C212/A592 Spring 17 Lab 12

Intro to Software Systems

Instructions:

- Review the requirements given below and Complete your work. Please submit all files through Canvas (including all input and output files).
- The grading scheme is provided on Canvas

Lab12: Code Refactoring, Junit, Java.awt.Rectangle Class

- There is a .jar file attached with the lab, run the file to see the functionality of the lab
    - Pressing the *c* key draws a circle, *r* draws a rectangle, and *s* draws a square
- One goal of this lab is to refactor existing code
- Here is the wiki description of code refactoring

  https://en.wikipedia.org/wiki/Code_refactoring
    - A lot of time spent being a Software engineer/developer is refactoring existing programs or programs you have previously written
- The other goal is to introduce the *Rectangle* class as well as use Junit to test

Adding Functionality and Refactoring the moving Shapes application from Lab 9

- We will be adding the Rectangle and Square shapes to move and bounce around with the Circles
- You can remove any of the methods that are not being used such as *perimeter and area* from the previous HW assignment that were the building blocks for this lab
- The refactoring part of the assignment is open ended, but the goal is to simplify the code as much as possible while changing the code to use the java awt Rectangle class
    - You can change as much code and logic as you want
    - e.g. I removed the *Point* class and added an *x, y* field directly in the Shape class (did not need a Point class for the *distanceTo()* method for collision detection)
- To do this we will add a *Java.awt.Rectangle* field to the *abstract Shape* class, and its subclasses (you will need to rename the Rectangle class you previously wrote to avoid a naming conflict)

- This field does not need to be initialized in the constructor
- Add the following abstract method in the Shape class, and Override it for each shapes

  abstract Rectangle getRect();
  - *getRect()* should return a new rectangle with the, *Rectangle(int x, int y, int width, int height)* constructor
- Now that each Shape has a rectangle field, we can use the *intersects* method and simply check:

  if (shape1.intersects(shape2)) { }
- Now it does not matter what shape shape1 or shape 2 is, but we can test whether a Square just collided with a Rectangle or Circle

- You will then add appropriate Junit tests to your program to test various classes.