LAB S05 – CLICK, CLICK, CLICK…GOTTA BOUNCE
PART 1

With this lab we make a return to the land of graphics and incorporate ArrayLists.

Using the starter code on Canvas, create a new project that has a ClickDriver and a ClickGame class. This code should run immediately, showing a blue background with a timer that counts up.

Enhance your code based on the following instructions.

ClickDriver: Do not make any changes to the driver class!

Sphere: Create a Sphere class. The Sphere class should have private instance variables for the position of the sphere (x and y), the incremental movement of the sphere (xShift and yShift), and the size of the sphere. The Sphere class should have a default constructor (i.e., no parameters) that randomly sets the position of a sphere (make sure it is on the screen) and the speed of the sphere. For Part 1, the size of each sphere should be 100 pixels and the color should be red.

The Sphere class should also have a move method and a draw method. A Sphere moves based on its speed (incremental movement variable) in each direction. If a Sphere encounters a wall it should bounce off of the wall and continue moving on the screen.

The Sphere class will also need getter methods.

ClickGame: In the ClickGame class, make the following additions.

- Create and initialize an ArrayList that will hold Sphere objects.
- Modify the paintComponent method so that each Sphere object in the ArrayList will be drawn on the screen (Hint: use a for loop).
- Every two seconds, create a new Sphere object and add it to the ArrayList.
- Whenever the mouse button is clicked, compare the location of the cursor to every sphere on the screen. If the cursor is inside of any sphere, that sphere should be removed from the ArrayList of spheres and disappear from the screen.

Enhanced for loops must be used where appropriate.

PART 2

Use the code you created from Part 1 of this lab. *Hint: You will be modifying the code substantially. Consider creating a new project for this part so that you do not lose your functioning code from Part 1*.

Make the following upgrades to your functioning program. Make these changes in this order, or you will encounter difficulties.

1) Modify the Sphere class so that each sphere has a random size between 50 and 150, inclusive.
   Items 2 – 4 should be done together
2) In the Sphere class, make an instance variable for color, so that each sphere will be one of six random colors.
3) In the Game class, create a static instance variable for an array of Color objects. Fill the array with six different colors.
4) Create a static method in the Game class that will return a random color from the array; this method should be called from the constructor of the Sphere class.

5) Currently, when a Sphere is clicked, it is removed from the ArrayList and the screen. Modify the Game class so that whenever a Sphere object is clicked, all of the other spheres of the same color are also removed. After the mouse is pressed, whether or not a sphere was removed, all remaining spheres on the screen should be changed to a new random color from the Color array.
6) Create a Cube* class that will create moving cubes on the screen; cubes should behave in the same way as spheres. However, **Cubes should not ever be removed**. And clicking on a Cube will not make anything disappear. Whenever a new shape is created, your program should randomly determine if it adds a Sphere or a Cube.

7) Create a Shape class to be used as a superclass for both Sphere and Cube objects. Put as much common code as possible into the Shape class.
8) Modify the ArrayList in your Game class to hold Shape objects (so that it can hold both Sphere and Cube objects). You should only have one ArrayList in
9) Modify the Game class so that the program ends when there are fifty spheres on the screen (*Hint: use a lower number for testing purposes*).
10) After the program ends, display on the screen the total number of spheres that the player removed during the game.

*Since I started talking in three dimensions already; it is really just a square.

PART 3

Personalize your program by making at least five modifications to individuate your version of this game from everyone else's version. Part 3 should be shown to me in class.

PRE and POST conditions are not required for this lab exercise. This lab may be submitted as a pair rather than individually.

Lab S05 should be shown to Holm in person before 8:00 AM on Tuesday, February 11th.