IT 2045C Computer Programming II  
Prof. Tom Wulf   
Fall 2024 v1.5

# Lab 05 Rock, Paper, Scissors (20 pts)

For this assignment, you will implement a GUI-based version of the simple Rock Paper Scissors game.

## Review of how the game is played:

This is a two player game. (Here the computer is the second player.)   
Each player simultaneously picks one of the three gestures:

1. Rock ( A closed fist)
2. Paper ( an open hand)
3. Scissors (first two fingers extended like scissors)

Each game is decided by the following rules:

* Rock breaks scissors (Rock wins)
* Paper covers rock (Paper wins)
* Scissors covers paper (Scissors wins)
* If both players choose the same gesture it is a tie

## Procedure:

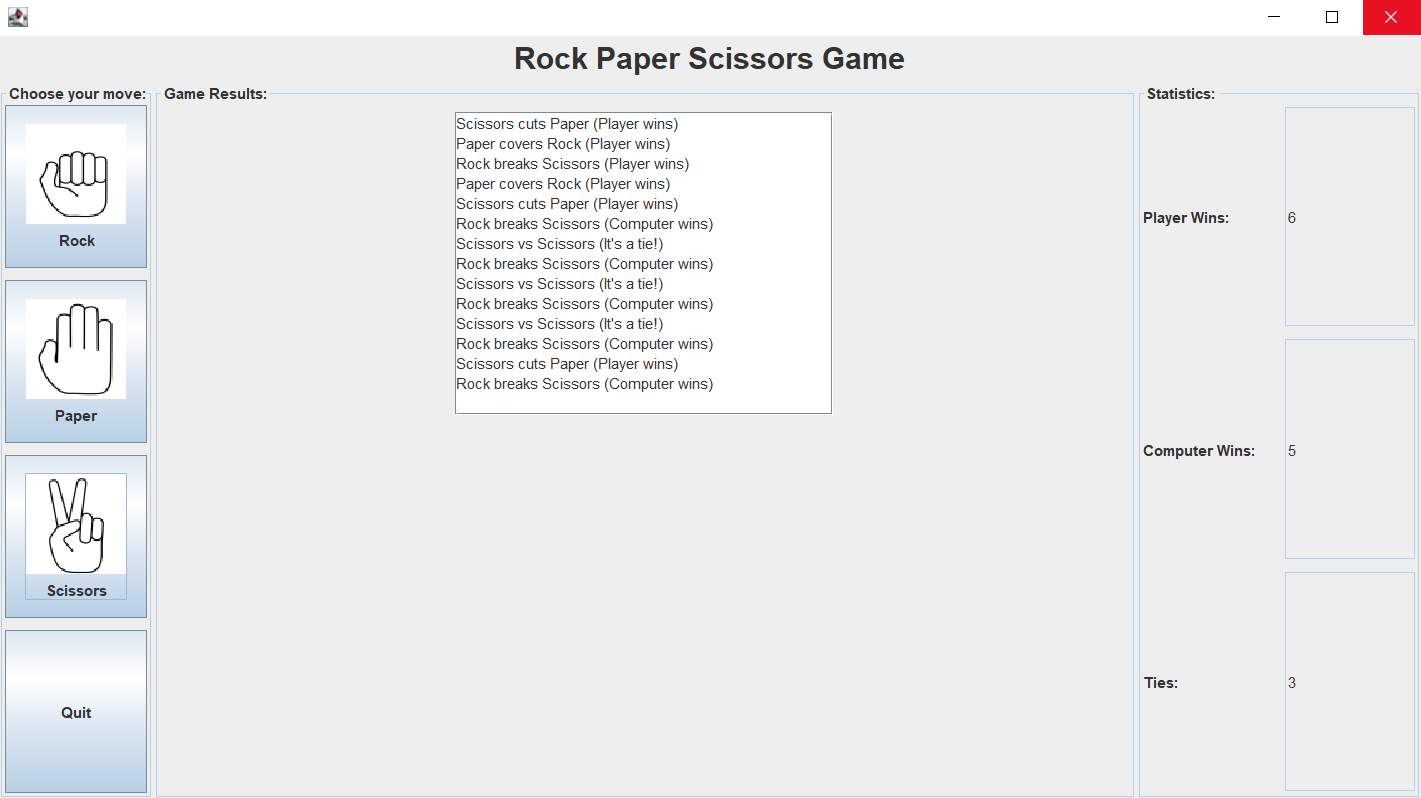
Create a IntelliJ IDEA project called **Rock\_Paper\_Scissors** and put it under GitHub source control**.**

Implement a Java GUI application that extends JFrame (**RockPaperScissorsFrame.java** and **RockPaperScissorsRunner.java**) and has the following GUI components:

* A title for your game’s frame “Rock Paper Scissors Game” or some such.
* A JPanel with three JButton components for Rock, Paper, Scissors, and an additional for Quitting the game. (Each should have an appropriate ImageIcon. Add your images to the IntelliJ project in the src directory.) Put a border around this panel.
* A stats panel with 3 JLabels and JTextFields (Player Wins, Computer Wins, Ties) each should have the count of the wins etc.
* A panel with a JTextArea with JScrollPane that displays the results of each game one per line as a text string:  
  ***Rock breaks scissors (Player wins)***

or  
 ***Paper covers Rock (Computer Wins)***This should accumulate and display the results for each game in the session, one per line, not just show the results for the last game played. (In other words, you can scroll through all of the game results for the session.)

1. Be sure to use the javadoc coding conventions.
2. The player will click on one of the buttons to play the game. The computer will **randomly** determine which symbol it will use and then determine the results. The JTextArea will be updated with the results display string. One line for each game. The stats panel keeping track of the computer and player wins and the ties should also be updated and should show a running total for all the games played during the session.



1. The game continues until the player closes the application window or presses the Quit key.

1. **Extra credit**: You will develop several different algorithmic schemes/strategies for choosing the computers move and then set the computer to randomly use your schemes.   
   1. **Least Used**: Keep track of the number of times that player uses each of the 3 symbols. Develop an algorithm scheme for the computer to pick the symbol that will win against the symbol used the least by the player. (The assumption here is that the player will be most likely to use that symbol that they have not used recently.)
   2. **Most Used**: As in the previous approach keep track of the player choices but this time assume that the player will tend to pick the symbol that they use the most so the computer will pick the symbol that will beat it.
   3. **Last Used:** Use the symbol that the player used on the last round. (Be careful not to call this on the first round of play.) This approach is actually the solution to a famous problem in computer science known as *the prisoner’s dilemma*.
   4. **Random:** Randomize the computers choice as in the normal game described previously.
   5. **Cheat:** no more than 10% of the time, have the computer cheat and pick the winning symbol based on the choice the player already made.

Add the strategy used each time to the display in the JTextArea e.g. :  
**“Rock breaks scissors. (Player wins Least Used)”**

A bit of analysis here shows that this is essentially just a different method to determine the result for each game. One rather clever way to do this would be to use an interface.   
  
**Double Extra credit, if you reasonably do this using an interface! (Consider an Interface called Strategy with an abstract method called determineMove().)**

**This is actually an implementation of the Strategy Design pattern!**

1. Thoroughly test your game making sure that everything works correctly. Try for a nice looking layout. Make sure you completed the javadoc.
2. Make sure you commit and push all your files to GitHub.  
     
   Provide several screen shots that establish that you completed the assigned work.
3. Submit the link to your project GiHb repo and this word doc (renamed **Lastname\_Firstname\_Lab 04.docx using your name)** with the Canvas Assignment mechanism.   
     
   Note because I have trouble keeping track of it. If you did the extra credit, also submit a second time using the extra credit link I have provided.