Tutorial of Polymorphism

> Based on the tutorial of "2020S-Java-A" designed by teaching group in SUSTech (Designed by ZHAO Yao)
>
> Modified (only change to markdown file) by ZHU Yueming in 2021. April. 25th
>
> Minor changes by Yida Tao, Nov. 17 2022

Objective

- Learn polymorphism.
- Learn abstract class.
- Learn implementing an interface.

Polymorphism Demo

In this lab, we'll use the same classes used in the previous lab.

First, create `PolymorphismTest.java` as follows:

```
public class PolymorphismTest {
    public static void main(String[] args) {
        ArrayList<Shape> shapeList = new ArrayList<Shape>();

        Shape.setScreenSize(9);
        StdDraw.setXscale(-Shape.getScreenSize(), Shape.getScreenSize());
        StdDraw.setYscale(-Shape.getScreenSize(), Shape.getScreenSize());

        for (int i = 0; i < 3; i++) {
            shapeList.add(new Circle(1, 4 * i + 1, 1));
            shapeList.add(new Rectangle(4 * i + 1, -1, 1,1));
        }

        for (int i = 0; i < shapeList.size(); i++) {
            shapeList.get(i).checkColor();
            System.out.print(shapeList.get(i));
            shapeList.get(i).draw();
        }
    }
}
```

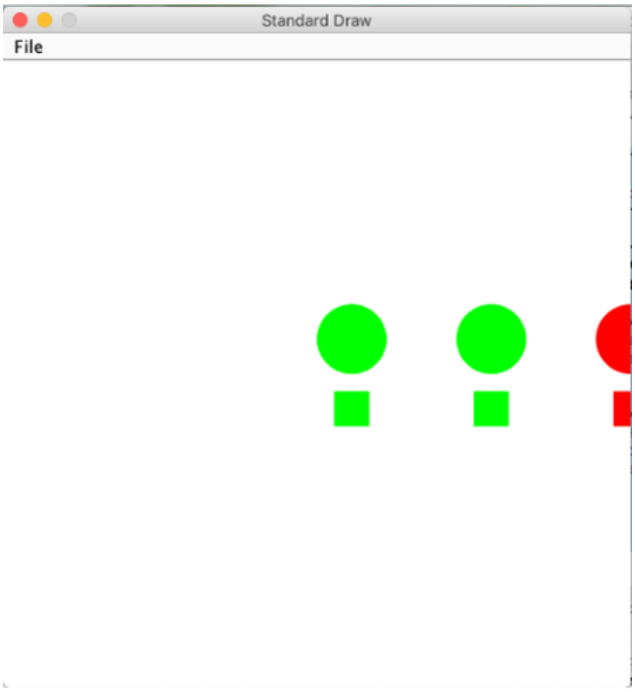
Two compilation errors would arise in `checkColor()` and `draw()`. Although these two methods have been defined in both `Circle` and `Rectangle` classes, we cannot invoke them directly if they haven't been defined in their super class `Shape`, since the compiler only knows that the element type in `shapeList` is `Shape`, therefore it only checks whether `Shape` has `checkColor()` and `draw()`.

To resolve these errors, define these two methods in `Shape`:

```
public void checkColor(){}
public void draw(){}
```

Run above code, observe the result:

```
Circle{radius=1.0 x=1.0, y=1.0, color=GREEN}
Rectangle{width=1.0, height=1.0 x=1.0, y=-1.0, color=GREEN}
Circle{radius=1.0 x=5.0, y=1.0, color=GREEN}
Rectangle{width=1.0, height=1.0 x=5.0, y=-1.0, color=GREEN}
Circle{radius=1.0 x=9.0, y=1.0, color=RED}
Rectangle{width=1.0, height=1.0 x=9.0, y=-1.0, color=RED}
```



Abstract Class Demo

Start from the code you finished in the previous task. We can see that there are two public methods, which have no valid code.

```
public void checkColor(){}
public void draw(){}
```

In fact, we don't need to instantiate `Shape`. In this case, we could change `Shape` to be an abstract class.

- Add `abstract` keyword before `class`: `public abstract class Shape`
- Change `draw()` to an abstract method: `public abstract void draw()`

Run `Polymorphism.java` again and observe the results.

Interface Demo

Create `ColorDraw.java` to define an interface that allows users to choose a color from a color scheme. `ColorScheme` is an enum type (see `ColorScheme.java`).

```
public interface ColorDraw {
    public void customizedColor(ColorScheme colorScheme, int index);
}
```

Let `Circle` implements `ColorDraw`, which means that we need to override `customizedColor` method in `Circle`:

```
    @Override
    public void customizedColor(ColorScheme colorScheme, int index) {
        Color[] colorList = colorScheme.getColorScheme();
        if (index < 0){
            index = 0;
        }
        if (index >= colorList.length){
            index = index % colorList.length;
        }
        StdDraw.setPenColor(colorList[index]);
        StdDraw.filledCircle(x, y, radius);
    }
```

Similarly, let `Rectangle` implements `ColorDraw` and override its `customizedColor` method:

```
    @Override
    public void customizedColor(ColorScheme colorScheme, int index) {
        Color[] colorList = colorScheme.getColorScheme();
        if (index < 0) {
            index = 0;
        }
        if (index >= colorList.length) {
            index = index % colorList.length;
        }
        StdDraw.setPenColor(colorList[index]);
        StdDraw.filledRectangle(x, y, this.width / 2, this.height / 2);
    }
```

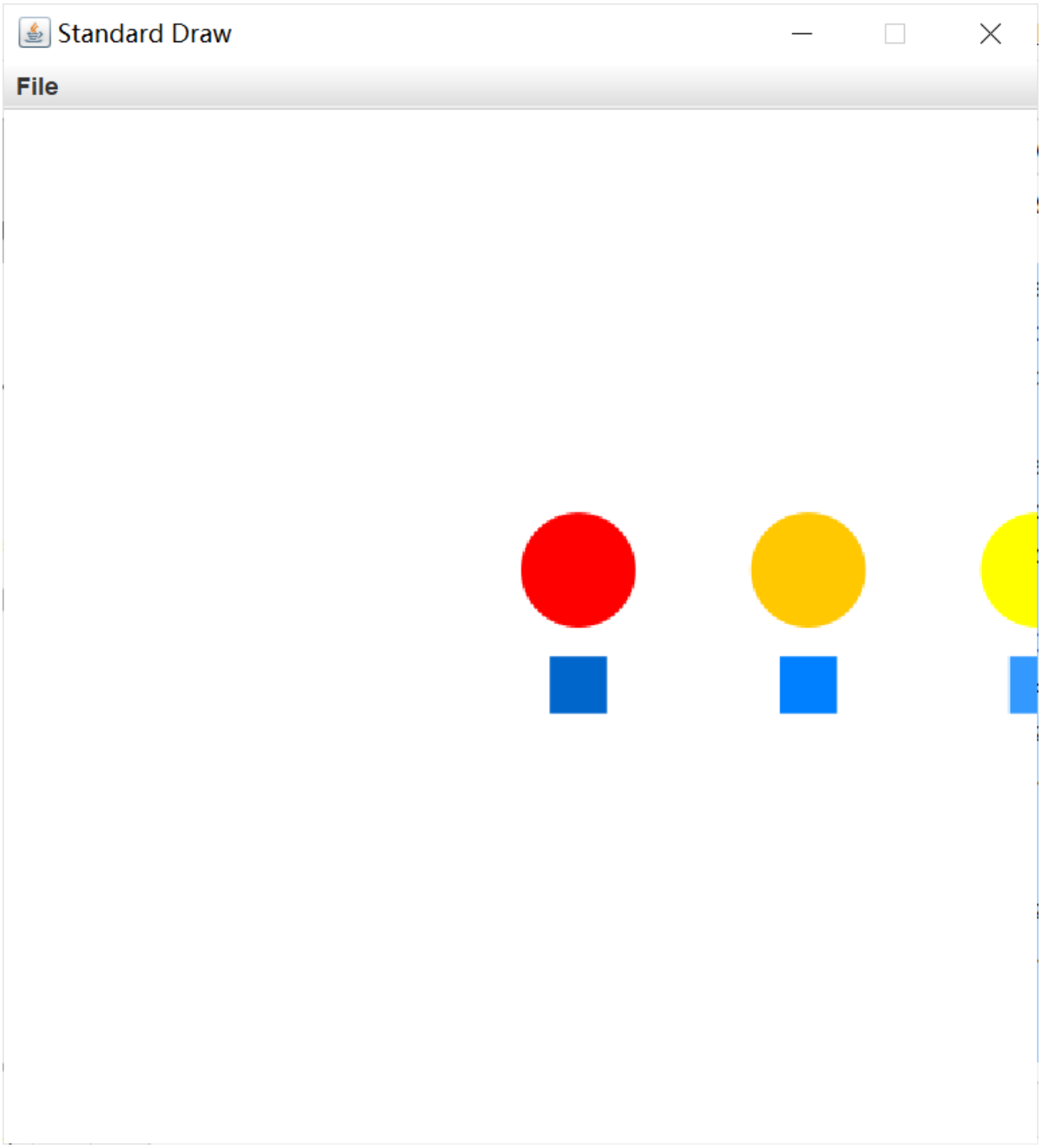
Now, executing the following code to observe the result.

```
public class InterfaceTest {
    public static void main(String[] args) {
        ArrayList<Shape> shapeList = new ArrayList<Shape>();
```

```
        Shape.setScreenSize(9);
        StdDraw.setXscale(-Shape.getScreenSize(), Shape.getScreenSize());
        StdDraw.setYscale(-Shape.getScreenSize(), Shape.getScreenSize());

        for (int i = 0; i < 3; i++) {
            Circle c = new Circle(1, 4 * i + 1, 1);
            c.customizedColor(ColorScheme.RAINBOW, i);
            shapeList.add(c);

            Rectangle r = new Rectangle(4 * i + 1, -1, 1,1);
            r.customizedColor(ColorScheme.SKY, i);
            shapeList.add(r);
        }
    }
}
```



Exercise

Exercise 1

Step 1: Create a class `Monkey`, which contains a public instance method `speak()` that simply print "aaaa" to the console to simulate how monkeys make sound.

Step 2: Human beings evolve from monkeys. So please create a `Human` class that extends the `Monkey` class. Since human beings have languages, please override the `speak()` method in the `Human` class and make it print "Hello World!" to the console.

Step 3: Create a class `Exercise1`, which contains a `main` method doing the following things:

- The `main` method creates a `Monkey` array of size 6, named mArray
- For each array element, if the index is an even number (i.e., 0, 2, 4), make the element point to a new `Monkey` object; otherwise, make the element point to a new `Human` object.
- Iterate through the array using the following for loop: `for (Monkey m : mArray) { m.speak();}`

If your code is correct, the main method should print the following content:

```
aaaa
Hello World!
aaaa
Hello World!
aaaa
Hello World!
```

Exercise 2

Modify the code your write in the above exercise.

Step 1: Create an abstract class `Animal`, which contains a public abstract method `speak()` that has no return values.

Step 2: Create a class `Monkey`, which extends from `Animal`. Please implement the abstract method `speak()`, and make it simply print "aaaa" to the console to simulate how monkeys make sound.

Step 3: Create a class `Human`, which extends from `Animal`. Please implement the abstract method `speak()`, and make it print "Hello World!" to the console.

Step 4: Create a class `Exercise2`, which contains a `main` method doing the following things:

- The main methods creates an `Animal` array of size 6, named `animals`
- For each array element, if the index is an even number (i.e., 0, 2, 4), make the element point to a new `Monkey` object; otherwise, make the element point to a new `Human` object.
- Iterate through the array using the following for loop: `for (Animal a : animals) { a.speak();}`

If your code is correct, the `main` method should print the same content as in the above exercise.