Assignment 9 – Creating classes

Problem 1

In the file `Circle.java`, fill in the code to define a class `Circle` which represents a circle of a given radius and has methods which provide information about its current radius, area, and circumference. In particular, it has the following constructor and methods:

- Its constructor takes a single double, representing the radius.
- A method `getRadius()`, which takes no arguments and returns the radius as a double.
- A method `getCircumference()`, which takes no arguments and returns the circumference as a double (the circumference of a circle is $2\pi r$, where r is the radius).
- A method `getArea()`, which takes no arguments and returns the area as a double (the area of a circle is πr^2, where r is the radius).
- A method `equals()`, which takes another `Circle` as its argument and returns true if the other circle has the same radius.

The class should have no public instance variables. In particular, there is no way to change the radius of the circle after the circle has been created.[1]

The file `TestCircle.java` contains a ConsoleProgram which can be used to test your circle implementation to see if it is working correctly. You can feel free to examine or modify this code.

Problem 2

In lecture, we saw how to create a `MovingBall` class. Along similar lines, you should now fill in code in `BouncingBall.java` to create a new class called `BouncingBall` which represents a GOval bouncing around the screen. You must implement the following constructor and method:

- A constructor which takes three arguments: a GOval which is to bounce around the screen, and two doubles representing the initial x- and y-velocity.
- A method `move()`, which takes no arguments. When this method is called, the GOval should be moved on the screen according to the current velocity. If this would cause the GOval to contact any side of the window, then the GOval should "bounce" off that side, reversing the x-velocity if on the left- or right-edge, and reversing the y-velocity if on the top- or bottom-edge.

That is, when the `move()` method is called repeatedly, the given GOval should appear to travel around the window, bouncing off sides when it makes contact.

Once you have defined your class, you should write a GraphicsProgram in `BouncingBallsProgram.java` which displays several balls of different colors bouncing around the screen. You may want to randomize their start locations and/or their initial velocities.

[1] Because of this, we say that `Circle` is *immutable*.

Challenge problem

Extend the bouncing balls program from the previous part in the following ways:

- Add an additional method to `BouncingBall` class called `isContacting()`, which takes in another `BouncingBall` as an argument and returns a boolean which is true if the two circles are overlapping, and false if not.
- Using this new method, modify your `BouncingBallsProgram` so that something happens if two balls run into each other. This could be as simple as changing their colors, or you could have the balls bounce off of each other in opposite directions.

A hint: two circles are overlapping if the distance between their centers is less than the sums of their radii. The distance between two points (x_1, y_1) and (x_2, y_2), according to the Pythagorean theorem, is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.