

Assignment4

Design a class named **Rectangle**. The class contains:

- a. Seven private data fields [10 points]
 - **x** (double) : The center of the rectangle in a range (-10, 10)
 - **y** (double) : The center of the rectangle in a range (-10, 10)
 - **width** (double) : The width of the rectangle a range [0, 5]
 - **height**(double) : The height of the rectangle a range [0, 5]
 - **red**(double) : The red color of the rectangle a range [0, 255]
 - **green** (double) : The green color of the rectangle in a range [0, 255]
 - **blue** (double) : The blue color of the rectangle in a range [0, 255]
- b. Design three **constructors**: [10 points]
 1. A constructor with two parameters including x and y
 2. A constructor with four parameters including x, y, width and height.
 3. A constructor with all parameters in private data fields.
 - If a rectangle has no width and height, in constructor method we need to set width=1 and height=1 as default value.
 - If a rectangle has no red, green and blue color, in constructor method we need to set red=100, green=100 and blue=200 as default value
 - For the third constructor, if the red green and blue value is larger than 255 or smaller than 0, we need to set it as its closest boundary value.
- c. Design the **getter** and **setter** method for each private field of Rectangle. For setter method, if the parameter value of red, green and blue is out of its boundary, we need to set it as the closest boundary value of parameter. [10 points]

- d. Design a `toString()` method, which can return a String value as follows output format. [10 points]

The first test:

```
public class Test1 {  
    public static void main(String[] args) {  
        Rectangle r1=new Rectangle(0,1,2,2,100,150,100);  
        Rectangle r2=new Rectangle(3,1,3,1,-1,350,100);  
        Rectangle r3=new Rectangle(2,2);  
        Rectangle r4=new Rectangle(3,-1,2,2);  
        System.out.println(r1);  
        System.out.println(r2);  
        System.out.println(r3);  
        System.out.println(r4);  
    }  
}
```

```
Centre[0.0,1.0] Shape[2.0,2.0]  
Color[100,150,100]  
Centre[3.0,1.0] Shape[3.0,1.0]  
Color[0,255,100]  
Centre[2.0,2.0] Shape[1.0,1.0]
```

- e. Design a public method named `intersect()` with a Rectangle parameter and a return value (boolean). If the current rectangle is intersected with the parameter rectangle, it will return true, otherwise it will return false. [10 points]
- f. Design a public static method named `intersect()` with two Rectangle parameters and a return value (boolean). If two rectangles are intersected, it will return true, otherwise it will return false. [10 points]

The Second Test:

```
public class Test2 {  
    public static void main(String[] args) {  
        Rectangle r1=new Rectangle(0,0,2,2,100,50,100);  
        Rectangle r2=new Rectangle(1.5,2,3,1,-1,350,100);  
        Rectangle r3=new Rectangle(1.6,-1,1,3,50,100,200);  
        Rectangle r4=new Rectangle(1,1,1,2,180,100,40);  
  
        System.out.println(r1.intersect(r2));  
        System.out.println(r1.intersect(r3));  
    }  
}
```

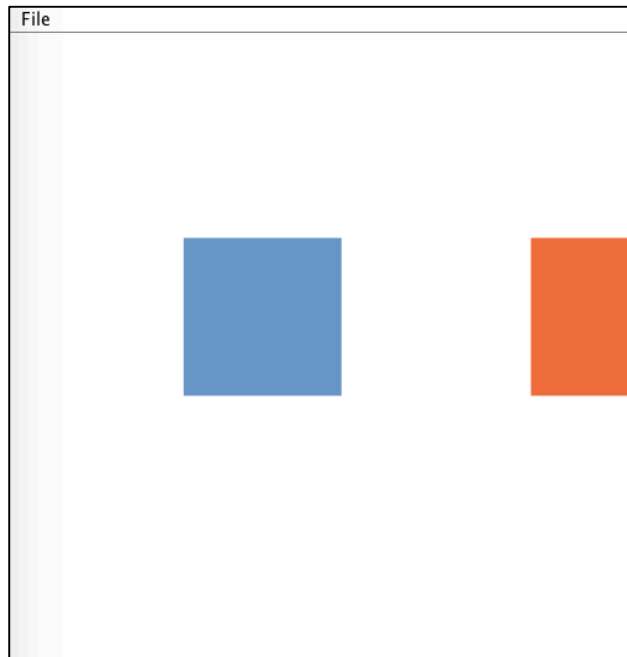
```
        System.out.println(r1.intersect(r4));  
        System.out.println(Rectangle.intersect(r3, r4));  
    }  
}
```

```
false  
false  
true  
true
```

- g. Design a public method named `isValid()` with a return value (boolean). Only when the width and the height of current rectangle are in the range `[0, 5]` (include 0 and 5) and the x and y are in the range `(-10,10)` (`-10` and `10` is not included), it will return true, otherwise it will return false. [10 points]
- h. Design a public method named `isInBoundary()` with a return value (boolean). If the whole rectangle can be drawn in the screen `(-10,10)`, it will return true, otherwise it will return false. [10 points]

The third Test:

```
Rectangle r1=new Rectangle(-2,1,5,5,100,150,200);  
Rectangle r2=new Rectangle(9,1,5,5,250,100,40);  
System.out.println(r1.isInBoundary());  
System.out.println(r2.isInBoundary()); '  
(The whole rectangle of the blue one can be drawn in the screen,  
so it is in boundary, but the yellow one is not.)
```



- i. In same package, we create a class named **RectangleTest**, in which there is a main method. In main method, we will read several lines as following format. For each line, we need create an object of Rectangle according to the data in line. After that, we need to output all **valid rectangles**, which are not **intersect** with the **first valid rectangle** in the line order. [10 points]

The fourth Test:

Sample input:

```
-10,-10,1,2
0,0,3,3,100,100,100
0,1,2,5,130,22,444
5,5,-3,-5
5,5,2,2,30,100,250
-3,5,5,10,40,20,330
2,3
```

Sample output:

```
Centre[0.0,0.0] Shape[3.0,3.0]
Color[100,100,100]
Centre[0.0,1.0] Shape[2.0,5.0]
Color[130,22,255]
```

j. Design a public method named `draw()` , in which we need to draw the current rectangle in screen. We need to make sure that: [10 points]

- Only draw the valid rectangle.
- The color of the rectangle is according to its red, green and blue value.
- We need to fill the rectangle in the screen.

In main method, adding following two statements, just before you using draw method.

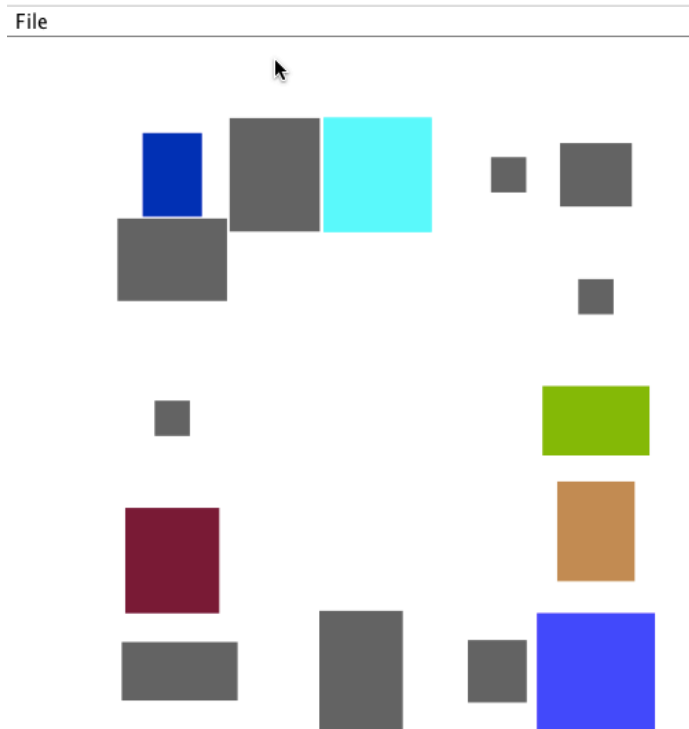
```
StdDraw.setXscale(-10,10);  
StdDraw.setYscale(-10,10);
```

The fifth Test:

Sample input:

```
-4.98,-8.00,3.29,1.66  
0.17,-8.00,2.37,3.43  
4.04,-8.00,1.67,1.77  
6.84,-8.00,3.35,3.30,87,9,260  
6.84,-4.03,2.20,2.83,200,139,78  
6.84,-0.89,3.04,1.97,122,195,14  
6.84,2.63  
6.84,6.09,2.04,1.80  
4.36,6.09  
0.64,6.09,3.07,3.25,-10,299,288  
-2.28,6.09,2.56,3.22  
-5.19,6.09,1.68,2.37,-20,-30,184  
-5.19,3.68,3.11,2.34  
-5.19,-0.82  
-5.19,-4.86,2.67,2.99,129,7,49
```

Sample output:



Sample input:

```
-1.96,-8.00,2.99,-2.39,230,119,148
2.80,-8.00
7.23,-8.00,2.44,3.20
7.23,-4.07,2.63,2.04,241,27,267
7.23,0.57,8.84,1.98
7.23,5.91
7.23,9.18,-2.15,2.26
3.26,9.18
1.15,9.18,1.86,2.16,69,204,101
-2.51,9.18,2.52,3.41
-5.18,9.18,-1.02,3.17
-6.18,6.75,-2.51,2.22,81,26,176
-7.18,2.01,2.43,2.88,116,85,169
-8.18,-1.42
```

Sample output:

