CS212: Paradigms
Dr. Gross, Spring 2016

Lab 9 - Roshambo Rampage

1. Create a new package called edu.blackburn.cs.cs212sp16.roshambo.lastnamelastname
2. When you start on your homework, copy your files your homework repo into a new package: edu.blackburn.cs.cs212sp16.roshambo.lastname
3. Copy the images directory from Documents\Lab09 to a subpackage **(only in homework!)** edu.blackburn.cs.cs212sp16.roshambo.lastname.images

Due Thursday, April 7 before lab

A. Roshambo Rampage

So there's this website (http://brunching.com/psr.html) where you can start an email game of Roshambo (rock-paper-scissors). It sends a link to whoever you want to play against, and they click the link, pick their weapon, and you both get an email response with who won. But it doesn't work anymore. That's entirely irrelevant to this project, except that's where I got the assignment name. I'm really, really tired today.

You're going to make a Roshambo game that will allow you to play against the computer and keep track of who wins more (and other interesting stats).
You're going to want a couple of links handy:

- https://docs.oracle.com/javafx/2/api/ (API for JavaFX)
- http://docs.oracle.com/javase/8/javase-clienttechnologies.htm (tutorials, SceneBuilder info, etc.)
- http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html (download a copy of the API so you don't have to wait for it, but don't put it in the repo)

As always, you will need to import packages. **Never import packages that begin java.awt or javax.swing!**

B. Start the Game

1. Create a new Empty FXML file in your package called Roshambo, along with a controller (which will automatically be named RoshamboController.java)
2. Double-click Roshambo.fxml to open the
3. Open Controls on the left side and drag a Button onto the screen
4. Drag a TextField next to the Button for output
5. Save
6. Open RoshamboController.java
7. Create a TextField variable (don't forget the @FXML tag) for the output field
8. Create a handler (don't forget the @FXML tag) for the button being pressed
9. Create a variable to hold the counter

10. Connect the code in RoshamboController to the TextField and Button in the Scene Builder
11. Create a JavaFX Main Class
12. Replace the code in the start() method with this code (don't copy/paste, type it or it won't work):

```
Parent root = FXMLLoader.load(getClass().getResource("Roshambo.fxml"));
Scene scene = new Scene(root);
primaryStage.setScene(scene);
primaryStage.show();
```

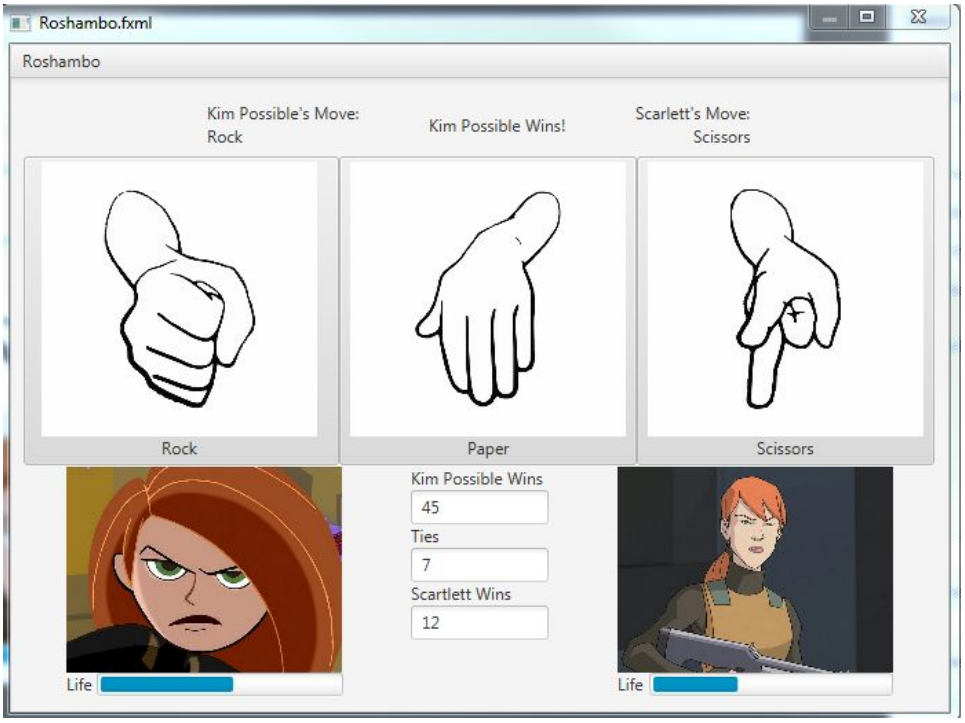
13. You'll need to do some imports, etc.
14. Try it out!
15. Fix things
16. Keep going

C. Finish the Game

This is what the final game should look like. Easy, right? Read all of the hints below before you start.

Some hints:

1. Build one piece at a time
2. You will need to use BorderPane (or GridPane, where elements can span multiple rows/columns)
3. You will need to use VBox and HBox (a ton)
4. If you right-click on an element, you can Wrap In->whatever container you want
5. Build one piece at a time
6. Seriously, put a component on, connect it with code, then add another component
7. You didn't do that, did you? You went ahead and made lots of components and spent lots of time on layout, and now you have no code done. Sigh. What am I going to do with you?
8. Maybe start over.



D. Building the Game Properly

Add elements piece-by-piece. Don't forget to break things into their own methods, and from there into their own classes/objects. If you look at the rubric, you'll see this is required for several elements of the game. Build, test, revise. Build, test, revise.

Probably the first thing you want to do is make a button each for rock, paper, and scissors. Make sure you can handle each button press and detect each button press separately. The easiest way to do this is to make three different button handlers. You don't have to do it that way; there's always a harder way to do things.

The left player is the human player, and the right player is the computer player. In order to get full credit, each of these players should exist in a separate class, but hey, wait a sec, aren't they largely the same class? And if so, don't we have tools to show those similarities?

Once you have a computer player, you can determine the winner and loser of each of the rounds and report that. Once you are reporting the winner/loser of each round, then it would be useful to count how many time the human player won and report that. That's something that could easily be added into its own statistics class. That class could later be extended to track other statistics.

E. Add Stats & Dialog Boxes

When you have the game running and tested, it's time to add some additional features.

Opening Dialog Box

When the player starts the game, let them choose their player and their opponent. Load the correct names and images. All of the images are 320x240 pixels, 72 dpi. You don't have to let them choose the same character for both sides. Also let them select a number of rounds to play. This is worth 5 points.

Additional Statistics

Add some additional statistics that might be interesting. For example, you might keep track of how often each player used each weapon (but don't do that). Each properly implemented stat earns you 2.5 points.

Closing Dialog Box

Don't add all of these four new statistics to the existing game screen (it's already cluttered). When the number of rounds selected by the user at the beginning has elapsed, then print out all of the statistics and give the user the option of restarting (going back to the open dialog) or quitting. Even if you don't have all four additional statistics done, this dialog box is also worth 4 points.

F. Nearly Done

Finally, read through the rubric and make sure you aren't missing anything!

Rubric

Standards/comments	5
Time estimate/accounts	5
Use multiple classes to represent the functions of the game (no JavaFX)	15
Isolate JavaFX controller functions properly	10
Get UI design pretty much correct	5
Implement all mandatory game features	10
Isolating statistics/records into a class	15
Isolating the computer player and human player as classes	15
Having a beginning and/or end dialog box	10
Implementing additional statistics	10
Total	**100**
Having FXML elements not connected to code	-20