Lab #1 – Pong Game

Total Points: 50

**Introduction:**

You have just been hired by the Object-Oriented Language Development Company (OLD Co.) They have begun development of what they believe to be a radically new computer game, with the wholly original name of Pong. Figure 1 shows a picture of the OLD Co. Pong game:

A screenshot of a cell phone

Description automatically generated

**Figure 1:** The OLD Co. Pong game.

Pong is a two-player game in which each player attempts to prevent the pong-ball from reaching the wall behind their pong-paddle. A player deflects the pong-ball before it reaches the wall by placing their pong-paddle into the path of the pong-ball. This causes the ball to bounce off the paddle in the opposite direction. If the ball reaches the wall behind a player's paddle, the opponent scores. The number of points scored depends on the region of the wall that is contacted by the ball. The closer to the center of the wall the greater the number of points scored.

The highly paid software engineers at OLD Co. have spent years designing the Pong program. They have identified the classes that describe the objects the program will use. The specifications for these classes were handed off to the programmers who completed the majority of the implementation. However, just before the project was completed, the programmers were all offered huge salaries by start-up companies and quit on the spot. In exchange for a big check, you have been hired to complete the implementation.

**Getting Started:**

Create your GitHub repository and import the code into Eclipse as described on the "How to…" webpage, available from the course homepage. Expanding the Pong package will reveal the six Java source files you will be working on in this lab.

To run the Pong game, run the main method In the PongGame class. This will bring up a window similar to the one in Figure 1. However, because some important parts of the Pong game have not yet been implemented, the game does not quite look right, nor does it work correctly. Your assignment, as described below, is to complete the implementation of the Pong game, ultimately leading to a fully functional game.

**The Assignment:**

The *Pong* project contains 6 classes. In this assignment you will be concerned with only three classes: PongBall, PongScore, and PongPaddle. You will be writing the source code that defines the fields, constructors, and methods of these classes.

However, before you can write the source code for these classes you will need to understand what they are required to do. You will also need to determine what fields each class will need to contain (i.e. what *state* each object will need).

Begin with the PongBall class. Each method signature is accompanied by a short comment describing what the method must do.

One important hint for this assignment is that the coordinate system (that is, the and values) is not the same as is usually used in mathematics. In math, we normally have the origin positioned at the bottom left. Positive -values are to the right of the origin, and positive -values are above the origin. In computer graphics, the -direction is usually flipped. The origin is positioned at the *top* left. Positive -values are to the right of the origin (just as in math), but positive -values are *below* the origin and as the value of increases, the corresponding point moves *down* the screen.

Study the Javadoc comments in the PongBall class until you understand what each of the methods is supposed to do. You will need to use your understanding of the class to infer the fields that it will need. **Note: do not change any of the method signatures.** You should simply add your field definitions and fill in the body of the constructor and each of the methods so that they perform their required functions. **You must create a new JUnit Test Case for the PongBall class. Tests must be written for the constructor(s) and for each method.** Information on creating a new test case can be found under “Creating a new JUnit Test Case” in the How To… document linked from the course webpage.

If you have correctly implemented the PongBall class, the ball in the Pong game should become operational. To find out, run the PongGame class. Pressing the 'B' key on the keyboard should set the ball in motion. If the ball doesn't move, or doesn't seem to move correctly, you'll have to go back and make sure that your source code correctly implements the behavior required of PongBall.

**Pushing Changes to GitHub**

Periodically while you are working, and definitely at the end of each work session before you log out, you should push your changes to GitHub. This makes a backup copy of your work, will allow you to retrieve (pull) them down to another machine, and makes them accessible to your professor. Ultimately, your last push for a lab will turn it in for grading. To push your changes, follow the directions for "Committing and Pushing your code to GitHub” in the How to… document on the course homepage. After you push your changes, log in to GitHub and confirm that the changes now appear there. Remember, you can—and should—push your changes to GitHub frequently, and definitely at the end of each work session before you log out.

After **implementing and testing** the PongBall class, go on to the PongScore and PongPaddle classes. As you complete the **implementation and testing** for each of these classes, the corresponding part of the Pong game will become operational. When you finish the PongScore class, the score for each player will be kept at the bottom of the window. When you finish coding the PongPaddle class, you will be able to move the paddles. Note that the left paddle is moved using the 'A' (up) and 'Z' (down) keys. The right paddle is moved using the 'K' (up) and 'M' (down) keys.

**Submitting Your Solution:**

Turn in your source code by pushing your modifications to Lab 01 on GitHub as described in the How to… page. Don't forget that for every lab you must also submit to Moodle a lab report, incorporating a self-assessment report. For this lab, the lab report will contain nothing but the self-assessment report, which should take you only 5 to 10 minutes to write.