Eric Foote
3502094
CS 4983

Final Project Paper

Introduction

The goal of this project was to create a network application. I decided to create a suite of card games over a network. I am going to outline some aspects of this project, such as the background information on the card games, some of the networking features that I implemeted, the methods used for creating this project with the design process and finally outline some future goals of this project. I am going to begin with some of the background information about the card games of choice.

Background Information

The games that I wanted to implement were blackjack, poker, war, crib and hearts. I am going to quote the rules of the game, that I aimed to implement but not go into a lot of detail on them just enough to state the winning conditions of the game and maybe the individual games scoring system. For blackjack, “the most important blackjack rule is simple: beat the dealer’s hand without going over 21. If you get 21 points exactly on the deal, that is called a blackjack.” [1] Where each card has a particular weight corresponding to the value listed, with the “Ace” corresponding to 11, the “King”, ”Queen”, ”Jack” and 10 all having the same weight of 10. Poker has a more complicated rule set so here is an exact quote from Carnegie Mellon Universities Basic Poker Rules page “Poker is played from a standard pack of 52 cards. (Some variant games use multiple packs or add a few cards called jokers.) The cards are ranked (from high to low) Ace, King, Queen, Jack, 10, 9, 8, 7, 6, 5, 4, 3, 2, Ace. (Ace can be high or low, but is usually high). There are four suits (spades, hearts, diamonds and clubs); however, no suit is higher than another. All poker hands contain five cards, the highest hand wins. “[2] There is a variety of winning conditions that rank hands as better then others but for simplicity I am only going to state a few: high card, if you have the highest rank card you win and two pair, if you have 2 of the same card. The game play of war as stated by Erik Arenson of the spruce crafts website is “Each player simultaneously turns over their top card. The higher card wins the pair; the winning player takes both cards and puts them at the bottom of their face-down pile. (Ace is high card; 2 is low card.) In the event of a tie, the players have a "war." Each player places three cards face down in the middle of the table and then turns a fourth card face up. The player with the higher of these cards takes all 10 cards which are now in the middle. If these two cards are also a tie, additional cards are turned face-up, one at a time, until one player wins; that player takes the entire set of cards. “ You win if you control the entire deck of cards after each of the players start with 26 of the 52 cards. For Hearts, the objective of the game is to have the lowest score as opposed to some other games such as blackjack whose goal is to have a score of as close to 21 as possible without going beyond it, with the hearts having a weight of 1 point and the queen having a weight of 13. Finally, for Cribbage, the rules are quite difficult but the objective of the game is to be the first player to score 121 points. The remainder of the rules for this game is not really necessary. Now that we have some background information on the card games we are now going to talk about some of the networking features applied in this project.

Networking Features

The networking features that I applied in this program was the Java socket class, with the connect method to connect to the server, the Java server socket class, with the accept method to accept connections. I also created an inner class inside my server class that implements the runnable class to

allow the server to be multi-threaded. For the communication between the client and the server I utilized the buffered reader and buffered writer, typically the things that were sent across the network were either Strings or integers. Now that we have listed our network features our next step is to talk about the design process.

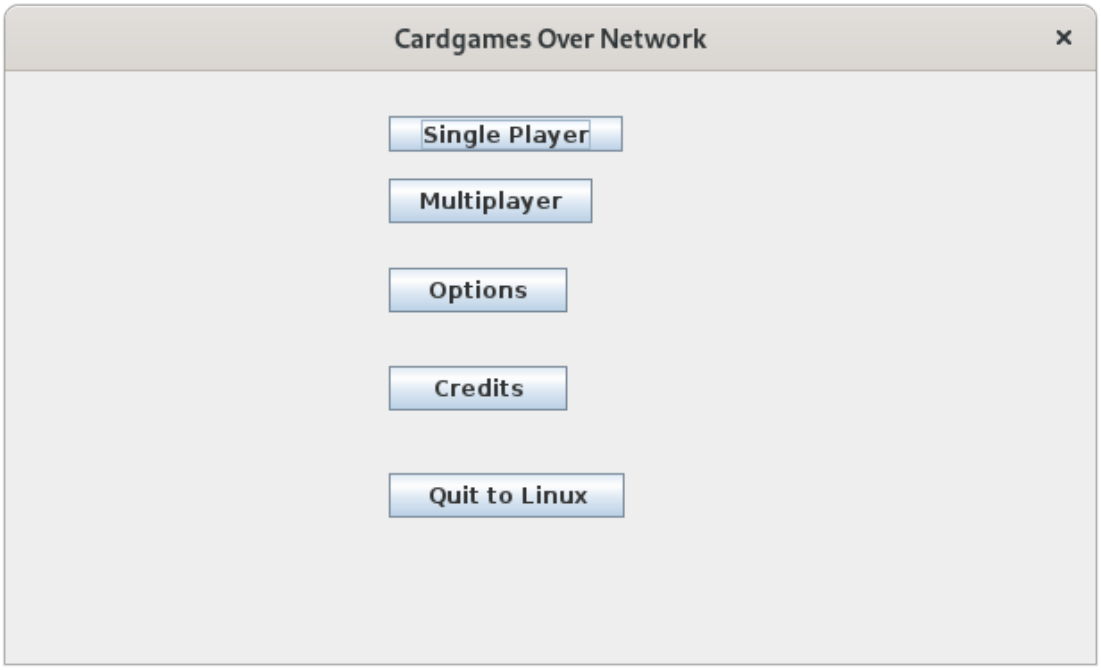
Design Process

The design process I employed for this project was to utilize a helper program called gui genie which is a method for designing the graphical user interfaces, this program was mostly used for designing where exactly in 2D space I wanted buttons, or labels, the remaining functionality was coded by hand since the provided code was valid in my case (i.e. assumes every class that I created contained a main method). The next step was to code the client side up to the games themselves. The final step was to code the server side of the program. The next thing I am going to discuss is how everything was implemented.

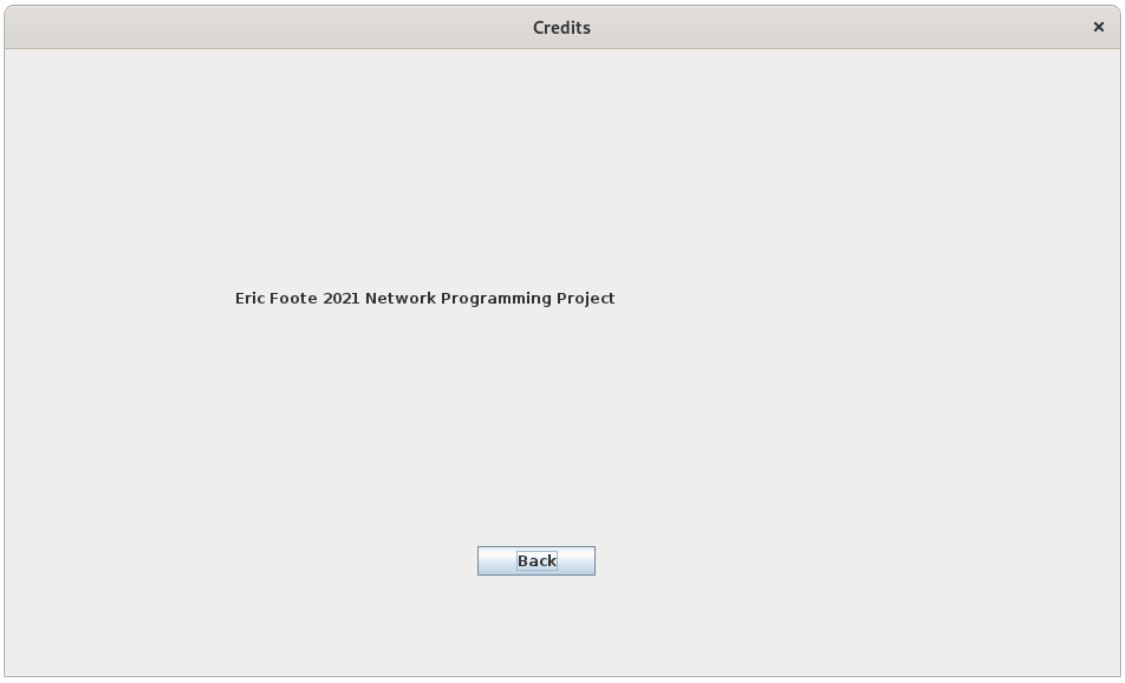
Implementation

We are going to begin our discussion on the implementation of the program with first talking about the structure of the program. The structure of the program was this, the main method of the client side class, creates an instance of the title screen class which simply contains the GUI for the title screen seen below:

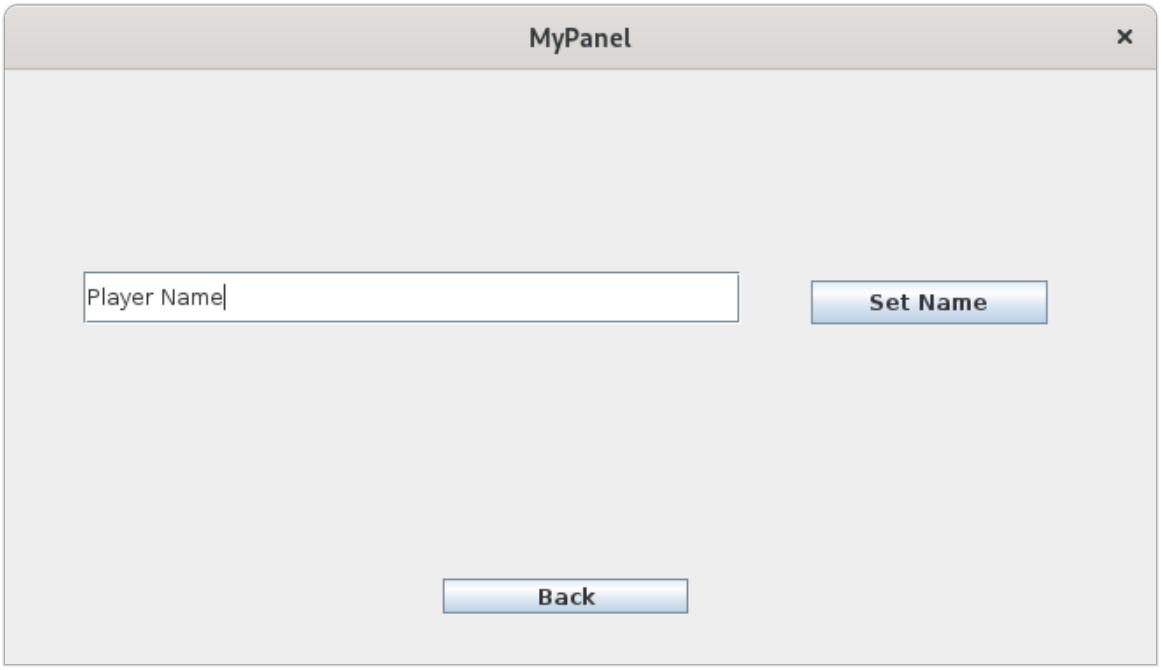
When you hit the start button, it closes and disposes of this frame and calls an instance of the main menu class, which can be seen below:



The main menu contains a few different buttons, starting from the bottom the “Quit to Linux” button simply disposes of the frame and runs system.exit(0) which halts operation of the program. The credits button which opens up this frame:

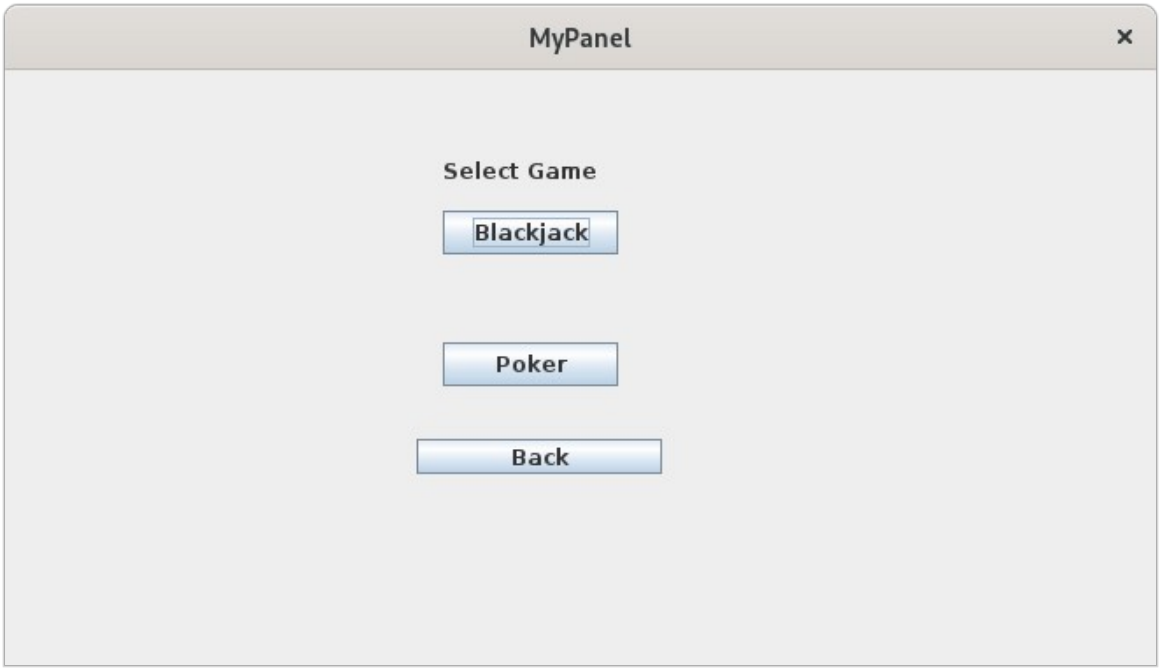


and it also disposes of the previous frame. All the credits frame contains is my name, the class project and a back button which disposes of this frame and returns us back to the main menu by creating a new instance of the main menu class. Next is the options button which creates a new instance of the options class and disposes of the main menu frame, the options frame looks like this:

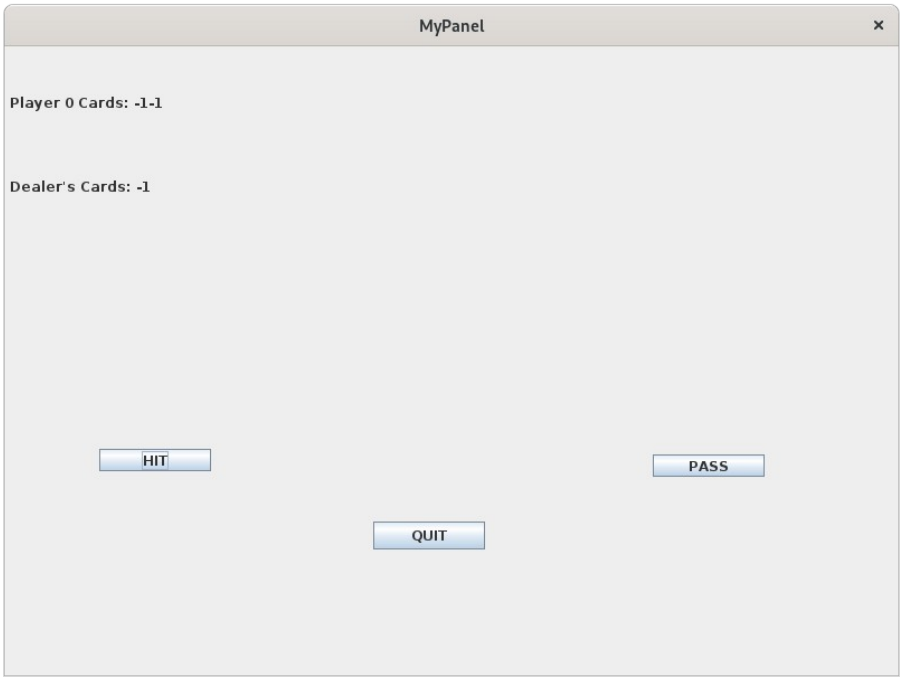


The options frame allows you to set the player name for this particular instance of the client class and the back button does the same as usual, it disposes of the current frame and creates a new instance of the main menu class. I am going to skip the multi player implementation to talk about the single player implementation first. When you press the corresponding button the following frame is created using the single player class:

I never finished the functionality of the single player class but in theory it would upon pressing either the black jack button or the poker button allow the player to play the game locally against the computer. The back button would dispose of this frame and create a new instance of the main menu class. Now we consider the multi player button which disposes of the main menu frame and creates an instance of the multi player class whose frame looks like this:

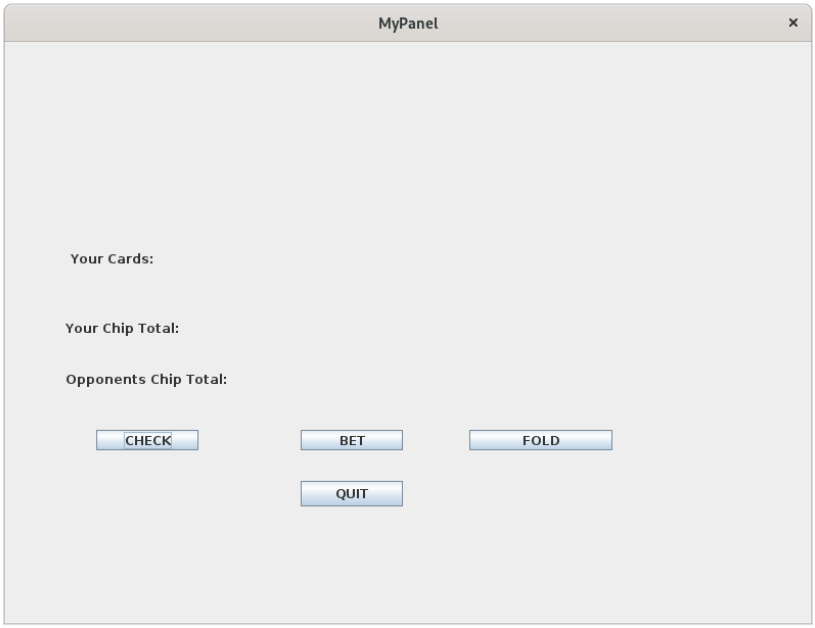


Once this frame is created the client socket tries to connect to the server socket on the specified port and IP address. (for the purposes of testing I used local host for the IP and port 6666). The back button works identically to how we have defined it previously, while the black jack button disposes of this frame and creates an instance of the black jack client side class and creates the frame which looks like this:



The first thing that is supposed to happen is the player number and the 3 cards are supposed to be sent over the server and the buffered reader is supposed to read them in, the hit button is supposed to write the string hit over the network and server side, a new card value is supposed to be sent over the network and read in to client side. Pass is supposed to stop you from making moves and just checks

your totals against the dealer and the other player(s). The quit button is supposed to close the socket connection, dispose of the frame and create a new instance of the main menu. The final thing I am going to discuss is the poker frame which is created when the poker button is hit, it disposes of the multi player frame and creates an instance of the poker frame:



this frame contains an empty space to show the tables cards, a label to contain your cards, your chip total and a one for the opponent's chip totals. The check button's functionality is to write the string check to the server, the fold button's functionality is to write the string fold to the server, upon hitting the bet button, a j input dialog box is created and it allows the player to input the amount of chips they want to bet which is checked client side to make sure that its a valid input then its sent over the network to the server. The last portion of the program to discuss is the server side. On the server side of this program the server socket is created and then we check to see if we are at the maximum amount of players if we are not we create a new thread to handle the individual players moves. A majority of the game logic, when to draw a card, checking for winning or losing is handled by the server rather then the client for a simple reason, to stop any possible manipulation by a player. Our next step is to talk about some future work for this project.

Future Work

One area of future work is allowing the individual clients to set their own names using the options menu, I am thinking that when the set name button is pressed a file is created locally where the java files are stored which contains the name of the player and it is read upon starting the game. Another area where the options menu can be changed is to be able to set the pixel resolution of the j frames going forward, for example, allow the user to set the dimensions of a new j frame to be 800 width x 600 height or 1920 width x 1080 height. Another area of future work is to get the games to work locally against the computer. Another area of potential future work is the creation of a chat room for players to be able to communicate with each other. I would also like to look into writing this in a graphics game engine such as Open GL in order to get some graphics in the game instead of just frames, labels and text boxes. I would also like to enhance the server side of the program to allow for logging of moves to potentially allow players to pick up a game at a later time if one player has to leave the match. However, the main area of future work that I want to consider is actually getting the games to work

because under the current conditions I don't have any of the games working and I am going to elaborate on this further in the conclusion.

Conclusion

In conclusion, I was very over ambitious in the pursuit of this project and the time line for events in the project demonstrates this 5 hours on research, 15 hours designing the GUI's on paper, designing the GUI's in software, implementing the GUI's and fixing any issues, 15 hours working on the server (this includes multiple restarts), 10 hours working on the client-server connection, 5 hours researching and trying possible bug fixes and 6 hours creating this paper and presentation. I was unable to get any of the games to work under this current configuration and in retrospect I would have clearly spent some additional time in order to really figure out where my issues lie in the client server communication, however, it was an enjoyable experience overall in learning how to actually design and start to implement larger scale software from the ground up.

Sources

[1] https://www.blackjack.org/blackjack-rules/
[2] http://www.contrib.andrew.cmu.edu/~gc00/reviews/pokerrules
[3] Erik Arneson October 21 2019 https://www.thesprucecrafts.com/war-card-game-rules-411145
[4] https://www.officialgamerules.org/hearts
[5] https://www.officialgamerules.org/cribbage

Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.*;
import java.io.*;
import java.util.logging.Level;
import java.util.logging.Logger;

// this is the client side for the project


class Introduction extends JPanel implements ActionListener {
   private JButton jcomp1;
   private JLabel jcomp2;
   private JLabel jcomp3;
   JFrame frame = new JFrame ("Cardgames Over Network");
   public Introduction() {
      frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
      frame.pack();

      //construct components
      jcomp1 = new JButton ("START");
      jcomp2 = new JLabel ("Card Games over Network");
```

```
        jcomp3 = new JLabel ("Eric 2021");

        //adjust size and set layout
        frame.setPreferredSize (new Dimension (667, 369));
        frame.setLayout (null);

        //add components
        frame.add (jcomp1);
        frame.add (jcomp2);
        frame.add (jcomp3);

        //set component bounds (only needed by Absolute Positioning)
        jcomp1.setBounds (255, 250, 100, 20);
        jcomp2.setBounds (210, 45, 191, 50);
        jcomp3.setBounds (265, 135, 100, 25);
        jcomp1.addActionListener(this);
        frame.setLocationRelativeTo(null);
        frame.setSize(new Dimension(667,369));
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        new MainMenu();
        frame.setVisible(false);
        frame.dispose();

    }
}

class MainMenu extends JPanel implements ActionListener {
    private JButton jcomp1;
    private JButton jcomp2;
    private JButton jcomp3;
    private JButton jcomp4;
    private JButton jcomp5;
    JFrame frame = new JFrame ("Cardgames Over Network");
    public MainMenu() {

        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.pack();

        //construct components
        jcomp1 = new JButton ("Single Player");
        jcomp2 = new JButton ("Multiplayer");
        jcomp3 = new JButton ("Options");
        jcomp4 = new JButton ("Credits");
        jcomp5 = new JButton ("Quit to Linux");

        //adjust size and set layout
        frame.setPreferredSize (new Dimension (611, 369));
```

```
    frame.setLayout (null);

    //add components
    frame.add (jcomp1);
    frame.add (jcomp2);
    frame.add (jcomp3);
    frame.add (jcomp4);
    frame.add (jcomp5);

    //set component bounds (only needed by Absolute Positioning)
    jcomp1.setBounds (215, 25, 131, 20);
    jcomp2.setBounds (215, 60, 114, 25);
    jcomp3.setBounds (215, 110, 100, 25);
    jcomp4.setBounds (215, 165, 100, 25);
    jcomp5.setBounds (215, 225, 132, 25);
    jcomp1.addActionListener(this);
    jcomp2.addActionListener(this);
    jcomp3.addActionListener(this);
    jcomp4.addActionListener(this);
    jcomp5.addActionListener(this);
    frame.setLocationRelativeTo(null);
    frame.setSize(new Dimension(611,369));
    frame.setVisible (true);
}

public void actionPerformed(ActionEvent e) {
    if(e.getSource() == jcomp1){
        frame.setVisible(false);
        frame.dispose();
        new SinglePlayer();
    }
    if(e.getSource() == jcomp2){
        frame.setVisible(false);
        frame.dispose();
        new MultiPlayer();
    }
     if(e.getSource() == jcomp3){
        frame.setVisible(false);
        frame.dispose();
        new Options();
    }
    if(e.getSource() == jcomp4){
        frame.setVisible(false);
        frame.dispose();
        new Credits();
    }
    if(e.getSource() == jcomp5){
        frame.setVisible(false);
        frame.dispose();
```

```
            System.exit(0);
        }
}
}

class Credits extends JPanel implements ActionListener {
    private JLabel jcomp1;
    private JButton jcomp2;
    JFrame frame = new JFrame ("Credits");

   public Credits(){
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.pack();
        jcomp1 = new JLabel ("Eric Foote 2021 Network Programming Project");
        jcomp2 = new JButton ("Back");
        frame.setPreferredSize (new Dimension (944, 567));
        frame.setLayout (null);
        frame.add(jcomp1);
        frame.add(jcomp2);
        jcomp1.setBounds (195, 70, 500, 280);
        jcomp2.setBounds (400, 420, 100, 25);
        frame.setLocationRelativeTo(null);
        frame.setSize(new Dimension(944,567));
        jcomp2.addActionListener(this);
        frame.setVisible (true);
    }

    public void actionPerformed(ActionEvent e) {
        new MainMenu();
        frame.setVisible(false);
        frame.dispose();

    }
}
class Options extends JPanel implements ActionListener {
    public static String playerName;
    private JButton jcomp1;
    private JButton jcomp2;
    private JTextField jcomp3;
    JFrame frame = new JFrame ("MyPanel");

    public Options() {
        //construct components
        jcomp1 = new JButton ("Set Name");
        jcomp2 = new JButton ("Back");
        jcomp3 = new JTextField ("Player Name");

        //adjust size and set layout
        frame.setPreferredSize (new Dimension (667, 369));
```

```
        frame.setLayout (null);

        //add components
        frame.add (jcomp1);
        frame.add (jcomp2);
        frame.add (jcomp3);

        //set component bounds (only needed by Absolute Positioning)
        jcomp1.setBounds (460, 120, 135, 25);
        jcomp2.setBounds (250, 290, 140, 20);
        jcomp3.setBounds (45, 115, 375, 30);

        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setSize(new Dimension(667,369));
        frame.setLocationRelativeTo(null);
        jcomp1.addActionListener(this);
        jcomp2.addActionListener(this);
        jcomp3.addActionListener(this);
        frame.setVisible (true);

    }

        public void actionPerformed(ActionEvent e) {
        if(e.getSource() == jcomp2) {
            frame.setVisible(false);
            frame.dispose();
            new MainMenu();
        }
        if(e.getSource() == jcomp1) {
            playerName = jcomp3.getText();
        }
        }

}

class SinglePlayer extends JPanel {
    private JButton jcomp1;
    private JButton jcomp2;
    private JButton jcomp3;
    private JLabel jcomp4;
    private JButton jcomp5;
    private JButton jcomp6;
    private JButton jcomp7;
    JFrame frame = new JFrame ("MyPanel");

    public SinglePlayer() {
        //construct components
        jcomp1 = new JButton ("Blackjack");
```

```
        jcomp2 = new JButton ("Poker");
        jcomp3 = new JButton ("Back");
        jcomp4 = new JLabel ("Select Game");


        //adjust size and set layout
        frame.setPreferredSize (new Dimension (667, 369));
        frame.setLayout (null);

        //add components
        frame.add (jcomp1);
        frame.add (jcomp2);
        frame.add (jcomp3);
        frame.add (jcomp4);

        //set component bounds (only needed by Absolute Positioning)
        jcomp1.setBounds (245, 80, 100, 25);
        jcomp2.setBounds (245, 120, 100, 25);
        jcomp3.setBounds (240, 310, 140, 20);
        jcomp4.setBounds (255, 20, 100, 25);
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.setSize(new Dimension (667,369));
        frame.setLocationRelativeTo(null);
        frame.pack();
        frame.setVisible (true);
    }
}

class MultiPlayer extends JPanel implements ActionListener{
    private JButton jcomp1;
    private JLabel jcomp2;
    private JButton jcomp3;
    private JButton jcomp4;
    private Socket socket;
    JFrame frame = new JFrame ("MyPanel");
    public MultiPlayer() {
        try {
            socket = new Socket("127.0.0.1", 6666);
        }
         catch (IOException ex) {
           ex.printStackTrace();
        }
        GUI();
    }

    @Override
    public void actionPerformed(ActionEvent e) {

    if(e.getSource() == jcomp1) {
```

```
    try {
       socket.close();
    } catch (IOException ex) {
       ex.printStackTrace();
    }
    frame.setVisible(false);
    frame.dispose();
    new MainMenu();
}
if(e.getSource() == jcomp3){
        frame.setVisible(false);
        frame.dispose();
        new BlackjackClientSide("Multiplayer",socket);
}
if(e.getSource() == jcomp4) {
            frame.setVisible(false);
            frame.dispose();
            new PokerClientSide("Multiplayer",socket);

        }

    }

public void GUI(){
 //construct components
    jcomp1 = new JButton ("Back");
    jcomp2 = new JLabel ("Select Game");
    jcomp3 = new JButton ("Blackjack");
    jcomp4 = new JButton ("Poker");

    //adjust size and set layout
    frame.setPreferredSize (new Dimension (667, 369));
    frame.setLayout (null);

    //add components
    frame.add (jcomp1);
    frame.add (jcomp2);
    frame.add (jcomp3);
    frame.add (jcomp4);

    //set component bounds (only needed by Absolute Positioning)
    jcomp1.setBounds (235, 210, 140, 20);
    jcomp2.setBounds (250, 45, 100, 25);
    jcomp3.setBounds (250, 80, 100, 25);
    jcomp4.setBounds (250, 155, 100, 25);
    jcomp1.addActionListener(this);
    jcomp3.addActionListener(this);
    jcomp4.addActionListener(this);
```

```
        //set component bounds (only needed by Absolute Positioning)
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setSize(new Dimension (667,369));
        frame.setLocationRelativeTo(null);
        frame.setVisible (true);
        }

}


class BlackjackClientSide implements ActionListener{
        private JButton jcomp1;
        private JButton jcomp2;
        private JButton jcomp3;
        private JLabel jcomp4;
        private JLabel jcomp5;
        private JOptionPane jcomp6;
        public String playerName;
        private Socket s;
        private BufferedReader reader;
        private BufferedWriter writer;
        private int playerNumber;
        private int opponentNumber;
        String initialString = "Player "  + playerNumber + " Cards: ";

        String player2Name = ""; // Get this from the network!
        JFrame frame = new JFrame ("MyPanel");

        public BlackjackClientSide(String gameMode, Socket socket){
        s = socket;
        if(gameMode.equals("SinglePlayer")){
           // Single Player Game Code
        playerName = JOptionPane.showInputDialog("Enter Name"); // this was a feature that if the name
was not set in the options then the user could set it here but I never came back to it
        frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
        frame.pack();

        jcomp1 = new JButton ("HIT");
        jcomp2 = new JButton ("PASS");
        jcomp3 = new JButton ("QUIT");
        jcomp4 = new JLabel (playerName + "s " + "Cards: ");
        jcomp5 = new JLabel ("Dealer's Cards: ");

        //adjust size and set layout
        frame.setPreferredSize (new Dimension (773, 464));
        frame.setLayout (null);
```

```
//add components
frame.add (jcomp1);
frame.add (jcomp2);
frame.add (jcomp3);
frame.add (jcomp4);
frame.add (jcomp5);

//set component bounds (only needed by Absolute Positioning)
jcomp1.setBounds (85, 360, 100, 20);
jcomp2.setBounds (580, 365, 100, 20);
jcomp3.setBounds (330, 425, 100, 25);
jcomp4.setBounds (5, 10, 745, 80);
jcomp5.setBounds (5, 85, 760, 80);
frame.setSize(new Dimension(773,464));
frame.setVisible (true);


}

if(gameMode.equals("Multiplayer")){
   try{
      try{
         reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
         writer = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
      }
      catch(Exception e){
         e.printStackTrace();
      }
      playerNumber = reader.read();
      if(playerNumber == 1){ // sets the player numbers
         opponentNumber = 2;
      }
      else
         opponentNumber = 1;
      frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
      frame.pack();

      jcomp1 = new JButton ("HIT");
      jcomp2 = new JButton ("PASS");
      jcomp3 = new JButton ("QUIT");
      jcomp4 = new JLabel(initialString);
      jcomp5 = new JLabel ("Dealer's Cards: ");


      frame.add(jcomp4);

      //adjust size and set layout
      frame.setPreferredSize (new Dimension (800, 600));
      frame.setLayout (null);
```

```
            //add components
            frame.add (jcomp1);
            frame.add (jcomp2);
            frame.add (jcomp3);
            frame.add (jcomp4);
            frame.add (jcomp5);


            //set component bounds (only needed by Absolute Positioning)
            jcomp1.setBounds (85, 360, 100, 20);
            jcomp2.setBounds (580, 365, 100, 20);
            jcomp3.setBounds (330, 425, 100, 25);
            jcomp4.setBounds (5, 10, 745, 80);
            jcomp5.setBounds (5, 85, 760, 80);
            jcomp3.addActionListener(this);
            frame.setSize(new Dimension(800,600));
            frame.setVisible(true);

        }
        catch(IOException ex){
            ex.printStackTrace();
        }

    }




    try{
    int card1 = reader.read();
    int card2 = reader.read();
    int card3 = reader.read();

    jcomp4.setText(initialString + card2 + "" + card3);
    jcomp5.setText("Dealer's Cards: " + card1);

    }
   catch(Exception e){
       e.printStackTrace();
    }

    }
    @Override
public void actionPerformed(ActionEvent e){
    if(e.getSource() == jcomp3){
        frame.setVisible(false);
        frame.dispose();
        try {
```

```
                s.close();
            } catch (IOException ex) {
              ex.printStackTrace();
            }
            new MainMenu();
        }
    }
    // ran out of time to get the 2 buttons implemented fully

}
class PokerClientSide implements ActionListener{
private JButton jcomp1;
private JButton jcomp2;
private JButton jcomp3;
private JButton jcomp4;
private JLabel jcomp5;
private JLabel jcomp6;
private JLabel jcomp7;
private JLabel jcomp8;
JFrame frame = new JFrame ("MyPanel");
private String bet;
private int Bet;
private Socket socket;
    public PokerClientSide(String gameMode, Socket s){
        socket = s;
        GUI();
}
    public void GUI() {
        //construct components
        jcomp1 = new JButton ("BET");
        jcomp2 = new JButton ("CHECK");
        jcomp3 = new JButton ("FOLD");
        jcomp4 = new JButton ("QUIT");
        jcomp5 = new JLabel (""); // This is going to be the Table
        jcomp6 = new JLabel ("Your Cards:");
        jcomp7 = new JLabel ("Your Chip Total:");
        jcomp8 = new JLabel ("Opponents Chip Total:");

        //adjust size and set layout
        frame.setPreferredSize (new Dimension (800,600));
        frame.setLayout (null);

        //add components
        frame.add (jcomp1);
        frame.add (jcomp2);
        frame.add (jcomp3);
        frame.add (jcomp4);
        frame.add (jcomp5);
        frame.add (jcomp6);
```

```
    frame.add (jcomp7);
    frame.add (jcomp8);

    //set component bounds (only needed by Absolute Positioning)
    jcomp1.setBounds (290, 380, 100, 20);
    jcomp2.setBounds (90, 380, 100, 20);
    jcomp3.setBounds (455, 380, 140, 20);
    jcomp4.setBounds (290, 430, 100, 25);
    jcomp5.setBounds (155, 30, 335, 65);
    jcomp6.setBounds (65, 190, 520, 45);
    jcomp7.setBounds (60, 260, 495, 40);
    jcomp8.setBounds (60, 310, 475, 40);
    jcomp1.addActionListener(this);
    jcomp2.addActionListener(this);
    jcomp3.addActionListener(this);
    jcomp4.addActionListener(this);
    frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setSize(new Dimension(800,600));
    frame.setVisible (true);
}
    @Override
public void actionPerformed(ActionEvent e){
    if(e.getSource() == jcomp1){ // BET LOGIC
      bet = JOptionPane.showInputDialog("Enter Betsize");
      Bet = Integer.parseInt(bet); //get the integer representation of the string
      if(Bet < 0 ){ // OR GREATER THEN CHIP TOTAL
         bet = JOptionPane.showInputDialog("Enter Betsize");
         Bet = Integer.parseInt(bet); //get the integer representation of the string
         //writer.write(Bet);
         //chipTotal -= Bet;
      }
    }
    if(e.getSource() == jcomp2){ // CHECK LOGIC
       //writer.write("CHECK");
    }
    if(e.getSource() == jcomp3){ // FOLD LOGIC
       //writer.write("FOLD")
    }
    if(e.getSource() == jcomp4){ // QUIT LOGIC
       frame.setVisible(false);
       frame.dispose();
      //  try {
      //      s.close();
      //  } catch (IOException ex) {
      //     ex.printStackTrace();
      //  }
       new MainMenu();
    }
```

```
}
}
class Card { // This was my card class, I was going to enhance this further once I got the client server connections working properly in order to allow for the creation of the whole deck of 52 cards rather then each time the card needing to be created again.
    private String Suit;
    private String Value;
        public Card(String suit, String value){
            suit = this.Suit;
            value = this.Value;
        }
        public String getSuit(){
            return Suit;
        }
        public String getValue(){
            return Value;
        }
        public int valueOfCard(){
            if(Value.equals("2"))
                return 2;
            if(Value.equals("3"))
                return 3;
            if(Value.equals("4"))
                return 4;
            if(Value.equals("5"))
                return 5;
            if(Value.equals("6"))
                return 6;
            if(Value.equals("7"))
                return 7;
            if(Value.equals("8"))
                return 8;
            if(Value.equals("9"))
                return 9;
            if(Value.equals("Jack"))
                return 10;
            if(Value.equals("Queen"))
                return 12;
            if(Value.equals("King"))
                return 13;
            else
                return 1;
        }
    @Override
        public String toString(){
            return Value + " of " + Suit;
        }
}
public class CardgamesOverNetwork{
```

```
    public static void main (String[] args){
      new Introduction();
    }

}

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Server {
    private ServerSocket s;
    private int numPlayers;
    private ServerRunnable player1;
    private ServerRunnable player2;
    String gameChoice = "Blackjack"; // This was going to be able to be set for different games, I had it
set to blackjack for debugging the connection
    int [] deckOfCards = {0,1,2,3,4,5}; // this was just temporary
    int playerTotal;
    int DealerTotal;

    Random r = new Random();
    int numberOfCardsDrawn = 0;

    class Card {
    private String Suit;
    private String Value;
        public Card(String suit, String value){
            suit = this.Suit;
            value = this.Value;
        }
        public String getSuit(){
            return Suit;
        }
        public String getValue(){
            return Value;
        }
        public int valueOfCard(){
            if(Value.equals("2"))
                return 2;
            if(Value.equals("3"))
                return 3;
            if(Value.equals("4"))
                return 4;
            if(Value.equals("5"))
                return 5;
```

```
        if(Value.equals("6"))
            return 6;
        if(Value.equals("7"))
            return 7;
        if(Value.equals("8"))
            return 8;
        if(Value.equals("9"))
            return 9;
        if(Value.equals("Jack"))
            return 10;
        if(Value.equals("Queen"))
            return 12;
        if(Value.equals("King"))
            return 13;
        else
            return 1;
    }
}

public Server(){
    System.out.println("Starting Server");
    numPlayers = 0;
    try{
        s = new ServerSocket(6666);
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

public void playerConnections(){
    try{
        System.out.println("Waiting for player connections");
        while(numPlayers < 2 ){
            Socket socket = s.accept();
            numPlayers++;
            ServerRunnable srun = new ServerRunnable(socket,numPlayers);
            if (numPlayers == 1){
                player1 = srun;
            }
            else
                player2 = srun;
            Thread t = new Thread(srun);
            t.start();
            System.out.println("Player " + numPlayers + " Connected");
        }
        System.out.println("Lobby is full. No longer accepting players");
    }
    catch(Exception e){
```

```
            e.printStackTrace();
        }
    }


    private class ServerRunnable implements Runnable {
        private Socket socket;
        private BufferedReader reader;
        private BufferedWriter writer;
        private int playerNumber;

        int turnNumber = 0;
        int Total = 0;


        public boolean winningConditions(){
            if(playerTotal >= 21 | DealerTotal >= 21){ // Implements the winning conditions for an
individual player
                try {
                    if(playerTotal >= 21)
                        writer.write(playerNumber + " Loses");
                    return false;
                } catch (IOException ex) {
                    Logger.getLogger(Server.class.getName()).log(Level.SEVERE, null, ex);
                }
            }
        return true;
        }

        public void BlackJack(){
            while(!winningConditions()) // Checks to make sure that the game is not already lost
            try{
            if (turnNumber == 0){ // Sets up the initial state of the games
                    if(playerNumber == 1){
                        writer.write(deckOfCards[0]); // Send the second dealer card
                        writer.flush();
                        writer.write(deckOfCards[1]); // Send the first player card
                        writer.flush();
                        writer.write(deckOfCards[2]); // Send the second player card
                        writer.flush();
                        Total = deckOfCards[1] + deckOfCards[2];
                    }
                    else if(playerNumber == 2){
                        writer.write(deckOfCards[0]); // Send the second dealer card
                        writer.flush();
                        writer.write(deckOfCards[3]);
                        writer.flush();
                        writer.write(deckOfCards[4]);
                        writer.flush();
```

```
                Total = deckOfCards[3] + deckOfCards[4];
            }

            turnNumber++;
        }
            else{
                String response = "";
                response = reader.readLine();
                if(response.equals("HIT")){
                    if(playerNumber == 1){
                    writer.write(deckOfCards[5]);
                    writer.flush();
                    }
                    else if(playerNumber == 2){
                        writer.write(deckOfCards[6]);
                    writer.flush();
                    }
                }

}
    }
    catch(Exception e){
        e.printStackTrace();
}
}

public void Poker(){

}

public ServerRunnable(Socket s, int pNum){
    socket = s;
    playerNumber = pNum;
    try{
        reader = new BufferedReader(new InputStreamReader(s.getInputStream()));
        writer = new BufferedWriter(new OutputStreamWriter(s.getOutputStream()));
    }
    catch(Exception e){
        e.printStackTrace();
    }
}

public void run(){
    try{
        writer.write(playerNumber);
        writer.flush();
            if(gameChoice.equals("BlackJack")){
                BlackJack();
             }
```

```
            //        else if(gameChoice.equals("Poker")){
             //           Poker();
             //       }

                }
            catch(Exception e){
                e.printStackTrace();
            }
        }
    }

public static void main(String[] args){
    Server server = new Server();
    server.playerConnections();
}

}
```