22. Inheritance :: Declaring an Abstract Class

All it takes to make *Shape* an abstract class is:

```
public abstract class Shape {
  ...
}
```

Once *Shape* is declared abstract, it is no longer legal to instantiate a *Shape* object:

```
public class ShapeCreate {
  public static void main(String[] args) {
    Shape newShape = new Shape(10, 20);
  }
}
```

When I build this code I get a syntax error:

```
ShapeCreate.java:3: error: Shape is abstract; cannot be instantiated
      Shape newShape = new Shape(10, 20);
                       ^
1 error
```

22. Inheritance :: Abstract Methods

Currently our abstract *Shape* class does not contain any abstract methods. As an example of an abstract method, suppose wish to add to the various shape-related classes a *draw*() method which will be called on an object to draw the shape on the graphical window.

```
public Main() {
  public static void main(String[] args) {
    Rectangle rect = new Rectangle(10, 20, 40, 50);
    Oval oval = new Oval(5, 73, 18, 92);
    Square sq = new Square(230, 470, 55);
    rect.draw();
    oval.draw();
    sq.draw();
  }
}
```

22. Inheritance :: Abstract Methods (continued)

Can *draw*() be implemented in the abstract *Shape* class, i.e., is this okay?

```
public abstract class Shape {
  ...
  void draw() {
    // Code is here that draws the Shape on the graphical window.
  }
}
```

How would we implement *draw*()? If you think about it, drawing a *Shape* is impossible because a *Shape* can have many different forms. Drawing a *Rectangle* is not the same as drawing an *Oval*. They look different on the window and the Java Class Library methods we call to draw a rectangle and an oval are different. It makes sense that *draw*() **cannot** be implemented in *Shape* and **must** be implemented in every subclass of *Shape*, i.e., *Rectangle* will have a *draw*() method that gets called to draw rectangles on the window, *Oval* will have a *draw*() method that gets called to draw ovals on the window. *Line* will have a *draw*() method that gets called to draw lines on the window.

22. Inheritance :: Abstract Methods (continued)

To specify that *draw*() is not implemented in *Shape* and that *draw*() **must** be implemented in every subclass of *Shape* we make *draw*() an **abstract method** in *Shape*. This is done by declaring the method but not providing a method body.

```
public abstract class Shape {
  ...
  public void draw();  // No method body means draw() is an abstract method.
}
```

Abstract methods **must** be implemented in subclasses and if not, the subclass itself becomes an abstract class. This rule is how the compiler forces the subclasses to implement *draw*().