Object Oriented Programing(OOP) Concepts

Av: Jonas Andrée

Constructor

Modify the class `Circle` to include a third constructor for constructing a `Circle` instance with two arguments - a `double` for `radius` and a `String` for `color`.

```
// 3rd constructor to construct a new instance of Circle with the given radius and color
public Circle (double r, String c) { ..... }
```

Modify the test program `TestCircle` to construct an instance of `Circle` using this constructor.

Getter

Add a getter for variable `color` for retrieving the `color` of this instance.

```
// Getter for instance variable color
public String getColor() { ..... }
```

Modify the test program to test this method.

public vs. Private

In `TestCircle`, can you access the instance variable `radius` directly (e.g., `System.out.println(c1.radius)`); or assign a new value to `radius` (e.g., `c1.radius=5.0`)? Try it out and explain the error messages.

Setter

Is there a need to change the values of `radius` and `color` of a `Circle` instance after it is constructed? If so, add two `public` methods called *setters* for changing the `radius` and `color` of a `Circle` instance as follows:

```
// Setter for instance variable radius
public void setRadius(double newRadius) {
   radius = newRadius;
}

// Setter for instance variable color
public void setColor(String newColor) { ..... }
```

Modify the `TestCircle` to test these methods, e.g.,

```
Circle c4 = new Circle();   // construct an instance of Circle
c4.setRadius(5.0);          // change radius
System.out.println("radius is: " + c4.getRadius()); // Print radius via getter
c4.setColor(......);        // Change color
System.out.println("color is: " + c4.getColor());   // Print color via getter

// You cannot do the following because setRadius() returns void,
// which cannot be printed.
System.out.println(c4.setRadius(4.0));
```

Keyword "this"

Instead of using variable names such as `r` (for `radius`) and `c` (for `color`) in the methods' arguments, it is better to use variable names `radius` (for `radius`) and `color` (for `color`) and use the special keyword "`this`" to resolve the conflict between instance variables and methods' arguments. For example,

```
// Instance variable
private double radius;

// Constructor
public Circle(double radius) {
   this.radius = radius;   // "this.radius" refers to the instance variable
                           // "radius" refers to the method's parameter
   color = .......
}

// Setter of radius
public void setRadius(double radius) {
   this.radius = radius;   // "this.radius" refers to the instance variable
                           // "radius" refers to the method's argument
}
```

Modify ALL the constructors and setters in the `Circle` class to use the keyword "`this`".

Method `toString()`:

Every well-designed Java class should contain a `public` method called `toString()` that returns a short description of the instance (in a return type of `String`). The `toString()` method can be called explicitly (via *`instanceName`*`.toString()`) just like any other method; or implicitly through `println()`. If an instance is passed to the `println(`*`anInstance`*`)` method, the `toString()` method of that instance will be invoked implicitly. For example, include the following `toString()` methods to the `Circle` class:

```
// Return a description of this instance in the form of
// Circle[radius=r,color=c]
public String toString() {
   return "Circle[radius=" + radius + " color=" + color + "]";
}
```

Try calling `toString()` method explicitly, just like any other method:

```
Circle c1 = new Circle(5.0);

System.out.println(c1.toString());   // explicit call
```

`toString()` is called implicitly when an instance is passed to `println()` method, for example,

```
Circle c2 = new Circle(1.2);

System.out.println(c2.toString());  // explicit call
System.out.println(c2);             // println() calls toString() implicitly,
same as above
System.out.println("Operator '+' invokes toString() too: " + c2);   // '+'
invokes toString() too
```

The final class diagram for the `Circle` class is as follows: