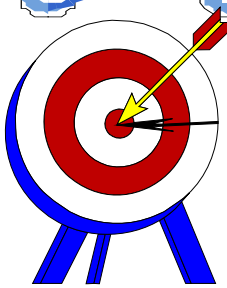
Object-Oriented Systems Development:
Using the Unified Modeling Language

Chapter 10:
Designing Classes

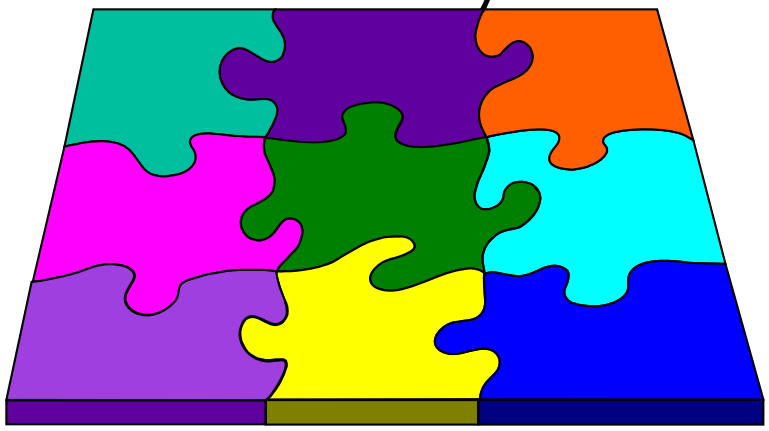


Goals

- Designing classes.
- Designing protocols and class visibility.
- Defining attributes.
- Designing methods.

Object-Oriented Design Philosophy

- **The first step in building an application should be to design a set of classes, each of which has a specific expertise and all of which can work together in useful ways.**



Designing Class: the Process

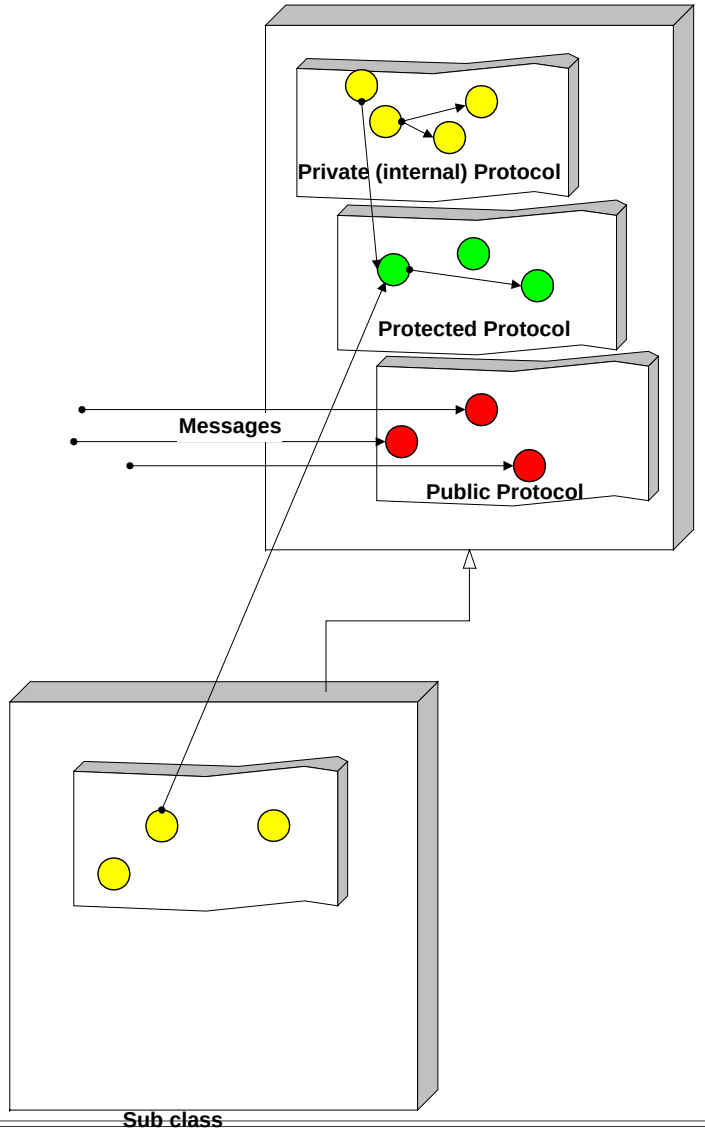
1. **Apply design axioms to design classes, their attributes, methods, associations, structures, and protocols.**
   - **1.1. Refine and complete the static UML class diagram (object model) by adding details to that diagram.**
     - **1.1.1. Refine attributes.**
     - **1.1.2. Design methods and the protocols by utilizing a UML activity diagram to represent the method's algorithm..**
     - **1.1.3. Refine the associations between classes (if required).**
     - **1.1.4. Refine the class hierarchy and design with inheritance (if required).**
   - **1.2. Iterate and refine again.**

Class Visibility

- **In designing methods or attributes for classes, you are confronted with two issues.**
  - **One is the *protocol*, or interface to the class operations and its visibility;**
  - **and how it should be implemented.**

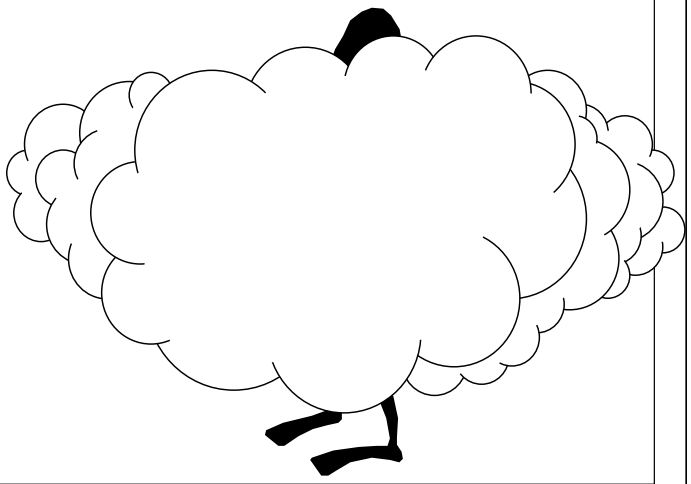
Class Visibility (Con't)

- **Public protocols define the functionality and external messages of an object, while private protocols define the implementation of an object.**



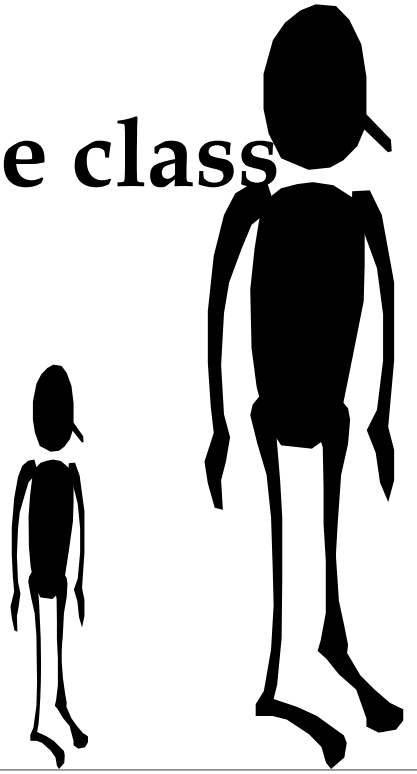
Private Protocol (Visibility)

- **A set of methods that are used only internally.**
- **Object messages to itself.**
- **Define the implementation of the object (Internal).**
- **Issues are: deciding what should be private.**
  - **What attributes**
  - **What methods**



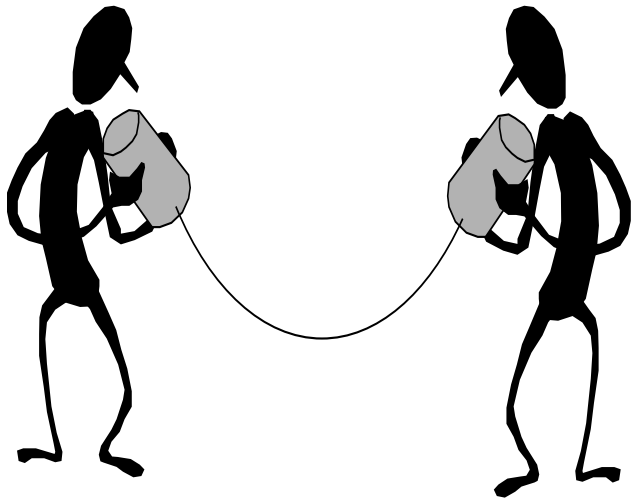
Protected Protocol (Visibility)

- **In a protected protocol, subclasses can use the method in addition to the class itself.**
- **In private protocols, only the class itself can use the method.**



Public Protocol (Visibility)

- **Defines the functionality of the object**
- **Decide what should be public (External).**



Guidelines for Designing Protocols

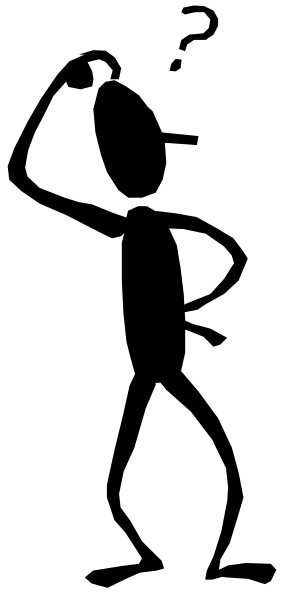
- **Good design allows for polymorphism.**
- **Not all protocols should be public, again apply design axioms and corollaries.**

Guidelines for Designing Protocols (Con't)

- **The following key questions must be answered:**
  - **What are the class interfaces and protocols?**
  - **What public (external) protocol will be used or what external messages must the system understand?**

Questions (Con't)

- **What private or protected (internal) protocol will be used or what internal messages or messages from a subclass must the system understand?**



Attribute Types

- **The three basic types of attributes are:**
  - **1. Single-value attributes.**
  - **2. Multiplicity or multivalue attributes.**
  - **3. Reference to another object, or instance connection.**

Designing Methods and Protocols

- **A class can provide several types of methods:**
  - ***Constructor*. Method that creates instances (objects) of the class.**
  - ***Destructor*. The method that destroys instances.**
  - ***Conversion method*. The method that converts a value from one unit of measure to another.**

Designing Methods and Protocols (Con't)

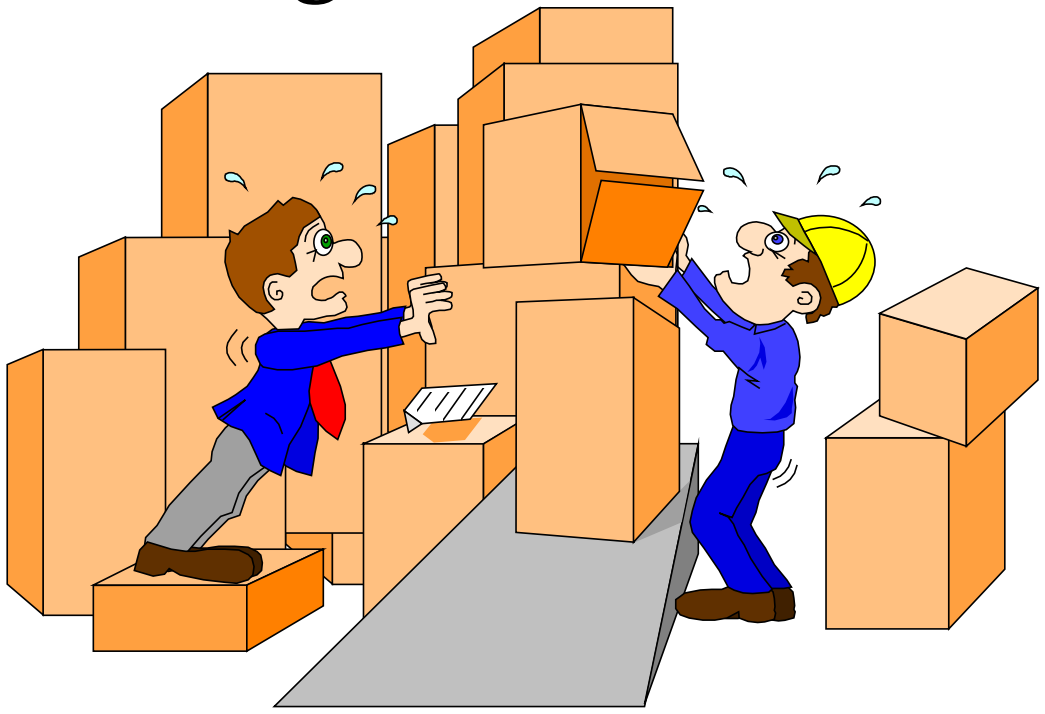
- ***Copy method*. The method that copies the contents of one instance to another instance.**
- ***Attribute set*. The method that sets the values of one or more attributes.**
- ***Attribute get*. The method that returns the values of one or more attributes.**

Designing Methods and Protocols (Con't)

- ***I/O methods*. The methods that provide or receive data to or from a device.**
- ***Domain specific*. The method specific to the application.**

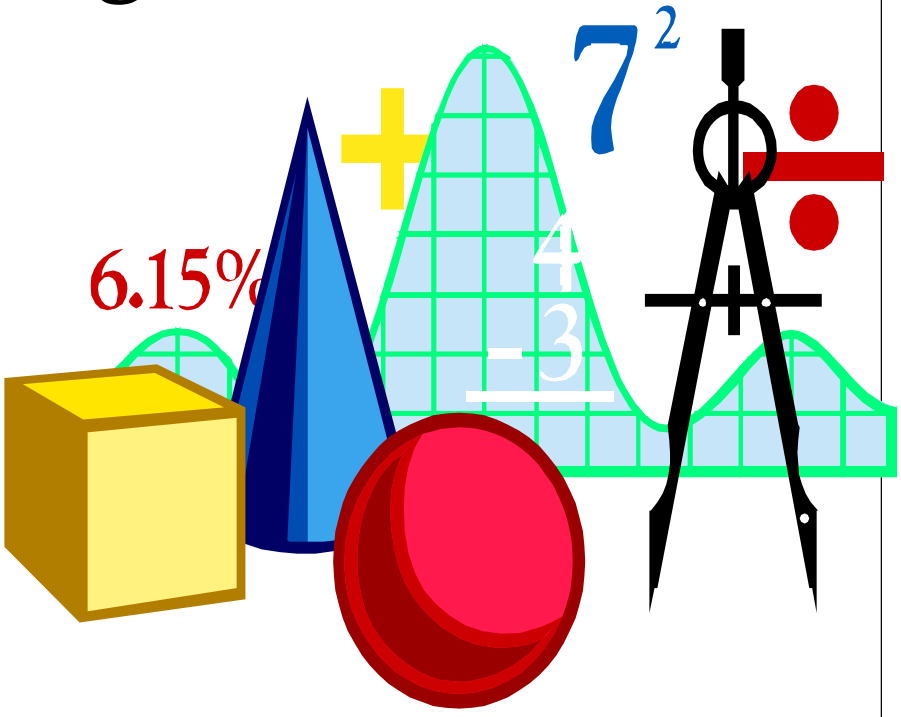
Five Rules For Identifying Bad Design

- **I. If it looks messy then it's probably a bad design.**



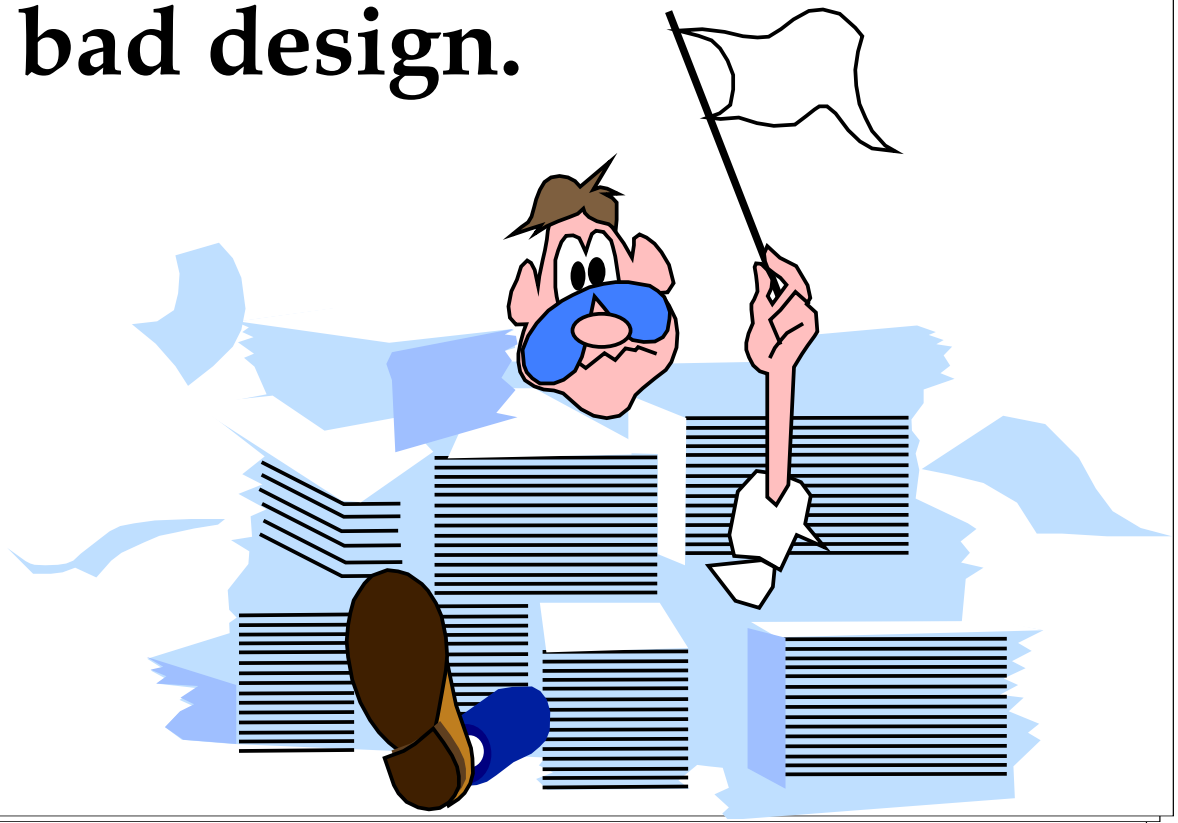
Five Rules For Identifying Bad Design (Con't)

- **II. If it is too complex then it's probably a bad design.**



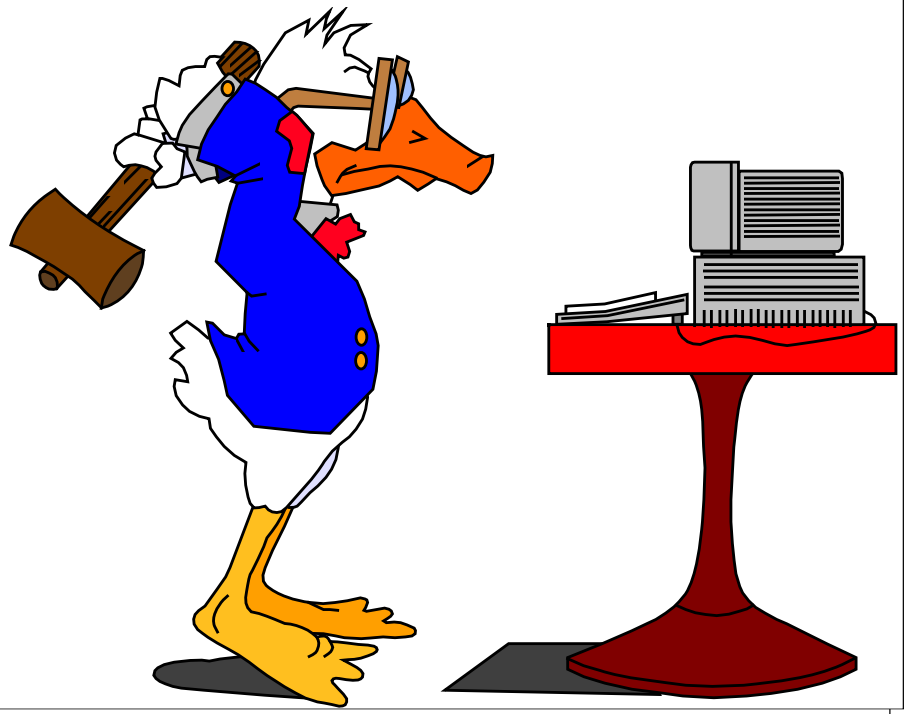
Five Rules For Identifying Bad Design (Con't)

- **III. If it is too big then it's probably a bad design.**



Five Rules For Identifying Bad Design (Con't)

- **IV. If people don't like it then it's probably a bad design.**



Five Rules For Identifying Bad Design (Con't)

- **V. If it doesn't work then it's probably a bad design.**

Avoiding Design Pitfalls

- **Keep a careful eye on the class design and make sure that an object's role remains well defined.**
- **If an object loses focus, you need to modify the design.**
- **Apply Corollary 2 (single purpose).**

Avoiding Design Pitfalls (Con't)

- **Move some functions into new classes that the object would use.**
- **Apply Corollary 1 (uncoupled design with less information content).**
- **Break up the class into two or more classes.**
- **Apply Corollary 3 (large number of simple classes).**

Summary

- **This chapter concentrated on the first step of the object-oriented design process, which consists of applying the design axioms and corollaries to design classes, their attributes, methods, associations, structures, and protocols; then, iterating and refining.**

Summary (Con't)

- **Object-oriented design is an iterative process.**
- **Designing is as much about discovery as construction.**
- **Do not be afraid to change a class design, based on experience gained, and do not be afraid to change it a second, third, or fourth time.**