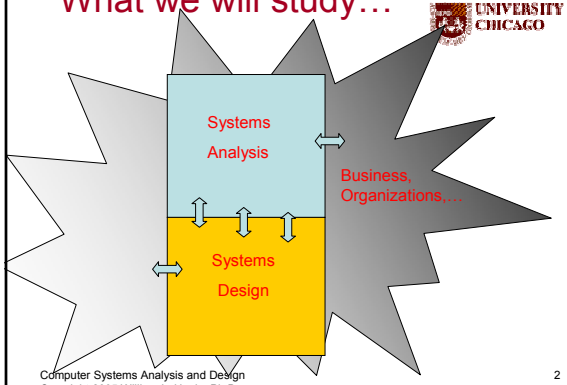
Comp 320 – Computer Systems Analysis and Design

Design Class Diagrams

Dr. William L. Honig
Associate Professor
Department of Computer Science

What we will study…



Design Class Diagrams

- Chapter Nine in Text
  - The real template or mock-up for implementation
  - Be able to “see the final system structure”
- Learn Tools
  - Design Class Diagrams and Signatures
- Learn Skills
  - Coupling and Cohesion; Law of Demeter

Overview

Interaction diagrams are dynamic models of object-oriented software. They show in sequence the internal messages triggered when the façade object receives a message from an actor.

Step 2 of object-oriented program design produces a design class diagram based on the interaction diagrams.

Overview
(continued)

A design class diagram is a static model of the entire system.

It combines the interaction diagrams to show only the links between classes and the direction of visibility between classes.

Overview
(continued)

Step 3 of object-oriented program design specifies precisely the composition of each message and the algorithm for each operation in the program.

At this point, the design of the software in the business layer is complete.

Components of a Design Class Diagram

A design class diagram follows the same UML graphic conventions as a domain model.

A design class diagram shows:

- Classes and class hierarchies
- Attributes
- Operations
- Whole-to-part associations
- Qualified associations

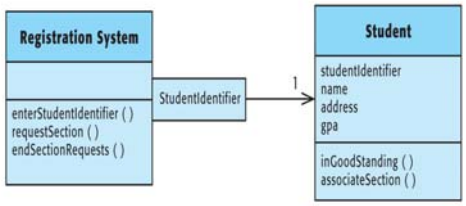
Qualified Associations

A qualified association associates two objects using a qualifier to select objects at the other end of the association.

A qualifier is an attribute or set of attributes which has a unique value for each object in the class.

Qualified Associations
(continued)

FIGURE 9.2



Visibility

Visibility is the ability of one object to have a reference to (to "see") another object. This reference is the object's internal identifier.

Visibility
(continued)

There are four ways to obtain visibility in an object-oriented system:

1. **Reference visibility (Navigability)**: The client object has a pointer or reference to the server object.
2. **Parameter visibility**: An object is provided by a message as a parameter.
3. **Local visibility**: An object obtains visibility to another object by declaring it inside one of its methods.
4. **Global visibility:** An object is obtained from a class by an object requiring visibility to it.

Procedure for Object-Oriented Program Design

Step 1. Produce an interaction diagram for each system operation identified during analysis.

Step 2. Produce a design class diagram showing the operations from the interaction diagrams.

Step 3. Specify the signature and the algorithm for each operation.

Procedure for Object-Oriented Program Design (continued)

Step 4. Design the graphical user interface.

Step 5. Define the interface to the presentation layer.

Step 6. Define the interface to the storage layer.

Step 7. Place the classes in packages.

Step 2 of Object-Oriented Program Design

Produce a design class diagram showing the operations from the interaction diagrams:

Step 2.1: Identify **classes** with their **behavior**.

Step 2.2: Add **inheritance** to the class diagram.

Step 2.3: Add **associations** to the class diagram.

Step 2.4: Create a final class diagram with **qualified associations**.

Goals and Feeling

You remember…

1/3 of what you read,

½ of what people tell you,

But 100% of what you feel

Source: Raytheon CEO Bill Swanson, The CEO's Secret Handbook, Business 2.0, July 2005, pg 69-74.

Show Off Time

LOYOLA UNIVERSITY CHICAGO

- Who here is a good designer?
  - Willing to lead the class in a design class diagram creation
- Let's pick part of the University Registration System
  - One sequence diagram that is your favorite???

Object-Oriented Design Quality

There are three common guidelines for assessing and improving the quality of object-oriented program design:

- **Cohesion**
- **Coupling**
- **The Law of Demeter**

Cohesion

Cohesion measures how strongly related and focused the responsibilities of a class are – that is, how diverse an object's attributes and behaviors are.

It is desirable to have objects with high cohesion.

Using the Expert pattern helps increase cohesion.

An Example of Poor Cohesion

FIGURE 9.16

EXPERT OBJECT	RESPONSIBILITIES
section	create ( ) associate professor ( ) associate student ( ) in good standing ( ) – nothing to do with section

Coupling

Coupling measures how strongly one class is **connected to, has knowledge of,** or **relies on** other objects.

It is desirable to have objects with **low coupling**.

Giving too much responsibility to the **façade** object leads to poor coupling.

Sources of Coupling

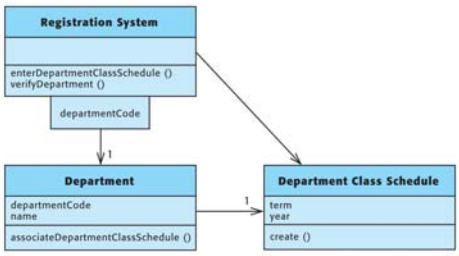
Subclasses are strongly connected to their **superclasses**.

Other sources of coupling are classes which have:

- A **reference to another class** or an object in that class.
- A **method which references another class** or an object in that class.
- An **association with another class**.

An Example of Increased Coupling

FIGURE 9.17



The Law of Demeter

The Law of Demeter states that a client should give its server the responsibility for collaborating with other objects.

An object should send messages only to:

- **Itself**
- **An object to which it contains a reference**
- **A parameter of one of its methods**
- **One of its local objects**
- **A class**

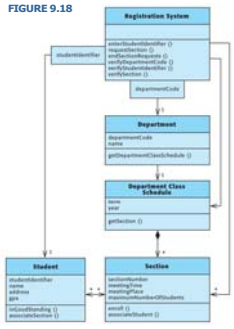
Classical Studies Short Course

LOYOLA UNIVERSITY CHICAGO

- **Dêmêtêr** (or **Demetra**) (DEH-MEH-ter) ("mother-goddess" or perhaps "distribution-mother") is the Greek goddess of agriculture,
  - the pure nourisher of youth and the green earth, the health-giving cycle of life and death, and preserver of marriage and the sacred law.
  - She is invoked as the "bringer of seasons" in the Homeric hymn, a subtle sign that she was worshiped long before the Olympians arrived.
  - She and her daughter Persephone were the central figures of the Eleusinian Mysteries that also predated the Olympian pantheon.
- The Roman equivalent is Ceres.

A Violation of The Law of Demeter

FIGURE 9.18



Show Off Time

LOYOLA UNIVERSITY CHICAGO

- Who here is a good designer?
  - Willing to debate some simple design quality examples with the class
- Here's some examples from the university registration system
  - Let's give them a grade…..

Specifying Data Types

Data type is the UML term for the description of an attribute of a class.

Data types are either primitive data types or value objects.

A primitive data type can be represented directly in computer hardware – integer, floating point, or character.

Specifying Data Types
(continued)

A value object is an object whose unique identity is not meaningful.

A reference object is an object whose unique identity is meaningful.

Reference objects are never shown as attributes in a class diagram; they are shown as associations.

Signatures

A signature is a specification of an operation which describes completely the interface of the operation.

This comprises the operation's:

- name, and
- every input and output with its data type.

Specifying Attributes in the UML

Attributes are described as:

- name : data type

Where the first symbol represents the attribute's visibility:

- \- for private
- \+ for public
- # for protected

Specifying Operations in the UML

Operations are described in the sequence operation visibility, operation name, parameter list, and type returned by the operation:

+ operation name (in parameter name : parameter type … ,
out parameter name : parameter type … ,
inout parameter name : parameter type ,) : return type

Example of Specifying an Operation

FIGURE 9.23

Contract Name:	**+ enroll (studentIdentifier : int, courseNumber : int, sectionNumber : int) : boolean**
Class:	Department
Use Case:	Register for Classes
Responsibilities:	Enroll the student if the section is not filled.
Exceptions:	None
Preconditions:	Student is known to the system and in good standing. Section is known to the system.
Postconditions:	If the student was enrolled, the association between Student and Section was saved and true was returned; else, false was returned.

Example of Specifying an Operation (continued)

FIGURE 9.25

Contract Name:	**+ makeSection (courseNumber : int, maximumNumberOfStudents : int, meetingTime : String, meetingPlace : String, professor : Professor) : Section**
Class:	Department Class Schedule
Use Case:	Submit Department Class Schedule
Responsibilities:	Add a new Section to Department Class Schedule.
Exceptions:	None
Preconditions:	Department Class Schedule is known to the system.
Postconditions:	The instance of **section** was saved. The association between Section and Department Class Schedule was saved.

Review Questions
Design Class Diagrams and Method Signatures

LOYOLA UNIVERSITY CHICAGO

- See text Review Questions:
  - 9-1, 9-2, 9-4, 9-6*, 9-10, 9-11 in particular
- Key Ideas (Deep Knowledge):
  - How to design a class – skills and tools
  - How to interconnect classes – tools
- Possibly the single most important chapter in the course!

Learning Objectives

- **Understand the components of a design class diagram and how it differs from a domain model.**
- **Determine which associations are required to support the messages from the interaction diagrams.**
- **Know when to use qualified associations.**

Learning Objectives
(continued)

- **Understand visibility and why it is important.**
- **Draw a complete design class diagram.**
- **Measure the design quality using the properties of cohesion, coupling, and the Law of Demeter.**
- **Write the method signatures.**

Summary

- Step 2 of object-oriented program design produces a design class diagram.
- This diagram shows the interactions from the interaction diagrams, visibilities, and may show qualified associations.
- Coupling, cohesion, and the Law of Demeter are used to refine the diagram.
- Defining the signatures and algorithms for all the operations completes the design of the business layer.