CS 342 **Project#3** **Spring 2019**

RPSLS JavaFX GUI

Due Date: Sunday March 31 2019, @ 11:59pm

Description:

You will implement the game Rock, Paper, Scissors, Lizard, Spock. This is just an augmented version of the traditional Rock, Paper, Scissors game. Your implementation will be a two player game where each player is a separate client and the game is run by a server. Your server and clients will use the same machine; with the server choosing a port on the local host and clients knowing the local host and port number (just as was demonstrated in class). Each round of the game will be worth one point. Games will be played until one of the players has three points. At the end of each game, each user will be able to play again or quit.

Implementation Details:

You will create separate programs, each with a GUI created in JavaFX, for the server and the client.

For the server GUI:

- A way to chose the port number to listen to
- Have buttons to turn on the server and turn it off.
- Display the state of the game:
    - how many clients are connected to the server.
    - what each player played.
    - how many points each player has.
    - if someone won the game.
    - are they playing again.
- Any other GUI elements you feel are necessary for your implementation.

For the server logic:

- It will only allow a game to start if there are two clients connected.
- It will notify a client if they are the only one connected.
- It will keep track of what each player played.
- It will evaluate who won each hand.
- It will evaluate if a client has won the game.
- It will update each client with the above items in time.
- It will do all things necessary to run the game.

CS 342 Project#3 Spring 2019

It is expected that your server code will open, manage and close all resources needed and handle all exceptions in a graceful way. For game play, each client will chose to play either Rock, Paper, Scissors, Lizard or Spock and send that choice to the server. The server will determine who won and then update each client with what the other played and the resulting state of the game. If a client has won the hand and has reached three points, the server will send what the other player played, the resulting state of the game and require each client to make a choice as to play again or quit. If a player quits, the server will end that connection. If one player quits and the other wants to play again, the server will notify the client that they must wait for another person to connect. If both want to play again, the server will start a new game.

For the client GUI:

- A way for the user to enter the port number and ip address to connect to
- A button to connect to the server.
- A way to display the points each player has.
- A way to display what the opponent played each round.
- Clickable images to choose what to play.
- A way to display messages from the server.
- Buttons to chose to play again or quit.
- Any other GUI elements you feel are necessary for your implementation.

For the client logic:

After entering the port number and ip address, the user will click to connect to the server. When there is another client to play, the game will start. The user will select which item to play and send to the server. The server will respond with what the opponent played, who won and what points were distributed. The client GUI will update with this information and allow the user to either keep playing or, if someone has won, chose to either play again or quit. Quit will end the client program. It is expected that your client code will open, manage and close all resources needed and handle all exceptions in a graceful way.

Testing Code:

You are required to include a folder containing the JUnit 5 test suite you created to test the classes and methods in your program. Minimum of 10 total cases.

UML Diagram:

You are required to include a UML Diagram of your project; including all classes and interactions between them. Format this as a PDF.

Electronic Submission:

Include your client and server programs in different folders plus the testing code in a separate folder and the UML pdf in a .zip file. Submit to the link on Blackboard.

CS 342 Project#3 Spring 2019

Assignment Details:

Late work **is accepted**. You may submit your code up to 24 hours late for a 10% penalty and 24 to 48 hours late for a 20% penalty. Anything later than 48 hours will not be graded and result in a zero.
Unless stated otherwise, all work submitted for grading *must* be done individually. While we encourage you to talk to your peers and learn from them, this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is available here:

https://dos.uic.edu/conductforstudents.shtml.

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing

your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. It is also considered academic dishonesty if you click someone else's iClicker with the intent of answering for that student, whether for a quiz, exam, or class participation. Academic dishonesty is unacceptable, and penalties range from a letter grade drop to expulsion from the university; cases are handled via the official student conduct process described at https://dos.uic.edu/conductforstudents.shtml.