09.03 Virtual Lecture Notes

Let's use the shapes example using rectangles and boxes to explore polymorphism.

In the tester class, if a polymorphic static method is written for the rectangle and box classes, the code would look like this:

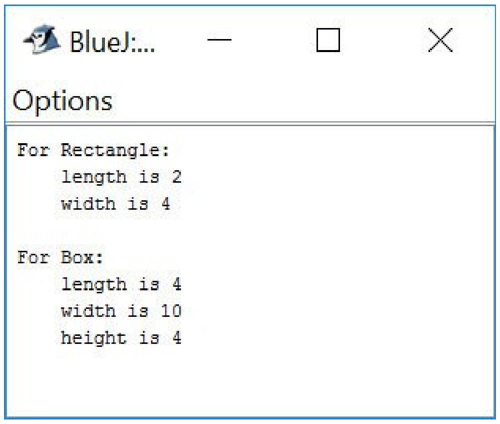
```
public static void polyMorph(Rectangle2 r)
{
   System.out.println("For " + r.getName() + ": ");
   System.out.println(" length is " + r.getLength());
   System.out.println(" width is " + r.getWidth());
}
```

When the method is involved in the `main` method, it can apply to objects of the `Rectangle2` or `Box2` classes.

Note that the `polyMorph` method cannot call methods the `Rectangle2` class does not have!

The `getName` method is used to get the name of each object. For an instance of the `Rectangle2` class, we want the method to return just `Rectangle2`.

Run the program and look at the output. Do you see anything odd? In the client class, were two rectangles declared or only one? One of the objects should be a box. But the output says both are rectangles.



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

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

To fix this, go to the `Box2` class and uncomment the `getName` method. Now run the program again.

This time, the output displays information for one rectangle and one box! Java knows which `getName` method to invoke based on the type of object declared.

The output shows which object was used in each call to the `polyMorph` method.

Look at the code in the client class. Notice that no reference can be made to the `getHeight` method in the `polyMorph` method, as that is not part of the `Rectangle2` class. You may be tempted to call `getHeight` in the `main` method. The program statement commented out below shows how you may try. Uncomment to view the error message it produces.

```
polyMorph(two);
//System.out.println(" height is " + two.getHeight());
System.out.println(" height is " + ((Box2)two).getHeight() );
```

Instead, since the object needs to be a `Box2` to access the `getHeight` method, we can utilize casting. This is demonstrated in the second print statement in the code above.

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

Print