Java – OOP, Objects and Classes

4.01 Rectangle.java

Write a class called Rectangle.java that has two attributes of type double called length and width that represent the length and width of a rectangle respectively.

The class has 2 constructors, the no-arg sets the length to 2 and width to 1 and an initialization constructor.

Add setters, getters, toString(), getPerimeter() and getArea() methods.

Sample Data in RectangleRunner.java

```
// Create a 2x1 rectangle
Rectangle rectangle1 = new Rectangle();
out.println(rectangle1);

 // Create a 10x3 rectangle
Rectangle rectangle2 = new Rectangle(3, 10);
out.println(rectangle2);

// Create a 4.5 by 1.5 rectangle
Rectangle rectangle3 = new Rectangle(1.5, 4.5);
out.println(rectangle3);

 // Modify rectangle
rectangle3.setLength(15);
rectangle3.setWidth(15);
out.println(rectangle3);
```

Sample Execution

```
Rectangle{length=1.0, width=2.0, perimeter=6.0, area=2.0}

Rectangle{length=3.0, width=10.0, perimeter=26.0, area=30.0}

Rectangle{length=1.5, width=4.5, perimeter=12.0, area=6.75}

Rectangle{length=15.0, width=15.0, perimeter=60.0, area=225.0}
```

4.02 Circle.java

Write a class called Circle.java that has one attribute of type double called radius.

The class has 2 constructors, the no-arg (default or non-parameterized) sets the radius to 1 and an initialization constructor.

Use Math.max(0, r) to guarantee the radius can't be negative.

this.radius = Math.max(0, r);

Add instance methods getRadius(), setRadius(double r), toString(), getCircumference() and getArea().

The class has 1 static(class) variable that keeps the count of the number of objects created (private static int numObjects = 0;). Increment the class variable numObjects in the constructors (numObjects += 1; or numObjects++). Add 1 static method called getNumerOfObjects that gives access to numObjects.

Sample Data in CircleRunner.java

```
// Create a circle with radius 1 using no-arg constructor
Circle circle1 = new Circle();
out.printf("The circumference is: %.2f\n", circle1.getCircumference());
out.printf("The area of the circle is: %.2f\n", circle1.getArea());
out.printf("%s\n\n", circle1);

// Create a circle with radius 25 using initialization constructor
Circle circle2 = new Circle(25);
out.printf("The circumference is: %.2f\n", circle2.getCircumference());
out.printf("The area of the circle is: %.2f\n", circle2.getArea());
out.printf("%s\n\n", circle2);

// Create a circle with radius 125 using initialization constructor
Circle circle3 = new Circle(100);
out.printf("The circumference is: %.2f\n", circle3.getCircumference());
out.printf("The area of the circle is: %.2f\n", circle3.getArea());
out.printf("%s\n\n", circle3);

// Modify circle3 radius by invoking mutator method
circle3.setRadius(225);
out.printf("Circle 3's radius is now: %s\n\n", circle3.getRadius());

// verify negative values aren't allowed
circle3.setRadius(-25);
out.printf("Circle 3's radius is now: %s\n\n", circle3.getRadius());

// Check the number of objects that have been created
out.printf("Created %2d objects.\n\n", Circle.getNumberOfObjects());
```

Sample Execution

```
The circumference is: 6.28
The area of the circle is: 3.14
Circle{radius=1.0}

The circumference is: 157.08
The area of the circle is: 1963.50
Circle{radius=25.0}

The circumference is: 628.32
The area of the circle is: 31415.93
Circle{radius=100.0}

Circle 3's radius is now: 225.0
Circle 3's radius is now: 0.0
Created  3 objects.
```

4.03 Color.java

Write a class called Color.java that has 3 private attributes of type int called red, green, and blue in the range of 0-255 inclusive. The class has 2 constructors, the no-arg (default or non-parameterized) sets the color to black (all zeros) and an initialization constructor. Add mutators and accessors for all 3 fields and a toString(). The toString should return the r, g, b components int the format: "#(125,30,210)". Any RGB values out of range should use 0 for negatives and 255 for values greater than 255 (hint: r = Math.max(0, Math.min(255, red));)

Sample Data

```
Color black = new Color();
System.out.printf("%s\n", black);

black.setRed(500);
black.setGreen(-500);
black.setBlue(75);
System.out.printf("%s\n", black.getRed());
System.out.printf("%s\n", black.getGreen());
System.out.printf("%s\n", black.getBlue());
System.out.printf("%s\n\n", black);

Color hotPink = new Color(255, 105, 180);
System.out.printf("%s\n", hotPink);
```

Sample Execution

```
#(0,0,0)
255
0
75
#(255,0,75)
#(255,105,180)
```

4.04 Fan.java

Write/design a class called Fan.java that has 2 attributes, a boolean 'on' representing the fan being on or off and a speed of type int in the range of 0 to 10.

The Fan class has 3 class constants for LOW(1), MEDIUM(5) AND HIGH(10).

public static final int LOW = 1; // follow the same format for the other 2

The class has 2 constructors, the no-arg constructor creates a fan turned on low, and the initialization constructor accepts an integer representing the speed of the fan and a boolean representing the fan being on or off.

Add a setter for speed, getters for both speed and on, a toString and a power method. The power method turns the fan on/off.

Sample Data

```
Fan fan1 = new Fan();
System.out.println(fan1); // invokes toString

Fan fan2 = new Fan(Fan.HIGH, true);
System.out.println(fan2);

fan2.power();
System.out.println(fan2);

fan2.setSpeed(-5);      // invalid range
System.out.println(fan2);
```

Sample Execution

```
Fan{speed=1, on=true}
Fan{speed=10, on=true}
Fan{speed=10, on=false}
Fan{speed=1, on=false}
```

4.05 TV.java (advanced)

Write/design a class called TV.java that represents the state of a TV that has a channel(1 to 100 inclusive), volume(0 to 9 inclusive), and behaviors changeChannel(int channel), channelUp(), channelDown(), changeVolume(int volume), volumeUp(), volumeDown(), and power().

The volume and channel can only be changed if the TV is on.

Provide a no-arg constructor that turns the tv on with a volume of 3 and channel 5.

Provide an initialization constructor that accepts 2 integers and a boolean representing the channel, volume and the tv being on.

Sample Data

```
TV tv1 = new TV(); // create a TV
out.println(tv1);  // The tv is on and is set to channel 5 with a volume of 3.

// change the volume to 8 and channel 6 then turn off
tv1.setChannel(6);
tv1.setVolume(8);
tv1.power();
out.println(tv1); // The tv is off and is set to channel 6 with a volume of 8.

TV tv2 = new TV(3, 0, true);
tv2.channelUp();
tv2.channelUp();
tv2.volumeUp();
out.println(tv2); // The tv is on and is set to channel 5 with a volume of 1.

// check invalid states and edge cases
TV badTv = new TV(300, -5, true); // invalid data uses default values
out.println(badTv);    // The tv is on, channel set to 5 with a volume of 3.
badTv.setVolume(0);    // Volume is now 0
badTv.volumeDown();    // Volume still 0
badTv.setChannel(120); // Channel doesn't change
out.println(badTv);    // The tv is on, channel set to 5 with a volume of 0.
badTv.setChannel(100); // Channel is now 100
badTv.channelUp();     // Channel wraps to 0
out.println(badTv);    // The tv is on, channel set to 0 with a volume of 0.
badTv.power();         // turn off the power
badTv.setChannel(4);   // can't change the channel to a tv that is off
badTv.setVolume(6);    // can't change the volume to a tv that is off
out.println(badTv);    // The tv is off, channel set to 0 with a volume of 0.
```

Sample Execution

```
The tv is on and is set to channel 5 with a volume of 3.

The tv is off and is set to channel 6 with a volume of 8.

The tv is on and is set to channel 5 with a volume of 1.

The tv is on and is set to channel 5 with a volume of 3.

The tv is on and is set to channel 5 with a volume of 0.

The tv is on and is set to channel 1 with a volume of 0.

The tv is off and is set to channel 1 with a volume of 0.
```

4.06 LinearEquations2by2.java (advanced)

Write a program LinearEquatins2by2.java that represents a 2x2 system of equations. The class has one constructor that takes 6 doubles and a toString() that outputs if the system has a solution and the corresponding x and y solution.

Using Cramer's Rule:

ax + by = e

cx + dy = f

$x = \frac{ed-bf}{ad-b}$ and $y = \frac{af-ec}{ad-bc}$ where $D = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \neq 0$ or ad – bc ≠ 0.

Sample Data

```
LinearEquation2by2 test1 = new LinearEquation2by2(2, -2, 1, 1, 8, 1);
System.out.println(test1);

test1 = new LinearEquation2by2(3, 2, -2, -1, 3, -1);
System.out.println(test1);

test1 = new LinearEquation2by2(3, 2, 4, -5, -1, 14);
System.out.println(test1);
```

Sample Execution

```
2.0x + -2.0y = 8.0
1.0x + 1.0y = 1.0
Solvable.
x = 2.5
y = -1.5

3.0x + 2.0y = 3.0
-2.0x + -1.0y = -1.0
Solvable.
x = -1.0
y = 3.0

3.0x + 2.0y = -1.0
4.0x + -5.0y = 14.0
Solvable.
x = 1.0
y = -2.0
```