Design Patterns

- The Composite Pattern
- The Strategy Pattern
- The Union Pattern
1. **Objects are the only things that can perform computations.**

2. **Encapsulate data that which varies (a variant) into a class, and make all related variants into concrete subclasses of an “abstract class.”**

3. **Program to the interface (or abstract class).**

• E.g., `Shape.delete()`, where `Shape` is A `Shape`, not `Rectangle` or `Circle`.

• E.g., `Rectangle` and `Circle` extend `Shape`.

**Object-Oriented Programming Principles**

February 4, 2000
This is an example of the simplest yet most fundamental OO design which has the abstract capability of computing its area. 

Circle, and make their subclasses of an abstract class, called AShape, and make them subclasses of AShape, such as Rectangle and circles, etc.

The variants for this problem are the infinitely many shapes: rectangles, these areas.

Suppose I face the problem of computing the areas of geometrical shapes such as rectangles and circles.

The Union Pattern

#0 and #0 #1.

pattern called the Union Pattern. It is the result of applying #0 #1.

- #0 #1 drives me to define concrete classes such as Rectangle and circles, etc.

- The variants for this problem are the infinitely many shapes: rectangles, these areas.

- Suppose I face the problem of computing the areas of geometrical shapes such as rectangles and circles.

The Union Pattern
The union of these subsets equals the superset.

- The union of the above superset, subsets of the above superset, several concrete subclasses (Variant1, Variant2) representing disjoint
  - of interest, an abstract class (AClass) representing the superset of all the objects
  - problem domain into disjoint subsets and consists of

The Union Pattern is the result of partitioning the sets of objects in the

The Union Pattern (cont.)
Reducing code complexity and making the code easier to maintain.

- Conditional statements to distinguish the various cases are gone.

- Concrete instances it is working with.

- The client class should only concern itself with the public methods of AClass and should not need to check for the class type of the (Variant1, Variant2), but should only see them as AClass objects.

- A client of the Union Pattern uses instances of the concrete subclasses.

The Union Pattern (cont.)
The Union Pattern (cont.)
The pizza uses its shape as a "strategy" to compute its area.

- This is an example of what is called the Strategy Pattern.
- It only knows that its shape is capable of computing the appropriate computation of its area to its shape.
- It does not care what the exact type of its shape is.
- Recall the Pizza problem. The pizza has a shape and delegates the

The Strategy Pattern
A shape plays the role of the (abstract) strategy.

- In our Pizza example, the context is the Pizza class, and the abstract strategy delegates the work to this strategy reference.

- The context delegates the work to this strategy reference.

- Strategy in the union.

- In general, the Strategy pattern consists of a union pattern of strategies.

The Strategy Pattern (cont.)
The Strategy Pattern (cont.)
This design pattern is called the **Composite Pattern**.

- **The recursive object structural design gives rise to recursive algorithms**.
- **A polynomial is an example**.
- **Composition, in particular, is a common object design**.
- **Often, we combine (or compose) objects to form new objects. Recursive**.

The Composite Pattern
The Composite Pattern (cont.)
The method operator() for Composite is mostly recursive.

- In the previous diagram, classes Basic1 and Basic2 correspond to the non-base cases of the recursion, and Composite corresponds to the Composite Pattern (cont.)