means a class can represent any of its subclasses. Any of its subclasses, but not the other way around. Polymorphism

— In general, a variable of a superclass can be assigned an instance of a subclass at any time in a program. A shape is said to be polymorphic.

A variable of class Ashape can be assigned any instance of subclasses of Ashape:

```
circle u = new Rectangle(5, 6); // OK
```

```
t = s; // OK, the old Rectangle is gone.
```

```
Ashape t = new Rectangle(3, 4); // OK.
```

```
Ashape s = new Circle(2.7); // OK.
```

Polymorphism
1. Objects are the only things that can perform computations.

2. Encapsulate 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` is a shape, `Rectangle` or `Circle` are shapes.

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

Object-oriented Programming Principles
This is an example of the simplest yet most fundamental OO design which has the abstract capability of computing its area. Circle, and make them subclasses of an abstract class, called AShape, 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.

#0 and 00pp #1.

The Union Pattern

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

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

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

  (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.)
What are the (basic) operations supported on a list? • An ordered collection of zero or more objects.

A Scheme-like list
public abstract ArrayList getRest() // Returns the tail ("rest") of this ArrayList, if any.

public abstract Object getFirst(); // Returns the first object in this ArrayList, if any.

public abstract class ArrayList

A Scheme-like List (cont.)
...}
{
    return -rest;
}

public ArrayList getRest()
    // Returns the tail ("rest") of this ArrayList.
    {
    return -first;
}

public Object getFirst()
    // Returns the first object in this ArrayList.
    {
    return -first;
}

private ArrayList rest -rest;
private Object first -first;

public class NELIST extends ArrayList
{
    A Scheme-like List (cont.)}
public class EmptyList extends List

// Throws an IllegalArgumentException
throw new IllegalArgumentException("Empty List has no tail.");

// Throws an IllegalArgumentException
throw new IllegalArgumentException("Empty List has no data.");

A Scheme-like List (cont.)
public abstract int helpGetLength(int acc);
  // in the Altist.
  // Returns the sum of ,,acc,, and the number of elements

public abstract int helpGetLength();
  // the "helper" method helpGetLength()
  // Returns the number of elements in this Altist.

  ...

public abstract class Altist

A Scheme-like List (cont.)
{ return rest.head(accumulate(acc + 1)); }

public int head(accumulate(int acc)
  // tail for help to compute the length.
  adds I to the accumulated length and pass it down to the

{ return rest.head(accumulate(1)); }

public int getLength()
  // passing it an accumulated length of 1.
  asks the tail for help to compute the length,
  ...
}

public class NELList extends ArrayList

A Scheme-like List (cont.)
A Scheme-like List (cont.)