13.03 Virtual Lecture Notes

If we were to write a polymorphic method for our rectangle and box classes, the code would look like this:

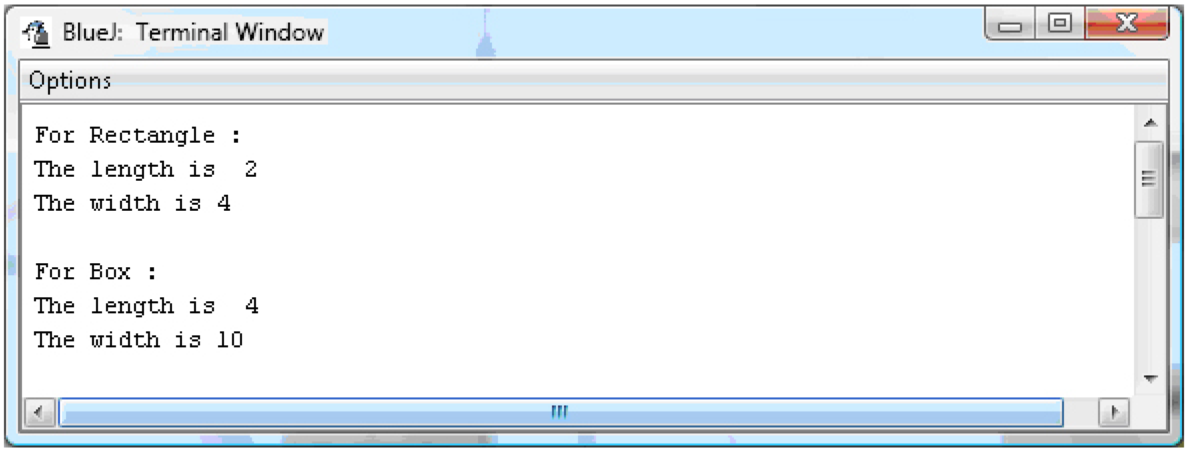
```
public void polyMorph(Rectangle r)
{

    System.out.println("For " + r.getClass().getName() + " : ");
    System.out.println("The length is  " + r.getLength() );
    System.out.println("The width is " + r.getWidth());
}
```

If this method is involved in a program, it can apply to objects of Rectangle, Box, or cube types. Note that polyMorph cannot refer to methods that Rectangle does not have! The **`getClass()`** and **`getName()`** methods are inherited from the Object class, which is a superclass for every class created in Java! With **`r.getClass()`**, we will get the actual class of the parameter **`r`**, and **`r.getClass().getName()`** will give use the actual name of the class. This will help us verify that polymorphism is taking place.

- Download the Rectangle.java file to your module 13 Lessons folder and open it.
- Download the Box.java file to your module 13 Lessons folder and open it.
- Download the Test2.java file to your module 13 Lessons folder and run it.

The output from running the test2.java demo program will be:



```
For Rectangle :
The length is  2
The width is 4

For Box :
The length is  4
The width is 10
```

Notice that the output tells us which object was used in each call. Just remember that no reference can be made to **`getHeight()`** in **`polymorph()`**, as that is not part of the Rectangle class.

Also, we could have used the **`toString()`** method of Object if we wanted a direct reference to the object passed (instead of just its name). You could do that by switching:

```
System.out.println(“For “ + r.toString()); or
System.out.println(“For “ + r);
```

with:

```
r.getClass().getName();
```

Experiment with the demo programs until you are confident that you understand what polymorphism is doing in this example.