Comp 320 – Computer Systems Analysis and Design Onto Design…..

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

Design Tools

- Chapter 8 in Text
  - Professional, complete, OO designs….
    - Steps or "Roadmap" for implementation
      - Make the Objects Live, "Seeable"
    - Implement all the requirements from analysis
- Design tool = UML
  - Complete Objects = Design Class Diagram
  - Collaboration = Sequence Diagram
  - Who does what when and how they do it

Overview

Design is a critical intermediate step between a statement of requirements and the construction of a solution.

It produces a description of the solution – not the solution itself. This description is sufficiently complete and accurate to assure that the solution can be constructed.

Design models allow the behavior of proposed solutions to be evaluated and compared.

Responsibilities

The principal task of object-oriented program design is to assign responsibilities to classes.

A responsibility is an obligation of an object to other objects.

Responsibilities
(continued)

An object may be responsible for knowing:

- What it knows – its attributes
- Who it knows – the objects associated with it
- What it knows how to do – the operations it can perform

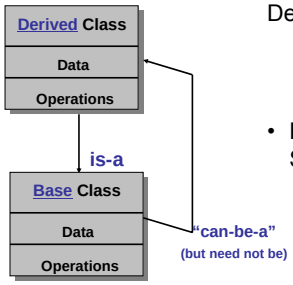
OO Review

LOYOLA UNIVERSITY CHICAGO

- Class and Encapsulation (system parts)
  - Attributes (Private Information)
  - Methods (Public Behavior)
  - Inheritance
  - Polymorphism
- Interactions (doing things)
  - Messages
  - Parameters
  - "access" to or visibility of other objects
- Instances are NOT classes

Types of Relationships - Inheritance

LOYOLA UNIVERSITY CHICAGO



Derived Class Object
- "is-a" Base Class Object
- "is-a-kind-of"
- "must-be-a"

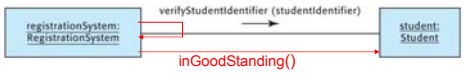
- Remember the Substitution Rule
  - **Any object of the derived class must be usable in place of a base-class object.**

Fundamental Concepts – Messages (continued)

Corrections Added WLHonig

Visibility: For an object (the client) to send a message to another object (the server), the receiving object must be visible to the sending object. (That is, it must know the server's identity).

FIGURE 8.4



NEVER ask someone to confirm their own identity

Overview (continued)

Object-oriented program design is a seven-step process.

Chapter 8 presents the first step – producing an interaction diagram for each system operation identified during analysis.

Overview
(continued)

The interaction diagrams are developed on the basis of system operation contracts produced during analysis.

This overall approach is called "design by contract."

Overview
(continued)

As discussed in Chapter 5, design by contract assumes a commitment to a contract on the part of the object which receives a message.

The preconditions and postconditions of the system operation contracts drive the program design.

Overview
(continued)

In developing the interaction diagram for each system operation, we must assure that the operation:

- first checks whether every precondition of the contract is true, and then
- makes every postcondition of the contract come true.

Patterns for Object-Oriented Program Design

A pattern is a named statement of a design problem together with its solution and guidance for applying the pattern. Patterns include:

- Façade
- Creator
- Expert
- Singleton

The Façade Pattern

Problem: Who should be responsible for handling a system operation message from an actor?

Solution: Assign this responsibility to an object representing the system as a whole.

The Creator Pattern

Problem: Who should be responsible for requesting the creation of a new object, i. e.,
who sends the *create* message to the appropriate class?

Solution: Assign this responsibility to a class which is in some way closely involved with the class. (See Figure 8.4 in text for details.)

The Expert Pattern

Problem: What is the most basic principle for assigning responsibilities to objects?

Solution: Assign the responsibility to the class which has the information necessary to fulfill it.

Interaction Diagrams

An interaction diagram depicts the messages between objects or classes in a program. It shows collaborations between objects.

The UML includes two types of interaction diagrams – collaboration diagrams and sequence diagrams.

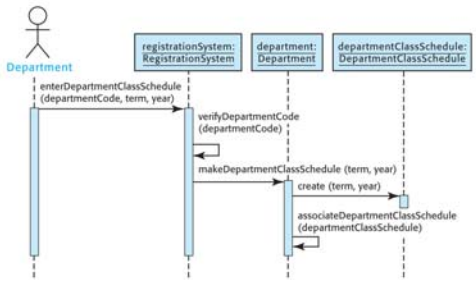
Sequence Diagrams

A sequence diagram shows interactions in a fence format.

The messages appear from top to bottom in the sequence in which they occur.

Sequence Diagrams
(continued)

FIGURE 8.6



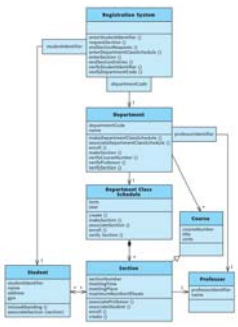
Design Class Diagrams

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

However, it also shows the operations of the classes.

Design Class Diagrams
(continued)

FIGURE 8.7



Design Sequence Diagrams

System sequence diagrams show only messages between the system and actors.

Design sequence diagrams show all the messages between objects inside the system.

Learning Objectives

- Explain fundamental object-oriented concepts.
- Understand what patterns are and how they are used.
- Learn how to assign responsibilities to classes using the **Façade, Creator,** and **Expert** patterns.

Learning Objectives
(continued)

- Understand the differences between **collaboration** and **sequence** diagrams.
- Create **interaction diagrams**.