09.04 Virtual Lecture Notes

What happens when you try to print an object of the `Rectangle3` class? Try it by running the `ShapesTester3` program. The print statement simply asks for a `Rectangle3` object to be printed.

Notice the output displays values such as `Rectangle3@c21495` and `Box3@1d5550d`.

This looks very cryptic! What you see is a reference to the memory location where the object's values are stored in the computer's memory. A method named `toString` of the `Object` class is called when the print statement is executed.

What if we could override the `toString` method? Wouldn't it be nice to control the values displayed when an object is printed? For instance, what if we could print the class name followed by the dimensions for a rectangle or box? Well, we can!

To accomplish this, the `toString` method of the `Object` class needs to be overridden.

Adding a `toString` method to the `Rectangle3` class, will look like this:

```
public String toString()
{
     return "The rectangle's dimensions are " + length + " X " +
width;
}
```

In the `Rectangle3` class, uncomment the code for the `toString` method. Run the project. Notice the output now displays the dimensions of the rectangle. This provides more useful information than the default `toString` method we would have inherited from the `Object` class.

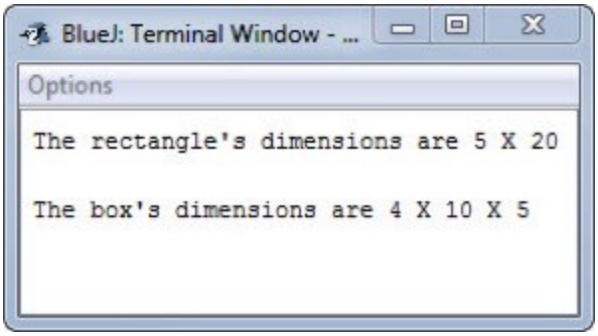
The `Rectangle3` class `toString` method takes no parameters and returns a `String` value. The returned value is a description of a rectangle, including its length and width.

The question remains: what happens with a `Box3` object? The output currently it is a rectangle and only has two dimensions. Why? Since we have not added a `toString` method to the `Box3` class, we would get a message starting with, "The rectangle." Since a `Box3` object is a `Rectangle3`, the `toString` method in `Rectangle3` is used. Obviously, this is not what we want. The fix is easy! Override

`toString` again.

Uncomment the `toString` method in `Box3` and run the project again.

The output generated should look like the following:



Polymorphism and overriding can be used in combination. For example, the `showEffectBoth` method is created to demonstrate the usage of both polymorphism and overriding.

```
public static void showEffectBoth(Rectangle3 r)
{
     System.out.println(r);
}
```

Notice the method is polymorphic and makes use of the `toString` method overriding in its call to `System.out.println()`.

In the `main` method, uncomment the call to `showEffectBoth` and run the program again. The output should look the same.

Print