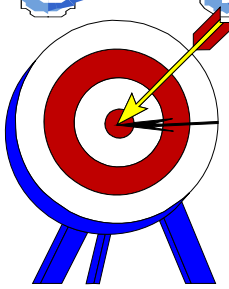
Object-Oriented Systems Development:
Using the Unified Modeling Language

Chapter 9:

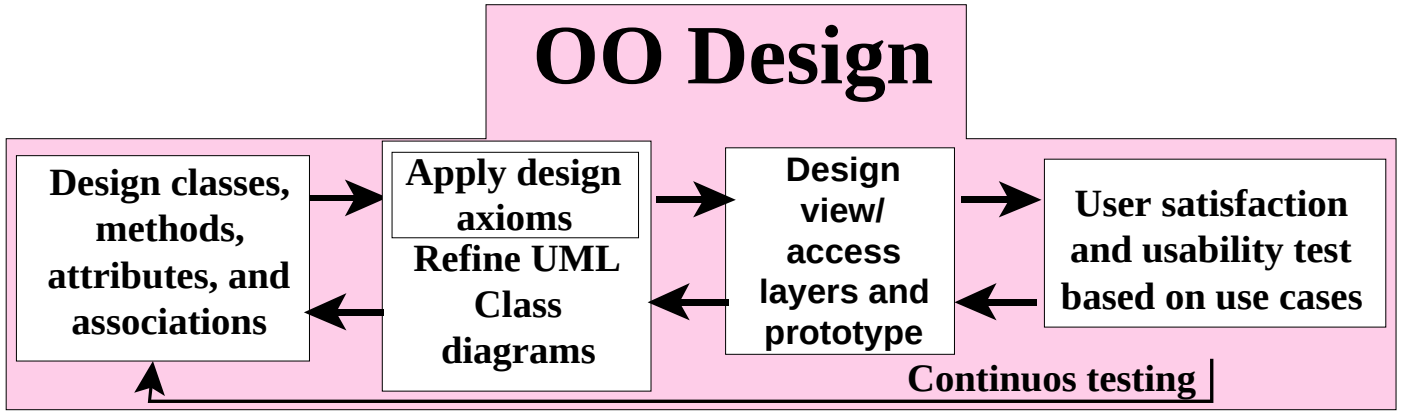
The Object-Oriented Design Process and Design Axioms



Goals

- **The object-oriented design process.**
- **Object-oriented design axioms and corollaries.**
- **Design patterns.**

Object-Oriented Design Process in the Unified Approach



OO Design 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 the UML class diagram. This step consists of the following activities:
  - 1.1.1. Refine attributes.
    - 1.1.2. Design methods and protocols by utilizing a UML activity diagram to represent the method's algorithm.
    - 1.1.3. Refine associations between classes (if required).
    - 1.1.4. Refine class hierarchy and design with inheritance (if required).
  - 1.2. Iterate and refine again.

OO Design Process (Con't)

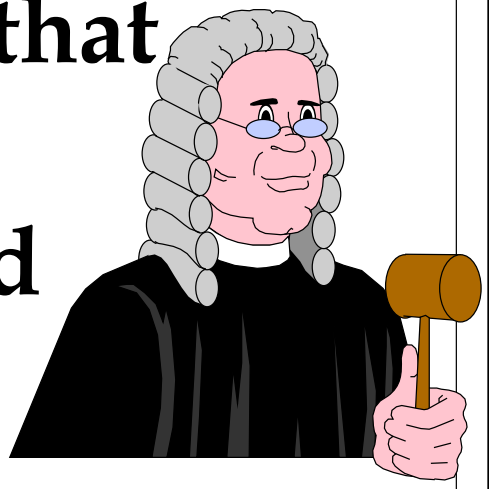
- **2. Design the access layer**
- **2.1.** Create mirror classes. For every business class identified and created, create one access class.
  - **2.2.** define relationships among access layer classes.
  - **2.3.** Simplify the class relationships**.** The main goal here is to eliminate redundant classes and structures.
    - 2.3.1. Redundant classes: Do not keep two classes that perform similar *translate request* and *translate results* activities. Simply select one and eliminate the other.
    - 2.3.2. Method classes: Revisit the classes that consist of only one or two methods to see if they can be eliminated or combined with existing classes.
  - 2.4. Iterate and refine again.

OO Design Process (Con't)

- **3. Design the view layer classes.**
  - 3.1. Design the macro level user interface, identifying view layer objects.
  - 3.2. Design the micro level user interface, which includes these activities:
    - 3.2.1. Design the view layer objects by applying the design axioms and corollaries.
    - 3.2.2. Build a prototype of the view layer interface.
  - 3.3. Test usability and user satisfaction (Chapters 13 and 14).
  - 3.4. Iterate and refine.
- **4. Iterate and refine the preceding steps. Reapply the design axioms and, if needed, repeat the preceding steps.**

Object-Oriented Design
Axioms, Theorems and Corollaries

- **An *axiom* is a fundamental truth that always is observed to be valid and for which there is no counterexample or exception.**
- **A *theorem* is a proposition that may not be self-evident but can be proven from accepted axioms.**



Axioms, Theorems and Corollaries (Con't)

- **A *Corollary* is a proposition that follows from an axiom or another proposition that has been proven.**

Design Axioms

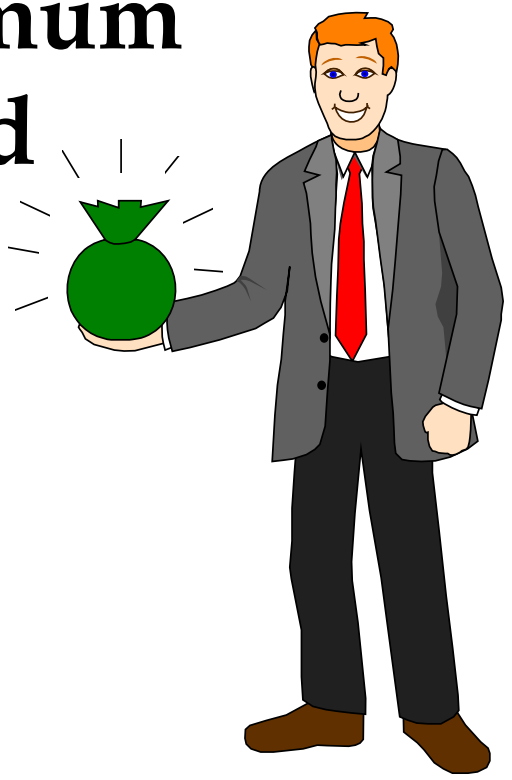
- **Axiom 1 deals with relationships between system components (such as classes, requirements, software components).**
- **Axiom 2 deals with the complexity of design.**

Axioms

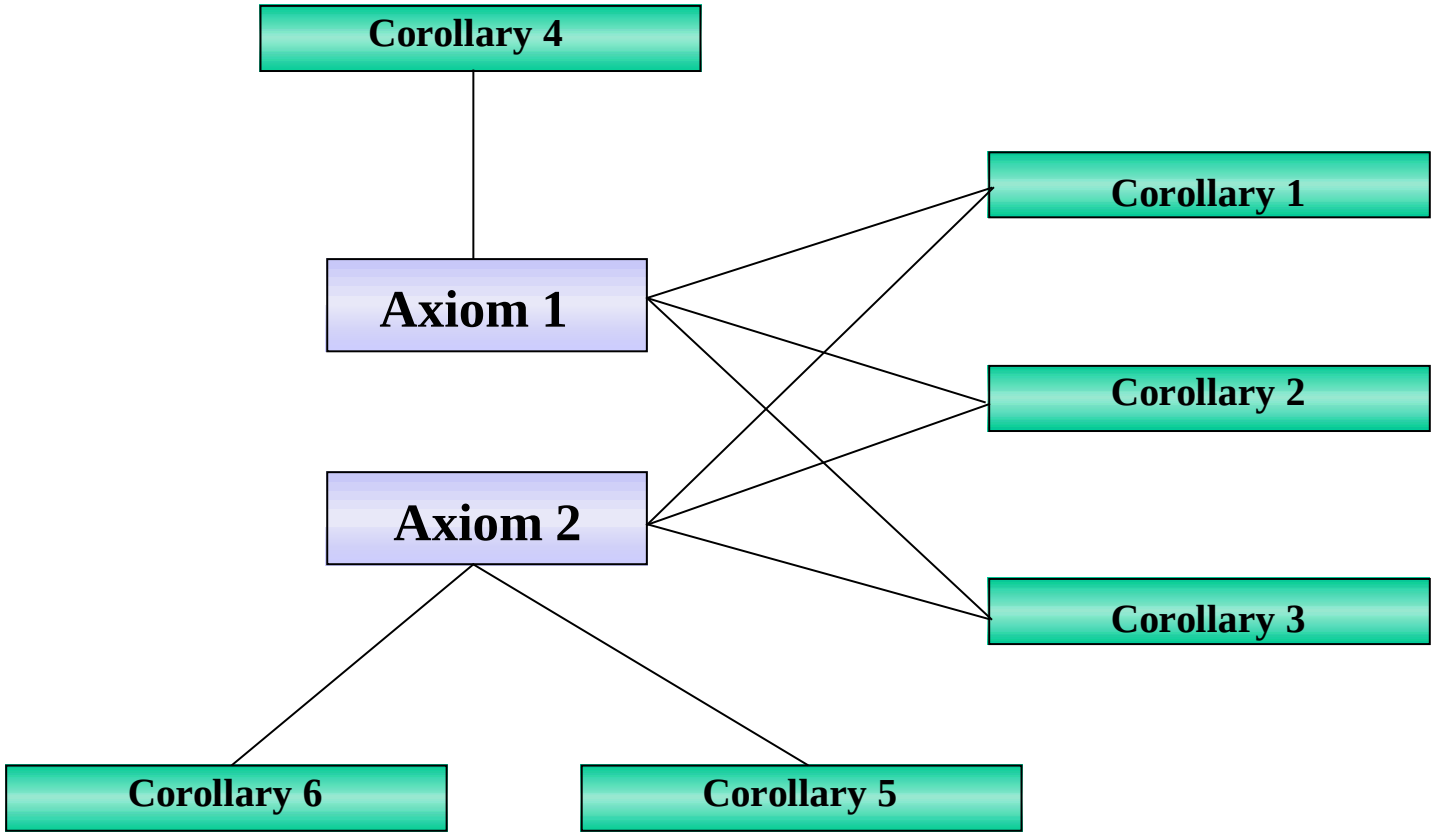
- **Axiom 1.** ***The independence axiom*****. Maintain the independence of components.**
- **Axiom 2.** ***The information axiom*****. Minimize the information content of the design.**

Occam's Razor

- **The best theory explains the known facts with a minimum amount of complexity and maximum simplicity and straightforwardness.**



The Origin of Corollaries



Corollaries

- **Corollary 1.** ***Uncoupled design with less information content*****.**
- **Corollary 2.** ***Single purpose*****. Each class must have single, clearly defined purpose.**
- **Corollary 3.** ***Large number of simple classes*****. Keeping the classes simple allows reusability.**

Corollaries (Con't)

- **Corollary 4. *Strong mapping*. There must be a strong association between the analysis's object and design's object.**
- **Corollary 5. *Standardization*. Promote standardization by designing interchangeable components and reusing existing classes or components.**

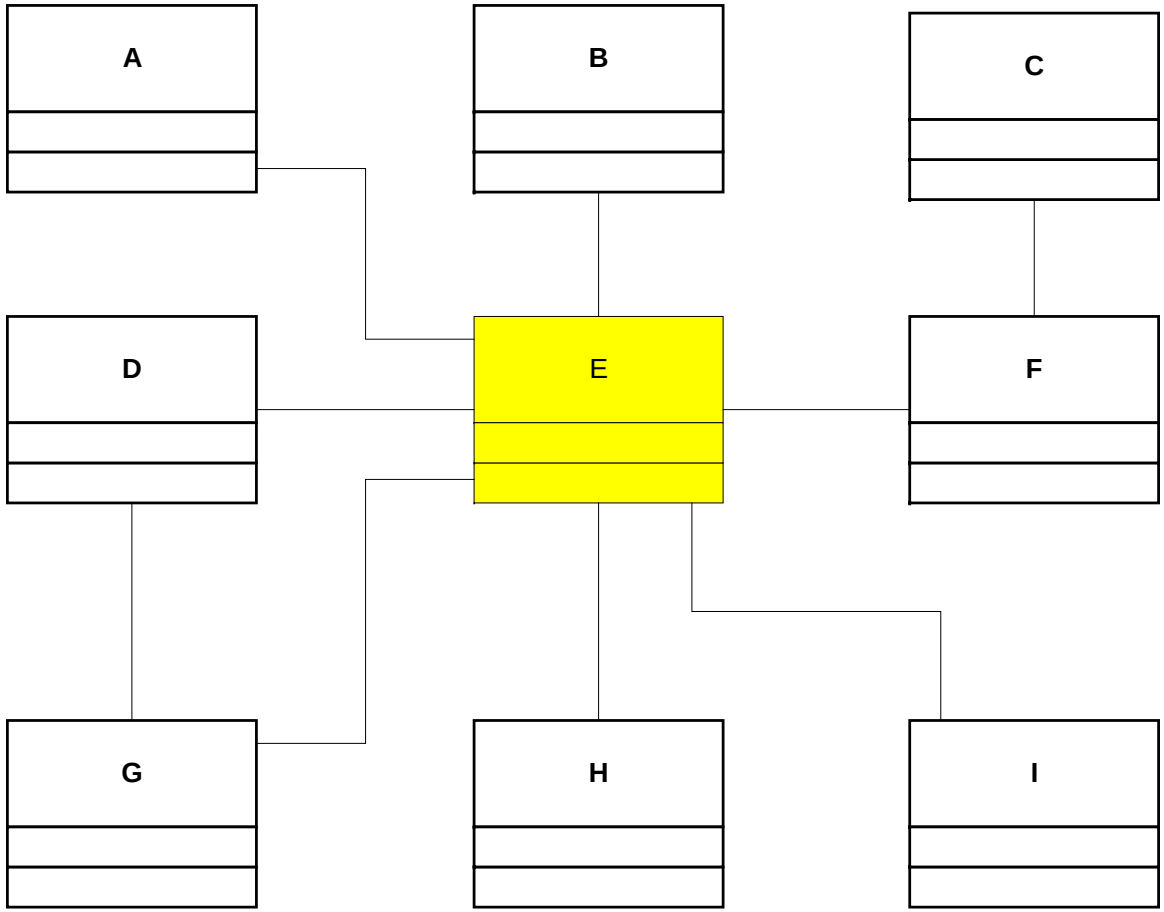
Corollaries (Con't)

- **Corollary 6.** ***Design with inheritance*****. Common behavior (methods) must be moved to superclasses.**
- **The superclass-subclass structure must make logical sense.**

Coupling and Cohesion

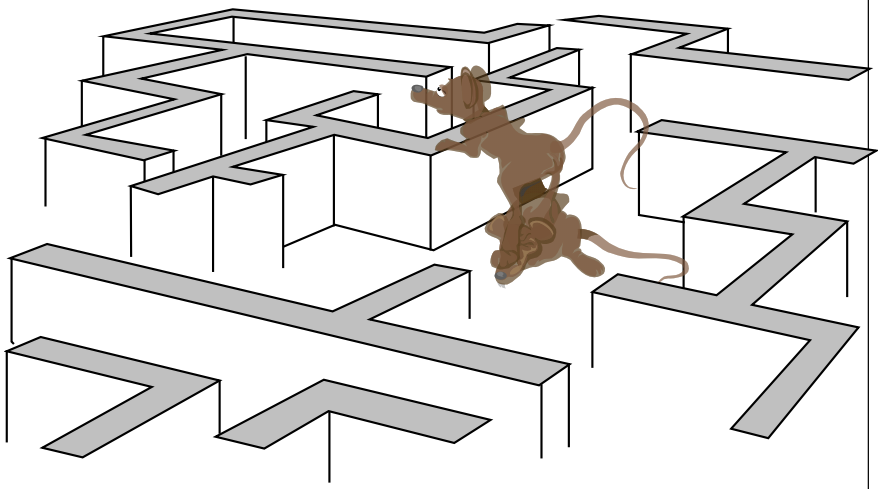
- **Coupling is a measure of the strength of association among objects.**
- **Cohesion is interactions within a single object or software component.**

Tightly Coupled Object



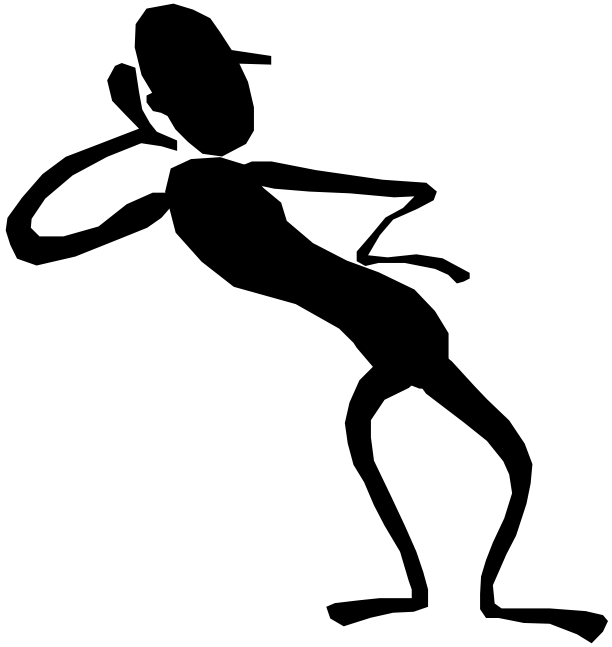
Corollary 1- Uncoupled Design with Less Information Content

- **The main goal here is to maximize objects (or software components) cohesiveness.**



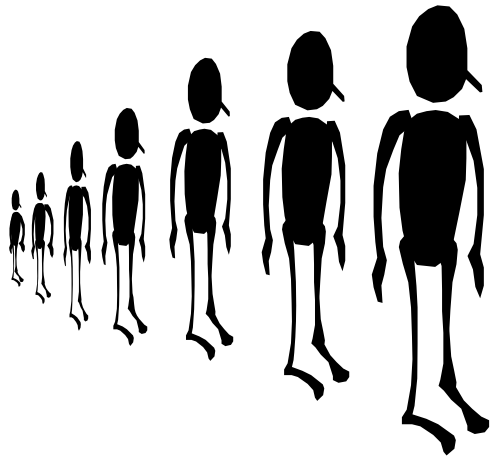
Corollary 2 - Single Purpose

- **Each class must have a purpose, as was explained in a previous chapter.**
- **When you document a class, you should be able to easily explain its purpose in a sentence or two.**



Corollary 3- Large Number of Simpler Classes, Reusability

- **A great benefit results from having a large number of simpler classes.**
- **The less specialized the classes are, the more likely they will be reused.**



Corollary 4. Strong Mapping

- **As the model progresses from analysis to implementation, more detail is added, but it remains essentially the same.**
- **A strong mapping links classes identified during analysis and classes designed during the design phase.**

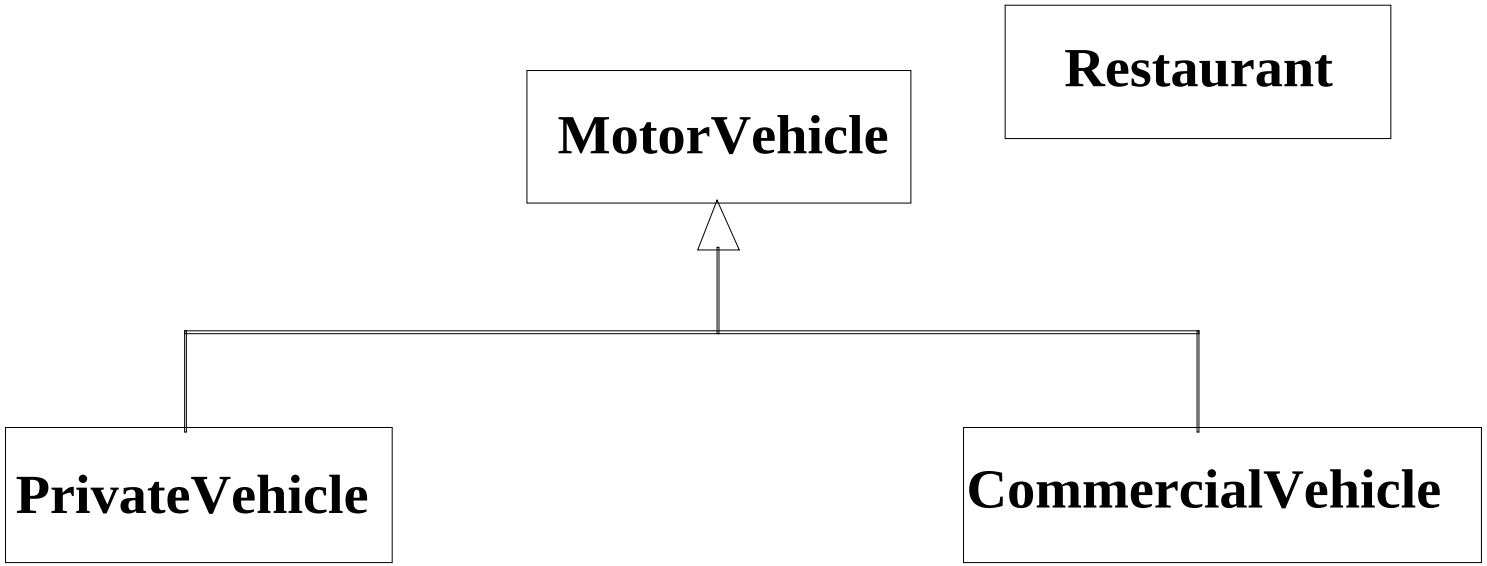
Corollary 5. Standardization

- **The concept of design patterns might provide a way for standardization by capturing the design knowledge, documenting it, and storing it in a repository that can be shared and reused in different applications.**

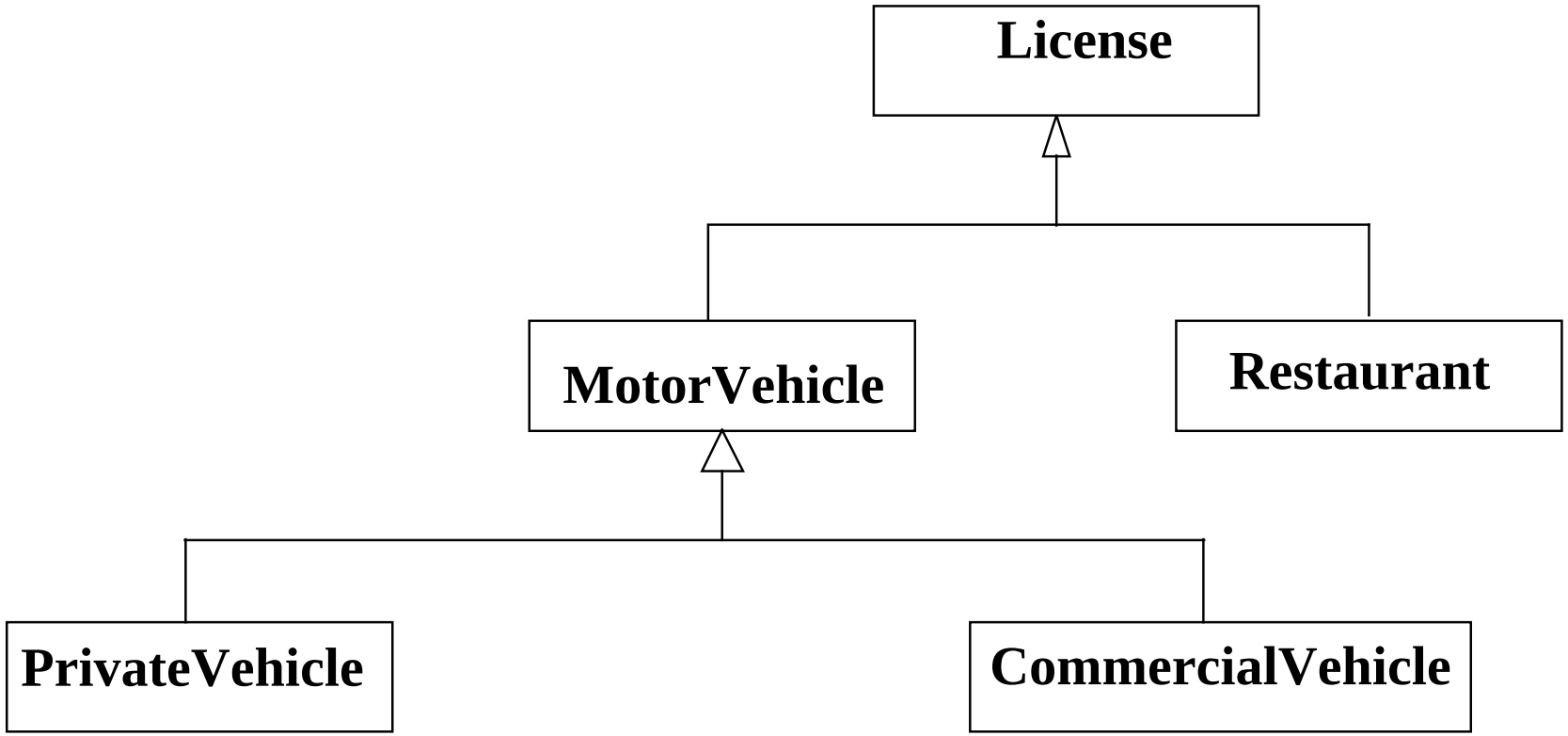
Corollary 6. Designing with Inheritance

- **Say we are developing an application for the government that manages the licensing procedure for a variety of regulated entities.**
- **Let us focus on just two types of entities: motor vehicles and restaurants.**

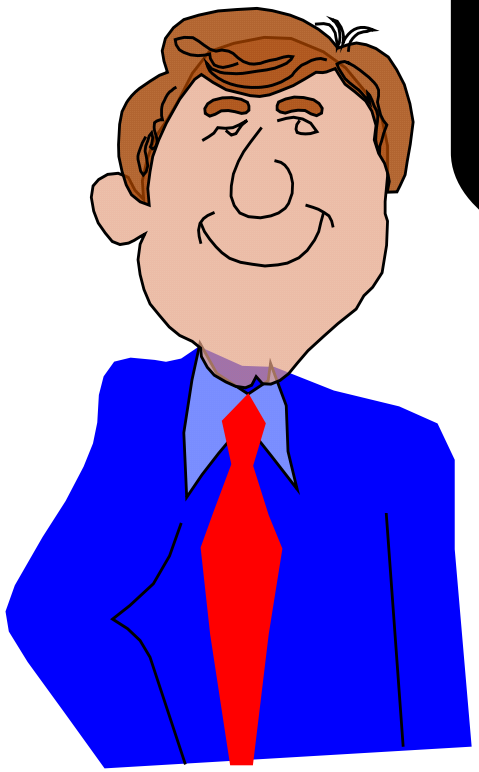
Designing With Inheritance (Con't)



Designing With Inheritance (Con't)



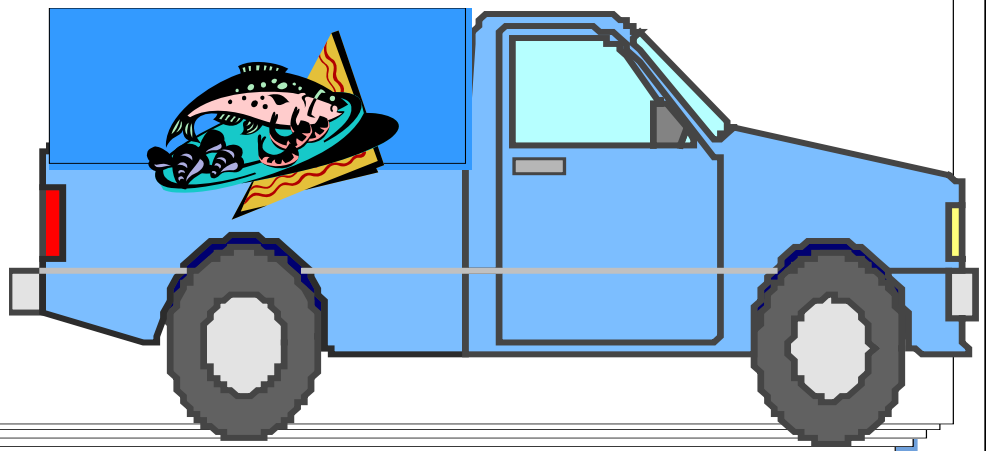
Six Months Later



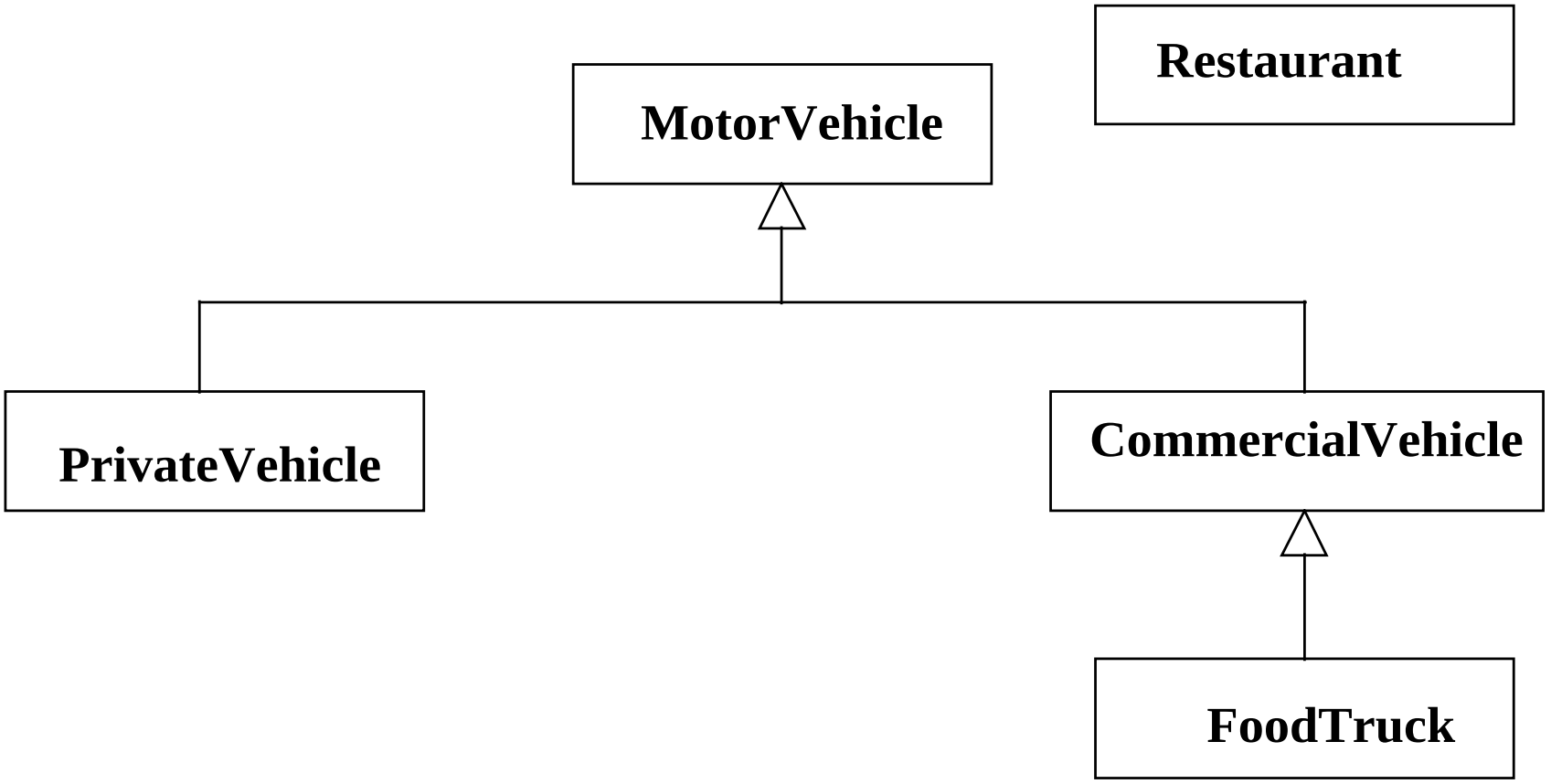
"What about coffee wagons, food trucks, and ice cream vendors? We're planning on licensing them as both restaurants and motor vehicles."

Designing With Inheritance
Weak Formal Class

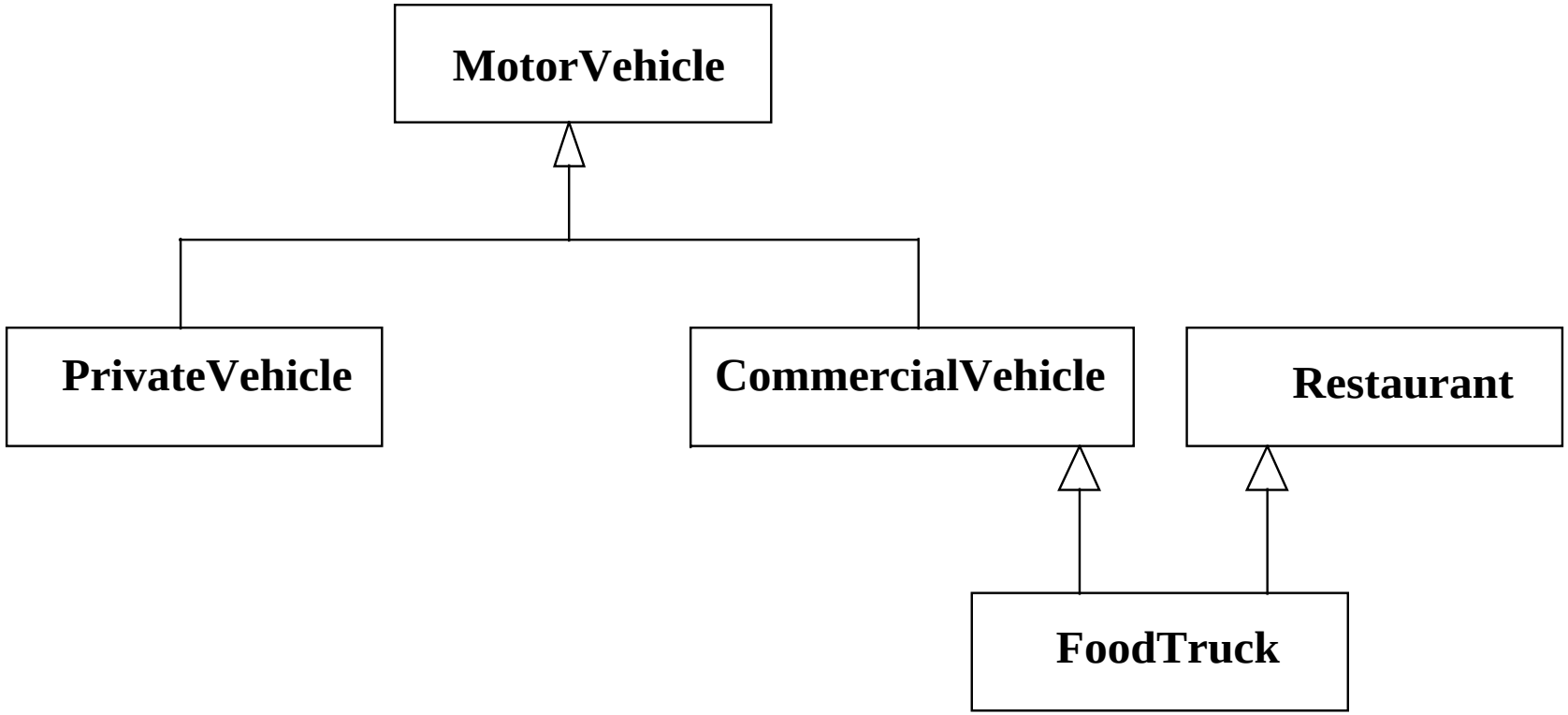
- **MotorVehicle and Restaurant classes do not have much in common.**
- **For example, of what use is the gross weight of a diner or the address of a truck?**



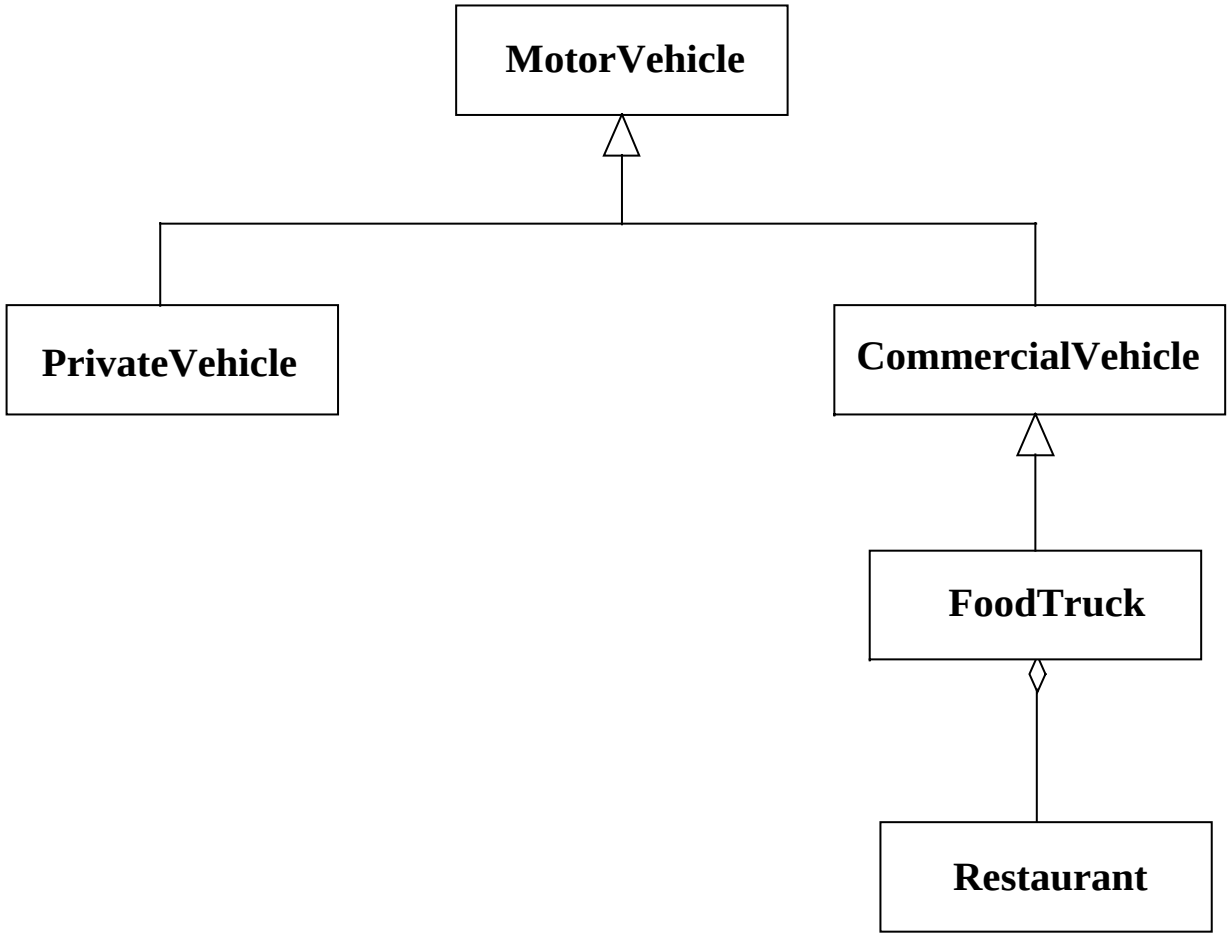
Designing With Inheritance (Con't)



Designing With Inheritance (Con't)



Achieving Multiple Inheritance using Single Inheritance Approach

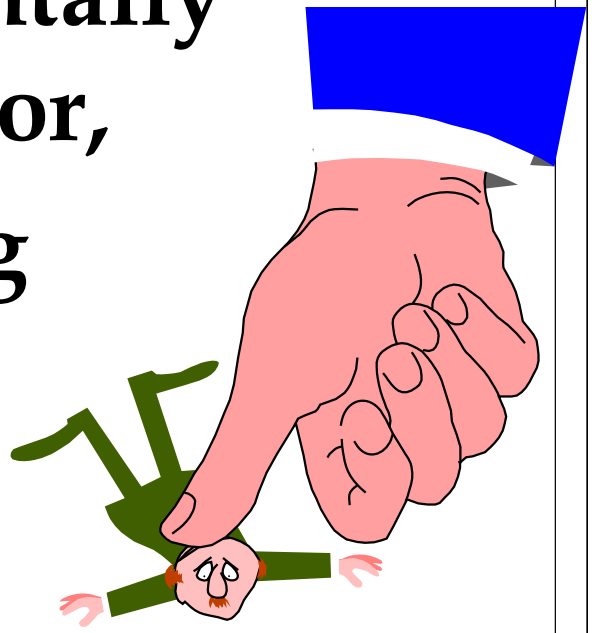


Avoid Inheriting Inappropriate Behaviors

You should ask the following questions:

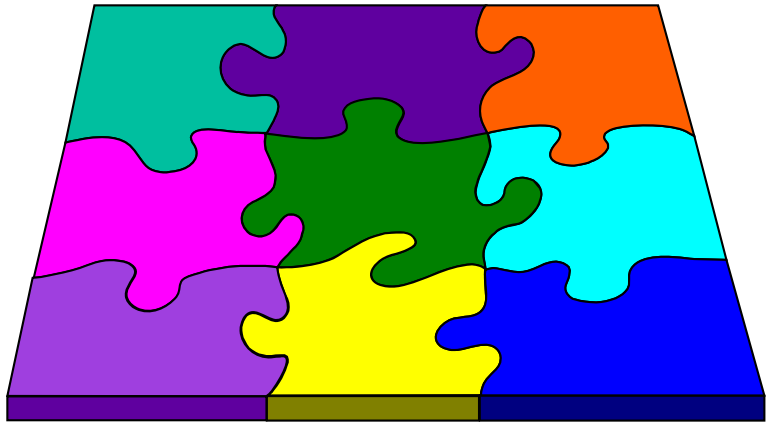
I. Is the subclass fundamentally similar to its superclass? or,

II. Is it entirely a new thing that simply wants to borrow some expertise from its superclass?

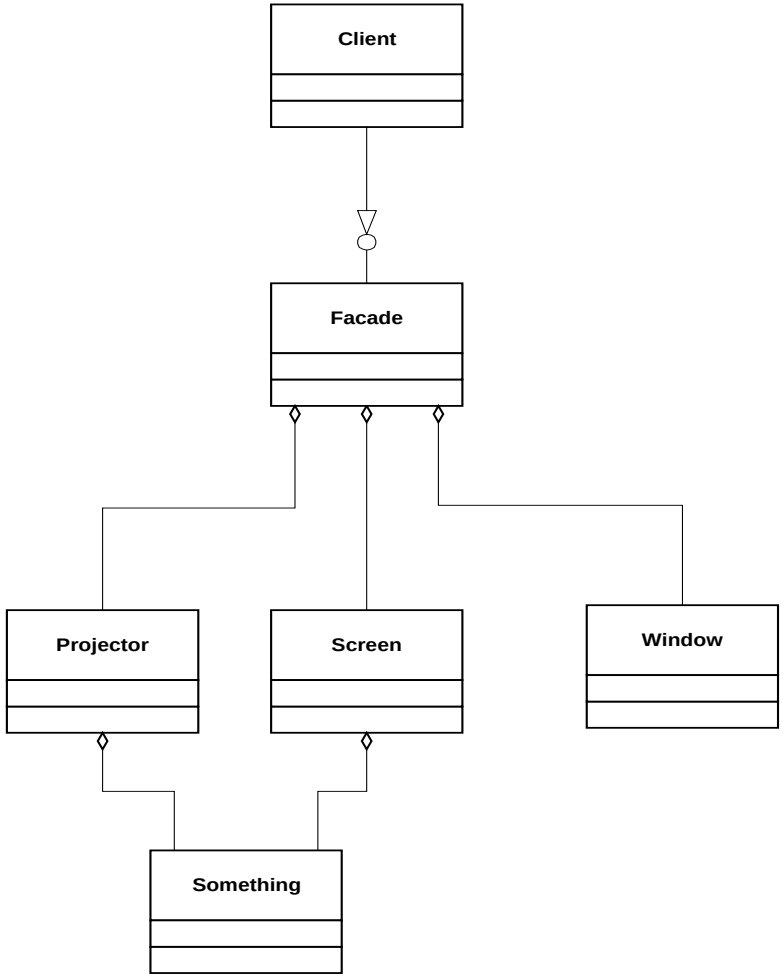


Design Patterns

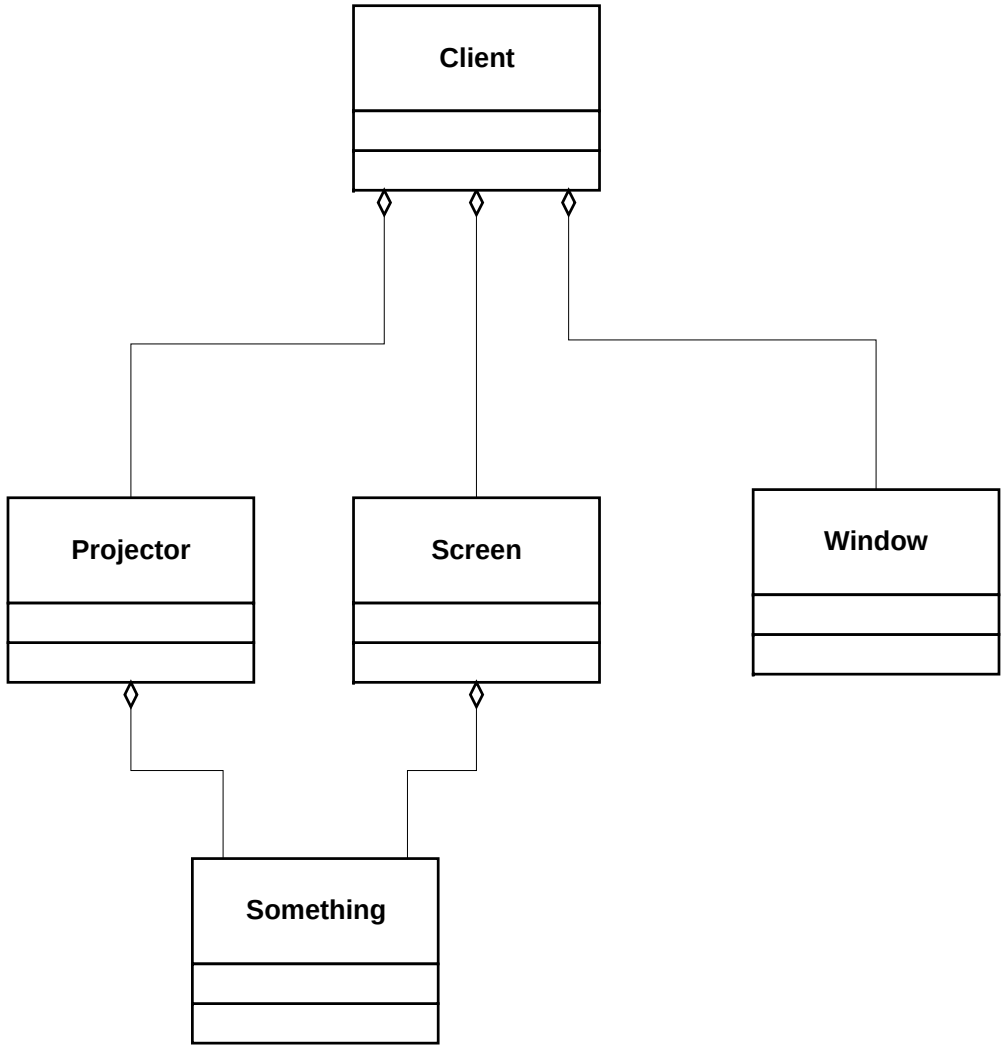
- **Patterns provide a mechanism for capturing and describing commonly recurring design ideas that solve a general design problem.**



Design Patterns (Con't)



Design Patterns (Con't)



Summary

- **We studied the object-oriented design process and axioms.**
- **The two design axioms are**
  - **Axiom 1. The independence axiom. Maintain the independence of components.**
  - **Axiom 2. *The information axiom. Minimize the information content of the design.***

Summary (Con't)

- **The six design corollaries are**
  - **Corollary 1. Uncoupled design with less information content.**
  - **Corollary 2. Single purpose.**
  - **Corollary 3. Large number of simple classes.**
  - **Corollary 4. Strong mapping.**
  - **Corollary 5. Standardization.**
  - **Corollary 6. Design with inheritance.**

Summary (Con't)

- **We also studied the concept of design patterns, which allow systems to share knowledge about their design.**