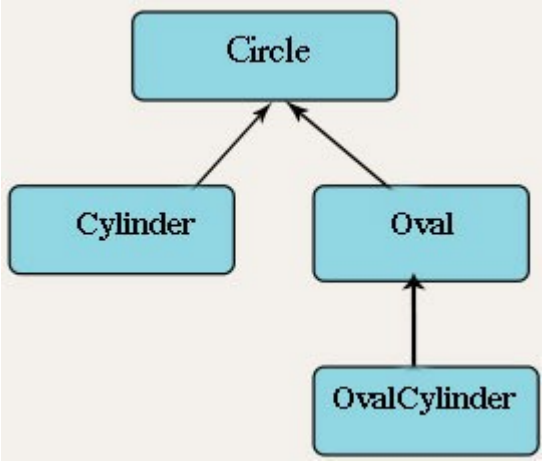
09.02 Virtual Lecture Notes



Class inheritance can be modeled using classes related to round shapes such as a circle and oval.

Take a moment to look over the `Circle` class. Now examine its subclasses.

For this example, the `Circle` class has been kept simple. `Circle` has instance variables for the `x` and `y` values of the center point, and the radius.

`Cylinder` extends `Circle` and adds height.

`Oval` extends `Circle` and adds on a second radius.

The `OvalCylinder` extends `Oval` and adds height.

In the `Circle` class, notice the public methods are `getRadius` and `getCenter`.

When `Circle` is extended to create the `Cylinder` class, it will need to include a call to the `super` method as the first statement of the constructor. Since the `Cylinder` class adds a height attribute, it needs an instance variable `height` and the method `getHeight`.

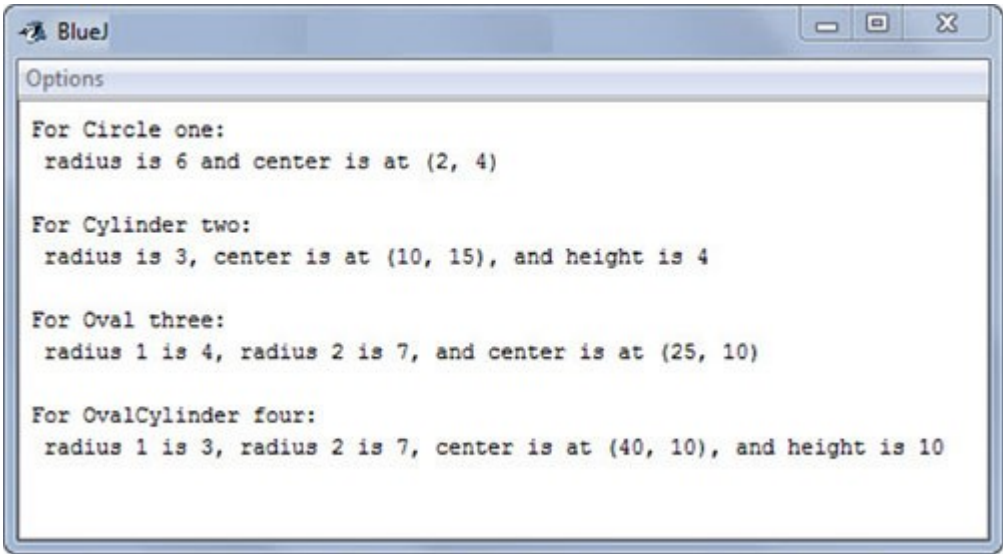
Class `Oval` extends `Circle`, as well; it adds a second radius. This will require an instance variable `radius2` and the method `getRadius2`.

Finally, `OvalCylinder` will extend the `Oval` class. It will add one attribute for height. This will require an instance variable `height` and the method `getHeight`.

Note that `OvalCylinder` extends `Oval` and not `Circle`. This means that `OvalCylinder` is the direct subclass of `Oval` and `Oval` is the direct subclass of `Circle`. However, both `OvalCylinder` and `Oval` are subclasses of `Circle`.

That is all there is to creating a class hierarchy. First, you create a diagram indicating the relationships between your classes. Second, you write the classes, making sure that each class extends the correct class from your diagram.

Run the `CircleTester1` program. The output should look similar to the image below:



Make changes and explore the classes to see how they behave.

Print