collaboration diagram is a way to represent the messages exchanged by a set of objects to realize a use case.**
- **The diagram shows object interactions organized around the objects and their links to each other.**
- **A collaboration diagram reflects:**
  - objects
  - links between objects
  - messages exchanged between objects
  - conditions on message responses, if any
  - data flowing between objects, if any

Sample Collaboration Diagram



Representing Objects on a Collaboration Diagram

- Objects are drawn as class symbols with underlined names

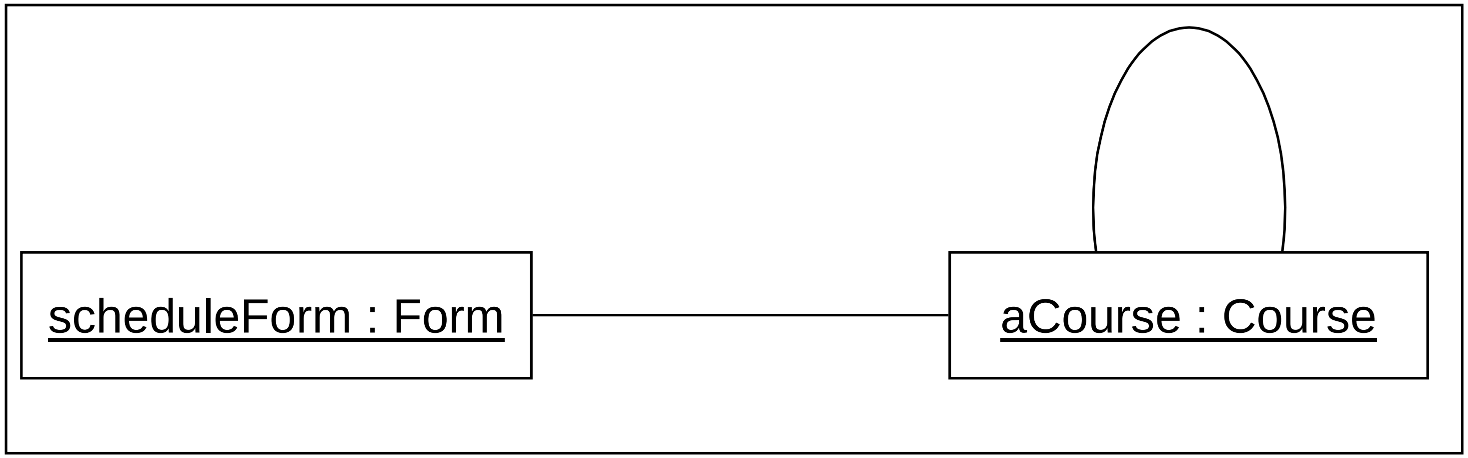


Object only **Object and Class** **Class only**

Note: we will draw the class symbols using the stereotype shapes ~ entity, boundary, controller.

Representing Links on a Collaboration Diagram

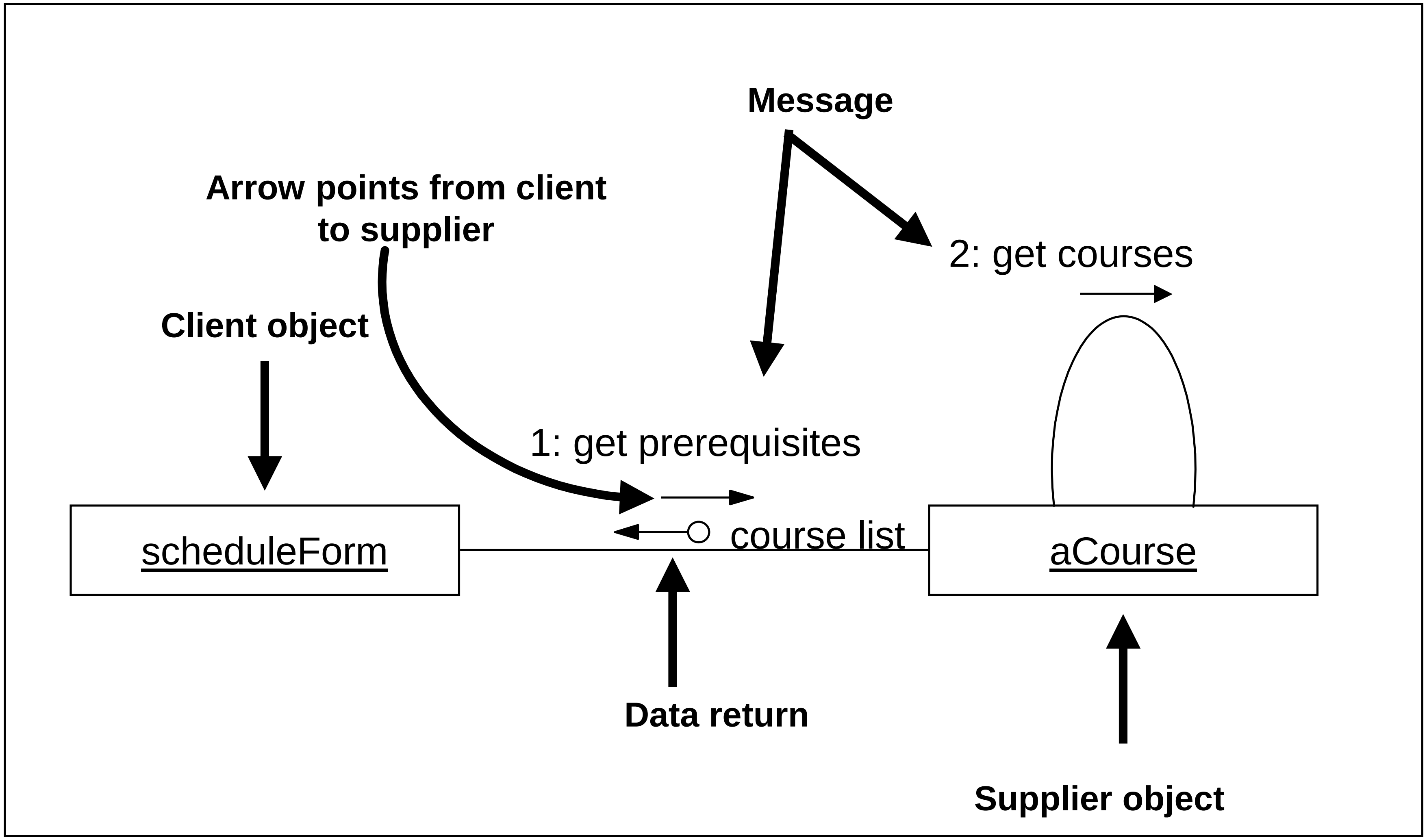
- **A interaction link in a collaboration diagram is represented as a line connecting object symbols**
- **A link indicates that there is a pathway for communication between the connected objects**



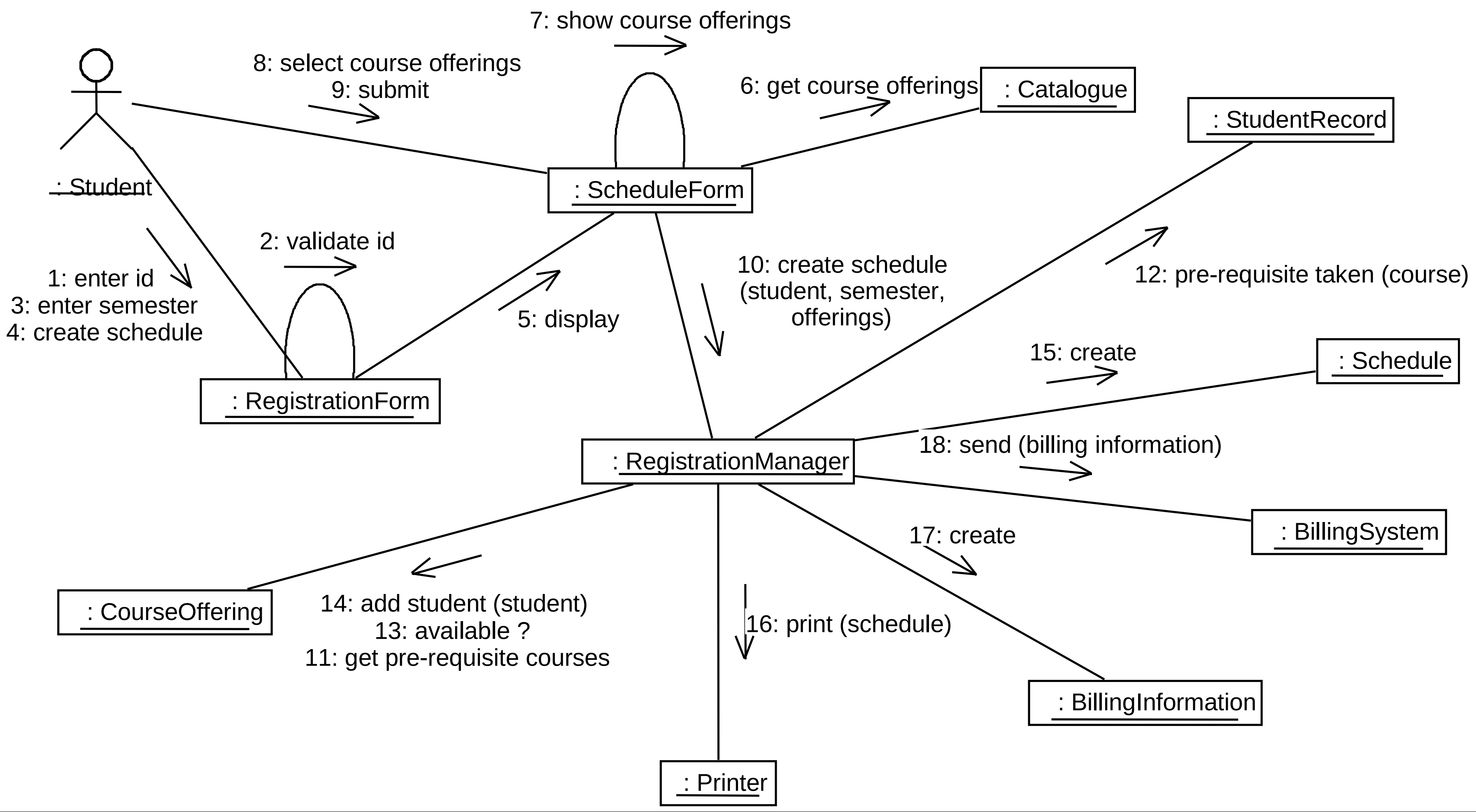
Link Annotations

- **An interaction link in a collaboration diagram can be annotated with:**
  - An arrow pointing from the client object to the supplier object
  - The name of the message with an optional list of parameters and/or a data return value
  - An optional sequence number showing the relative order with which the messages are sent

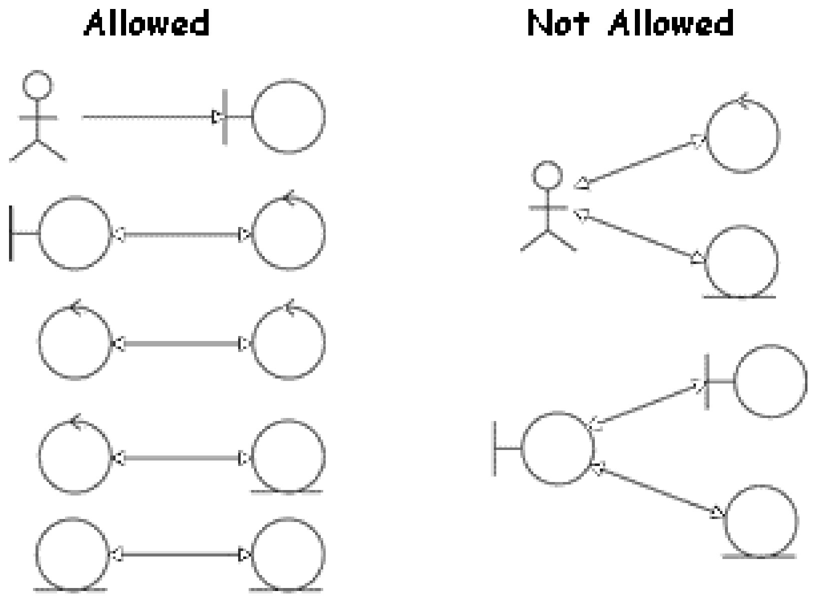
Link Notation



Collaboration Diagram for the “Register for Courses” Scenario



We will follow conventions for linking the collaborating objects



1. Actors can talk only to Boundary objects.
2. Boundary objects can talk only to Controllers or Actors.
3. Entity objects can talk only to Controllers or Entities.
4. Controllers can talk to anyone except Actors.

Summary: Collaboration Diagrams

- **A collaboration diagram is a graphical representation of object interactions**
  - Objects are represented by labeled class symbols.
  - A line (interaction link) is drawn between communicating objects.
    - *The link is annotated with an arrow containing the message name which points from the client object to the supplier object.*
    - *The link may also be annotated with a data return arrow.*