Tutorial of Interface

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech
>
> Modified (only change to markdown file) by ZHU Yueming in 2021. May. 5th
>
> Minor changes by Yida Tao, Dec.5th 2022.

Objective

- Learn how to define and implement an interface.
- Learn how to use the interface `java.long.Comparable<T>`.

Demo

The `Comparable` interface is very useful. This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's *natural ordering*, and the `compareTo` method in it is referred to as its *natural comparison method*.

Lists (and arrays) of objects that implement this interface can be sorted automatically by `Collections.sort` (and `Arrays.sort`).

Let **Circle** implements **Comparable**

```
public class Circle extends Shape implements Comparable<Circle>
```

After implements the interface `Comparable`, we got a compiler error since if a class implements an interface, it must override all abstract methods in it.

Therefore, let's override the method `compareTo` defined in `Comparable`.

```
@Override
public int compareTo(Circle o) {
    if(this.radius < o.radius){
        return -1;
    }else if(this.radius > o.radius){
        return 1;
    }
    return 0;
}
```

The `compareTo` method compares current object with the parameter object to determine a sort order. The return value of the method can be a negative integer, zero, or a positive integer, which means current object would in former place (<), all equal(=), or in latter place (>) than parameter, respectively.

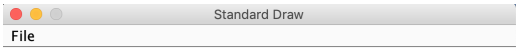
Create a class named **CircleTest**, using the following code. Note that `Collections.sort` sort list element from smallest to largest, according to the logic defined in `compareTo`. However, in this example, we need to draw larger circles first. That's why we used `Collections.sort(circleList, Collections.reverseOrder());` to reverse the order of circles.

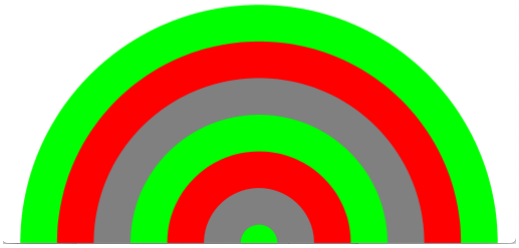
```
public class CircleTest {
  public static void main(String[] args) {
    List<Circle> circleList = new ArrayList<>();
    Circle.setScreenSize(14);
    StdDraw.setScale(-Shape.getScreenSize(), Shape.getScreenSize());
    for (int i = 0; i < Shape.getScreenSize(); i += 2) {
      circleList.add(new Circle(i, 0, -Shape.getScreenSize()));
    }

    Collections.sort(circleList, Collections.reverseOrder());

    for (int i = 0; i < circleList.size(); i++) {
      circleList.get(i).setColor(ShapeColor.values()[i%3]);
      circleList.get(i).draw();
    }
  }
}
```

Execution result:





We may also use the color scheme defined previously to set the color:

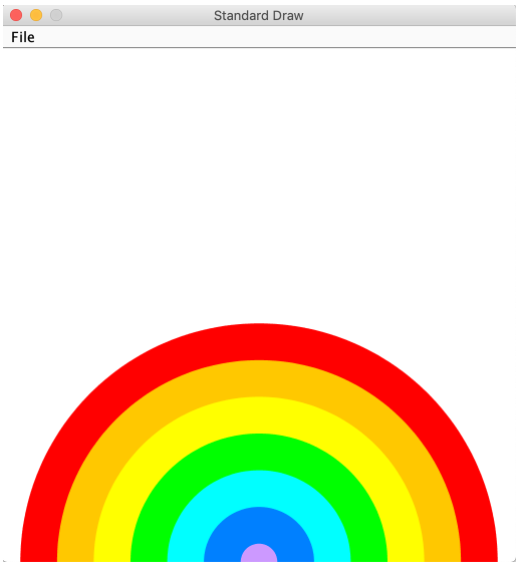
```
public class CircleTest {
  public static void main(String[] args) {
    List<Circle> circleList = new ArrayList<>();
    Circle.setScreenSize(14);
    StdDraw.setScale(-Shape.getScreenSize(), Shape.getScreenSize());
    for (int i = 0; i < Shape.getScreenSize(); i += 2) {
      circleList.add(new Circle(i, 0, -Shape.getScreenSize()));
```

```
      }

      Collections.sort(circleList, Collections.reverseOrder());

      for (int i = 0; i < circleList.size(); i++) {
        circleList.get(i).customizedColor(ColorScheme.RAINBOW, i);
      }
    }
  }
```

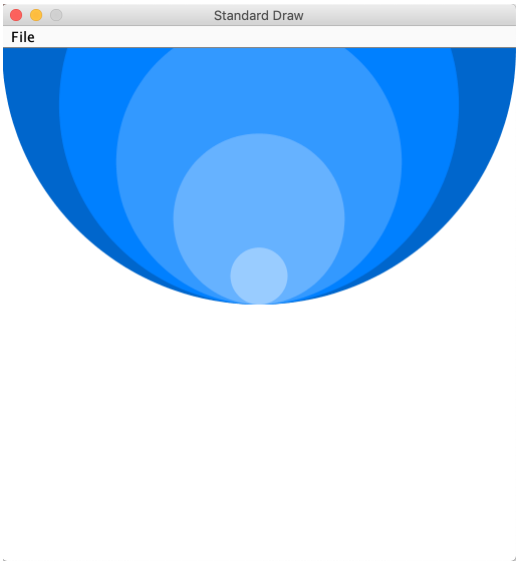
Execution result:



Exercises

Exercise1

Modify `CircleTest` to draw some circles like the following image:



Hint: you should use `ColorScheme.Sky` for colors and the following properties of circles. Also, set `screen size 9`

x	y	radius
0	1	1
0	3	3
0	5	5
0	7	7
0	9	9

Exercise 2

Modify the class `Rectangle` given to you. Make `Rectangle` implements `Comparable`, override the method `compareTo` to order the rectangles from the largest to smallest according their area. If two rectangles have the same area, order the rectangles from smallest to largest according to `x`.

Create a class `RectangleTest` for test.

```
public class RectangleTest {
    public static void main(String[] args) {
        Shape.setScreenSize(9);
        StdDraw.setScale(-Shape.getScreenSize(), Shape.getScreenSize());

        List<Rectangle> rectanglList = new ArrayList<Rectangle>();
        for (int i = -5; i < 5; i ++) {
            rectanglList.add(new Rectangle(i,2*i,Math.abs(i), 2*Math.abs(i)));
        }
        Collections.sort(rectanglList);

        for (int i = 0; i < rectanglList.size(); i++) {
            rectanglList.get(i).customizedColor(ColorScheme.GRAY, i);
            System.out.println(rectanglList.get(i));
        }
    }
}
```

Here is a sample run: