Introduction to Computer Programming (Java A)
Lab 11

[Objective]

- Learn polymorphism.
- Learn abstract class.

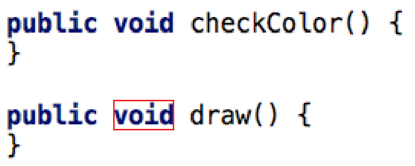
Demo 1: Polymorphism

Create a class **PolymorphismTest:**

```
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();
        }
    }
}
```

Obviously, two errors would arise in checkColor() and draw(). Although we understand that those two methods have been defined in both Circle and Rectangle class, we cannot invoke them directly if they haven't been defined in their super class Shape. It is because we are using subclass to instantiate their super class.

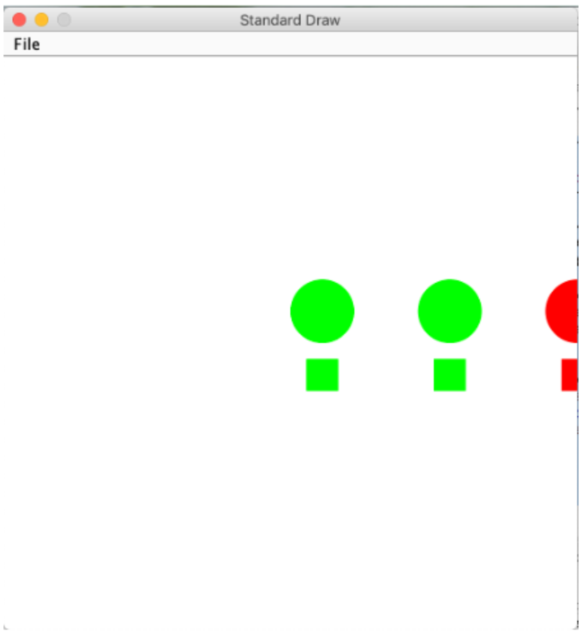
Define these two methods in Shape:



```
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}
```



Demo 2: Abstract class

Step1: Start from the codes 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() {
}
```

The most important thing is that we have no need to instantiate **Shape**. In this case, we should change the **Shape** class to an abstract class.

(1) Add “abstract” before “class”:

```
  public abstract class Shape
```

(2) Change draw() to abstract method:

```
     abstract public void draw();
```

Step2: In **ShapeTest**, we try to write the following code in main():

```
Shape shape=new Shape();
```

There will be an error: Cannot instantiate the type Shape

However, the following code can work:

```
public class ShapeTest {
   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();
        }
    }
}
```

[Exercises]

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 methods 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 shoud print the following content:

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

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.