CS 342 **Project#3** **Fall 2024**

Networked 3 Card Poker

Due Dates:

Part one: UML class diagrams for server and client programs and wireframe for the client program:
due: Tuesday November 26th @11:59pm

Part two: code for server and client programs:
due: Friday December 6th @11:59pm

Description:

In this project you will implement a networked version of the popular casino game 3 Card Poker; the same game you implemented in project #2. This is a somewhat simple game to understand and play which should allow you to focus on event driven programing and networking with Java Sockets.

Your implementation will allow up to multiple players to log on to the server and play at once; 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 client is playing a separate game with the server. So, each game has its own deck and own history.

All networking must be done utilizing Java Sockets (as shown in class). **The server must run on its own thread (not the JavaFX application thread) and handle each client on a separate thread. Each client should have their connection to the server on a separate thread other than the JavaFX application thread.**

This project will be developed as two Maven projects. You should use the same Maven JavaFX found in the Project #3 folder on our BB course page. So, download this twice and in one of them, the artifact ID in the POM file for the server Maven project should be "projectThreeServer" and in the other download, the artifact ID in the POM file for the client Maven project should be "projectThreeClient"

*****You may work in teams of two but do not have to***.**

CS 342 Project#3 Fall 2024

How the game is played:
Keep in mind: there are different variations of this game you will find on the web; the following is how your version will play

In three card poker, each player only plays against the dealers hand, not each other:

- Both players will start by placing an ante wager. We will limit the ante bet to $5 or greater, up to $25.
- There is one optional bet the players can make called the Pair Plus wager. We will also limit this bet to $5 or greater, up to $25. This is a separate bet that will win if a players hand is at least a pair of 2's. The payoff for this bet applies regardless of the dealers hand and what happens in the rest of the game. (See below for payouts).
- After all bets are made(ante and/or pair plus), the cards are dealt out. Each player and the dealer receive three cards each. The players cards are face up and the dealers hand is face down.
- Each player must decide if they will play or fold. If they fold, they lose their ante wager and pair plus wager(if they made one).
- If the player wants to continue, they will make a play wager (this must be equal to the amount of the ante wager).
- At this point, the dealer will show their cards. If the dealer does not have at least a Queen high or better, the play wager is returned to the players who did not fold and the ante bet is pushed to the next hand.
- If the dealer does have at least a Queen high or better, then each players hand, that did not fold, is evaluated against the dealers hand (see below for order of winning hands). If the dealer wins, the player loses both the ante and play wager. If the player wins, they get paid out 1 to 1 (they get back double what they wagered). Say the player bet $5 each for the ante and play wager and won, they would get back $20.

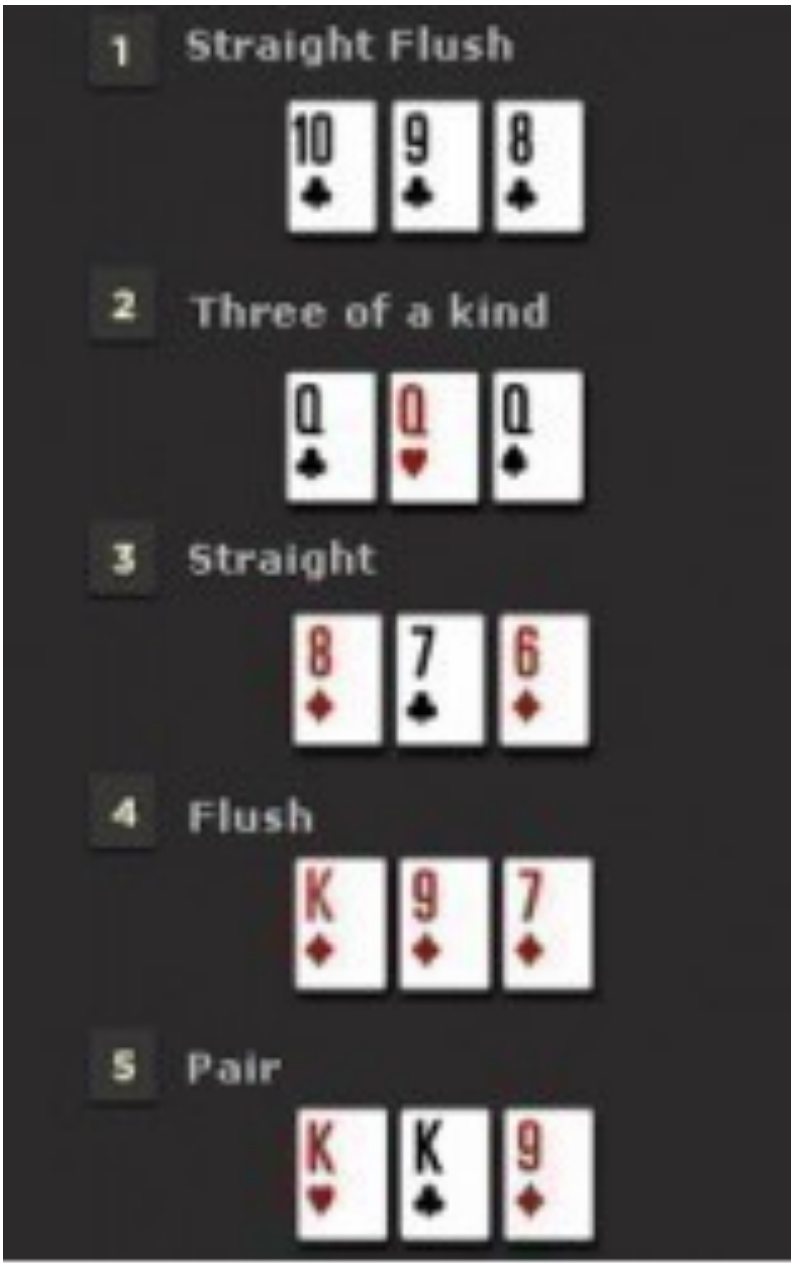
For the Pair Plus wager:

As long as the player does not fold, the Pair Plus wager gets evaluated regardless of if their hand beat the dealers hand; it is a separate bet based solely on the players hand. If the player does not have at least a pair of 2's, they lose this bet. Otherwise, the payouts are as follows:

- **Straight Flush** **40 to 1**
- **Three of a Kind** **30 to 1**
- **Straight** **6 to 1**
- **Flush** **3 to 1**
- **Pair** **1 to 1**

CS 342 **Project#3** **Fall 2024**

Order of winning three card poker hands:



CS 342 Project#3 Fall 2024

Implementation Details:

You will create two separate programs, each with a GUI created in JavaFX, one for the server and one for the client. In this project, you are free to use scene builder, FXML and .CSS files but you can also do so programmatically.

For the server program GUI:

- A way to chose the port number to listen to
- Have a button to turn on the server.
- Have a button to turn off the server.
- Display the state of the game(you can display more info, this is the minimum):
    - how many clients are connected to the server.
    - The results of each game played by any client.
    - how much the a client bet on each game
    - how much a client won or lost on each game
    - if a client drops off the server.
    - if a new client joins the server.
    - is the client playing another hand.
- Any other GUI elements you feel are necessary for your implementation.

Notes: Your server GUI must have a minimum of two scenes: an intro screen that allows the user to input the port number and start the server and another that will display the state of the game information. To display the game information, you must incorporate a ListView (as seen in class) with any other widgets used. Keep in mind, you can dynamically add items to the listView without using an ArrayList.

For the server program logic:

It is expected that your server code will open, manage and close all resources needed and handle all exceptions in a graceful way.
The server will manage each clients game on a separate thread.
For each game the server will maintain a standard deck of 52 playing cards. Before each game, the server will shuffle those cards into a random order before dealing them to the client.
All computations for the game including winning, losing, bets, and calculating winnings will be done on the server. The server should have a separate class that performs these calculations.

CS 342 Project#3 Fall 2024

For the client program GUI:

You are welcome to use/discover any widget, pane, node, layout or other in JavaFX to implement your GUI. **For this project, you are free to use Scene Builder or FXML layout files**. The following three scenes are required:

1) **Your program must start with a welcome screen that is it's own JavaFX scene. It will consist of:**
   - Your welcome screen should welcome players to the game and have some sort of design other than the default color and style of JavaFX. In past semesters, this is a good opportunity to use images as background and play with the style of the graphical elements.
   - It will also allow the user to input the port number and ip address to connect to the server. Once connected to the server, the client program will change the GUI to the game play screen.

2) For the client game play screen:
   - There should be an area to display both Players cards and Dealers cards with each clearly labeled. You may use images or text to display the cards.
   - Each player must have some way to make all of the available game wagers.
   - Each player should have a separate area to display the Ante, Pair Plus and Play wager.
   - Each player should have a separate area to display total winnings.
   - There should be an area that displays info for the game. For example:

     "Player one loses Pair Plus"
     "Player one beats dealer"
     "Player two loses to dealer"
     "Player two wins Pair Plus"
     "Dealer does not have at least Queen high; ante wager is pushed"

   - You will need to have a menu bar in your program with one tab: **Options**

Under options you will have **Exit, Fresh Start and NewLook**. **Exit** will end the program while **Fresh Start** will reset each players current winnings to zero and allow the user to start a new game. **NewLook** will change the look of the GUI; such as new colors, fonts, images….etc. While there is no minimum for elements to change, the new look must be noticeable to the average user.

3) A separate JavaFX Scene that is displayed when a player wins or loses the game. This scene will have four elements:

- A message announcing if they won or lost the game.
- An option to play another game(a button is fine for this)
- An option to exit the program (could also be a button)
- How much they won or lost in that single game.

Playing the game between your client and server programs:

The client program will start with the welcome screen. Once connected to the server, it will switch to the game play screen. Initially, it will allow the client to place all their bets for one hand of poker. After all bets are made(ante and/or pair plus), the client will send that information to the server. The server will send back both the players cards and dealers cards to the client. The client GUI will display them as noted above.
The client will be given the option to play or fold.

- If they fold, the server will be notified that the game is over and the client will go to the win/lose screen. If they choose to play another game, the server will be notified and the client will return to the game play screen to make their next bets.
- If they choose to play, the client makes the play wager and sends that info to the server. The client can then see the dealers cards. The server will calculate the result of the game and send back to the client. The client will go to the win/lose screen and see the results of the game.

*****All calculations are done in the server and sent to the client. Failure to do this will result in a large reduction of points for the project*****

Passing info between clients and server:

You must implement the PokerInfo class:

class PokerInfo implements Serializable{}.

This class will be used to send information back and forth between the server and clients. This is the only way you are allowed to send and receive information. The data fields are up to you and can vary depending on students individual implementations.

Testing Code:

You are required to include JUnit 5 test cases for your program. Add these to the src/test/java directory of your Maven Project. You must test the class where the game logic methods in the server are located. Here is an example of some methods that you might put in that class:

CS 342 **Project#3** **Fall 2024**

```
public class ThreeCardLogic

        public static int evalHand(ArrayList<Card> hand);
        public static int evalPPWinnings(ArrayList<Card> hand, int bet);
        public static int compareHands(ArrayList<Card> dealer,
ArrayList<Card> player);
```

UI and UX design:

You are required to use the best practices we discussed in lecture for designing the client programs user interface. The client should know what is happening and what to do at all times. Your interface should be intuitive and not cluttered. The client must have time to see and understand the dealers cards once they are “flipped over” before going to the win/lose screen.

How to Start:

- **Deal a few hands of three card poker, or play online, to understand the logic and game flow as well as the payouts for a winning bid. Note that the online version might be slightly different than what we are doing here.**
- **Create your wireframes and UML class diagrams. Use these to think through how the game and networking will be done**
- **Work on the classes in the server first without worrying about networking or GUI. If you get all those to work, you know you can play the game on the server.**
- **Work on the client GUI. How will you display cards and information? How will you add and delete cards from your interface?**
- **Start with one client and the server.**
- **Try passing simple objects back and forth between the server and your client before trying to pass all of the game information.**
- **Figure out all of the fields in your GameInfo class. How will game information be represented?**
- **Get it working with one client then work on multiple clients at once.**
- **Test everything as you go, do not wait till the end; it will save you a lot of debugging time!**

CS 342 Project#3 Fall 2024

Part 1: Wireframe and UML Class Diagrams:

You are required to create a wireframe for the client GUI and UML Class Diagrams for the server and client programs; including all classes, data members and methods of those classes, interfaces and interactions between them.
For the client wireframe, include a mapping to javafx graphical elements and any programatic notes you want to include. Format these as PDFs.

Part 2: two Maven projects (a server program and a client program):

Your server program will be its own Maven project and your client program will be its own Maven project. You will zip both together for your submission.

Electronic Submission:

If you worked in a group:

- only one of you needs to submit part #1 and part #2.
- You must submit a PDF file called Collaboration.pdf to the collaboration doc submission link on BB. In that document, put both of your names and netIds as well as a description of who worked on what in the project.
- If you worked alone, no need for the Collaboration.pdf.

Part #1: UML diagrams and wireframe

You will create a wireframe of the client GUI as well as class diagrams for the client and the server programs. Put these all in a single PDF file and submit to the assignment link on BB.

Part #2: Two full programs Zip both Maven projects

The server program and the client program and name it with your netid + Project3: for example, I would have a submission called mhalle5Project3.zip, and submit it to the link on Blackboard course website.

Assignment Details:

- You may NOT submit part #1 late.
- You CAN submit part #2 up to 24 hours late for a 10% penalty. Anything later than 24 hours will not be graded and result in a zero.

We will test all projects on the command line using Maven 3.6.3. You may develop in any IDE you chose but make sure your project can be run on the command line using Maven commands. Any project that does not run will result in a zero. If you are unsure about using Maven, come see your TA or Professor.

CS 342 Project#3 Fall 2024

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.