1 Objectives

1. Practice using subroutines
2. Practice using graphics
3. Practice animations

2 Tasks

1. Create a new Java project called “Lab12”.
2. Download the content of the SampleAnimationStarter.java file from the course webpage and copy/paste it into a new file in the project.
3. Rename the SampleAnimationStarter class to Lab12 in the class definition and inside the main function (there are three total places it needs to be replaced).
4. (Optional, as a warm-up) Try to modify the drawFrame function to do the following:
   (a) Draw a blue rectangle somewhere on the screen, with side lengths of 100 and 200 pixels.
   (b) Draw a red circle somewhere on the screen with radius 50 pixels
   (c) Print the text “Hello World” on the display in black letters.
   (d) Make sure none of the above elements overlap (for now).
   (e) Make your “Hello World” text scroll to the right until it is completely off the screen, then start back on the left edge, scrolling to the right again, and repeating.
5. For the main tasks of the lab, do the following (which can also get you started on some of the basic mechanics needed for Project 3). The main objective of these is to generate two randomly parameterized circles (size, shape, color, location, velocity) that bounce around the window and report a message when they overlap with one another.
   (a) Create a class called Circle that fits the following class structure:
      ```java
      public class Circle {
          // Holds the location of the center of the circle
          private int center_x;
          private int center_y;
      }
      ```
// the radius of the circle
private int radius;

// the velocity/direction of the circle's movement
private int velocity_x;
private int velocity_y;

// the color of the circle (using java.awt.Color class)
private Color color;

// The default constructor
public Circle(int maxX int maxY){
    // randomize the location of the circle so that the center has x coordinate
    // within [0, maxX] and y coordinate within [0, maxY].

    // also set the radius to be a random amount, but not too big or small

    // Then set the color randomly (your choice how to implement this)

    // Set each velocity to something reasonable, the range [-3, 3] is good
}

/**
 * Moves the circle by adding its velocity to each respective coordinate
 * of the center. If the edge of the circle would fall outside the x range
 * [0, maxX] or the y coordinate outside [0, maxy], it should reverse
 * its velocity along that coordinate.
 * For debugging, you may output messages to System.out to help detect
 * what your logic is doing.
 */
public void move(int maxX, int maxY){
    // your code here
}

/**
 * Render the circle to the canvas. Note that the java.awt.Graphics.fillCircle
 * uses the top-left coordinates along with width and height. You'll
 * have to translate your center_x, center_y and radius to values that will
 * work for this.
 */
public void draw(Graphics canvas){
    // draw the circle with the appropriate color, in the appropriate place
    // For debug, print out (to System.out) the calculated top-left coordinates
    // along with the circle attributes to see that it's done correctly.
}

/**
* Returns true if this circle is logically overlapping with the one
* pass as other. Keep in mind you can directly access the circle
* attributes of other because this function is in the scope of the
* Circle class.
*/

public boolean overlapsWith(Circle other){
    // your code here
}

(b) In the Lab12 class, you should create two circles and allow them to bounce around the canvas.

(c) To construct the circles, you can add a Lab12 constructor, initialize them when they are declared, or initialize them in the drawFrame function (but make sure this only happens once per circle, not every frame).

(d) On each frame:
   i. Move each of the two circles, allowing the Circle.move function to handle collision with the walls.
   ii. Check to see if the two circles overlap and, if they do, print a message to System.out to indicate this.

3 Turn In

Upload your source files to the MyClasses assignment submission page.

4 Bonus

You may try adding some of the following extra features for 5 points each:

1. Instead of two circles, do an array of 10 circles and use loops to do the construction, movements, and overlap checking.

2. Add some visual indication when circles collide, such as changing the colors of the circles.

3. Use the Color constructor when creating a circle (instead of the pre-loaded Color.RED, etc.) and randomize the red, blue, and green values of the color used. See the java.awt.Color documentation for how to use this constructor.

4. When two circles collide, try allowing them to bounce off each other. This doesn’t have to be perfect, and to get right can involve a lot of extra calculation. But if you at least change the directions so that they go away from each other, that is sufficient. What goes wrong in this process? Be sure to document possible bugs that you notice.

5. Instead of drawing each circle with the drawCircle function, you could make the circle look like it’s “wobbling” by instead, drawing an oval each time, but slowly changing the lengths and widths. For instance, instead of length 5 and width 5, you could draw the circle with length 20 and width 20, you could repeat the pattern of, length 25 width 15, length 20 width 20, length 15 width 25, length 20 width 20, and so on. Try adjusting the increment size to make the animation look smoother by increasing the number of transitional states.