Review: Chapter 4 Programming

4.30

```java
    @Override
    public int compareTo(Circle o) {
        // Return the difference between the input and this circle
        return (int)(this.radius - o.radius);
    }
```

4.33

```java
public class Square extends Shape implements Comparable<Square>{

    private double sideLength;

    public Square( double _sideLength ) throws IllegalArgumentException {
        if (_sideLength < 0)
            throw new IllegalArgumentException();

        this.sideLength = _sideLength;
    }

    @Override
    public int compareTo(Square o) {
        return (int)(this.sideLength - o.sideLength);
    }

    @Override
    public double area() {
        return this.sideLength * this.sideLength;
    }

    @Override
    public double perimeter() {
        return this.sideLength * 4;
    }
}
```

4.27

```java
    public Circle( double rad ) throws IllegalArgumentException {
        if (rad < 0)
            throw new IllegalArgumentException();

        radius = rad;
    }
```

```java
    public Rectangle( double len, double wid ) throws IllegalArgumentException {
        if (len < 0 || wid < 0)
            throw new IllegalArgumentException();

        length = len; width = wid;
    }
```

```java
    public Square( double _sideLength ) throws IllegalArgumentException {
        if (_sideLength < 0)
            throw new IllegalArgumentException();

        this.sideLength = _sideLength;
    }
```

Review: Chapter 4 Programming

4.47

```
10      // Store the origin coords for the shape
11      public double originX;
12      public double originY;
13
14      // Define the shape origin
15      public void putShapeHere(double x, double y) {
16          this.originX = x;
17          this.originY = y;
18      }
19
20      // Calculate the distance between two shapes
21      public static double distance(Shape s1, Shape s2) {
22          double x1 = s1.originX;
23          double y1 = s1.originY;
24          double x2 = s2.originX;
25          double y2 = s2.originY;
26
27          double distance = ((x2 - x1) * (x2 - x1));
28          distance += ((y2 - y1) * (y2 - y1));
29
30          return Math.sqrt(distance);
31      }
```

Testing Output