

[Card Game Multiplayer]

A MINI PROJECT REPORT

Submitted by

[Kunjan Kanani] [92100103216]

[Jimish Khokhar] [92100103228]

[Darshan Rathod] [92100103226]

BACHELOR OF ENGINEERING

in

Computer Engineering



Marwadi University, Rajkot

[Month, Year]



Marwadi University
Rajkot

CERTIFICATE

This is to certify that the project report submitted along with the project entitled < Card-Game Multiplayer> has been carried out by [**Kunjan Kanani**][92100103216] , [**Jimish Khokhar**][92100103228] , [**Darshan Rathod**][92100103226] under my guidance in partial fulfilment for the degree of Bachelor of Technology in Computer Engineering, 3rd Semester of Marwari University, Rajkot during the academic year 2022-23.

Sign

Sign

Name of Internal Guide

Name of Head of the Department

Internal Guide

Head of the Department



Marwadi University
Rajkot

DECLARATION

We hereby declare that the Mini Project-I report submitted along with the Project entitled **Card-Game Multiplayer** submitted in partial fulfilment for the degree of Bachelor of Technology in <Name of the Branch> to Marwadi University, Rajkot, is a bonafide record of original project work carried out by me / us at Marwadi University under the supervision of **Ravi Kumar Natrajan** and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

Name of the Student

Sign of Student

1. Kunjan Kanani
2. Jimish Khokhar
3. Darshan Rathod

List of Figures

Fig 1.1 Use case / Procedure Diagram	2
Fig 1.2 Activity / Process Diagram	5
Fig 2.1 Use case / Procedure Diagram	12
Fig 2.2 Activity / Process Diagram	15
Fig 2.3 Use case / Procedure Diagram	22
Fig 2.4 Activity / Process Diagram	25
Fig 3.1 Use case / Procedure Diagram	32
Fig 4.1 Activity / Process Diagram	35
Fig 4.2 Use case / Procedure Diagram	42
Fig 4.3 Activity / Process Diagram	43

List of Tables

Table 1.1 Popular Methods / Techniques	2
Table 1.2 User / Reading / Observation Table	5
Table 2.1 Popular Methods / Techniques	12
Table 2.2 User / Reading / Observation Table	15
Table 2.3 Popular Methods / Techniques	22
Table 2.4 User / Reading / Observation Table	25
Table 3.1 Popular Methods / Techniques	32
Table 4.1 User / Reading / Observation Table	35
Table 4.2 Popular Methods / Techniques	42
Table 4.3 User / Reading / Observation Table	43

Abbreviations

ALU	Arithmetical & Logical Unit
SDLC	Software Development Life Cycle

Table of Contents

Acknowledgement.....	i
Abstract	ii
List of Figures	iii
List of Tables	iv
List of Abbreviations	v
Table of Contents	vi
Chapter 1	1
1.1 Introduction to Java	2
1.1.1 Benefits of Java.....	3
1.2 Add required topic.....	4
Chapter 2	8
2.1 Introduction to Project Topic.....	9
2.1.1 How to do	10
2.2 Drawbacks in Existing System	11
2.3 Advantages of Proposed System	12
2.4 Functional Requirements.....	13
2.4.1 Tools	14
2.4.2 Front End and Back End	14
Chapter 3	15
3.1 Source code	14
Chapter 4	15
4.1 Screenshots	17
Chapter 5	15
4.1 Conclusion	17
4.2 Future Enhancement.....	17
References.....	40

CHAPTER 1

OVERVIEW OF JAVA

1.1 Introduction of Java

JAVA was developed by James Gosling at **Sun Microsystems** Inc in the year **1995**, later acquired by Oracle Corporation. It is a simple programming language. Java makes writing, compiling, and debugging programming easy. It helps to create reusable code and modular programs. Java is a class-based, object-oriented programming language and is designed to have as few implementation dependencies as possible. A general-purpose programming language made for developers to *write once run anywhere* that is compiled Java code can run on all platforms that support Java. Java applications are compiled to byte code that can run on any Java Virtual Machine. The syntax of Java is similar to c/c++.

History: Java's history is very interesting. It is a programming language created in 1991. James Gosling, Mike Sheridan, and Patrick Naughton, a team of Sun engineers known as the **Green team** initiated the Java language in 1991. **Sun Microsystems** released its first public implementation in 1996 as **Java 1.0**. It provides no-cost -run-times on popular platforms. Java1.0 compiler was re-written in Java by Arthur Van Hoff to strictly comply with its specifications. With the arrival of Java 2, new versions had multiple configurations built for different types of platforms.

In 1997, Sun Microsystems approached the ISO standards body and later formalized Java, but it soon withdrew from the process. At one time, Sun made most of its Java implementations available without charge, despite their proprietary software status. Sun generated revenue from Java through the selling of licenses for specialized products such as the Java Enterprise System.

On November 13, 2006, Sun released much of its Java virtual machine as free, open-source software. On May 8, 2007, Sun finished the process, making all of its JVM's core code available under open-source distribution terms.

The principles for creating java were simple, robust, secured, high performance, portable, multi-threaded, interpreted, dynamic, etc. In 1995 Java was developed by **James Gosling**, who is known as the Father of Java. Currently, Java is used in mobile devices, internet programming, games, e-business, etc.

CHAPTER 2

OVERVIEW OF PROJECT

2.1 Card-Game Multiplayer

- There were very few card game applications or applets written in Java.
- So, We made a game of card to play with your friends ,win this game and enjoy that great moment.
- In this game connect with host in same stream and play this game.
- In the game two player can play to build connection using Java Socket programming.
- The CardGame class is a very simple one: it contains values signaling the color and the value. It may also have images and similar entities that describe the card, when start game shuffle the card and distribute it to player and host.
- According that card player can do bit on it, increase bit, do claim to show cards or pack own game.
- All this things visual better we use Graphical User Interface (GUI).

2.2 Drawbacks of Existing System

- Only Played by two Players

2.3 Advantages of Proposed System

- Very easy to play
- Can play on lan network
- Avoid data manipulations.

2.4 Functional Requirements

2.4.1 Tools

- VS Code
- Net Beans for GUI

2.4.2 Front End and Back End

- Core JAVA
- Local System
- Swing For GUI

CHAPTER 3

PROJECT SOURCE CODE

3.1 CardGame.java

```
1. import javax.swing.*;
2. import java.net.*;
3. import java.io.*;
4. import java.awt.Font;
5. import java.util.ArrayList;
6. import java.util.*;
7.
8. import java.awt.Color;
9. import java.awt.event.*;
10.
11. public class CardGame {
12.     String nameValString;
13.
14.     JFrame MainFrame;
15.     JLabel Name;
16.     JLabel instructions;
17.     JTextArea NameValue;
18.     JButton b1;
19.     JButton b2;
20.     JFrame HostFrame;
21.     JLabel TH1;
22.     JLabel TH2;
23.     JButton Hplay;
24.     JFrame JoinFrame;
25.     JLabel TJ1;
26.     JButton Jplay;
27.     JLabel TJ2;
28.     JLabel TJ3;
29.     JLabel TJ4;
30.     JTextField IPAdd;
31.
32.     String HostName;
33.     String PlayerName;
34.
35.     ServerSocket ss;
36.     Socket s;
37.     DataOutputStream dout;
38.     DataInputStream din;
39.     ArrayList<String> isSelected;
40.     String ALLCARDS[]=new String[6];
41.
42.     String
card[]={ "2H", "3H", "4H", "5H", "6H", "7H", "8H", "9H", "0H", "JH", "QH", "KH", "AH", "2S", "3S",
"4S", "5S", "6S", "7S", "8S", "9S", "0S", "JS", "QS", "KS", "AS", "2C", "3C", "4C", "5C", "6C", "7C",
"8C", "9C", "0C", "JC", "QC", "KC", "AC", "2D", "3D", "4D", "5D", "6D", "7D", "8D", "9D", "0D", "
JD", "QD", "KD", "AD"};
43.     public static void main(String[] args) {
44.         new CardGame();
45.     }
```

```

46.
47.     String[] GetCards()
48.     {
49.         Random rand=new Random();
50.         String []AllCards=new String[6];
51.         isSelected=new ArrayList<String>();
52.         int n=6;
53.         int i=0;
54.         while(n!=0)
55.         {
56.             int randomNumber=rand.nextInt(52);
57.             if(isSelected.contains(card[randomNumber]))
58.             {
59.                 continue;
60.             }
61.             AllCards[i]=card[randomNumber];
62.             isSelected.add(card[randomNumber]);
63.             System.out.println(card[randomNumber]);
64.             i++;
65.             n--;
66.         }
67.         return AllCards;
68.     }
69.
70. CardGame()
71. {
72.     // shuffle(card);
73.     // String[] MyCardsHost={card[0],card[2],card[4]};
74.     // String[] MyCardsPlayer={card[1],card[3],card[5]};
75.
76.
77.
78.     this.ALLCARDS=GetCards();
79.
80.     MainFrame = new JFrame("Card Game");
81.     MainFrame.getContentPane().setBackground(Color.CYAN);
82.     MainFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
83.     Name=new JLabel("Enter Your Name:");
84.     Name.setBounds(50,100,100,30);
85.     MainFrame.add(Name);
86.
87.     instructions=new JLabel("Please HOST The Game First Then Join The Game");
88.     instructions.setBounds(50,300,300,30);
89.     MainFrame.add(instructions);
90.
91.     NameValue=new JTextArea();
92.     NameValue.setBounds(160,107,150,18);
93.     MainFrame.add(NameValue);
94.
95.     b1=new JButton("Host");
96.     b1.setBounds(50,250,95,30);
97.     MainFrame.add(b1);
98.
99.     b2=new JButton("Join");
100.     b2.setBounds(250,250,95,30);
101.     MainFrame.add(b2);
102.     TJ4=new JLabel("Enter The IP Address Of HOST:");
103.     TJ4.setBounds(50,150,180,30);
104.     MainFrame.add(TJ4);
105.     IPAdd=new JTextField();

```

```

106.         IPAdd.setFont(new Font("Serif", Font.PLAIN, 20));
107.         IPAdd.setBounds(230,150,150,25);
108.         MainFrame.add(IPAdd);
109.         JLabel ins=new JLabel("Insert Host IP Address only if You are
JOINING THE GAME!");
110.         ins.setBounds(50,200,350,30);
111.         MainFrame.add(ins);
112.
113.
114.
115.
116.
117.
118.
119.         HostFrame = new JFrame("Card Game");
120.         HostFrame.setSize(800,800);
121.         HostFrame.getContentPane().setBackground(Color.CYAN);
122.         HostFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
123.         TH1=new JLabel("Host Frame!");
124.         TH1.setFont(new Font("Serif", Font.BOLD, 20));
125.         TH1.setBounds(370,10, 300,15);
126.         HostFrame.add(TH1);
127.         TH2=new JLabel("WAITING FOR PLAYER TO JOIN!...");
128.         TH2.setFont(new Font("Monospace", Font.PLAIN, 35));
129.         TH2.setBounds(100,60,700,40);
130.         HostFrame.add(TH2);
131.         Hplay=new JButton("Play");
132.         Hplay.setBounds(320,600,100,30);
133.         HostFrame.add(Hplay);
134.
135.
136.         JoinFrame = new JFrame("Card Game");
137.         JoinFrame.setSize(800,800);
138.         JoinFrame.getContentPane().setBackground(Color.CYAN);
139.         JoinFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
140.         TJ1=new JLabel("Join Frame");
141.         TJ1.setFont(new Font("Serif", Font.BOLD, 20));
142.         TJ1.setBounds(370,10, 300,30);
143.         JoinFrame.add(TJ1);
144.         TJ2=new JLabel("");
145.         TJ2.setFont(new Font("Monospace", Font.PLAIN, 35));
146.         TJ2.setBounds(100,60,700,30);
147.         JoinFrame.add(TJ2);
148.
149.
150.
151.
152.         //populate your frames with stuff
153.         b1.addActionListener(new ActionListener(){
154.             public void actionPerformed(ActionEvent e){
155.                 nameValString=NameValue.getText().toString();
156.                 if(nameValString.length()==0)
157.                 {
158.                     JOptionPane.showMessageDialog(MainFrame, "Please Enter
The Name!");
159.                     return;
160.                 }
161.                 MainFrame.setVisible(false);
162.                 HostFrame.setLayout(null);
163.                 HostFrame.setVisible(true);

```

```

164.         if (MakeHost())
165.         {
166.
167.             TH2.setText("Player Joined !!");
168.             HostFrame.setTitle("Card Game");
169.         }
170.         else return;
171.
172.     }
173. });
174.
175.     b2.addActionListener(new ActionListener(){
176.         public void actionPerformed(ActionEvent e){
177.             String nameValString=NameValue.getText().toString();
178.
179.             if(nameValString.length()==0)
180.             {
181.                 JOptionPane.showMessageDialog(MainFrame, "Please Enter
The Name!");
182.                 return;
183.             }
184.             if(IPAdd.getText().toString().length()==0)
185.             {
186.                 JOptionPane.showMessageDialog(MainFrame, "Please Enter
The IP Address Of HOST's COMPUTER..... \n ASK THE HOST-PLAYER!!");
187.                 return;
188.             }
189.             MainFrame.setVisible(false);
190.             JoinFrame.setLayout(null);
191.             JoinFrame.setVisible(true);
192.             if(MakeJoin())
193.             {
194.                 TJ2.setText("FAILED TO CONNECT ! PLEASE TRY TO JOIN
AGAIN");
195.                 TJ3=new JLabel();
196.                 TJ3.setText("Waiting For Host TO Start The
GAME!!!....");
197.                 TJ3.setFont(new Font("Monospace", Font.PLAIN, 30));
198.                 TJ3.setBounds(100,120,600,50);
199.                 JoinFrame.add(TJ3);
200.             }
201.             else return;
202.
203.             JoinFrame.setTitle("WAITING FOR THE HOST TO START THE
GAME!!!!");
204.
205.             String IsStart="";
206.             try {
207.                 IsStart=din.readUTF().toString();
208.             } catch (Exception e2) {
209.                 // TODO: handle exception
210.             }
211.             if(IsStart.equals("S"))
212.             {
213.
214.                 JoinFrame.setVisible(false);
215.                 new PlayerGame(din,dout,ALLCARDS);
216.             }
217.
218.         }

```

```

219.         });
220.
221.         Hplay.addActionListener(new ActionListener(){
222.             public void actionPerformed(ActionEvent e){
223.                 HostFrame.setVisible(false);
224.
225.                 try {
226.                     dout.writeUTF("S");
227.                 } catch (Exception e1) {
228.                     // TODO: handle exception
229.                 }
230.                 HostFrame.setVisible(false);
231.                 new HostGame(din,dout,ALLCARDS);
232.             }
233.         });
234.
235.
236.
237.
238.         MainFrame.add(b1);
239.         MainFrame.add(b2);
240.         MainFrame.setSize(400,400);
241.         MainFrame.setLayout(null);
242.         MainFrame.setVisible(true);
243.
244.     }
245.
246.     private Boolean MakeHost()
247.     {
248.         HostFrame.setTitle("PLEASE JOIN THE GAME FROM OTHER PLAYER!!");
249.         HostFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
250.         try {
251.             ss=new ServerSocket(6666);
252.             s=ss.accept();
253.             din=new DataInputStream(s.getInputStream());
254.             dout=new DataOutputStream(s.getOutputStream());
255.             //Host Sends The Name First
256.             dout.writeUTF(nameValString);
257.
258.             HostName=nameValString;
259.
260.             return true;
261.
262.
263.         } catch (Exception e) {
264.             // TODO: handle exception
265.             return false;
266.         }
267.
268.
269.
270.     }
271.
272.     private boolean MakeJoin()
273.     {
274.         try {
275.             s=new Socket(IPAdd.getText().toString(),6666);
276.             din=new DataInputStream(s.getInputStream());
277.             dout=new DataOutputStream(s.getOutputStream());
278.

```

```

279.         //Player Accepts The HostName First
280.         HostName=din.readUTF();
281.
282.
283.
284.         PlayerName=nameValString;
285.
286.
287.
288.
289.
290.     } catch (Exception e) {
291.         // TODO: handle exception
292.         return false;
293.     }
294.
295.
296.         return true;
297.
298.     }
299.
300. }
301.
302.
303. class CardGetter
304. {
305.     static ArrayList<String> card = new ArrayList<String>(
306.         Arrays.asList("2H","3H","4H","5H","6H","7H","8H","9H","0H","JH",
"QH","KH","AH","2S","3S","4S","5S","6S","7S","8S","9S","0S","JS","QS","KS","AS","2C",
"3C","4C","5C","6C","7C","8C","9C","0C","JC","QC","KC","AC","2D","3D","4D","5D",""
"6D","7D","8D","9D","0D","JD","QD","KD","AD"));
307.
308.     public String[] GetMyCards()
309.     {
310.         Random rand=new Random();
311.         String MyCards[]=new String[3];
312.
313.         for(int i=0;i<3;i++)
314.         {
315.             int randomNumber=rand.nextInt(card.size());
316.             MyCards[i]=card.get(randomNumber);
317.             card.remove(randomNumber);
318.             System.out.println(card.size());
319.         }
320.         return MyCards;
321.     }
322. }
323.
324. class ImageIconGetter
325. {
326.     static Map<String,String> map=new HashMap<String,String>();
327.     ArrayList<String> card = new ArrayList<String>(
328.         Arrays.asList("2H","3H","4H","5H","6H","7H","8H","9H","0H","JH",
"QH","KH","AH","2S","3S","4S","5S","6S","7S","8S","9S","0S","JS","QS","KS","AS","2C",
"3C","4C","5C","6C","7C","8C","9C","0C","JC","QC","KC","AC","2D","3D","4D","5D",""
"6D","7D","8D","9D","0D","JD","QD","KD","AD"));
329.         //4H","5H","6H","7H","8H","9H","0H","JH","QH","KH","AH","2S","3S","4S",""
"5S","6S","7S","8S","9S","0S","JS","QS","KS","AS","2C","3C","4C","5C","6C","7C","8C",
"9C","0C","JC","QC","KC","AC","2D","3D","4D","5D","6D","7D","8D","9D","0D","JD","Q
D","KD","AD"

```

```

330.         ImageIconGetter()
331.         {
332.             for(int i=0;i<52;i++)
333.             {
334.                 map.put(card.get(i), "ALLCards\\"+card.get(i)+".png");
335.             }
336.         }
337.         String GetTheImageIcon(String k)
338.         {
339.             return map.get(k);
340.         }
341.     }
342.
343.
344.     class HostGame extends javax.swing.JFrame
345.     {
346.         private static javax.swing.JLabel C1;
347.         private static javax.swing.JLabel C2;
348.         private static javax.swing.JLabel C3;
349.         private javax.swing.JButton DecrementBit;
350.         private javax.swing.JButton IncrementBit;
351.         private javax.swing.JLabel Kaalein;
352.         private javax.swing.JLabel OppPlayerFace;
353.         private javax.swing.JLabel PoolBalance;
354.         private javax.swing.JLabel TurnIndicator;
355.         private javax.swing.JLabel YourBalance;
356.         private javax.swing.JButton btnbet;
357.         private javax.swing.JButton btnpack;
358.         private javax.swing.JButton btnsee;
359.         private javax.swing.JButton btnshow;
360.         private javax.swing.JLabel labcurrval;
361.         private javax.swing.JLabel oppbal;
362.
363.         //Connection components
364.         DataInputStream din;
365.         DataOutputStream dout;
366.
367.         //Game Variables
368.
369.         int YourBalanceValue;//host
370.         int PoolBalanceValue;
371.         int OppositeBalanceValue;//player
372.         int CurrentBitValue;
373.         int Increment;
374.         boolean BtnVisible;
375.         String ALLCARDS[]=new String[6];
376.         String MyCards[]=new String[3];
377.         boolean isVisibleCards;
378.         int k;
379.         boolean isSeen;
380.
381.         HostGame(DataInputStream din,DataOutputStream dout,String[] allcards)
382.         {
383.             this.din=din;
384.             this.dout=dout;
385.
386.             // CardGetter cardGetter=new CardGetter();
387.             // MyCards=cardGetter.GetMyCards();
388.             this.ALLCARDS=allcards;
389.             MyCards[0]=ALLCARDS[0];

```



```

390.         MyCards[1]=ALLCARDS[1];
391.         MyCards[2]=ALLCARDS[2];
392.         isSeen=false;
393.
394.         k=0;
395.
396.         isVisibleCards=false;
397.
398.         YourBalanceValue=1000;//host
399.         PoolBalanceValue=0;
400.         OppositeBalanceValue=YourBalanceValue;//player
401.         CurrentBitValue=10;
402.         Increment=0;
403.         BtnVisible=true;
404.
405.         IncrementBit = new javax.swing.JButton();
406.         btnbet = new javax.swing.JButton();
407.         btnshow = new javax.swing.JButton();
408.         btnpack = new javax.swing.JButton();
409.         C3 = new javax.swing.JLabel();
410.         C1 = new javax.swing.JLabel();
411.         C2 = new javax.swing.JLabel();
412.         labcurrval = new javax.swing.JLabel();//Current Bit Value
413.         OppPlayerFace = new javax.swing.JLabel();
414.         TurnIndicator = new javax.swing.JLabel();
415.         oppbal = new javax.swing.JLabel();
416.         PoolBalance = new javax.swing.JLabel();
417.         btnsee = new javax.swing.JButton();
418.         DecrementBit = new javax.swing.JButton();
419.         YourBalance = new javax.swing.JLabel();
420.         Kaalein = new javax.swing.JLabel();
421.
422.
423.
424.         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
425.         setTitle("Card Game ");
426.         setMinimumSize(new java.awt.Dimension(910, 584));
427.         getContentPane().setLayout(null);
428.
429.         IncrementBit.setBackground(new java.awt.Color(102, 255, 255));
430.         IncrementBit.setFont(new java.awt.Font("Segoe UI", 1, 36)); //
NOI18N
431.         IncrementBit.setText("+");
432.         IncrementBit.setAlignmentY(0.0F);
433.         IncrementBit.setAutoscrolls(true);
434.         IncrementBit.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
435.         IncrementBit.setMaximumSize(new java.awt.Dimension(100, 100));
436.         IncrementBit.setMinimumSize(new java.awt.Dimension(100, 100));
437.         IncrementBit.setPreferredSize(new java.awt.Dimension(100, 100));
438.         IncrementBit.addActionListener(new java.awt.event.ActionListener() {
439.             public void actionPerformed(java.awt.event.ActionEvent evt) {
440.                 IncrementBitActionPerformed(evt);
441.             }
442.         });
443.         getContentPane().add(IncrementBit);
444.         IncrementBit.setBounds(660, 420, 60, 50);
445.
446.         btnbet.setBackground(new java.awt.Color(255, 102, 51));
447.         btnbet.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

```

```

448.         btnbet.setText("BET");
449.         btnbet.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
450.         getContentPane().add(btnbet);
451.         btnbet.setBounds(760, 460, 100, 50);
452.
453.         btnshow.setBackground(new java.awt.Color(102, 255, 255));
454.         btnshow.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
455.         btnshow.setText("SHOW");
456.         btnshow.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
457.         btnshow.addActionListener(new java.awt.event.ActionListener() {
458.             public void actionPerformed(java.awt.event.ActionEvent evt) {
459.                 btnshowActionPerformed(evt);
460.             }
461.         });
462.         getContentPane().add(btnshow);
463.         btnshow.setBounds(60, 400, 100, 50);
464.
465.         btnpack.setBackground(new java.awt.Color(102, 255, 255));
466.         btnpack.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
467.         btnpack.setText("PACK");
468.         btnpack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
469.         btnpack.addActionListener(new java.awt.event.ActionListener() {
470.             public void actionPerformed(java.awt.event.ActionEvent evt) {
471.                 btnpackActionPerformed(evt);
472.             }
473.         });
474.         getContentPane().add(btnpack);
475.         btnpack.setBounds(60, 470, 100, 50);
476.
477.         C3.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
478.         getContentPane().add(C3);
479.         C3.setBounds(690, 190, 153, 220);
480.
481.         C1.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
482.         getContentPane().add(C1);
483.         C1.setBounds(350, 190, 153, 220);
484.
485.         C2.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
486.         getContentPane().add(C2);
487.         C2.setBounds(520, 190, 153, 220);
488.
489.         labcurrval.setBackground(new java.awt.Color(0, 0, 0));
490.         labcurrval.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
491.         labcurrval.setForeground(new java.awt.Color(255, 51, 51));
492.         labcurrval.setText("$ current bit value");
493.         labcurrval.setOpaque(true);
494.         getContentPane().add(labcurrval);
495.         labcurrval.setBounds(310, 420, 330, 50);
496.
497.         OppPlayerFace.setIcon(new javax.swing.ImageIcon("opposite-player-
final.png")); // NOI18N
498.         getContentPane().add(OppPlayerFace);
499.         OppPlayerFace.setBounds(50, 40, 211, 170);
500.
501.         TurnIndicator.setBackground(new java.awt.Color(0, 0, 0));
502.         TurnIndicator.setFont(new java.awt.Font("Segoe UI", 1, 36)); //
NOI18N
503.         TurnIndicator.setForeground(new java.awt.Color(0, 0, 255));
504.         TurnIndicator.setText("Player 1's TURN");
505.         TurnIndicator.setOpaque(true);

```

```

506.         getContentPane().add(TurnIndicator);
507.         TurnIndicator.setBounds(350, 40, 280, 50);
508.
509.         oppbal.setBackground(new java.awt.Color(0, 0, 0));
510.         oppbal.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
511.         oppbal.setForeground(new java.awt.Color(255, 51, 51));
512.         oppbal.setText("$ opposite val");
513.         oppbal.setOpaque(true);
514.         getContentPane().add(oppbal);
515.         oppbal.setBounds(60, 220, 190, 40);
516.
517.         PoolBalance.setBackground(new java.awt.Color(0, 0, 0));
518.         PoolBalance.setFont(new java.awt.Font("Segoe UI", 1, 48)); // NOI18N
519.         PoolBalance.setForeground(new java.awt.Color(0, 255, 51));
520.         PoolBalance.setText("$ PoolValue");
521.         PoolBalance.setOpaque(true);
522.         getContentPane().add(PoolBalance);
523.         PoolBalance.setBounds(300, 110, 420, 70);
524.
525.         btnsee.setBackground(new java.awt.Color(102, 255, 255));
526.         btnsee.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
527.         btnsee.setText("SEE CARDS");
528.         btnsee.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
529.         getContentPane().add(btnsee);
530.         btnsee.setBounds(50, 330, 130, 50);
531.
532.         DecrementBit.setBackground(new java.awt.Color(102, 255, 255));
533.         DecrementBit.setFont(new java.awt.Font("Segoe UI", 1, 48)); //
NOI18N
534.         DecrementBit.setText("-");
535.         DecrementBit.setAlignmentY(0.0F);
536.         DecrementBit.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
537.         DecrementBit.setMaximumSize(new java.awt.Dimension(100, 100));
538.         DecrementBit.setMinimumSize(new java.awt.Dimension(100, 100));
539.         DecrementBit.setPreferredSize(new java.awt.Dimension(100, 100));
540.         DecrementBit.addActionListener(new java.awt.event.ActionListener() {
541.             public void actionPerformed(java.awt.event.ActionEvent evt) {
542.                 DecrementBitActionPerformed(evt);
543.             }
544.         });
545.         getContentPane().add(DecrementBit);
546.         DecrementBit.setBounds(230, 420, 60, 50);
547.
548.         YourBalance.setBackground(new java.awt.Color(0, 0, 0));
549.         YourBalance.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
550.         YourBalance.setForeground(new java.awt.Color(153, 255, 153));
551.         YourBalance.setText("$ Your Balance");
552.         YourBalance.setOpaque(true);
553.         getContentPane().add(YourBalance);
554.         YourBalance.setBounds(310, 490, 330, 40);
555.
556.         Kaalein.setIcon(new javax.swing.ImageIcon("back.jpg")); // NOI18N
557.         getContentPane().add(Kaalein);
558.         Kaalein.setBounds(0, 0, 910, 580);
559.
560.
561.         //Initialising All The components As Per the Initial Values of Game
562.         YourBalance.setText("$ "+ YourBalanceValue);
563.         PoolBalance.setText("$ "+ PoolBalanceValue);

```



```

620. JOptionPane.showMessageDialog(f, "Hos
t\n You LOSE!\n"+Win[1]+" \n"+Win[2]);
621. setVisible(false);
622. return;
623. }
624. }
625. OppositeBalanceValue=din.readInt();
626. PoolBalanceValue=din.readInt();
627. CurrentBitValue=din.readInt();
628.
629. if(CurrentBitValue>YourBalanceValue)
630. {
631. //Show Dialog and Tell Him to Pack THE Game
632. JFrame f=new JFrame();
633. JOptionPane.showMessageDialog(f, "Low
Balance\nCurrent Bit Value:-"+CurrentBitValue+" \nYour Balance:-"+YourBalanceValue);
634. try {
635. dout.writeUTF("ilose");
636. } catch (Exception e) {
637. // TODO: handle exception
638. }
639. setVisible(false);
640. return;
641. }
642.
643. TurnIndicator.setText("Your Turn!");
644. YourBalance.setText("$ "+ YourBalanceValue);
645. PoolBalance.setText("$ "+ PoolBalanceValue);
646. oppbal.setText("$ "+ OppositeBalanceValue);
647. labcurrval.setText("$ "+CurrentBitValue);
648. } catch (Exception ex2) {
649. // TODO: handle exception
650. }
651.
652. btnbet.setVisible(true);
653. btnpack.setVisible(true);
654. btnshow.setVisible(true);
655. btnsee.setVisible(true);
656. IncrementBit.setVisible(true);
657. DecrementBit.setVisible(true);
658. }
659. }
660.
661. });
662. th.start();
663.
664.
665. Thread Send=new Thread(new Runnable() {
666. @Override
667. public void run() {
668. // TODO Auto-generated method stub
669.
670. //Button of Betting The Bit
671.
672. btnbet.addActionListener(new ActionListener(){
673. public void actionPerformed(ActionEvent e){
674. k++;
675. btnbet.setVisible(false);
676. btnpack.setVisible(false);
677. btnshow.setVisible(false);

```

```

678.         btnsee.setVisible(false);
679.         IncrementBit.setVisible(false);
680.         DecrementBit.setVisible(false);
681.         Increment=0;
682.         YourBalanceValue-=CurrentBitValue;
683.         PoolBalanceValue+=CurrentBitValue;
684.         try {
685.             if(k==1)
686.             {
687.                 dout.writeUTF("passing");
688.                 dout.writeUTF(ALLCARDS[0]);
689.                 dout.writeUTF(ALLCARDS[1]);
690.                 dout.writeUTF(ALLCARDS[2]);
691.                 dout.writeUTF(ALLCARDS[3]);
692.                 dout.writeUTF(ALLCARDS[4]);
693.                 dout.writeUTF(ALLCARDS[5]);
694.             }
695.             else{
696.                 dout.writeUTF("continue");
697.             }
698.             dout.writeInt(YourBalanceValue);
699.             dout.writeInt(PoolBalanceValue);
700.             dout.writeInt(CurrentBitValue);
701.
702.         } catch (Exception ex3) {
703.             // TODO: handle exception
704.         }
705.         TurnIndicator.setText("Player's Turn");
706.         YourBalance.setText("$ " + YourBalanceValue);
707.         PoolBalance.setText("$ " + PoolBalanceValue);
708.         oppbal.setText("$ " + OppositeBalanceValue);
709.         labcurrval.setText("$ " + CurrentBitValue);
710.
711.         try {
712.             th.notify();
713.         } catch (Exception ex5) {
714.             // TODO: handle exception
715.         }
716.     }
717. });
718.
719. IncrementBit.addActionListener(new
720.     ActionListener(){
721.         public void actionPerformed(ActionEvent e){
722.             if(Increment==1)return;
723.             else{
724.                 if(CurrentBitValue*2>YourBalanceValue)
725.                 {
726.                     return;
727.                 }
728.
729.                 CurrentBitValue*=2;
730.                 labcurrval.setText("$ " + CurrentBitValue);
731.                 Increment++;
732.             }
733.         }
734.     });
735.

```

```

736.         DecrementBit.addActionListener(new
        ActionListener(){
737.             public void actionPerformed(ActionEvent e){
738.                 if(Increment==0)
739.                     return;
740.                 else{
741.                     CurrentBitValue/=2;
742.                     labcurrval.setText("$
"+CurrentBitValue);
743.                     Increment--;
744.                 }
745.             }
746.         });
747.
748.         btnsee.addActionListener(new ActionListener(){
749.             public void actionPerformed(ActionEvent e){
750.                 //Implement The Logic
751.                 //(MyCards[0])ImageIconGetter.GetTheImageIcon
752.                 isSeen=true;
753.
754.                 if(isVisibleCards==false)
755.                 {
756.                     C1.setIcon(new ImageIcon(new
ImageIconGetter().GetTheImageIcon(MyCards[0])));
757.                     C2.setIcon(new ImageIcon(new
ImageIconGetter().GetTheImageIcon(MyCards[1])));
758.                     C3.setIcon(new ImageIcon(new
ImageIconGetter().GetTheImageIcon(MyCards[2])));
759.                     isVisibleCards=true;
760.                 }
761.                 else{
762.                     C1.setIcon(new
ImageIcon("card_back.png"));
763.                     C2.setIcon(new
ImageIcon("card_back.png"));
764.                     C3.setIcon(new
ImageIcon("card_back.png"));
765.                     isVisibleCards=false;
766.                 }
767.             }
768.         });
769.
770.         btnpack.addActionListener(new ActionListener(){
771.             public void actionPerformed(ActionEvent e){
772.                 try {
773.                     dout.writeUTF("ipack");
774.                 } catch (Exception ex6) {
775.                     // TODO: handle exception
776.                 }
777.                 setVisible(false);
778.
779.                 return;
780.             }
781.         });
782.
783.         btnshow.addActionListener(new ActionListener(){
784.             public void actionPerformed(ActionEvent e){
785.                 try {
786.                     dout.writeUTF("showingwinner");

```

```

787.         } catch (Exception ev) {
788.             // TODO: handle exception
789.         }
790.         String Result[]=new String[3];
791.
792.
793.         card_game x=new card_game();
794.         Result=x.GetResult(ALLCARDS);
795.
796.         try {
797.             if(Result[0].equals("2"))
798.             {
799.                 JFrame f=new JFrame();
800.                 JOptionPane.showMessageDialog(f,"Hos
t\n You LOSE!\n Player has:"+Result[1]+"\\n"+Result[2]);
801.                 setVisible(false);
802.             }
803.             else
804.             {
805.                 JFrame f=new JFrame();
806.                 JOptionPane.showMessageDialog(f,"Hos
t\n You WIN!\n"+Result[1]+"\\n"+Result[2]);
807.                 setVisible(false);
808.             }
809.             dout.writeUTF(Result[0]);
810.             dout.writeUTF(Result[1]);
811.             dout.writeUTF(Result[2]);
812.         } catch (Exception x7) {
813.             // TODO: handle exception
814.         }
815.
816.
817.         }
818.     });
819. }
820. });
821. Send.start();
822.
823.
824.
825.     //Over Of Host Code
826.
827. }
828.
829. private void btnshowActionPerformed(java.awt.event.ActionEvent evt)
830. {
831.     // TODO add your handling code here:
832. }
833. private void btnpackActionPerformed(java.awt.event.ActionEvent evt)
834. {
835.     // TODO add your handling code here:
836. }
837. private void IncrementBitActionPerformed(java.awt.event.ActionEvent evt)
838. {
839.     // TODO add your handling code here:
840. }

```



```

841.         private void DecrementBitActionPerformed(java.awt.event.ActionEvent evt)
842.     {
843.         // TODO add your handling code here:
844.     }
845.
846.
847.
848.     class PlayerGame extends javax.swing.JFrame
849.     {
850.         private static javax.swing.JLabel C1;
851.         private static javax.swing.JLabel C2;
852.         private static javax.swing.JLabel C3;
853.         private javax.swing.JButton DecrementBit;
854.         private javax.swing.JButton IncrementBit;
855.         private javax.swing.JLabel Kaalein;
856.         private javax.swing.JLabel OppPlayerFace;
857.         private javax.swing.JLabel PoolBalance;
858.         private javax.swing.JLabel TurnIndicator;
859.         private javax.swing.JLabel YourBalance;
860.         private javax.swing.JButton btnbet;
861.         private javax.swing.JButton btnpack;
862.         private javax.swing.JButton btnsee;
863.         private javax.swing.JButton btnshow;
864.         private javax.swing.JLabel labcurrval;
865.         private javax.swing.JLabel oppbal;
866.
867.         //connection Components
868.         DataInputStream din;
869.         DataOutputStream dout;
870.
871.         //Game Variables
872.         int YourBalanceValue;//player
873.         int PoolBalanceValue;
874.         int OppositeBalanceValue;//host
875.         int CurrentBitValue;
876.         int Increment;
877.         String ALLCARDS[]=new String[6];
878.         String MyCards[]=new String[3];
879.         boolean isVisibleCards=false;
880.         boolean isSeen;
881.
882.         PlayerGame(DataInputStream din,DataOutputStream dout,String[] allcards)
883.         {
884.             this.din=din;
885.             this.dout=dout;
886.
887.             // CardGetter cardGetter=nasdasdew CardGetter();
888.             // MyCards=cardGetter.GetMyCards();
889.
890.             isSeen=false;
891.
892.             YourBalanceValue=1000;//host
893.             PoolBalanceValue=0;
894.             OppositeBalanceValue=YourBalanceValue;//player
895.             CurrentBitValue=10;
896.             Increment=0;
897.
898.             IncrementBit = new javax.swing.JButton();
899.             btnbet = new javax.swing.JButton();

```

```

900.         btnshow = new javax.swing.JButton();
901.         btnpack = new javax.swing.JButton();
902.         C3 = new javax.swing.JLabel();
903.         C1 = new javax.swing.JLabel();
904.         C2 = new javax.swing.JLabel();
905.         labcurrval = new javax.swing.JLabel();
906.         OppPlayerFace = new javax.swing.JLabel();
907.         TurnIndicator = new javax.swing.JLabel();
908.         oppbal = new javax.swing.JLabel();
909.         PoolBalance = new javax.swing.JLabel();
910.         btnsee = new javax.swing.JButton();
911.         DecrementBit = new javax.swing.JButton();
912.         YourBalance = new javax.swing.JLabel();
913.         Kaalein = new javax.swing.JLabel();
914.
915.         setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
916.         setTitle("Card Game ");
917.         setMinimumSize(new java.awt.Dimension(910, 584));
918.         getContentPane().setLayout(null);
919.
920.         IncrementBit.setBackground(new java.awt.Color(102, 255, 255));
921.         IncrementBit.setFont(new java.awt.Font("Segoe UI", 1, 36)); //
NOI18N
922.         IncrementBit.setText("+");
923.         IncrementBit.setAlignmentY(0.0F);
924.         IncrementBit.setAutoscrolls(true);
925.         IncrementBit.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
926.         IncrementBit.setMaximumSize(new java.awt.Dimension(100, 100));
927.         IncrementBit.setMinimumSize(new java.awt.Dimension(100, 100));
928.         IncrementBit.setPreferredSize(new java.awt.Dimension(100, 100));
929.         IncrementBit.addActionListener(new java.awt.event.ActionListener() {
930.             public void actionPerformed(java.awt.event.ActionEvent evt) {
931.                 IncrementBitActionPerformed(evt);
932.             }
933.         });
934.         getContentPane().add(IncrementBit);
935.         IncrementBit.setBounds(660, 420, 60, 50);
936.
937.         btnbet.setBackground(new java.awt.Color(255, 102, 51));
938.         btnbet.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
939.         btnbet.setText("BET");
940.         btnbet.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
941.         getContentPane().add(btnbet);
942.         btnbet.setBounds(760, 460, 100, 50);
943.
944.         btnshow.setBackground(new java.awt.Color(102, 255, 255));
945.         btnshow.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
946.         btnshow.setText("SHOW");
947.         btnshow.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
948.         btnshow.addActionListener(new java.awt.event.ActionListener() {
949.             public void actionPerformed(java.awt.event.ActionEvent evt) {
950.                 btnshowActionPerformed(evt);
951.             }
952.         });
953.         getContentPane().add(btnshow);
954.         btnshow.setBounds(60, 400, 100, 50);
955.
956.         btnpack.setBackground(new java.awt.Color(102, 255, 255));
957.         btnpack.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N

```

```

958.         btnpack.setText("PACK");
959.         btnpack.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
960.         btnpack.addActionListener(new java.awt.event.ActionListener() {
961.             public void actionPerformed(java.awt.event.ActionEvent evt) {
962.                 btnpackActionPerformed(evt);
963.             }
964.         });
965.         getContentPane().add(btnpack);
966.         btnpack.setBounds(60, 470, 100, 50);
967.
968.         C3.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
969.         getContentPane().add(C3);
970.         C3.setBounds(690, 190, 153, 220);
971.
972.         C1.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
973.         getContentPane().add(C1);
974.         C1.setBounds(350, 190, 153, 220);
975.
976.         C2.setIcon(new javax.swing.ImageIcon("card_back.png")); // NOI18N
977.         getContentPane().add(C2);
978.         C2.setBounds(520, 190, 153, 220);
979.
980.         labcurrval.setBackground(new java.awt.Color(0, 0, 0));
981.         labcurrval.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
982.         labcurrval.setForeground(new java.awt.Color(255, 51, 51));
983.         labcurrval.setText("$ current bit value");
984.         labcurrval.setOpaque(true);
985.         getContentPane().add(labcurrval);
986.         labcurrval.setBounds(310, 420, 330, 50);
987.
988.         OppPlayerFace.setIcon(new javax.swing.ImageIcon("opposite-player-
final.png")); // NOI18N
989.         getContentPane().add(OppPlayerFace);
990.         OppPlayerFace.setBounds(50, 40, 211, 170);
991.
992.         TurnIndicator.setBackground(new java.awt.Color(0, 0, 0));
993.         TurnIndicator.setFont(new java.awt.Font("Segoe UI", 1, 36)); //
NOI18N
994.         TurnIndicator.setForeground(new java.awt.Color(0, 0, 255));
995.         TurnIndicator.setText("Player 1's TURN");
996.         TurnIndicator.setOpaque(true);
997.         getContentPane().add(TurnIndicator);
998.         TurnIndicator.setBounds(350, 40, 280, 50);
999.
1000.        oppbal.setBackground(new java.awt.Color(0, 0, 0));
1001.        oppbal.setFont(new java.awt.Font("Segoe UI", 1, 24)); // NOI18N
1002.        oppbal.setForeground(new java.awt.Color(255, 51, 51));
1003.        oppbal.setText("$ opposite val");
1004.        oppbal.setOpaque(true);
1005.        getContentPane().add(oppbal);
1006.        oppbal.setBounds(60, 220, 190, 40);
1007.
1008.        PoolBalance.setBackground(new java.awt.Color(0, 0, 0));
1009.        PoolBalance.setFont(new java.awt.Font("Segoe UI", 1, 48)); // NOI18N
1010.        PoolBalance.setForeground(new java.awt.Color(0, 255, 51));
1011.        PoolBalance.setText("$ PoolValue");
1012.        PoolBalance.setOpaque(true);
1013.        getContentPane().add(PoolBalance);
1014.        PoolBalance.setBounds(300, 110, 420, 70);
1015.

```

```

1016.         btnsee.setBackground(new java.awt.Color(102, 255, 255));
1017.         btnsee.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
1018.         btnsee.setText("SEE CARDS");
1019.         btnsee.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1020.         getContentPane().add(btnsee);
1021.         btnsee.setBounds(50, 330, 130, 50);
1022.
1023.         DecrementBit.setBackground(new java.awt.Color(0, 153, 153));
1024.         DecrementBit.setFont(new java.awt.Font("Segoe UI", 1, 48)); //
NOI18N
1025.         DecrementBit.setText("-");
1026.         DecrementBit.setAlignmentY(0.0F);
1027.         DecrementBit.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
1028.         DecrementBit.setMaximumSize(new java.awt.Dimension(100, 100));
1029.         DecrementBit.setMinimumSize(new java.awt.Dimension(100, 100));
1030.         DecrementBit.setPreferredSize(new java.awt.Dimension(100, 100));
1031.         DecrementBit.addActionListener(new java.awt.event.ActionListener() {
1032.             public void actionPerformed(java.awt.event.ActionEvent evt) {
1033.                 DecrementBitActionPerformed(evt);
1034.             }
1035.         });
1036.         getContentPane().add(DecrementBit);
1037.         DecrementBit.setBounds(230, 420, 60, 50);
1038.
1039.         YourBalance.setBackground(new java.awt.Color(0, 0, 0));
1040.         YourBalance.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
1041.         YourBalance.setForeground(new java.awt.Color(153, 255, 153));
1042.         YourBalance.setText("$ Your Balance");
1043.         YourBalance.setOpaque(true);
1044.         getContentPane().add(YourBalance);
1045.         YourBalance.setBounds(310, 490, 330, 40);
1046.
1047.         Kaalein.setIcon(new javax.swing.ImageIcon("back.jpg")); // NOI18N
1048.         getContentPane().add(Kaalein);
1049.         Kaalein.setBounds(0, 0, 910, 580);
1050.
1051.         YourBalance.setText("$ " + YourBalanceValue);
1052.         PoolBalance.setText("$ " + PoolBalanceValue);
1053.         oppbal.setText("$ " + OppositeBalanceValue);
1054.         TurnIndicator.setText("Host's Turn!");
1055.         labcurrval.setText("$ " + CurrentBitValue);
1056.
1057.         btnbet.setVisible(false);
1058.         btnpack.setVisible(false);
1059.         btnshow.setVisible(false);
1060.         btnsee.setVisible(false);
1061.         IncrementBit.setVisible(false);
1062.         DecrementBit.setVisible(false);
1063.
1064.         pack();
1065.         setVisible(true);
1066.
1067.         Thread th=new Thread(new Runnable() {
1068.             @Override
1069.             public void run() {
1070.                 // TODO Auto-generated method stub
1071.
1072.                 while(true)
1073.                 {

```

```

1074.         try {
1075.             String x=din.readUTF();
1076.
1077.             if(x.equals("ilose"))
1078.             {
1079.                 //Show Dialog and Tell Him to Pack THE
Game
1080.                 JFrame f=new JFrame();
1081.                 JOptionPane.showMessageDialog(f,"You
Win!\nHost Ran Out Of Money");
1082.                 setVisible(false);
1083.                 return;
1084.             }
1085.             else if(x.equals("ipack"))
1086.             {
1087.                 JFrame f=new JFrame();
1088.                 JOptionPane.showMessageDialog(f,"You
Win!\nHost PACKED!");
1089.                 setVisible(false);
1090.                 return;
1091.             }
1092.             else if(x.equals("passing"))
1093.             {
1094.                 ALLCARDS[0]=din.readUTF();
1095.                 ALLCARDS[1]=din.readUTF();
1096.                 ALLCARDS[2]=din.readUTF();
1097.                 ALLCARDS[3]=din.readUTF();
1098.                 ALLCARDS[4]=din.readUTF();
1099.                 ALLCARDS[5]=din.readUTF();
1100.
1101.                 MyCards[0]=ALLCARDS[3];
1102.                 MyCards[1]=ALLCARDS[4];
1103.                 MyCards[2]=ALLCARDS[5];
1104.             }
1105.             else if(x.equals("showingwinner"))
1106.             {
1107.                 String Win[]=new String[3];
1108.                 Win[0]=din.readUTF();
1109.                 Win[1]=din.readUTF();
1110.                 Win[2]=din.readUTF();
1111.                 if(Win[0].equals("1"))
1112.                 {
1113.                     JFrame f=new JFrame();
1114.                     JOptionPane.showMessageDialog(f,"Pla
yer \nYou LOSE!\nHost has:"+Win[1]+" \n"+Win[2]);
1115.                     setVisible(false);
1116.                     return;
1117.                 }
1118.                 else
1119.                 {
1120.                     JFrame f=new JFrame();
1121.                     JOptionPane.showMessageDialog(f,"Pla
yer \nYou WIN!\n"+Win[1]+" \n"+Win[2]);
1122.                     setVisible(false);
1123.                     return;
1124.                 }
1125.             }
1126.             OppositeBalanceValue=din.readInt();
1127.             PoolBalanceValue=din.readInt();
1128.             CurrentBitValue=din.readInt();

```

[illegible]

```

1187.                dout.writeInt(PoolBalanceValue);
1188.                dout.writeInt(CurrentBitValue);
1189.
1190.            } catch (Exception ex3) {
1191.                // TODO: handle exception
1192.            }
1193.            TurnIndicator.setText("Host's Turn");
1194.            YourBalance.setText("$ " + YourBalanceValue);
1195.            PoolBalance.setText("$ " + PoolBalanceValue);
1196.            oppbal.setText("$ " + OppositeBalanceValue);
1197.            labcurrval.setText("$ " + CurrentBitValue);
1198.            try {
1199.                btnbet.setVisible(false);
1200.                btnpack.setVisible(false);
1201.                btnshow.setVisible(false);
1202.                btnsee.setVisible(false);
1203.                IncrementBit.setVisible(false);
1204.                DecrementBit.setVisible(false);
1205.                th.notify();
1206.
1207.            } catch (Exception ex7) {
1208.                // TODO: handle exception
1209.            }
1210.        }
1211.    });
1212.
1213.
1214.    IncrementBit.addActionListener(new
1215.        ActionListener(){
1216.            public void actionPerformed(ActionEvent e){
1217.                if(Increment==1)return;
1218.                else{
1219.                    if(CurrentBitValue*2>YourBalanceValue)
1220.                    {
1221.                        return;
1222.                    }
1223.                    CurrentBitValue*=2;
1224.                    labcurrval.setText("$ " + CurrentBitValue);
1225.                    Increment++;
1226.                }
1227.            });
1228.
1229.    DecrementBit.addActionListener(new
1230.        ActionListener(){
1231.            public void actionPerformed(ActionEvent e){
1232.                if(Increment==0)
1233.                return;
1234.                else{
1235.                    CurrentBitValue/=2;
1236.                    labcurrval.setText("$
1237.                    "+CurrentBitValue);
1238.                    Increment--;
1239.                }
1240.            });
1241.
1242.    btnsee.addActionListener(new ActionListener(){
1243.        public void actionPerformed(ActionEvent e){
1244.            //Implement The Logic

```

```

1244.                                     //(ImageIcon)ImageIconGetter.GetTheImageIcon
      (MyCards[0])
1245.                                     isVisible=true;
1246.
1247.                                     if(isVisibleCards==false)
1248.                                     {
1249.                                         C1.setIcon(new ImageIcon(new
      ImageIconGetter().GetTheImageIcon(MyCards[0])));
1250.                                         C2.setIcon(new ImageIcon(new
      ImageIconGetter().GetTheImageIcon(MyCards[1])));
1251.                                         C3.setIcon(new ImageIcon(new
      ImageIconGetter().GetTheImageIcon(MyCards[2])));
1252.                                         isVisibleCards=true;
1253.                                     }
1254.                                     else{
1255.                                         C1.setIcon(new
      ImageIcon("card_back.png"));
1256.                                         C2.setIcon(new
      ImageIcon("card_back.png"));
1257.                                         C3.setIcon(new
      ImageIcon("card_back.png"));
1258.                                         isVisibleCards=false;
1259.                                     }
1260.                                 }});
1261.
1262. btnpack.addActionListener(new ActionListener(){
1263.     public void actionPerformed(ActionEvent e){
1264.         try {
1265.             dout.writeUTF("ipack");
1266.         } catch (Exception ex6) {
1267.             // TODO: handle exception
1268.         }
1269.         setVisible(false);
1270.         return;
1271.     }
1272. });
1273.
1274. btnclick.addActionListener(new ActionListener(){
1275.     public void actionPerformed(ActionEvent e){
1276.         try {
1277.             dout.writeUTF("showingwinner");
1278.         } catch (Exception ev) {
1279.             // TODO: handle exception
1280.         }
1281.         String Result[]=new String[3];
1282.         card_game x=new card_game();
1283.         Result=x.getResult(ALLCARDS);
1284.
1285.         try {
1286.             dout.writeUTF(Result[0]);
1287.             dout.writeUTF(Result[1]);
1288.             dout.writeUTF(Result[2]);
1289.         } catch (Exception x7) {
1290.             // TODO: handle exception
1291.         }
1292.
1293.         if(Result[0].equals("1"))
1294.         {
1295.             JFrame f=new JFrame();

```



```

1297.                                     JOptionPane.showMessageDialog(f, "Pla
    yer \nYou LOSE!\nHost has:"+Result[1]+"\\n"+Result[2]);
1298.                                     setVisible(false);
1299.                                     return;
1300.                                     }
1301.                                     else
1302.                                     {
1303.                                         JFrame f=new JFrame();
1304.                                         JOptionPane.showMessageDialog(f, "Pla
    yer \n You WIN!\n"+Result[1]+"\\n"+Result[2]);
1305.                                         setVisible(false);
1306.                                         return;
1307.                                     }
1308.                                     }
1309.                                     });
1310.                                     }
1311.                                     });
1312.                                     Send.start();
1313.
1314.                                     }
1315.
1316.                                     private void btnshowActionPerformed(java.awt.event.ActionEvent evt)
1317.                                     {
1318.                                         // TODO add your handling code here:
1319.                                     }
1320.                                     private void btnpackActionPerformed(java.awt.event.ActionEvent evt)
1321.                                     {
1322.                                         // TODO add your handling code here:
1323.                                     }
1324.                                     private void IncrementBitActionPerformed(java.awt.event.ActionEvent evt)
1325.                                     {
1326.                                         // TODO add your handling code here:
1327.                                     }
1328.                                     private void DecrementBitActionPerformed(java.awt.event.ActionEvent evt)
1329.                                     {
1330.                                         // TODO add your handling code here:
1331.                                     }
1332.                                     }
1333.
1334.
1335.                                     class card_game {
1336.
1337.
1338.
1339.                                         InputStreamReader x = new InputStreamReader(System.in);
1340.
1341.                                         int resuser_1,resuser_2,cc = 0;
1342.                                         String hands[] = { "High Card", "Pair","Colour", "Sequence", "Coloured
    Sequence", "Trail" };
1343.                                         //static String
    card[]={ "2H","3H","4H","5H","6H","7H","8H","9H","0H","JH","QH","KH","AH","2S","3S",
    "4S","5S","6S","7S","8S","9S","0S","JS","QS","KS","AS","2C","3C","4C","5C","6C","7C",
    "8C","9C","0C","JC","QC","KC","AC","2D","3D","4D","5D","6D","7D","8D","9D","0D",""
    JD","QD","KD","AD"};

```

```

1344.         //static String
        card[]={"JH","QH","KH","AH","JS","QS","KS","AS","JC","QC","KC","AC","JD","QD","KD",
        "AD"};
1345.         String win[] = { "", "", ""};
1346.
1347.
1348.         // public static void main(String[] args) {
1349.         //     cc = 0;
1350.         //     shuffle(card);
1351.         //     resuser_1 = checkcard(card[0], card[2], card[4]);
1352.         //     resuser_2 = checkcard(card[1], card[3], card[5]);
1353.         //     result(resuser_1, resuser_2);
1354.
1355.         //     for (int i = 0; i < 3; i++) {
1356.         //         System.out.print(win[i] + " ");
1357.         //     }
1358.
1359.         // }
1360.
1361.         String card[]=new String[6];
1362.
1363.         String[] GetResult(String ALLCARDS[])
1364.         {
1365.             card[0]=ALLCARDS[0];
1366.             card[2]=ALLCARDS[1];
1367.             card[4]=ALLCARDS[2];
1368.             card[1]=ALLCARDS[3];
1369.             card[3]=ALLCARDS[4];
1370.             card[5]=ALLCARDS[5];
1371.             resuser_1 = checkcard(card[0], card[2], card[4]);
1372.             resuser_2 = checkcard(card[1], card[3], card[5]);
1373.             result(resuser_1, resuser_2);
1374.             return win;
1375.         }
1376.
1377.         // public static void waitfr(int time) {
1378.         //     try {
1379.         //         Thread.sleep(time);
1380.         //     } catch (Exception e) {
1381.         //     }
1382.         // }
1383.
1384.         public int checkcard(String crd1, String crd2, String crd3) {
1385.
1386.             int crd1num = 0, crd2num = 0, crd3num = 0;
1387.             char crd1baghdo = crd1.charAt(1), crd2baghdo = crd2.charAt(1),
            crd3baghdo = crd3.charAt(1);
1388.             int res = 0;
1389.             crd1num = assignvalue(crd1.charAt(0));
1390.             crd2num = assignvalue(crd2.charAt(0));
1391.             crd3num = assignvalue(crd3.charAt(0));
1392.
1393.             if (crd1.charAt(0) == '0') {
1394.                 crd1num = 10;
1395.             }
1396.             if (crd2.charAt(0) == '0') {
1397.                 crd2num = 10;
1398.             }
1399.             if (crd3.charAt(0) == '0') {
1400.                 crd3num = 10;

```

```

1401.     }
1402.
1403.     int arr[] = { crd1num, crd2num, crd3num };
1404.     Arrays.sort(arr);
1405.     if (crd1num == crd2num && crd1num == crd3num)
1406.         res = 5;
1407.     else if (crd1baghdo == crd2baghdo && crd1baghdo == crd3baghdo) {
1408.         if (arr[0] + 1 == arr[1] && arr[1] + 1 == arr[2])
1409.             res = 4;
1410.         else
1411.             res = 2;
1412.     } else if (arr[0] + 1 == arr[1] && arr[1] + 1 == arr[2])
1413.         res = 3;
1414.     else if (crd1num == crd2num || crd2num == crd3num || crd3num ==
        crd1num)
1415.         res = 1;
1416.     else
1417.         res = 0;
1418.     return res;
1419. }
1420.
1421. public int assignvalue(int val) {
1422.     int res = 0;
1423.     if (val == 65)
1424.         res = 14;
1425.     else if (val == 74)
1426.         res = 11;
1427.     else if (val == 75)
1428.         res = 13;
1429.     else if (val == 81)
1430.         res = 12;
1431.     else
1432.         res = val - 48;
1433.     return res;
1434. }
1435.
1436. public void result(int user_1, int user_2) {
1437.
1438.     if (user_1 > user_2) {
1439.         win[0] = "1";
1440.         win[1] = hands[user_1];
1441.     } else if (user_2 > user_1) {
1442.         win[0] = "2";
1443.         win[1] = hands[user_2];
1444.     } else {
1445.         if ((user_1 == 0 && user_2 == 0) || (user_1 == 2 && user_2 == 2) ||
            (user_1 == 3 && user_2 == 3)
1446.             || (user_1 == 4 && user_2 == 4) || (user_1 == 5 && user_2 == 5))
1447.         {
1448.             int[] num1 = findhigh(card[0], card[2], card[4]);
1449.             int[] num2 = findhigh(card[1], card[3], card[5]);
1450.
1451.             if (num1[0] > num2[0]) {
1452.                 if (user_1 == 3 && user_2 == 3) {
1453.                     win[0] = "1";
1454.                     win[1] = hands[user_1];
1455.                     win[2] = "+ high card";
1456.                 } else if (user_1 == 2 && user_2 == 2) {
1457.                     win[0] = "1";

```

```

1458.         win[1] = hands[user_1];
1459.         win[2] = "+ high card";
1460.     } else
1461.         win[0] = "1";
1462.     win[1] = hands[user_1];
1463. } else if (num2[0] > num1[0]) {
1464.     if (user_1 == 3 && user_2 == 3) {
1465.         win[0] = "2";
1466.         win[1] = hands[user_2];
1467.         win[2] = "+ high card";
1468.     } else if (user_1 == 2 && user_2 == 2) {
1469.         win[0] = "2";
1470.         win[1] = hands[user_2];
1471.         win[2] = "+ high card";
1472.     } else{
1473.         win[0] = "2";
1474.         win[1] = hands[user_2];
1475.     }
1476. } else if (num1[0] == num2[0]) {
1477.     if (num1[1] > num2[1]) {
1478.         if (user_1 == 3 && user_2 == 3) {
1479.             win[0] = "1";
1480.             win[1] = hands[user_1];
1481.             win[2] = "+ high card";
1482.         } else if (user_1 == 2 && user_2 == 2) {
1483.             win[0] = "1";
1484.             win[1] = hands[user_1];
1485.             win[2] = "+ high card";
1486.         } else
1487.             win[0] = "1";
1488.         win[1] = hands[user_1];
1489.     } else {
1490.
1491.         if (user_1 == 3 && user_2 == 3) {
1492.             win[0] = "2";
1493.             win[1] = hands[user_2];
1494.             win[2] = "+ high card";
1495.         } else if (user_1 == 2 && user_2 == 2) {
1496.             win[0] = "2";
1497.             win[1] = hands[user_2];
1498.             win[2] = "+ high card";
1499.         } else
1500.             win[0] = "2";
1501.         win[1] = hands[user_2];
1502.     }
1503.
1504.     } else {
1505.         win[0]="It's a tie...";
1506.     }
1507. } else if (user_1 == 1 && user_2 == 1) {
1508.     int[] user_1crd1 = pairdis(card[0]);
1509.     int[] user_1crd2 = pairdis(card[2]);
1510.     int[] user_1crd3 = pairdis(card[4]);
1511.     int[] user_2crd1 = pairdis(card[1]);
1512.     int[] user_2crd2 = pairdis(card[3]);
1513.     int[] user_2crd3 = pairdis(card[5]);
1514.
1515.     // int user_1crd1=assignvalue(card[0].charAt(0));
1516.     // int user_1crd2=assignvalue(card[2].charAt(0));
1517.     // int user_1crd3=assignvalue(card[4].charAt(0));

```

```

1518.          // int user_2crd1=assignvalue(card[1].charAt(0));
1519.          // int user_2crd2=assignvalue(card[3].charAt(0));
1520.          // int user_2crd3=assignvalue(card[5].charAt(0));
1521.
1522.          int user_1c = 0, user_2c = 0;
1523.          if (user_1crd1[0] == user_1crd2[0] || user_1crd1[0] ==
            user_1crd3[0])
1524.              user_1c = user_1crd1[0];
1525.          else if (user_1crd2[0] == user_1crd3[0])
1526.              user_1c = user_1crd3[0];
1527.          if (user_2crd1[0] == user_2crd2[0] || user_2crd1[0] ==
            user_2crd3[0])
1528.              user_2c = user_2crd1[0];
1529.          else if (user_2crd2[0] == user_2crd3[0])
1530.              user_2c = user_2crd3[0];
1531.
1532.          if (user_1c > user_2c) {
1533.
1534.              win[0] = "1";
1535.              win[1] = hands[user_1];
1536.              win[2] = "+ high card";
1537.          } else if (user_2c > user_1c) {
1538.              win[0] = "2";
1539.              win[1] = hands[user_2];
1540.              win[2] = "+ high card";
1541.          } else {
1542.              int[] player1 = new int[2];
1543.              int[] player2 = new int[2];
1544.              if (user_1crd1[0] == user_1crd2[0]) {
1545.                  player1[0] = user_1crd3[0];
1546.                  player1[1] = user_1crd3[1];
1547.              } else if (user_1crd1[0] == user_1crd3[0]) {
1548.                  player1[0] = user_1crd2[0];
1549.                  player1[1] = user_1crd2[1];
1550.              } else if (user_1crd2[0] == user_1crd3[0]) {
1551.                  player1[0] = user_1crd1[0];
1552.                  player1[1] = user_1crd1[1];
1553.              }
1554.
1555.              if (user_2crd1[0] == user_2crd2[0]) {
1556.                  player2[0] = user_2crd3[0];
1557.                  player2[1] = user_2crd3[1];
1558.              } else if (user_2crd1[0] == user_2crd3[0]) {
1559.                  player2[0] = user_2crd2[0];
1560.                  player2[1] = user_2crd2[1];
1561.              } else if (user_2crd2[0] == user_2crd3[0]) {
1562.                  player2[0] = user_2crd1[0];
1563.                  player2[1] = user_2crd1[1];
1564.              }
1565.
1566.              if (player1[0] > player2[0]) {
1567.                  win[0] = "1";
1568.                  win[1] = hands[user_1];
1569.                  win[2] = "+ high card";
1570.              } else if (player1[0] < player2[0]) {
1571.                  win[0] = "2";
1572.                  win[1] = hands[user_2];
1573.                  win[2] = "+ high card";
1574.              } else {
1575.                  if (player1[0] > player2[0]) {

```

```

1576.         win[0] = "1";
1577.         win[1] = hands[user_1];
1578.         win[2] = "+ high card same but high card type";
1579.     } else {
1580.         win[0] = "2";
1581.         win[1] = hands[user_2];
1582.         win[2] = "+ high card same but high card type";
1583.     }
1584.
1585.     }
1586.
1587.     }
1588. } else {
1589.     System.out.println("\nIt's a tie...");
1590. }
1591.
1592. }
1593. }
1594.
1595. public int[] findhigh(String crd1, String crd2, String crd3) {
1596.     int[] val = new int[2];
1597.
1598.     int crd1_type = assignvalue(crd1.charAt(1));
1599.     int crd2_type = assignvalue(crd2.charAt(1));
1600.     int crd3_type = assignvalue(crd3.charAt(1));
1601.
1602.     int crd1num = assignvalue(crd1.charAt(0));
1603.     int crd2num = assignvalue(crd2.charAt(0));
1604.     int crd3num = assignvalue(crd3.charAt(0));
1605.
1606.     if (crd1.charAt(0) == '0') {
1607.         crd1num = 10;
1608.     }
1609.     if (crd2.charAt(0) == '0') {
1610.         crd2num = 10;
1611.     }
1612.
1613.     if (crd3.charAt(0) == '0') {
1614.         crd3num = 10;
1615.     }
1616.
1617.     if (crd1_type == 20) {
1618.         crd1_type -= 2;
1619.     }
1620.     if (crd2_type == 20) {
1621.         crd2_type -= 2;
1622.     }
1623.     if (crd3_type == 20) {
1624.         crd3_type -= 2;
1625.     }
1626.
1627.     if (crd1num > crd2num && crd1num > crd3num) {
1628.         val[0] = crd1num;
1629.         val[1] = crd1_type;
1630.     } else if (crd2num > crd3num) {
1631.         val[0] = crd2num;
1632.         val[1] = crd2_type;
1633.     } else {
1634.         val[0] = crd3num;
1635.         val[1] = crd3_type;

```


```
1636.         }
1637.         return val;
1638.     }
1639.
1640.     public int[] pairdis(String crd) {
1641.         int[] val = new int[2];
1642.         if (crd.charAt(0) == '0') {
1643.             val[0] = 10;
1644.             val[1] = assignvalue(crd.charAt(1));
1645.             if (val[1] == 20) {
1646.                 val[1] -= 2;
1647.             }
1648.         } else {
1649.             val[0] = assignvalue(crd.charAt(0));
1650.             val[1] = assignvalue(crd.charAt(1));
1651.             if (val[1] == 20) {
1652.                 val[1] -= 2;
1653.             }
1654.         }
1655.         return val;
1656.     }
1657.
1658. }
```

CHAPTER 4

SCREENSHOTS

4.1 Admin Screenshot

Host Hosting the game:



A screenshot of a web application window titled "Card Game". The background is blue. It contains the following elements:

- A text input field labeled "Enter Your Name:" with the text "Kunjan" entered.
- A text input field labeled "Enter The IP Address Of HOST:" which is currently empty.
- A line of text: "Insert Host IP Address only if You are JOINING THE GAME!"
- Two buttons: "Host" and "Join".
- A message at the bottom: "Please HOST The Game First Then Join The Game".

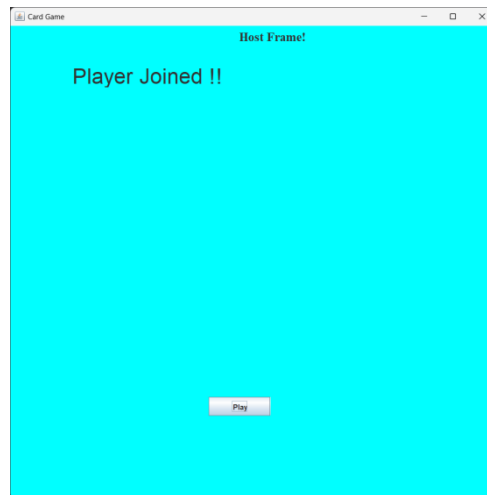
Player-2 Joining:



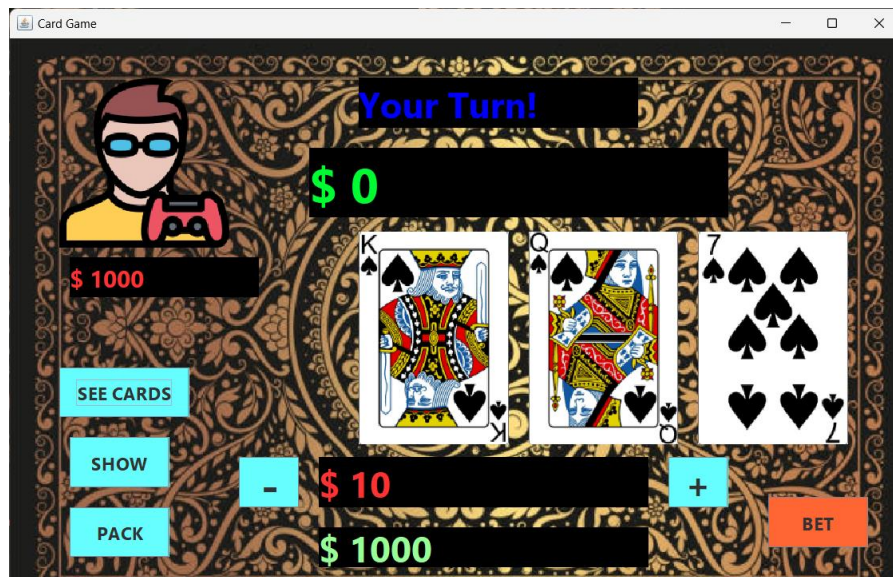
A screenshot of the same "Card Game" web application window. The background is blue. It contains the following elements:

- A text input field labeled "Enter Your Name:" with the text "player-2" entered.
- A text input field labeled "Enter The IP Address Of HOST:" with the text "127.0.0.1" entered.
- A line of text: "Insert Host IP Address only if You are JOINING THE GAME!"
- Two buttons: "Host" and "Join".
- A message at the bottom: "Please HOST The Game First Then Join The Game".

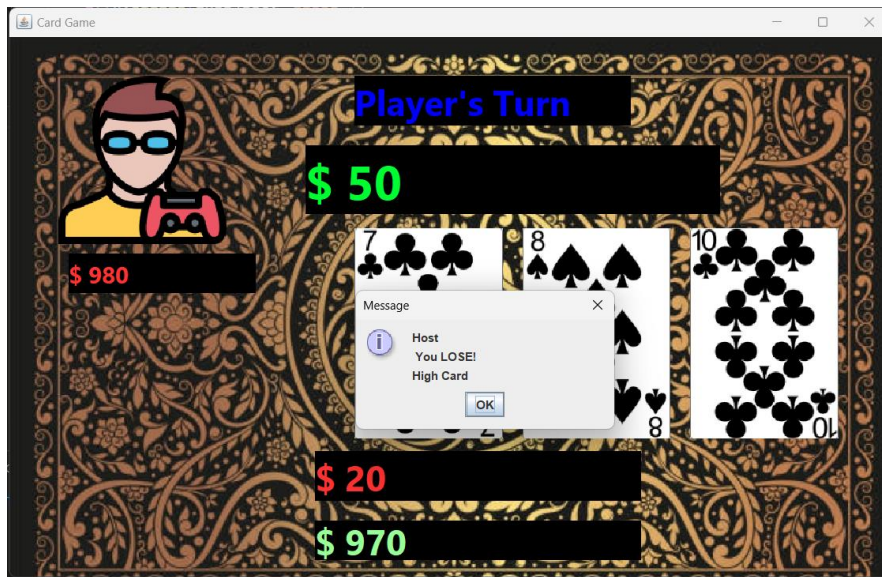
Player-2 Joined:



Host's TURN :



Host's SHOW :



CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

5.1 Conclusion

- We are going to add more than two players in future
- We will connect the database with this game in FUTURE.

5.2 Future Enhancements

- We can use GUI
- We can you database
- We can use more advance java
- We can add more future

REFERENCES

In IEEE Format – Add minimum 5 references

- [1] <https://google.com>
- [2] <https://www.geeksforgeeks.org>
- [3] <https://javatutorials.com/employee-management-system-project-in-java>
- [4] <https://github.com/topics/employee-management-system?l=c%2B%2B>
- [5] https://www.youtube.com/watch?v=E1ehMzKLKXk&ab_channel=NGTutorials