3. Polymorphism :: Why is Polymorphism Useful?

Polymorphism is a feature of all object oriented programming languages (Java, C++, Smalltalk, Objective C, etc) and a programming language cannot be considered an object oriented programming language (OOPL) if it does not support polymorphism.

Remember that inheritance facilitates **code reuse**. The design and implementation of our *Square* class was simplified by declaring *Square* a subclass of *Rectangle*. The design and implementation of our *Rectangle* class was simplified by declaring *Rectangle* a subclass of *Shape*. All OOPL's must support inheritance and inheritance, method overriding, and polymorphism go hand-in-hand.

Essentially, polymorphism is useful because it makes an inheritance hierarchy more easily extensible. To see why, consider this example where we will assume that Java does not support polymorphism.

Suppose we declare and create an *ArrayList* object *shapes* to store various *Shape* objects, i.e., *Rectangle*s, *Square*s, *Oval*s, etc:

```
ArrayList<Shape> shapes = new ArrayList<>();
```

We add various *Shape*s to *shapes*:

```
shapes.add(new Square(10, 20, 30));
shapes.add(new Rectangle(1, 2, 3, 4));
shapes.add(new Square(50, 70, 90));
shapes.add(new Oval(200, 300, 100, 50));
shapes.add(new Rectangle(3, 7, 11, 13));
```

3. Polymorphism :: Why is Polymorphism Useful (continued)?

Now suppose we write a *WindowManager* class that contains method to iterate over the elements of *shapes* and draws each *Shape* on the graphical window:

```
public class WindowManager() {
  private ArrayList<Shape> shapes;
  public void redrawWindow() {
    for (Shape shape : shapes) {
      if (shape instanceof Square) drawSquare((Square)shape);
      else if (shape instanceof Rectangle) drawRectangle((Rectangle)shape);
      else if (shape instanceof Oval) drawOval((Oval)shape);
    }
  }
  private void drawSquare(Square pSquare) { ... }
  private void drawRectangle(Rectangle pRect) { ... }
  private void drawOval(Oval pOval) { ... }
  ...
}
```

The type casts in *redrawWindow*() are necessary because the class of the *shape* object variable is *Shape* but the drawing methods require a subclass object to be passed. Remember that a superclass object may not be substituted for a subclass object.

Now assume in the next version of our painting program we want to add additional drawing shapes, in particular, *Line*, *Point*, and *Triangle*.

3. Polymorphism :: Why is Polymorphism Useful (continued)?

We design and implement the new classes, making all of them subclasses of *Shape*. We write methods *drawLine*(*Line*), *drawPoint*(*Point*), and *drawTriangle*(*Triangle*) to draw each type of shape on the graphical window. Almost done; now we just have to modify the **if** statement in *redrawWindow*():

```
public class WindowManager() {
  private ArrayList<Shape> shapes;
  public void redrawWindow() {
    for (Shape shape : shapes) {
      if (shape instanceof Square) drawSquare((Square)shape);
      else if (shape instanceof Rectangle) drawRectangle((Rectangle)shape);
      else if (shape instanceof Oval) drawOval((Oval)shape);
      else if (shape instanceof Line) drawLine((Line)shape);
      else if (shape instanceof Point) drawPoint((Point)shape);
      else if (shape instanceof Triangle) drawTriangle((Triangle)shape);
    }
  }
  private void drawSquare(Square pSquare) { ... }
  private void drawRectangle(Rectangle pRect) { ... }
  private void drawOval(Oval pOval) { ... }
  private void drawLine(Line pLine) { ... }
  private void drawPoint(Point pPoint) { ... }
  private void drawTriangle(Triangle pTriangle) { ... }
  ...
```

3. Polymorphism :: Why is Polymorphism Useful (continued)?

Whew! That was a lot of work. Now suppose in Version 3 we are going to add five additional drawing shapes: *A*, *B*, *C*, *D*, *E* (I am running out of names for them). We design and implement all of the required classes and drawing methods and before finishing, we have to (once again) modify the **if** statement in *redrawWindow*():

```
public class WindowManager() {
  private ArrayList<Shape> shapes;
  public void redrawWindow() {
    for (Shape shape : shapes) {
      if (shape instanceof Square) drawSquare((Square)shape);
      else if (shape instanceof Rectangle) drawRectangle((Rectangle)shape);
      else if (shape instanceof Oval) drawOval((Oval)shape);
      else if (shape instanceof Line) drawLine((Line)shape);
      else if (shape instanceof Point) drawPoint((Point)shape);
      else if (shape instanceof Triangle) drawTriangle((Triangle)shape);
      else if (shape instanceof A) draw((A)shape);
      else if (shape instanceof B) draw((B)shape);
      else if (shape instanceof C) draw((C)shape);
      else if (shape instanceof D) draw((D)shape);
      else if (shape instanceof E) draw((E)shape);
    }
  }
```

3. Polymorphism :: Why is Polymorphism Useful (continued)?

```
    private void drawSquare(Square pSquare) { ... }
    private void drawRectangle(Rectangle pRect) { ... }
    private void drawOval(Oval pOval) { ... }
    private void drawLine(Line pLine) { ... }
    private void drawPoint(Point pPoint) { ... }
    private void drawTriangle(Triangle pTriangle) { ... }
    private void drawA(A pA) { ... }
    private void drawB(B pB) { ... }
    private void drawC(C pC) { ... }
    private void drawD(D pD) { ... }
    private void drawE(E pE) { ... }
  }
```

I hope you will agree with me that *redrawWindow*() is starting to look a bit ugly with that huge **if** statement and all of those type casts. Furthermore, in each new version of *redraw Window*() I have been copying-and-pasting a lot of code to add the new functionality. And remember: when you find yourself copying-and-pasting code ... there is usually a better way to do what you are doing.