Study the Bounce Program

Obtain a copy of the program Bounce.java. Try it out. Figure out how it works. Experiment with it (for instance, try changing the values of the sleep method, the values of the x and y velocities, etc. to make the ball go faster or slower, or to change its direction from a 45 degree diagonal line to something else). By the time you are done experimenting with it you should completely understand how the ball moves, how collisions with the frame boundary are detected, what the velocity variables mean, etc. If you don’t, ask!

Implement Two Bouncing Balls

You are going to write a program that does almost the same as Bounce, except this program will have two balls, of different sizes, moving at different speeds. The balls will not only bounce off the walls of the frame, they will also bounce off each other. Furthermore, the user can drag either of the balls to a different location on the screen at any time. Details about “bouncing” are given below.

Here is how you are expected to set this up. If you deviate from this, you should get permission from me.

- Instead of using two “Ellipse” objects in the main application class, we will instead create a new class that is a subclass of Ellipse. I modified my Circle class from lab 4 to solve this; I encourage you to do the same. We need to do this if we want to have “draggable” circles.

- The Circle class should contain all information about a circle. In other words, its list of properties should consist of its center coordinates, its radius, its color, its x-velocity, and its y-velocity. Note that Color is an inherited property, so you don’t need to declare that explicitly. You may need other properties as well (for instance, variables to remember the previous location where the mouse was during a drag operation).

  You do not have to use “center” and “radius”. Instead you can use the usual “upper left corner of bounding rectangle” and the width and height. However, using center and radius is more symmetrical and, in my opinion, makes the code easier to understand. Besides, you already have some of the code for Circle, and it uses center and radius!

- The Circle class should also have accessor and mutator methods for all the properties except the ones that are inherited. (So, for instance, you will need to write getRadius() and setRadius(int r), but you don’t need setColor(Color c) or getColor().)
• The Circle class should also have a move method that has no parameters. This method should move the location of the circle by a distance of \(_xVelocity\) in the \(x\)-direction and by a distance of \(_yVelocity\) in the \(y\)-direction.

• Finally, the Circle class should have a mouseDragged method that lets the user change the location of a circle. This does not affect the velocity values for the circle.

• The main class (I named mine “Lab5.java”) should extend Frame and should have just two instance variables, both of class Circle. The constructor, which should have no parameters, should initialize these two circles to have different sizes, colors, starting locations, and velocities. Then it should enter a while loop that does the following forever:

  sleep for some fixed number of milliseconds
  See if the two circles are touching or overlapping. If they are, reverse the \(x\) and \(y\) velocities of both circles. (You will need to use methods like “\(getXVelocity()\)” and “\(setVelocity(...)\)” for each circle.)
  See if each circle needs to bounce off the boundary of the frame; update the appropriate velocity variable(s) for the circle(s).
  Call each circle’s move() method

You will probably need the distance method for doing your test for overlapping circles. You should also consider writing other methods. For instance, the code for testing whether a circle hits a wall is the same for both circles — only the name of the circle variable is different. You could write a method named “wallBounce” that has one parameter, a Circle, and use it like this inside your main application constructor:

```java
... 
wallBounce(_circle1);
wallBounce(_circle2);
... 
```

Your program should behave like the sample code posted on the web — download the files Bounce2.class and Circle.class and run Bounce2 to see how it works.

Optional

The way circles bounce off each other is not very realistic. A more realistic approach would consider angles at which they strike each other. Do the math; then implement a more realistic circle-bounce.

Re-do this assignment without using the wheels package (see separate handout).

Checklist:

- [ ] Header comments in each file with your name, the lab number, the date, and a brief description of the file’s purpose.
Descriptive comments about the instance variables, each method (including the constructor), and blocks of code that do a well-defined task. You don’t need to comment every line. Don’t repeat what the code says; instead, summarize and/or explain.

Circle class has all information about a circle — radius, center, x velocity, y velocity, color — as instance variables and has get and set methods for each one.

Circle class has a move method that moves circle’s center by amounts equal to x velocity and y velocity.

Circle has a mouseDragged method that allows repositioning of a circle.

Lab5 (or whatever you called you main application) has two Circle properties. It creates two circles of different sizes, colors, and velocities and then moves them around as demonstrated in sample file Bounce2.

Demonstrate your program to me.

Sign your name on every program; hand in program listings.