MULTI-PLAYER BINGO GAME WITH CHAT APPLICATION USING SOCKET PROGRAMMING IN JAVA

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE302: COMPUTER NETWORKS

Submitted by

VADLA KALYAN
(Reg. No.: 124003128, B.Tech Computer Science & Engineering)

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613401

SCHOOL OF COMPUTING

THANJAVUR – 613401

<u>Bonafide Certificate</u>

This is to certify that the report titled **“Multi-Player Bingo Game With Chat Application Using Socket Programming In Java”** submitted as a requirement for the course, **CSE302: COMPUTER NETWORKS** for B.Tech. is a bonafide record of the work done by **Mr. Vadla Kalyan (Reg. No.: 124003128, Computer Science and Engineering)** during the academic year 2022-23, in the School of Computing

Project Based Work *Viva voce* held on <u>14-12-2022</u>

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

First of all, I would like thank God Almighty for his endless blessings.

I would like to express my sincere gratitude to **Dr S. Vaidyasubramaniam, Vice-Chancellor** for his encouragement during the span of my academic life at SASTRA Deemed University.

I would forever remain grateful and I would like to thank **Dr A.Umamakeswri, Dean, School of Computing and R. Chandramouli, Registrar** for their overwhelming support provided during my course span in SASTRA Deemed University.

I am extremely grateful to **Dr. Shankar Sriram, Associate Dean, School of Computing** for his constant support, motivation , guidance and academic help while doing the project.

I also thank all the Teaching and Non-teaching faculty, and all other people who have directly or indirectly help me through their support, encouragement and all other assistance extended for completion of my project.

LIST OF FIGURES

ABREVATIONS

TCP	Transmission Control Protocol
IP	Internet Protocol
UDP	User Datagram Protocol
GUI	Graphical User Interface
IPConfig	Internet Protocol Configuration

ABSTRACT

Socket programming connects 2 nodes **client and server** on a network to communicate with each other. One socket listens to the IP address and the port number, while the other extends to form a connection. Most programming languages like java have bindings to the socket as API. They are connection between low level Application to Transport layer API's.

Socket is first created, associated with a type of **transport layer** (TCP/UDP), associated with an IP address and port, and then used to either listen to connections, or send or receive data, files.

TCP/IP protocol is the connection-oriented communication protocol used to connect devices on internet. It specifies how data is communicated over internet by providing end-to-end communication which identifies how it should be broken as packets, segments, addressed, routed and received at destination. It uses 3-way handshaking.

Online multiplayer games have become a fab among younger generations, being a good entertainment and a way to socialize with people all over the world using internet - computer network.

This project is to create a virtual game which we will be played online. The game is **BINGO**. This is a multiplayer game where minimum of two players are required. There will be a 5 x 5 grid provided to every player and they need to fill the blocks with numbers from 1 to 25 randomly. Anyone of the player will get a chance to start the game, player will select a number from the grid, The number will be popped out from the grids of all the players. As it is a **turn-based** game, Each player will get a chance to pop out a number. The player, who will make five lines each with five empty blocks either horizontally or vertically or diagonally. Rooms are created based on the code given by the players, each room has two players having same code and they are connected to play.

This game requires Socket programming to connect the players using their IP address. It can use TCP protocol in transport layer, it uses GUI for getting interaction between the system and player. And the players can also chat mutually in between the game using chat box that uses Socket Programming.

Multi-Threading concept is used such that server can interact with more clients and can create many rooms

Key words: Socket programming, Client and server, Transport layer, TCP, IP, Multi-Threading, strategical turn-based game.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In this **project** we are illustrating a multiplayer game using client-server architecture and by using socket programming, by using TCP/IP protocol to establish a connection between client and server.

1.1 COMPUTER NETWORKING

Computer network is defined as set of interconnected computers that exchange resources amongst themselves via a network such as the **Internet.** Multiplayer games, chat facility have rose to fame in the past decade. Computer networking provides multiple computers to communicate effortlessly with each other without having to be close by (connected via cables/wires).

1.2 OSI MODEL (Open System Interconnection)

The OSI model is a conceptual model that describes how information from a software application in one computer moves through physical medium to another computer.
This model is widely used as it is less complex and provides a reliable data transfer service.

The OSI model splits into seven layers: Physical, Data Link, Network, Transport, Session, Presentation and Application layer.
Each lower layer adds its services to the higher layer to provide services to control communications and to run the application. Each layer is independent and has a particular task.

The seven layer OSI model is converted into four layers TCP/IP protocol as shown in *Fig1.1*.

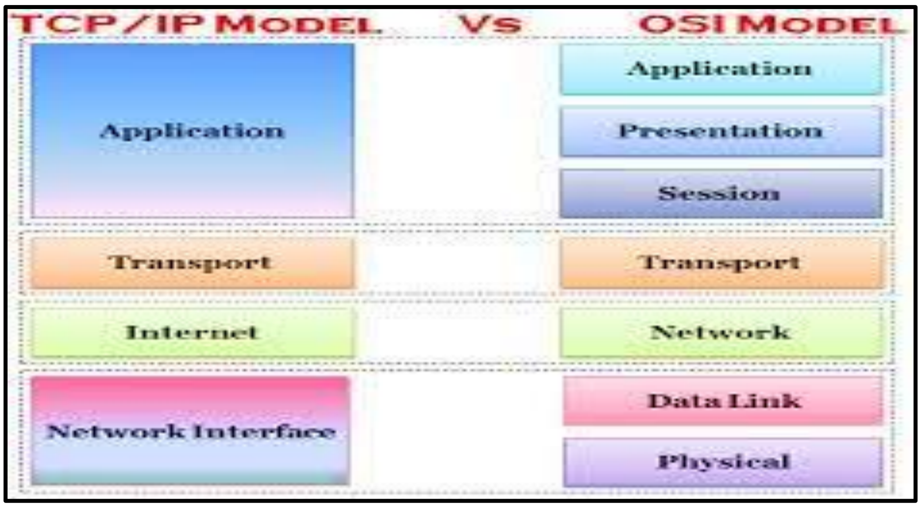


Fig 1.1.OSI model

1.3 TRANSMISSION CONTROL PROTOCOL (TCP)

It is a transport layer protocol that facilitates the transmission of packets from source to destination.

- The connection is established using three-way handshake.
- Connection oriented
- It is a reliable protocol
- It ensures that the data reaches intended destination in same order it was sent.
- It provides end to end communication
- It provides flow control and quality of service
- It provides error checking and recovery mechanism

1.4 TCP CLIENT SERVER ARCHITECTURE

A Client-Server Architecture consists of two types of components: clients and servers. A server component perpetually listens for requests from client components. When a request is received, the server processes the request, and then sends a response back to the client. The client sends a request for data to the server through the internet/local IP address. Multiple clients are able to communicate using the server.

In this project we are hosting the server locally on computer and both the clients/players are able to play the game, or players from any region can join and connect using IP address.

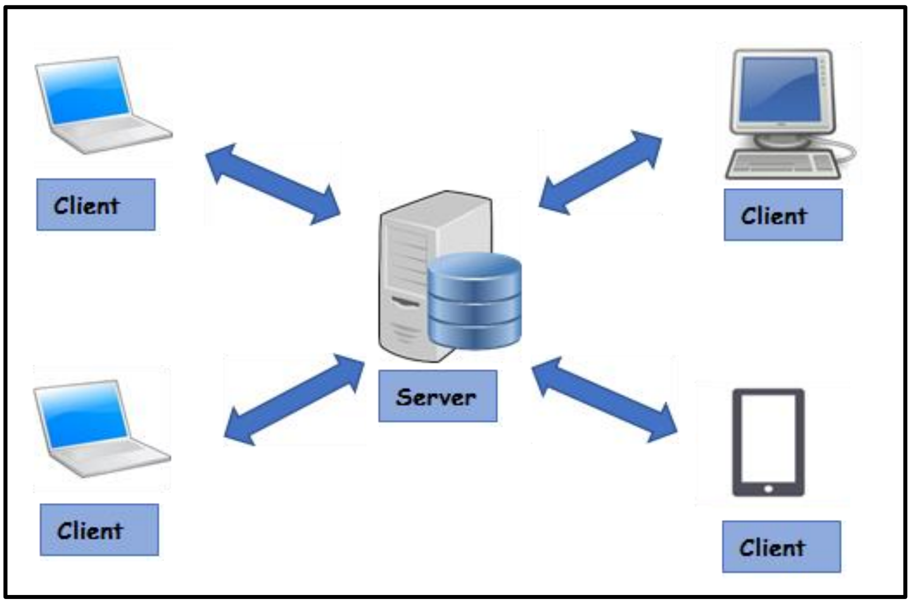


Fig 1.2 Client server architecture

1.5 COMMUNICATION PROCESS IN GAME

Server starts running and waiting for the clients/players to connect. Players will enter the game with a code, players having same code are enabled to be in same room i.e., TCP connection is established between those players having same entry code.

The players can chat – send messages(strings) through socket and, play the game forwarding numbers that are popped between the players. Winner will send a message that he won to the other player – this is the last message they share and the socket is closed.

These different types of messages are differentiated by prefixing some special symbols (like $, _) or a special string (like $Win) while sending, so that the receiver can understand the type of the message received and can perform the operations accordingly.

1.6 SOCKET PROGRAMMING IN BINGO

Socket Programming connects the server and players/clients using **multi-threading**.

Procedure on server side:

- The first step is to import java net package
  import java.net.;*
- *serversocket = new ServerSocket(Port_Number)* – server runs on the given port number
- *socket = serversocket.accept()* – this will allow server to wait for clients/players and accept them and server will store the socket and its details that are assigned to the instances of a PlayerThread class object and these objects are stored in an ArrayList
- for each PlayerThread object that is stored in ArrayList will have a BufferedReader object and PrintWriter object

  new BufferedReader(newInputStreamReader(socket.getInputStream));

  new PrintWriter(socket.getInputStream);

  BufferedReader object is used to read the incoming text from the socket and PrintWriter object is used to send the data in the form of string through the socket
- The PlayerThread starts running, if the player entered the game with the option join then server will search for the room with the code of the player, if it exists then server will forward a message to the player who joined the game also the player who created the game that will allow the players to start the game
- Hereafter the server will forward the data sent by one player to other player in the room by searching the player code in the ArrayList
- If any failure occurs like the player exits then the corresponding socket is closed
  socket.close();
- After closing the socket, this socket is removed from the ArrayList
  ArrayList.remove(socket);

Procedure on player/client side:

- Same as the server, it requires the net package, BufferedReader to receive the data and PrintWriter to send the data
- It contains the GUI part which helps for the player to communicate with the computer
- Player after executing, two options are displayed on the window, player will be asked to select one of those options. The options are CREATE and JOIN

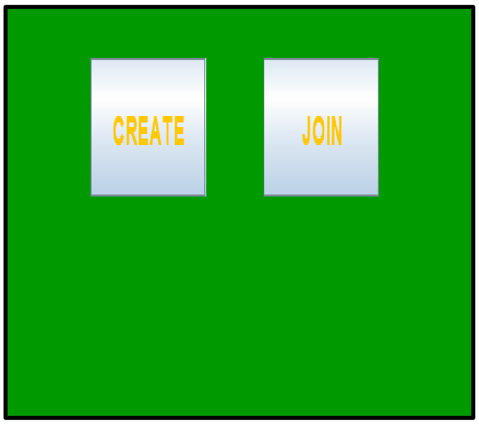
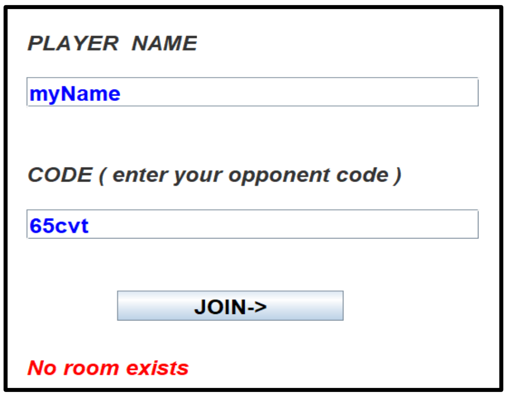


Fig 1.3 Create and join options

- Player will be asked to enter his name and a code in both cases, but the player who wants to join the game instead of creating he/she must give the existing code that is already created by another player. else an error message "No room exists" is displayed


PLAYER NAME

myName

CODE (enter your opponent code)

65cvt

JOIN->

No room exists

Fig 1.4 Error with code while joining

- Player who created the game will be waiting until other player joins.
- Now a socket is created that connects server and player

socket = new Socket("IP address of server",portNumOfServer);

- The details of the player are sent to the server and server will store the details in its ArrayList
- Now the two players with the same code are connected through the server are said to be in a room and starts the game

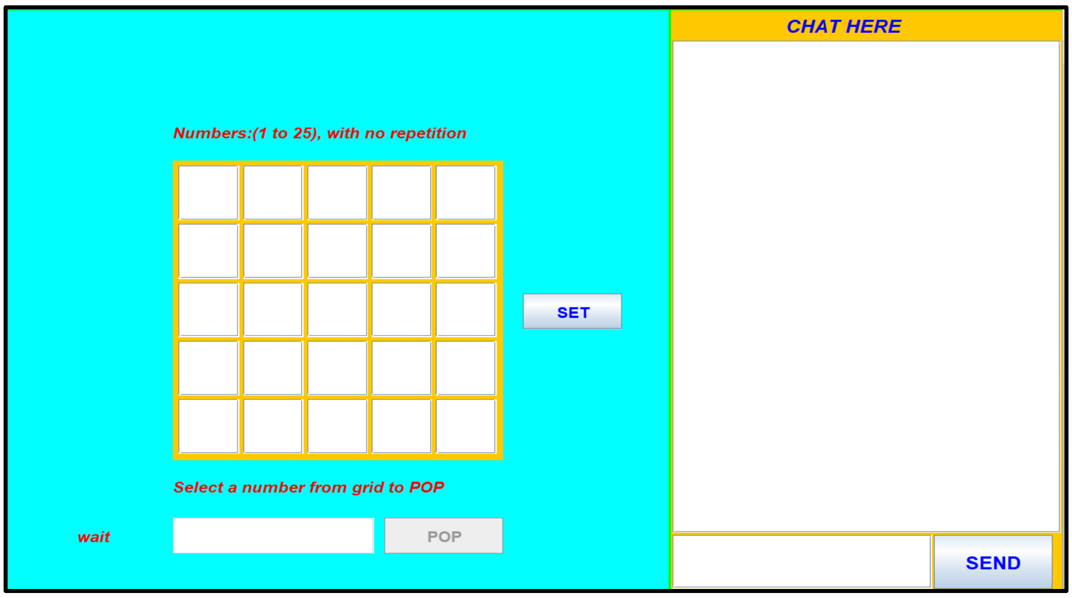

CHAT HERE
Numbers:(1 to 25), with no repetition
SET
Select a number from grid to POP
wait
POP
SEND

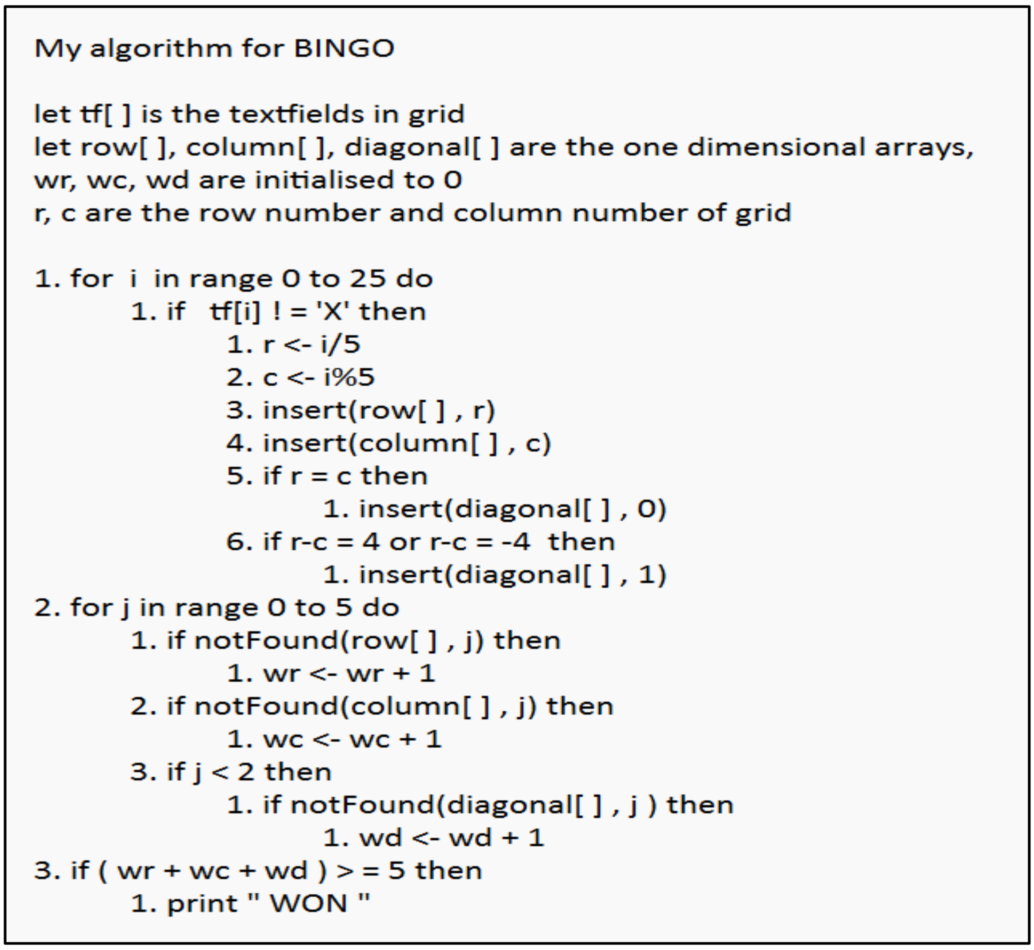
Fig 1.5 GUI of the game

- According to BINGO, players will enter the integers from 1 to 25 randomly with no repetition in the given 5x5 grid as shown in *fig 1.5* and clicks on the set button. If any given entry does not follow the rule, then that entry is not valid and player should enter the missing integer
- After both players completes the task of giving input in grid, the player who created the game will get first chance to pop a number from the grid and then the other player will get the chance
- The selected number will be popped from the grid and then it will check for BINGO (lets discuss the BINGO algorithm in *section 1.7*), if not then it will send the selected number to opponent player through server. If it is BINGO then it will send a message “$WON”, the opponent after receiving this message he will be stopped else if he receives a number then he will first check for BINGO, if yes he will do the same process else he will be allowed to pop a number form grid
- Simultaneously the players can even chat with each other in given chat area as shown in the *fig 1.5*, these messages are prefixed with underscore symbol to differentiate it from the numbers that are popped out
- After the end of the game the both sockets are closed, disconnected and are removed from the server storage and moves back to the create, join options
- Now the player can again able to repeat the whole process

- While these two players are playing in a room, at the same time many other rooms can be created each connects other two players

1.7 ALGORITHM FOR BINGO

- When a player pops a number from grid the textfield is marked with ‘**X**’
- The aim is to count the number of the vertical columns, horizontal rows and diagonal lines such that the complete line of 5 blocks must be filled with ‘**X**’
- The algorithm that is designed in this project to count the number of lines in grid is given in *fig 1.6*



```
My algorithm for BINGO

let tf[ ] is the textfields in grid
let row[ ], column[ ], diagonal[ ] are the one dimensional arrays,
wr, wc, wd are initialised to 0
r, c are the row number and column number of grid

1. for  i  in range 0 to 25 do
        1. if   tf[i] ! = 'X' then
                1. r <- i/5
                2. c <- i%5
                3. insert(row[ ] , r)
                4. insert(column[ ] , c)
                5. if r = c then
                        1. insert(diagonal[ ] , 0)
                6. if r-c = 4 or r-c = -4  then
                        1. insert(diagonal[ ] , 1)
2. for j in range 0 to 5 do
        1. if notFound(row[ ] , j) then
                1. wr <- wr + 1
        2. if notFound(column[ ] , j) then
                1. wc <- wc + 1
        3. if j < 2 then
                1. if notFound(diagonal[ ] , j ) then
                        1. wd <- wd + 1
3. if ( wr + wc + wd ) > = 5 then
        1. print " WON "
```

Fig 1.6 Algorithm for BINGO

- If the count reaches to 5 then the player wins the game

MERITS

- Players can chat with each other while playing the game
- Any number of players can connect with server since multi-threading concept is used in this project
- The connected players are grouped into rooms each having two players with same code
- Communication is done between two players who are held in same room
- Data is differentiated with special symbols such that the receiver can understand whether the data received is a chat message or popped number etc.,
- There will be no data loss as TCP is used

DEMERITS

- Players must wait until both the players completed the process of giving input in the grid
- There is no time limit
- It is not an offline game, so Internet is required while playing the game
- If there is any network failure then the game does not work properly

CHAPTER 2

SOURCE CODE

2.1 SERVER SOURCE CODE

```
import java.io.*;
import java.util.*;
import java.net.*;

/* PlayerThread objects are considered as players */
class PlayerThread implements Runnable
{
        Socket s;
        public static ArrayList<PlayerThread> lst = new ArrayList<PlayerThread>();
        // lst : list in which all players details are stored
        BufferedReader br;
        PrintWriter pw;
        String username;   //---------- player user name
        String code;      //------------ room code
        String purpose; // if player created game purpose=create else purpose=join

        public PlayerThread(Socket s)
        {
                try{
                this.s = s;
                this.br = new BufferedReader(new InputStreamReader(s.getInputStream()));
                this.pw = new PrintWriter(s.getOutputStream());
                /* receiving the details from socket and storing in list(lst) */
                this.purpose = this.br.readLine();
                this.username = this.br.readLine();
                this.code = this.br.readLine();
                lst.add(this);
                }catch(Exception e){  removeSocket(s);}
        }
        /* player with purpose "join" search for room with the code */
        public void searchForC()
        {
                int flag=0;
                try{
                for(PlayerThread pt : lst){
                        if(pt.purpose.equals("create->") && pt.code.equals(code)){
                                pw.println("1");  // if room exists , value 1 will be sent to player
so that he can join
                                pw.flush();
                                pt.pw.println("1");
                                pt.pw.flush();
                                flag=1;
                        }
                }
                if(flag==0){
                        pw.println("0");   // if room doesn't exists, value 0 will be sent to player,
he can't join
                        pw.flush();
                        removeSocket(s);
                }
                }catch(Exception e){removeSocket(s);}
        }

        @Override
        public void run()
        {
                String msg;
                if(purpose.equals("join->")){
                        searchForC();
                }
                while(s.isConnected()){
                        try{
                                msg = br.readLine();   // receives data from player
                                sendMsg(msg);  // sends data to other player
                        }
                        catch(Exception e){
                                removeSocket(s);
                                break;
                        }
                }
        }

        public void sendMsg(String msg)
        {
                for(PlayerThread pt : lst){
                        if(pt.code.equals(code) && !pt.username.equals(username)){
                                try{
                                pt.pw.println(msg); // sending the data
                                pt.pw.flush();
                                }
                                catch(Exception e){removeSocket(s);}
                        }
                }
        }

        /* if any exceptional errors occurs this method is called to close the connection */
        public void removeSocket(Socket s)
        {
                try{
                        lst.remove(this);  // removing the socket from list
                        s.close();
                }
                catch(Exception e){}
        }
}

/* Main Server class starts -----> */
public class MainServer
{
}

        ServerSocket ssk;   //------serversocket object
        public MainServer(ServerSocket ssk){
                this.ssk=ssk;
        }
        public void startServer(){
                System.out.println("\nServer started running ................\n");
                int playerCount=0;
                try{
                while(!ssk.isClosed()) {
                        Socket s = ssk.accept();          //waiting for a player to connect
                        System.out.println("A new Client has connected "+(playerCount+1));
                        PlayerThread ct = new PlayerThread(s); // creating PlayerThread object
                        Thread t = new Thread(ct);
                        t.start(); // thread starts running
                }}catch(Exception e){}
        }

        public void closeServer(){
                try{
                if(ssk!=null){
                        ssk.close();
                }}catch(Exception e){}
        }

        public static void main(String args[])
        {
                try{
                ServerSocket ss = new ServerSocket(9122);
                MainServer obj = new MainServer(ss);
                obj.startServer();
                obj.closeServer();}catch(Exception e){}
        }
}
```

2.2 PLAYER SOURCE CODE

```
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;

class PlayerGUI extends JFrame implements ActionListener
{
JFrame f = new JFrame("B I N G O");

/****entry panels***/
JPanel pa = new JPanel();
JPanel pa1 = new JPanel();
JPanel pa2 = new JPanel();
JButton create = new JButton("CREATE");
JButton join = new JButton("JOIN");
JLabel mydet1 = new JLabel("Developed by");
JLabel mydet2 = new JLabel("Vadla Kalyan");
JLabel mydet3 = new JLabel("124003128");
JLabel mydet4 = new JLabel("Sem 5, CSE");
JLabel userc = new JLabel("PLAYER  NAME");
JTextField userctf= new JTextField();
JLabel codec = new JLabel("CODE");
JTextField codectf = new JTextField();
JButton bcreate = new JButton("CREATE->");
JLabel waitc = new JLabel();
JLabel userj = new JLabel("PLAYER  NAME");
JTextField userjtf= new JTextField();
JLabel codej = new JLabel("CODE ( enter your opponent code )");
JTextField codejtf = new JTextField();
JButton bjoin = new JButton("JOIN->");
JLabel errorj = new JLabel();

/***game panel***/
JPanel pb = new JPanel();
JPanel pb1 = new JPanel();
JPanel pb2 = new JPanel();
JPanel gpan1 = new JPanel();
JPanel setP = new JPanel();
```

```
JButton setB = new JButton("SET");
JTextField tf[] = new JTextField[25];
JLabel note = new JLabel();
String notes = "Numbers:(1 to 25), with no repetition";
JTextField poptf = new JTextField();
JButton pop = new JButton("POP");
JLabel poplab = new JLabel("Select a number from grid to POP");
JLabel turn = new JLabel();
JLabel lc = new JLabel("CHAT HERE");
JButton bse = new JButton("SEND");
JTextArea tc1 = new JTextArea();
JTextArea tc2 = new JTextArea();
JScrollPane js1 = new JScrollPane(tc1);
JScrollPane js2 = new JScrollPane(tc2);

Socket s;
PrintWriter pwr;
BufferedReader br;
int ack=0;
int crtflag;
String pps;
String name,code;

/*** BINGO algorithm variables ***/
int row[] = new int[5],cr=0,wr=0;
int column[] = new int[5],cc=0,wc=0;
int diagonal[] = new int[2],cd=0,wd=0;
int r,c;

UIManager um = new UIManager();
int setme=0,setop=0;

/***constructor***/
PlayerGUI()throws Exception
{
        f.setVisible(true);
        f.setLayout(null);
        f.setSize(1082,690);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        entryPanel();
        playingPanel();
}
```

```
/***entry panels GUI***/
public void entryPanel()
{
pa.setSize(1075,655);
pa.setLayout(null);
pa.setVisible(true);
pa.setBackground(new Color(0,153,0));
create.setBounds(100,30,100,50);
create.addActionListener(this);
create.setFont(new Font("Areal",Font.BOLD,15));
create.setForeground(Color.orange);
pa.add(create);

join.setBounds(250,30,100,50);
join.addActionListener(this);
join.setFont(new Font("Areal",Font.BOLD,15));
join.setForeground(Color.orange);
pa.add(join);

mydet1.setBounds(850,480,150,30);
mydet1.setForeground(Color.blue);
mydet1.setFont(new Font("Times New Roman",Font.BOLD,18));
pa.add(mydet1);

mydet2.setBounds(850,520,150,25);
mydet2.setForeground(Color.blue);
mydet2.setFont(new Font("Times New Roman",Font.BOLD,18));
pa.add(mydet2);

mydet3.setBounds(850,548,150,25);
mydet3.setForeground(Color.blue);
mydet3.setFont(new Font("Times New Roman",Font.BOLD,18));
pa.add(mydet3);

mydet4.setBounds(850,576,150,25);
mydet4.setForeground(Color.blue);
mydet4.setFont(new Font("Times New Roman",Font.BOLD,18));
pa.add(mydet4);

/***pa1 panel***/
pa1.setBounds(70,120,650,470);
pa1.setLayout(null);
pa1.setVisible(false);
```

```
pa1.setBackground(Color.white);

userc.setBounds(20,30,150,30);
userc.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa1.add(userc);

userctf.setBounds(20,80,400,30);
userctf.setForeground(Color.blue);
userctf.setFont(new Font("Areal",Font.BOLD,20));
pa1.add(userctf);

codec.setBounds(20,160,400,30);
codec.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa1.add(codec);

codectf.setBounds(20,210,400,30);
codectf.setForeground(Color.blue);
codectf.setFont(new Font("Areal",Font.BOLD,20));
pa1.add(codectf);

bcreate.setBounds(100,290,200,30);
bcreate.setForeground(Color.black);
bcreate.addActionListener(this);
bcreate.setFont(new Font("Areal",Font.BOLD,20));
pa1.add(bcreate);

waitc.setBounds(20,350,500,30);
waitc.setForeground(Color.blue);
waitc.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa1.add(waitc);
waitc.setText("Share details and Wait for opponent....");

pa.add(pa1);

/***pa2 panel ***/
pa2.setBounds(70,120,650,470);
pa2.setLayout(null);
pa2.setVisible(false);
pa2.setBackground(Color.white);

userj.setBounds(20,30,150,30);
userj.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa2.add(userj);
```

```
userjtf.setBounds(20,80,400,30);
userjtf.setForeground(Color.blue);
userjtf.setFont(new Font("Areal",Font.BOLD,20));
pa2.add(userjtf);

codej.setBounds(20,160,400,30);
codej.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa2.add(codej);

codejtf.setBounds(20,210,400,30);
codejtf.setForeground(Color.blue);
codejtf.setFont(new Font("Areal",Font.BOLD,20));
pa2.add(codejtf);

bjoin.setBounds(100,290,200,30);
bjoin.setForeground(Color.black);
bjoin.addActionListener(this);
bjoin.setFont(new Font("Areal",Font.BOLD,20));
pa2.add(bjoin);

errorj.setBounds(20,350,500,30);
errorj.setForeground(Color.red);
errorj.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
pa2.add(errorj);

pa.add(pa2);
f.add(pa);
}

/*** GAME Panel GUI ***/
public void playingPanel()
{
pb.setSize(1075,655);
pb.setLayout(null);
pb.setVisible(false);
pb.setBackground(Color.green);

pb1.setBounds(2,2,668,646);
pb1.setLayout(null);
pb1.setBackground(Color.cyan);

gpan1.setBounds(167,167,334,334);
```

```
gpan1.setBackground(Color.orange);
gpan1.setBorder(BorderFactory.createLineBorder(Color.orange,6));
gpan1.setLayout(new GridLayout(5,5,4,4));
for(int i=0;i<25;i++){
tf[i] = new JTextField();
tf[i].setHorizontalAlignment(JTextField.CENTER);
tf[i].setBackground(Color.white);
tf[i].setForeground(Color.black);
tf[i].setFont(new Font("Areal",Font.BOLD,20));
gpan1.add(tf[i]);
}
pb1.add(gpan1);

setP.setBounds(521,315,100,40);
setP.setLayout(null);
setP.add(setB);
setB.setBounds(0,0,100,40);
setB.setForeground(Color.blue);
setB.setFont(new Font("Areal",Font.BOLD,17));
setB.addActionListener(this);
setP.setVisible(true);
pb1.add(setP);

note.setBounds(167,115,350,40);
note.setForeground(Color.red);
note.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,17));
pb1.add(note);

poplab.setBounds(167,510,350,40);
poplab.setForeground(Color.red);
poplab.setBackground(Color.white);
poplab.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,17));
pb1.add(poplab);

turn.setBounds(70,565,70,40);
turn.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,17));
turn.setText("Wait");
turn.setForeground(Color.red);
pb1.add(turn);

poptf.setBounds(167,565,203,40);
poptf.setForeground(Color.black);
poptf.setBackground(Color.white);
poptf.setFont(new Font("Areal",Font.BOLD,20));
poptf.setHorizontalAlignment(JTextField.CENTER);
poptf.setEditable(false);
pb1.add(poptf);

pop.setBounds(381,565,120,40);
pop.setForeground(Color.blue);
pop.setFont(new Font("Areal",Font.BOLD,17));
pop.addActionListener(this);
pop.setEnabled(false);
pb1.add(pop);

pb.add(pb1);

/*** CHAT panel GUI ***/
pb2.setBounds(672,2,396,646);
pb2.setLayout(null);
pb2.setBackground(Color.orange);

lc.setBounds(116,2,160,30);
lc.setFont(new Font("Areal",Font.BOLD | Font.ITALIC,20));
lc.setForeground(Color.blue);
pb2.add(lc);

tc1.setFont(new Font("Lucida Console",Font.BOLD,15));
tc1.setEditable(false);
tc1.setLineWrap(true);
tc1.setBackground(Color.white);
js1.setBounds(2,34,392,548);
pb2.add(js1);

tc2.setFont(new Font("Lucida Console",Font.BOLD,18));
tc2.setLineWrap(true);
js2.setBounds(2,584,262,60);
pb2.add(js2);

bse.setBounds(266,584,120,60);
bse.addActionListener(this);
bse.setFont(new Font("Areal",Font.BOLD,20));
bse.setForeground(Color.blue);
pb2.add(bse);
pb.add(pb2);
f.add(pb);
```

```
}

/****** Player receiving messages ******/
public void listenMsg()
{
        /* thread */
        new Thread(new Runnable(){
        @Override
        public void run(){
        String sms;
        try{
        while(true){
        int i=1;
        while(!(sms=br.readLine()).equals("NULL")){
        if(i==1){
        /*** '_' says that received message is chat text ***/
        if(sms.charAt(0)=='_'){
        i++;
        tc1.append("\n"+sms+"\n");  // appending the received message to TextArea
        }
        /*** received WIN message - opponent won ****/
        else if(sms.equals("$Win")){
                um.put("OptionPane.messageForeground",Color.red);
                JOptionPane.showConfirmDialog(f,"YOU
                LOST","BINGO",JOptionPane.DEFAULT_OPTION);
                pa.setVisible(true);
                pb.setVisible(false);
                Defaulties();
                s.close();   // closing connection
        }
        /**** the oppenent filled the grid ****/
        else if(sms.equals("$Set")){
        setop=1;
        if(pps.equals("create")){
                /*** as setme = 1 both can start play ****/
                if(setme==1){
                        turn.setText("---play---");
                        turn.setForeground(Color.blue);
                        pop.setEnabled(true);
                        poptf.setEditable(true);
                }
                else{
                        turn.setText("set grid");   // notification that to fill the grid fast
```

```
                    turn.setForeground(Color.red);
                    pop.setEnabled(false);
                    poptf.setEditable(false);
          }
}
/***** chance is given to created player joined player must wait for turn *****/
else if(pps.equals("join")){
          if(setme==1){
          turn.setText("---wait---");  // wait
          turn.setForeground(Color.red);
          pop.setEnabled(false);
          poptf.setEditable(false);
}
else{
          turn.setText("set grid");
          turn.setForeground(Color.red);
          pop.setEnabled(false);
          poptf.setEditable(false);
}}}
else{
searchGrid(sms); // received number is serched and popped out of grid
int winR = popFun();  // calling the BINGO algorithm
/*** if win >= 5 then player won the game ****/
if(winR>=5){
          pwr.println("$Win\nNULL");
          pwr.flush();
          um.put("OptionPane.messageForeground",Color.blue);
          JOptionPane.showConfirmDialog(f,"BINGO!!!\nYOUWON","BINGO",JOptio
nPane.DEFAULT_OPTION);
          pa.setVisible(true);
          pb.setVisible(false);
          Defaulties();
          try{
          s.close();}catch(Exception e){}
}
else{
          poptf.setEditable(true);
          pop.setEnabled(true);
          turn.setText("---play---");
          turn.setForeground(Color.blue);
}}}
else
tc1.append(""+sms+"\n");
```

```
        }
        }}catch(Exception e){}
        }
}).start();
}

/***** BINGO Algorithm check *******/
public int popFun()
{
        wr=0;wc=0;wd=0;
        cr=0;cc=0;cd=0;
        for(int i=0;i<25;i++){
                if(!tf[i].getText().equals("X")){
                r = i/5;
                c = i%5;
                insertInRow(r);
                insertInColumn(c);
                if(r==c)
                        insertInDiagonal(0);
                if(r-c==4 || r-c==-4)
                        insertInDiagonal(1);
                }
        }
        for(int j=0;j<5;j++){
                if(!findInArray(row,j,cr))
                        wr++;
                if(!findInArray(column,j,cc))
                        wc++;
                if(j<2)
                        if(!findInArray(diagonal,j,cd))
                                wd++;
        }

        return wd+wr+wc; // returning total count
}

public boolean findInArray(int arr[],int element,int count)
{
        for(int i=0;i<count;i++){
                if(arr[i]==element)
                        return true;
        }
        return false;
```

```
}
public void insertInRow(int insNum)
{
	for(int i=0;i<cr;i++){
		if(row[i]==insNum)
			return;
	}
	row[cr]=insNum;
	cr++;
}
public void insertInColumn(int insNum)
{
	for(int i=0;i<cc;i++){
	if(column[i]==insNum)
		return;
	}
	column[cc]=insNum;
	cc++;
}
public void insertInDiagonal(int insNum)
{
	for(int i=0;i<cd;i++){
	if(diagonal[i]==insNum)
		return;
	}
	diagonal[cd]=insNum;
	cd++;
}

/**** checks for any wrong input in grid */
public void wrongNum(int i)
{
	tf[i].setForeground(Color.red);
	tf[i].setText("X");
	crtflag=0;
}
public boolean findDot(String sd,int i)
{
	for(int j=0;j<sd.length();j++){
	if(sd.charAt(j)=='.')
		return true;
	}
	return false;
```

```
}
public boolean findSpace(String sd,int i)
{
        for(int j=0;j<sd.length();j++){
        if(sd.charAt(j)==' ')
                return true;
        }
        return false;
}
public boolean repetition(String stf,int i)
{
        for(int j=0;j<i;j++){
        if(tf[j].getText().equals(stf))
                return true;
        }
        return false;
}

/*** pops a number from grid making the value as blue coloured X ***/
public boolean searchGrid(String popNumS)
{
for(int i=0;i<25;i++){
if(tf[i].getText().equalsIgnoreCase(popNumS)){
tf[i].setText("X");
tf[i].setForeground(Color.blue);
return  true;
}}
return false;
}

public void Defaulties()
{
for(int i=0;i<25;i++){
tf[i].setEditable(true);
tf[i].setText("");
tf[i].setForeground(Color.black);
}
setP.setVisible(true);
turn.setText("wait");
turn.setForeground(Color.red);
pop.setEnabled(true);
poptf.setEditable(false);
tc1.setText("");
```

```
tc2.setText("");
}

/******** button action event *************/
public void actionPerformed(ActionEvent ae)
{
String bb = ae.getActionCommand();   // getting button name

/*** send message in chat ***/
if(bb.equals("SEND"))          {
String Smsg;
if(!(Smsg=tc2.getText()).equals(""))  // if text is not empty then proceed
{
try{
pwr.println("___[   "+name+"   ]___\n"+Smsg+"\nNULL");  // prefixing with '_'
pwr.flush();
tc2.setText("");
}catch(Exception e){}
tc1.append("\n_____ME_____\n"+Smsg+"\n");
}}

/**** Create button ready to create a game ****/
if(bb.equals("CREATE")){
pa2.setVisible(false);
pa1.setVisible(true);
}

/**** join button ****/
if(bb.equals("JOIN")){
pa1.setVisible(false);
pa2.setVisible(true);
}

/**** setting the grid ****/
// checks for incorrect input in the grid and marks it as red coloured X
// if all the input is correct then notifies the opponent
if(bb.equals("SET")){
crtflag=1;
for(int i=0;i<25;i++){
String stf=tf[i].getText();
if( !stf.equals("")){
        if(repetition(stf,i))
                wrongNum(i);
```

```
		else if(findSpace(stf,i))
			wrongNum(i);
		else if(findDot(stf,i))
			wrongNum(i);
		else if(stf.charAt(0)>='1' && stf.charAt(0)<='9'){
			int x = Integer.valueOf(stf);
			if(x<1||x>25)
				wrongNum(i);
			else{tf[i].setForeground(Color.black);}
		}
		else
			wrongNum(i);
	}
	else
		wrongNum(i);
	}
	if(crtflag==1){
	for(int i=0;i<25;i++){tf[i].setEditable(false);}
	setP.setVisible(false);
	setme=1;
	pwr.println("$Set\nNULL");
	pwr.flush();
	if(pps.equals("create")){
		if(setop==1){
			turn.setText("---play---");
			turn.setForeground(Color.blue);
			pop.setEnabled(true);
			poptf.setEditable(true);
		}
		else{
			turn.setText("wait");
			turn.setForeground(Color.red);
			pop.setEnabled(false);
			poptf.setEditable(false);
		}
	}
	if(pps.equals("join")){
		if(setop==1){
			turn.setText("---wait---");
			turn.setForeground(Color.red);
			pop.setEnabled(false);
			poptf.setEditable(false);
		}
```

```
        else{
                turn.setText("wait");
                turn.setForeground(Color.red);
                pop.setEnabled(false);
                poptf.setEditable(false);
        }
}
}
}

/**** POP button action removes number from grid *****/
// checks for bingo
// and sends the number to player
if(bb.equals("POP")){
String pops = poptf.getText();
if(!pops.equals("X")){
        if(searchGrid(pops)){
                poptf.setText("");
                poptf.setForeground(Color.black);
                poptf.setEditable(false);
                pop.setEnabled(false);
                int winR = popFun();
                if(winR<5){
                        pwr.println(pops+"\nNULL");
                        pwr.flush();
                }
                else{
                        pwr.println("$Win\nNULL");
                        pwr.flush();
                        um.put("OptionPane.messageForeground",Color.blue);
        JOptionPane.showConfirmDialog(f,"BINGO!!!\nYOU
        WON","BINGO",JOptionPane.DEFAULT_OPTION);
                        pa.setVisible(true);
                        pb.setVisible(false);
                        Defaulties();
                        try{s.close();}catch(Exception e){}
                }
                turn.setText("---wait---");
                turn.setForeground(Color.red);
        }
        else{
                poptf.setText("X");
                poptf.setForeground(Color.red);
```

```
		}
}
}

/** creates the socket and sends the player name and code to server **/
//waits for opponent to join
if(bb.equals("CREATE->")){
try{
s =  new Socket("localhost",9122);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
pwr = new PrintWriter(s.getOutputStream());
pps = "create";
create.setEnabled(false);
join.setEnabled(false);
userctf.setEditable(false);
codectf.setEditable(false);
bcreate.setEnabled(false);
String sn = userctf.getText();
String sc = codectf.getText();
name=sn;
code=sc;
// sending details to server
pwr.println("create->");
pwr.flush();
pwr.println(sn);
pwr.flush();
pwr.println(sc);
pwr.flush();
ack = Integer.valueOf(br.readLine());  // waiting for opponent
if(ack==1){
		turn.setText("wait");
		turn.setForeground(Color.red);
		pa.setVisible(false);
		pb.setVisible(true);
		note.setText(notes);
		create.setEnabled(true);
		join.setEnabled(true);
		userctf.setEditable(true);
		codectf.setEditable(true);
		bcreate.setEnabled(true);
		userctf.setText("");
		codectf.setText("");
		listenMsg();   // starts listening to player
```

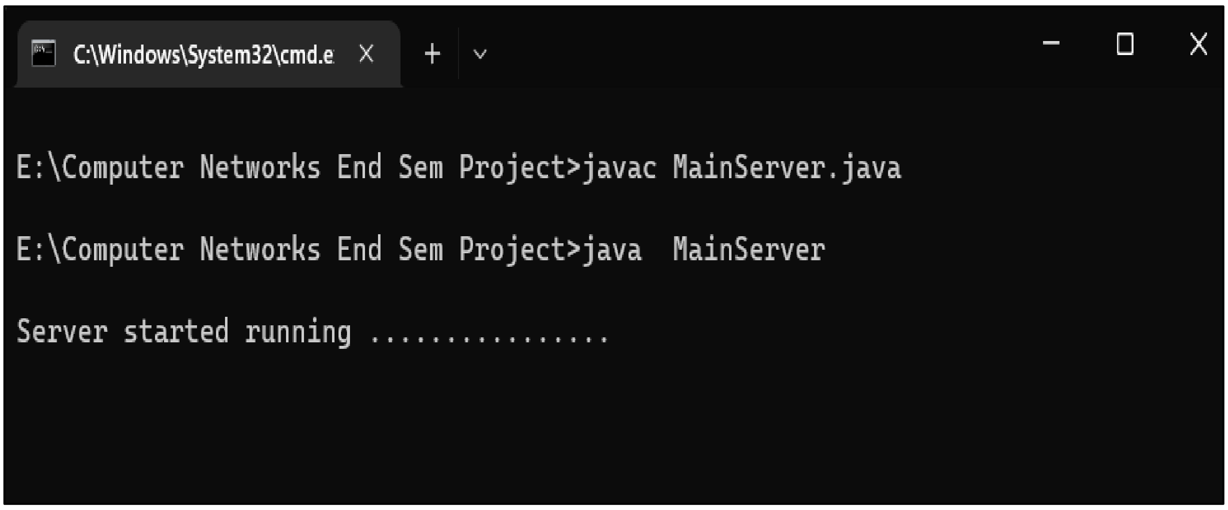
```
}
}
catch(Exception e){    }
}

/*** creates the socket connection with server ***/
/***sends the details to the server ***/
// server checks the code and sends the message
// if room exists then enters the game else re-enter the code
if(bb.equals("JOIN->")){
try{
s =  new Socket("localhost",9122);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
pwr = new PrintWriter(s.getOutputStream());
pps="join";
create.setEnabled(false);
join.setEnabled(false);
userjtf.setEditable(false);
codejtf.setEditable(false);
String sn = userjtf.getText();
String sc = codejtf.getText();
name=sn;
code=sc;
/** sending details to server
pwr.println("join->");
pwr.flush();
pwr.println(sn);
pwr.flush();
pwr.println(sc);
pwr.flush();
ack = Integer.valueOf(br.readLine());  // waiting for the server
// if room exists ack will be 1
if(ack==1){
        turn.setText("wait");
        turn.setForeground(Color.red);
        pa.setVisible(false);
        pb.setVisible(true);
        note.setText(notes);
        create.setEnabled(true);
        join.setEnabled(true);
        userjtf.setEditable(true);
        codejtf.setEditable(true);
        bjoin.setEnabled(true);
```

```
            userjtf.setText("");
            codejtf.setText("");
            errorj.setText("");
            listenMsg();   // start listening
}
// if room not exists ack will be 0, re-enter the code and socket is closed
else{
            create.setEnabled(true);
            join.setEnabled(true);
            userjtf.setEditable(true);
            codejtf.setEditable(true);
            bjoin.setEnabled(true);
            codejtf.setText("");
            errorj.setText("No room exists");
            s.close();
}
}
catch(Exception e){    }
}
}
}

class Player
{
            public static void main(String args[])throws Exception
            {
                        PlayerGUI obj = new PlayerGUI();
            }
}
```

CHAPTER 3

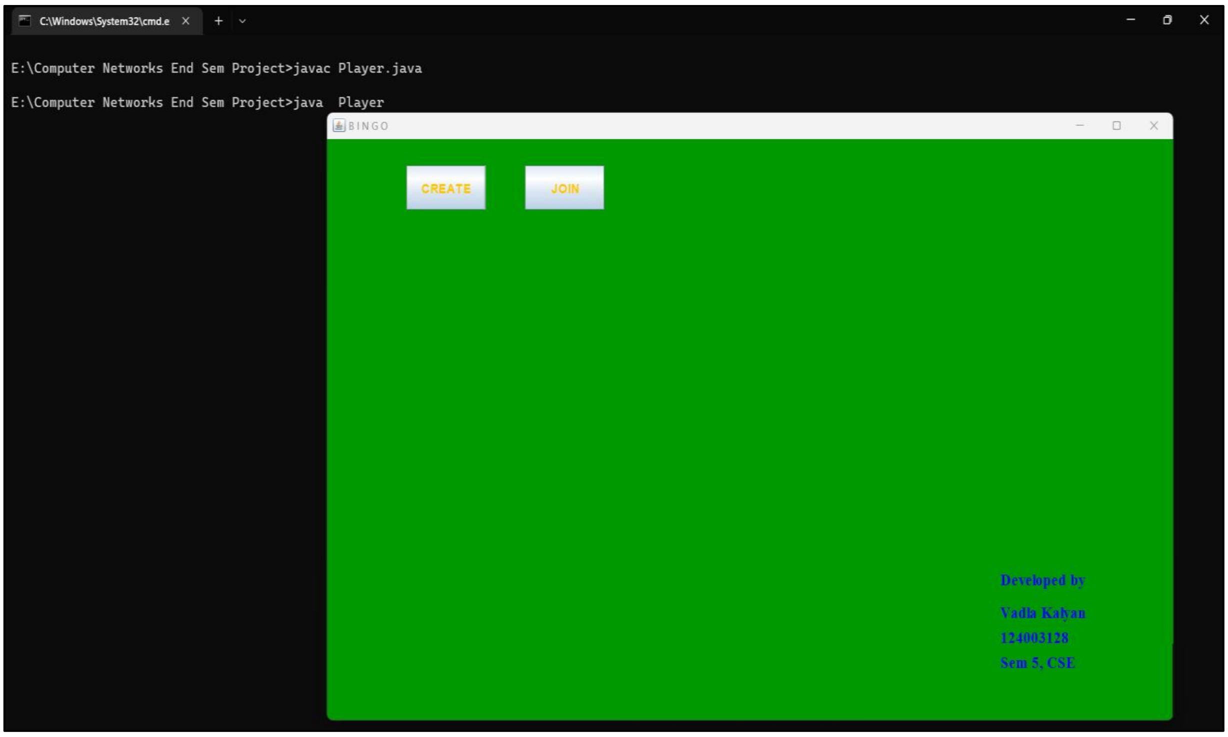
RESULTS

Fig 3.1 Server started running



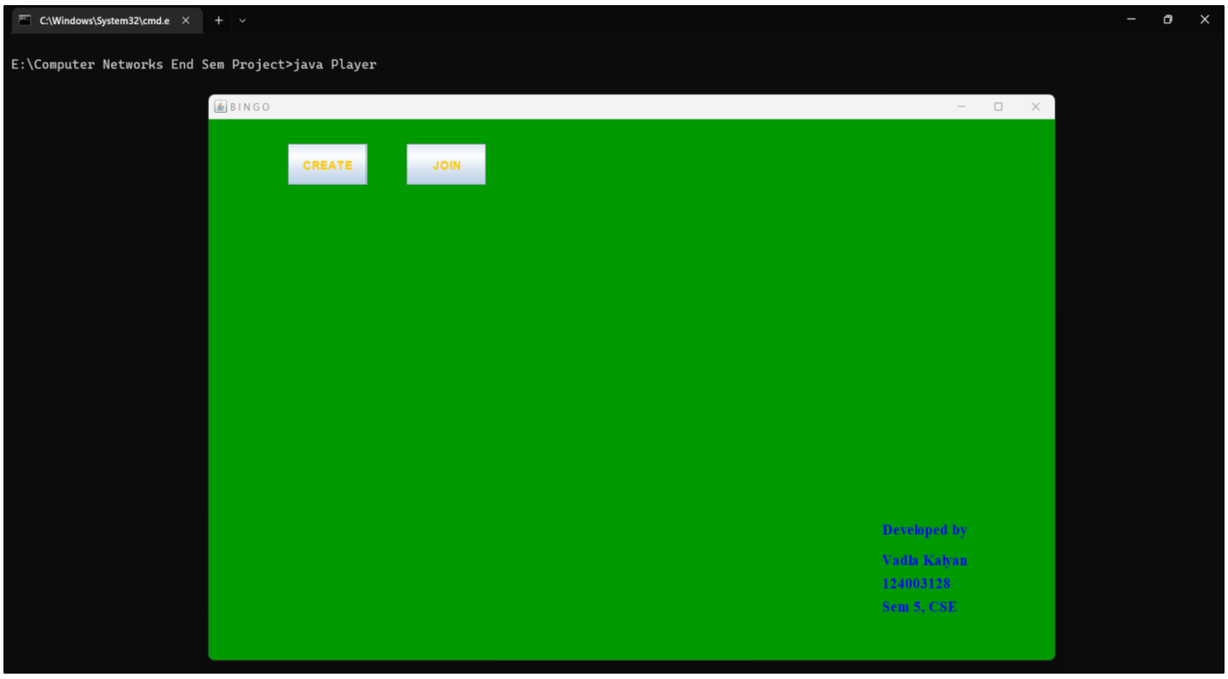
```
C:\Windows\System32\cmd.e

E:\Computer Networks End Sem Project>javac MainServer.java

E:\Computer Networks End Sem Project>java  MainServer

Server started running ................
```

Fig 3.2 Player 1 joined



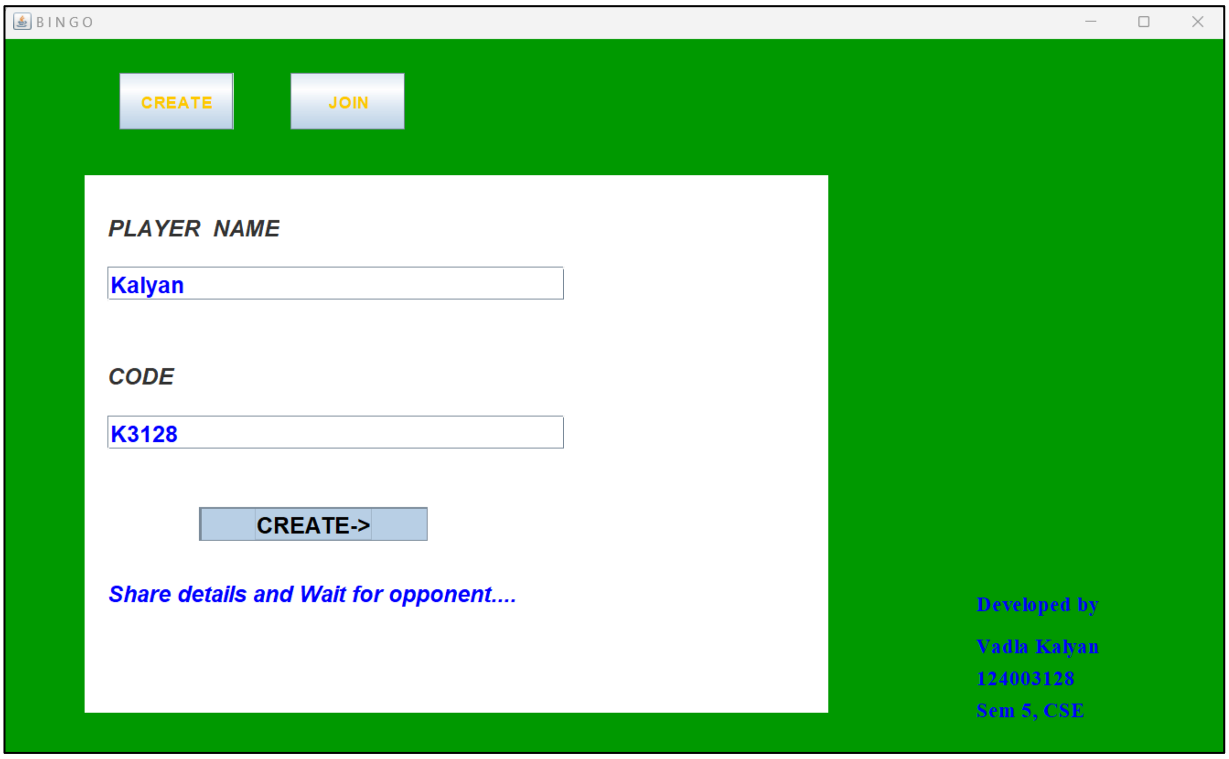
```
C:\Windows\System32\cmd.e

E:\Computer Networks End Sem Project>javac Player.java

E:\Computer Networks End Sem Project>java  Player
```

Fig 3.3 Player 2 joined



```
E:\Computer Networks End Sem Project>java Player
```

Fig 3.4 Player 1 creates the game and waits for opponent



PLAYER NAME

Kalyan

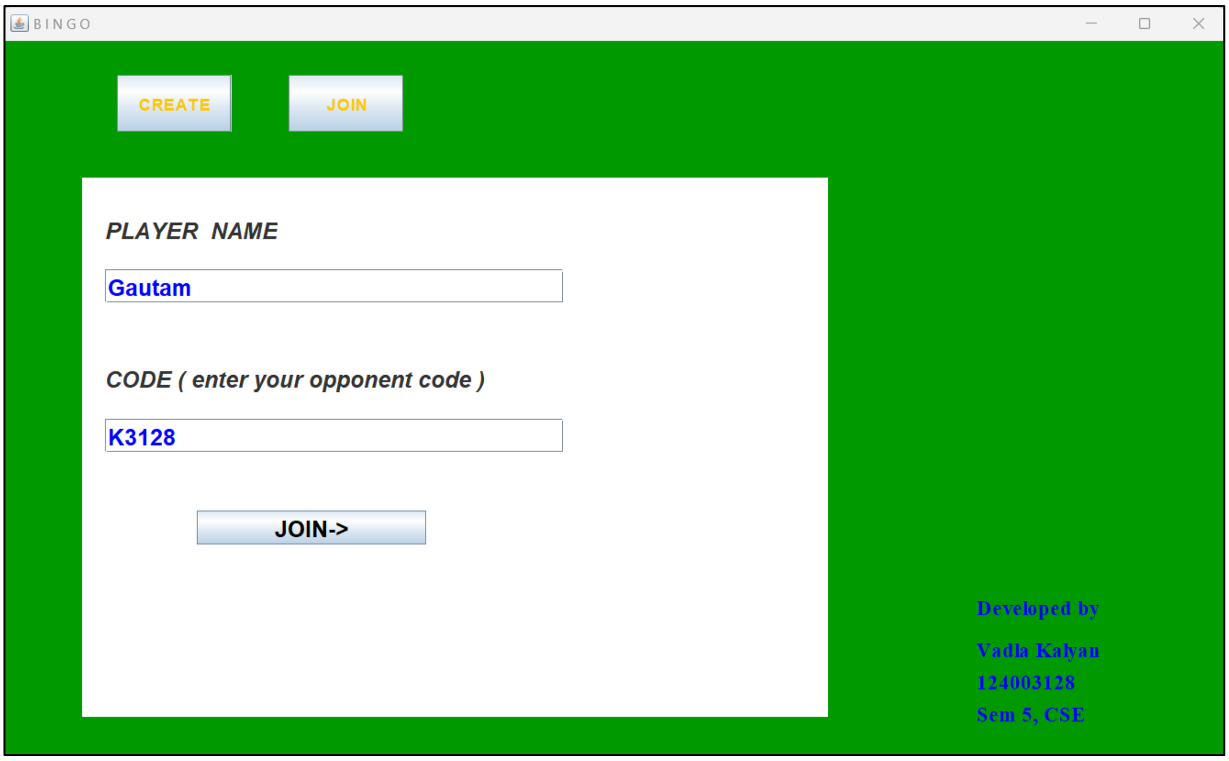
CODE

K3128

CREATE->

Share details and Wait for opponent....

Fig 3.5 Player 2 joines with the same code



PLAYER NAME

Gautam

CODE (enter your opponent code)

K3128

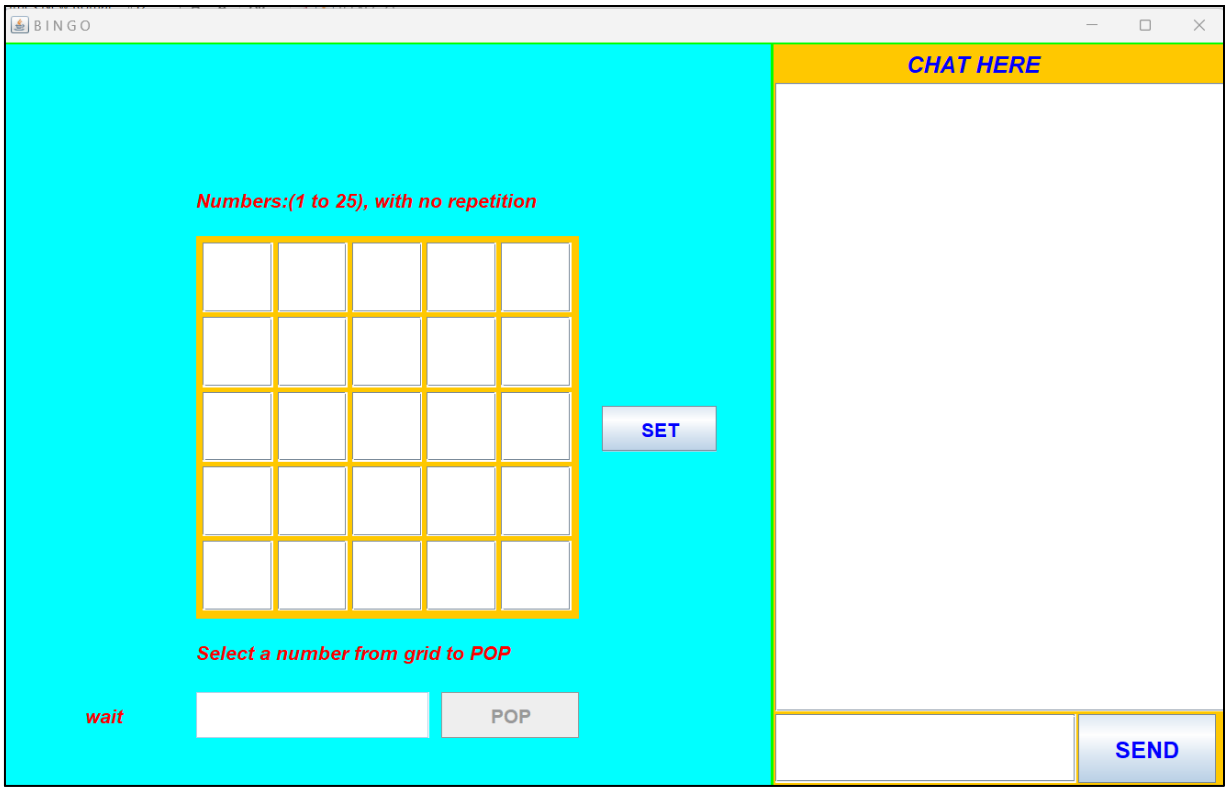
Developed by

Vadla Kalyan

124003128

Sem 5, CSE

Fig 3.6 Both players enters the game and are said to be in same room with same code

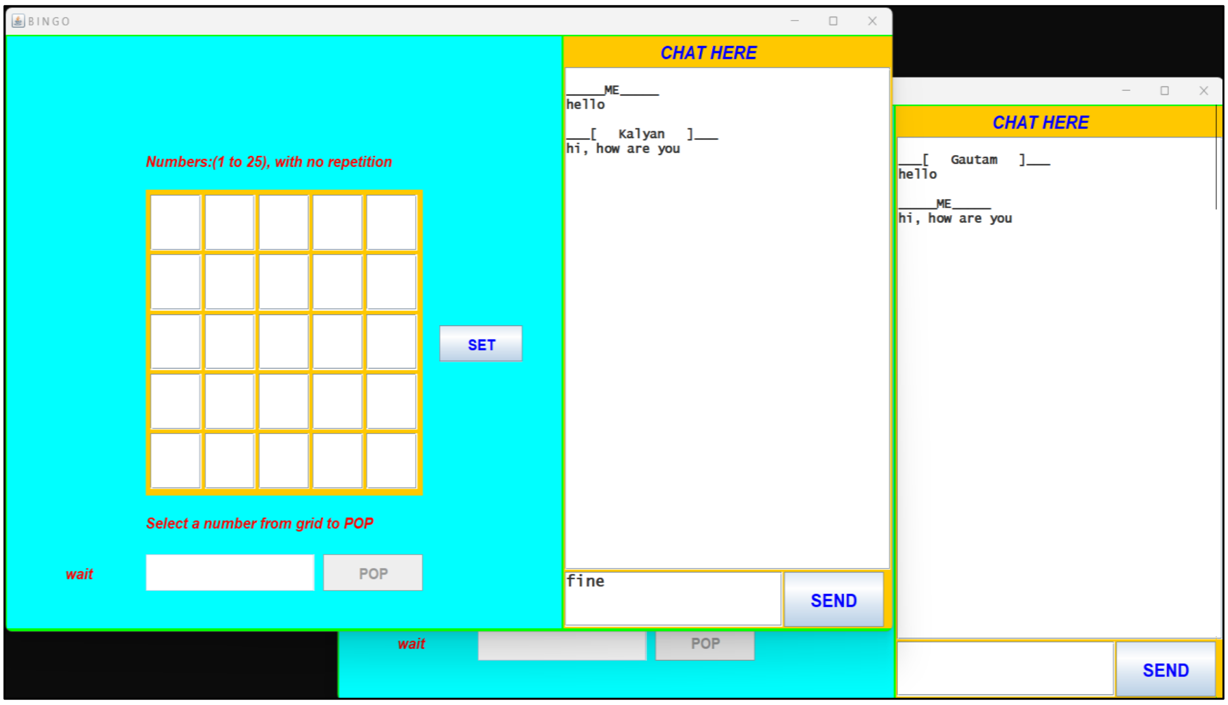


Numbers:(1 to 25), with no repetition

Select a number from grid to POP

wait

Fig 3.7 Players chatting

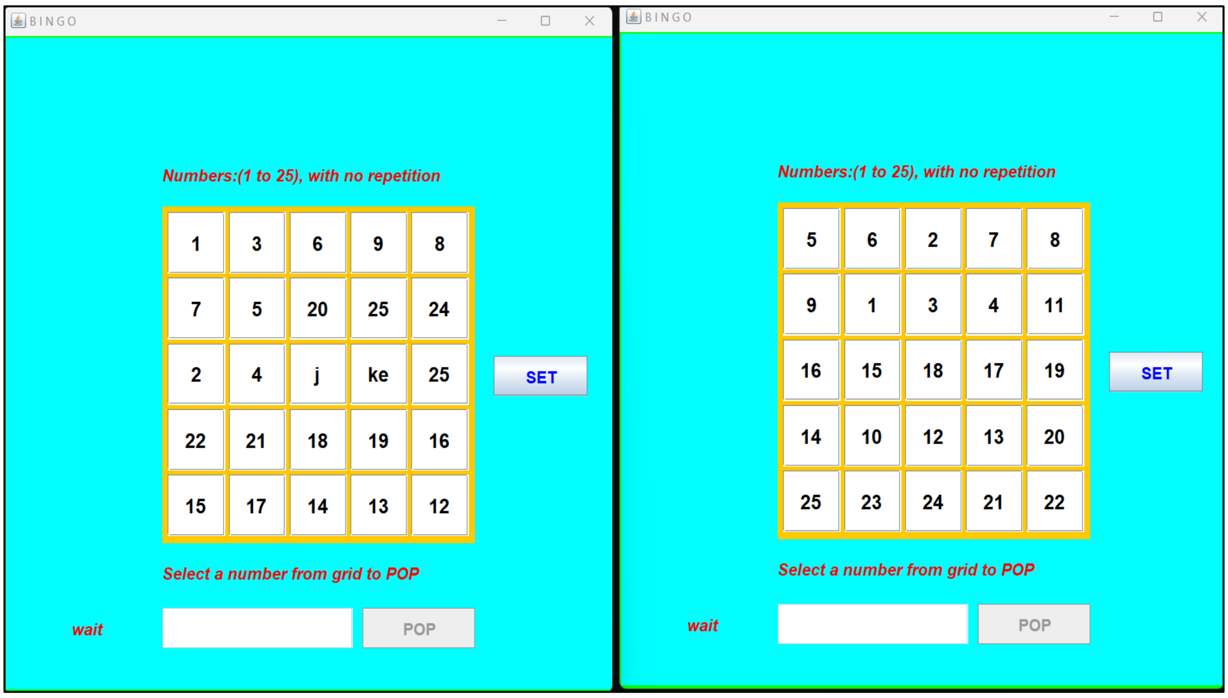

BINGO
Numbers:(1 to 25), with no repetition
SET
Select a number from grid to POP
wait
POP
CHAT HERE
SEND
wait
POP
CHAT HERE
SEND

```
_____ME_____
hello

___[  Kalyan  ]___
hi, how are you
```

```
fine
```

```
___[  Gautam  ]___
hello

_____ME_____
hi, how are you
```

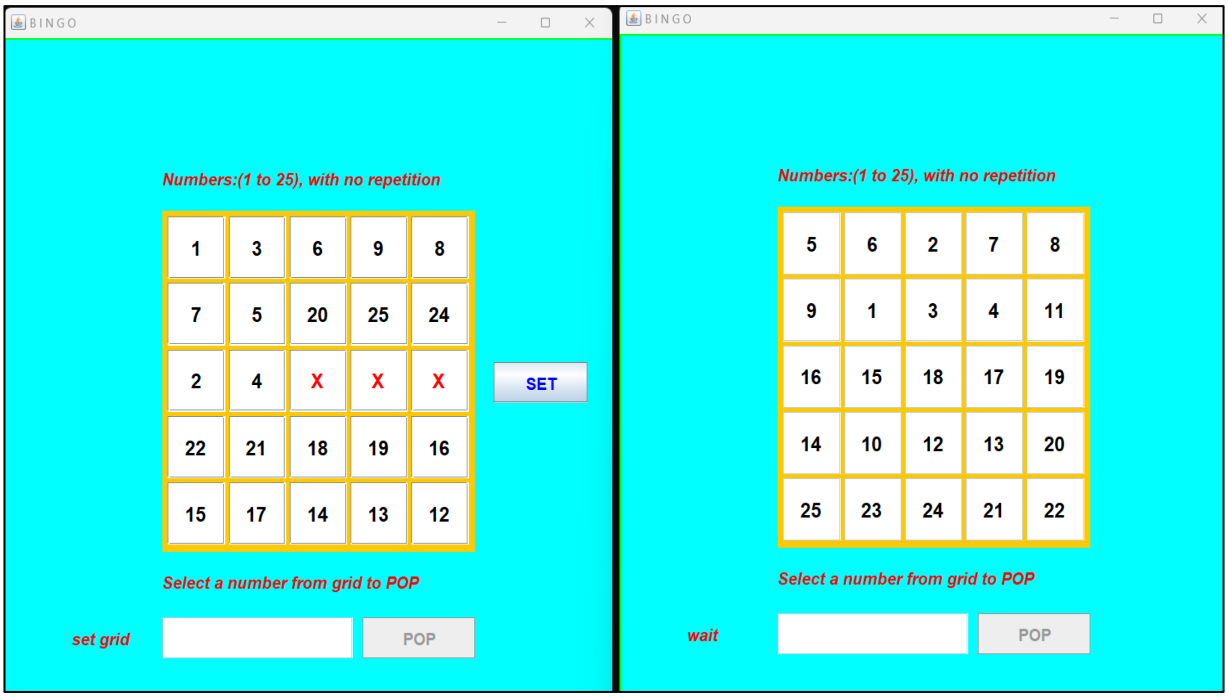
Fig 3.8 Players filling the grid with integers from 1 to 25


BINGO
Numbers:(1 to 25), with no repetition
SET
Select a number from grid to POP
wait
POP
BINGO
Numbers:(1 to 25), with no repetition
SET
Select a number from grid to POP
wait
POP

1	3	6	9	8
7	5	20	25	24
2	4	j	ke	25
22	21	18	19	16
15	17	14	13	12

5	6	2	7	8
9	1	3	4	11
16	15	18	17	19
14	10	12	13	20
25	23	24	21	22

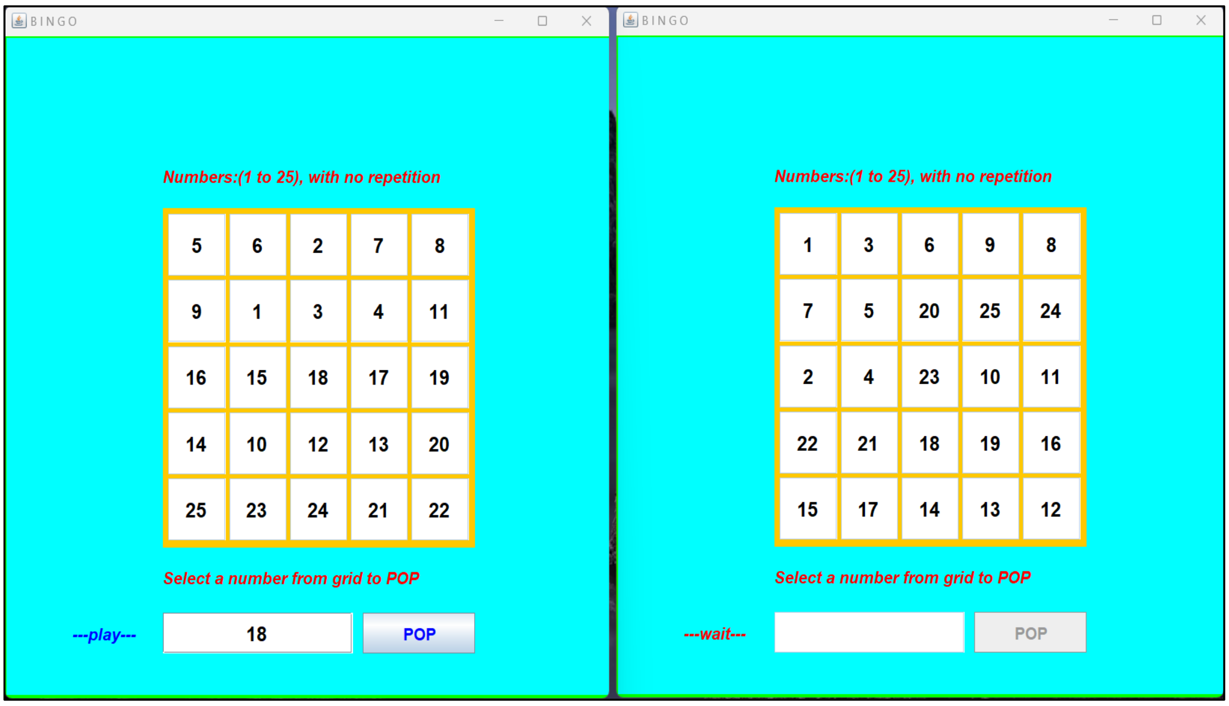
Fig 3.9 Eliminating the invalid entries


BINGO
Numbers:(1 to 25), with no repetition
SET
Select a number from grid to POP
set grid
POP
BINGO
Numbers:(1 to 25), with no repetition
Select a number from grid to POP
wait
POP

1	3	6	9	8
7	5	20	25	24
2	4	X	X	X
22	21	18	19	16
15	17	14	13	12

5	6	2	7	8
9	1	3	4	11
16	15	18	17	19
14	10	12	13	20
25	23	24	21	22

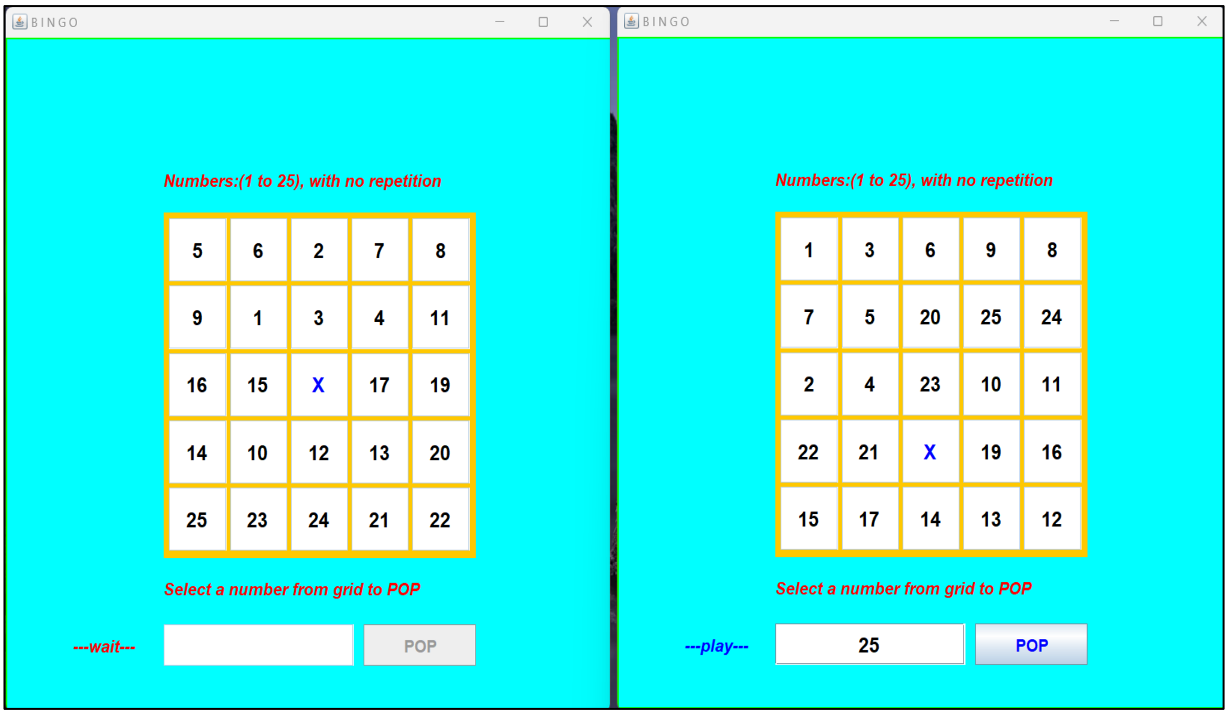
Fig 3.10 Player 1 gets a chance to pop


BINGO
Numbers:(1 to 25), with no repetition
Select a number from grid to POP
---play---
18
POP
BINGO
Numbers:(1 to 25), with no repetition
Select a number from grid to POP
---wait---
POP

5	6	2	7	8
9	1	3	4	11
16	15	18	17	19
14	10	12	13	20
25	23	24	21	22

1	3	6	9	8
7	5	20	25	24
2	4	23	10	11
22	21	18	19	16
15	17	14	13	12

Fig 3.11 Selected number is popped out from grids of both players and player 2 gets chance



Numbers:(1 to 25), with no repetition

5	6	2	7	8
9	1	3	4	11
16	15	X	17	19
14	10	12	13	20
25	23	24	21	22

Select a number from grid to POP

---wait---

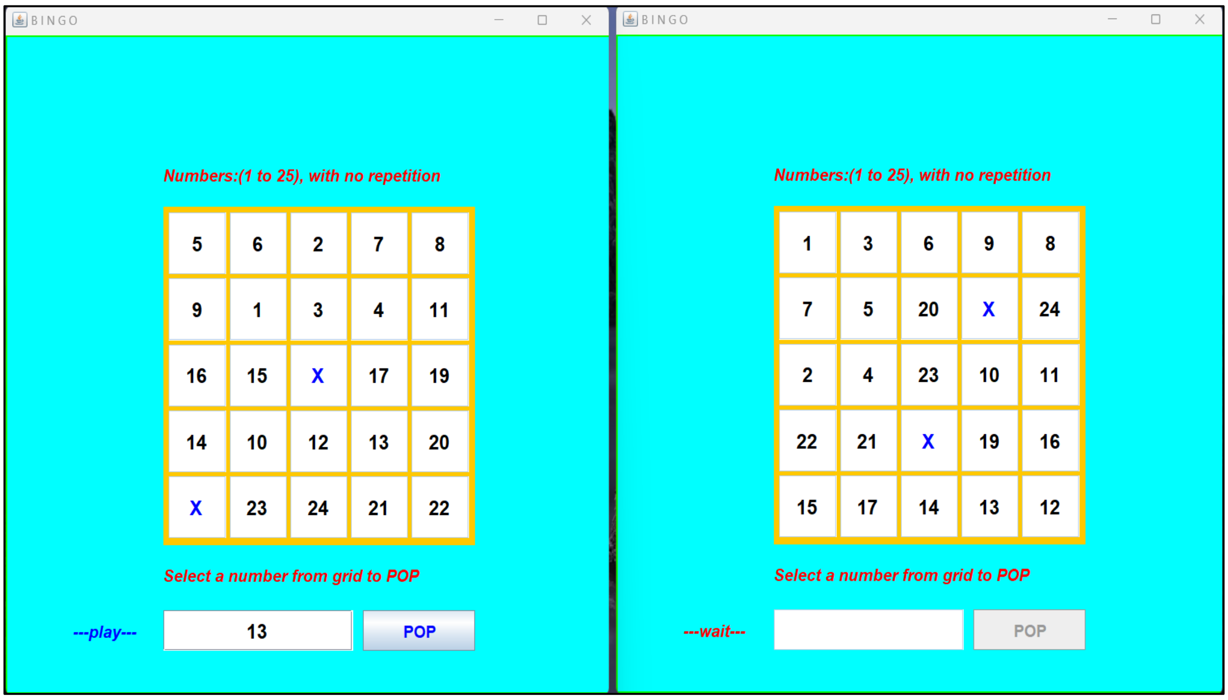
Numbers:(1 to 25), with no repetition

1	3	6	9	8
7	5	20	25	24
2	4	23	10	11
22	21	X	19	16
15	17	14	13	12

Select a number from grid to POP

---play--- 25

Fig 3.12 Player 2 pops a number from both grids and then player 1 gets the chance



Numbers:(1 to 25), with no repetition

5	6	2	7	8
9	1	3	4	11
16	15	X	17	19
14	10	12	13	20
X	23	24	21	22

Select a number from grid to POP

---play--- 13

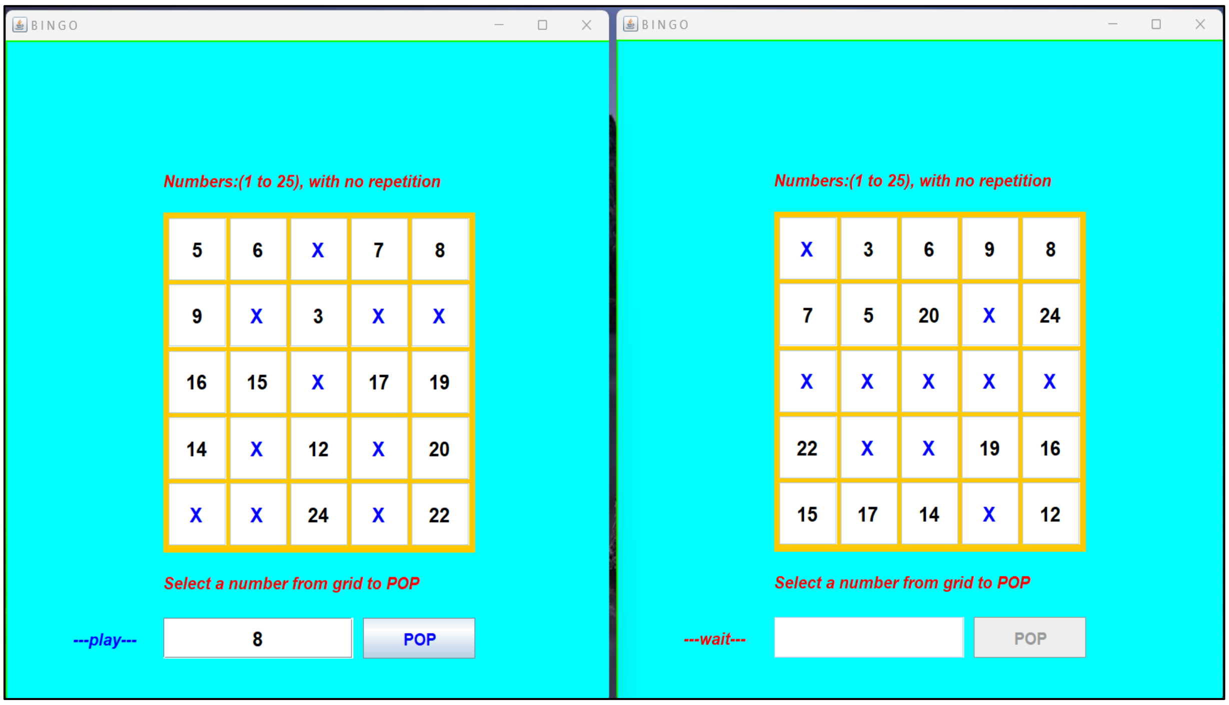
Numbers:(1 to 25), with no repetition

1	3	6	9	8
7	5	20	X	24
2	4	23	10	11
22	21	X	19	16
15	17	14	13	12

Select a number from grid to POP

---wait---

Fig 3.13 snapshot taken in middle of the game



Numbers:(1 to 25), with no repetition

5	6	X	7	8
9	X	3	X	X
16	15	X	17	19
14	X	12	X	20
X	X	24	X	22

Select a number from grid to POP

---play--- 8 POP

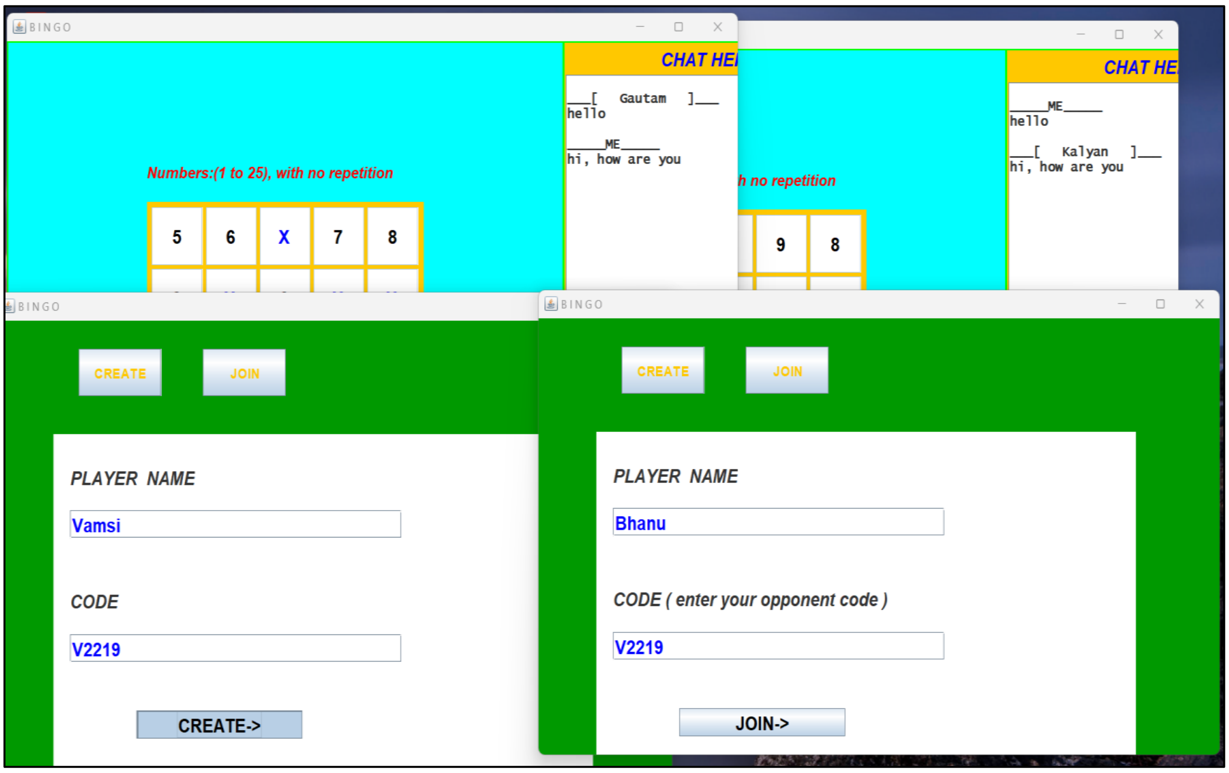
Numbers:(1 to 25), with no repetition

X	3	6	9	8
7	5	20	X	24
X	X	X	X	X
22	X	X	19	16
15	17	14	X	12

Select a number from grid to POP

---wait--- POP

Fig 3.14 New room with another code created by others



```
___[  Gautam  ]___
hello

_____ME_____
hi, how are you
```

Numbers:(1 to 25), with no repetition

5	6	X	7	8

```
_____ME_____
hello

___[  Kalyan  ]___
hi, how are you
```

h no repetition

9	8

PLAYER NAME

Vamsi

CODE

V2219

CREATE->

PLAYER NAME

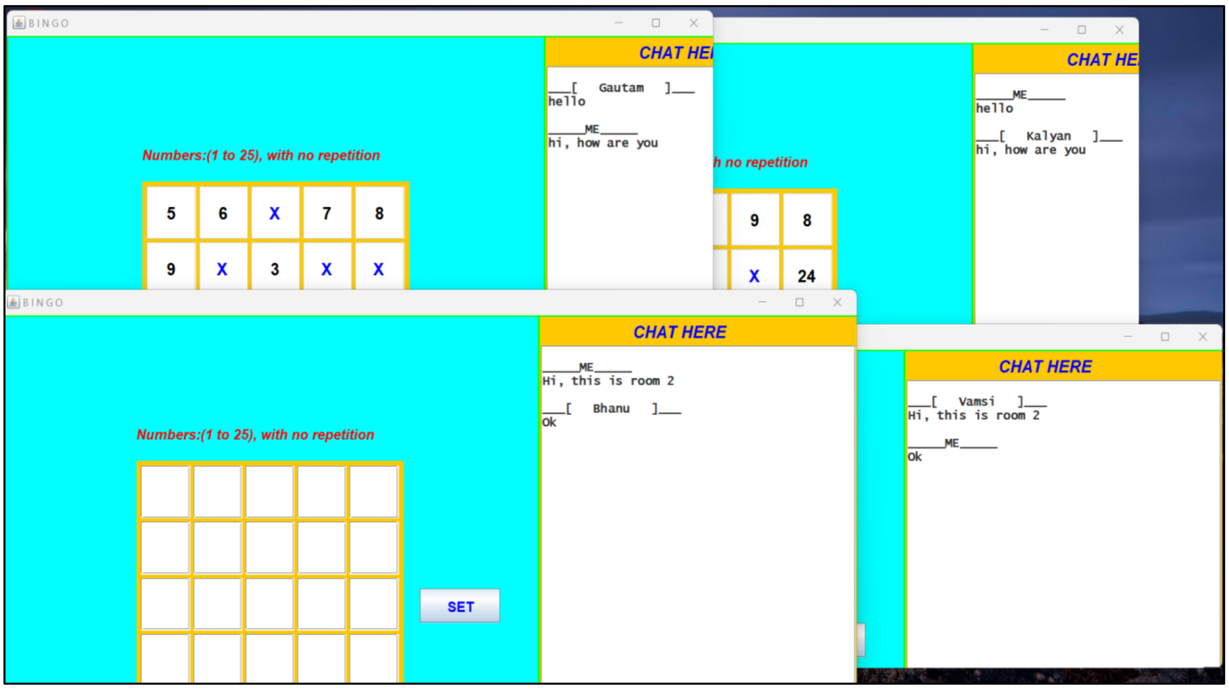
Bhanu

CODE (enter your opponent code)

V2219

JOIN->

Fig 3.15 Communication occurs between the players in same room



```
___[ Gautam ]___
hello

_____ME_____
hi, how are you
```

```
_____ME_____
hello

___[ Kalyan ]___
hi, how are you
```

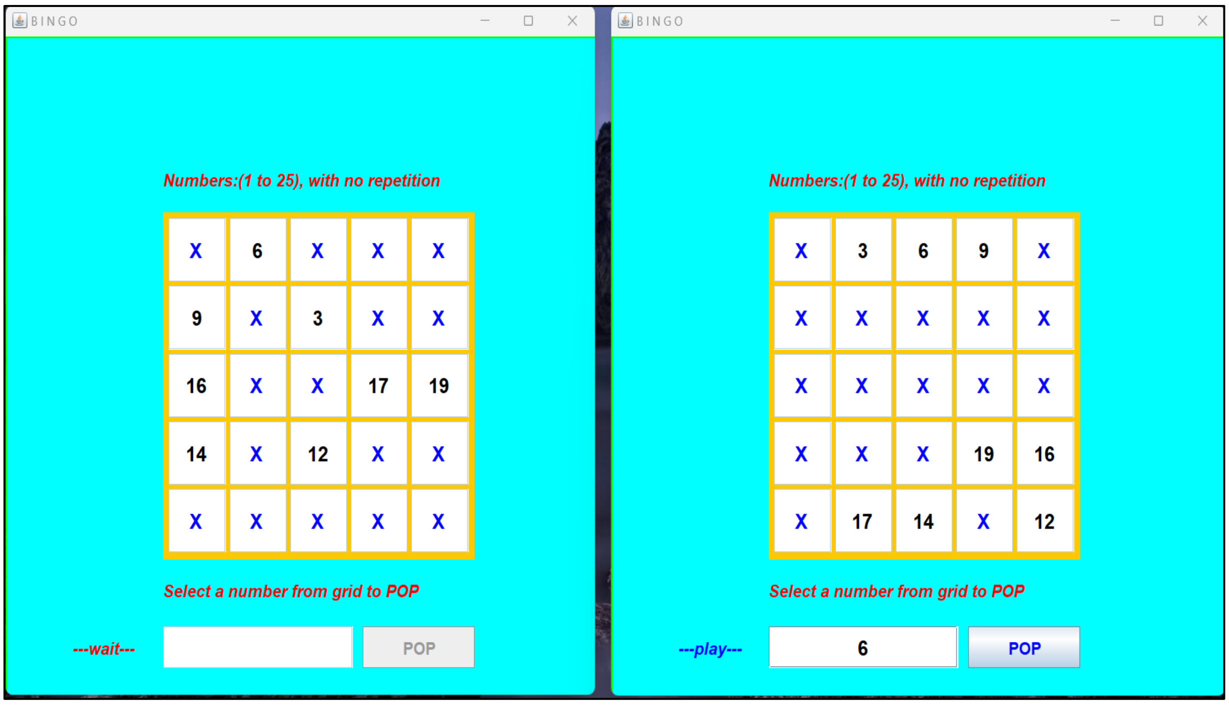
```
_____ME_____
Hi, this is room 2

___[ Bhanu ]___
Ok
```

```
___[ Vamsi ]___
Hi, this is room 2

_____ME_____
Ok
```

5	6	X	7	8
9	X	3	X	X

9	8
X	24

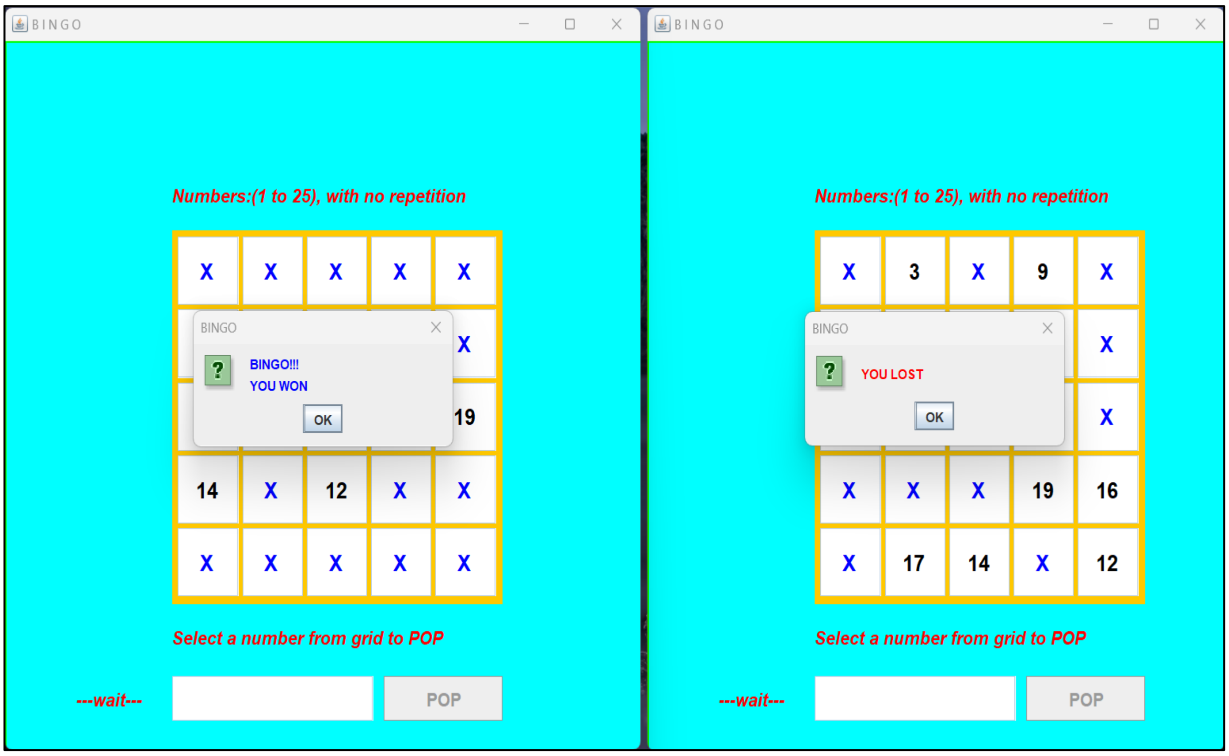
Fig 3.16 One step to finish game, if player2 pop out 6 from the grids then player1 will get 5 lines of empty blocks and player2 gets 4 lines,


BINGO
Numbers:(1 to 25), with no repetition
Select a number from grid to POP
---wait---
POP
BINGO
Numbers:(1 to 25), with no repetition
Select a number from grid to POP
---play---
6
POP

X	6	X	X	X
9	X	3	X	X
16	X	X	17	19
14	X	12	X	X
X	X	X	X	X

X	3	6	9	X
X	X	X	X	X
X	X	X	X	X
X	X	X	19	16
X	17	14	X	12

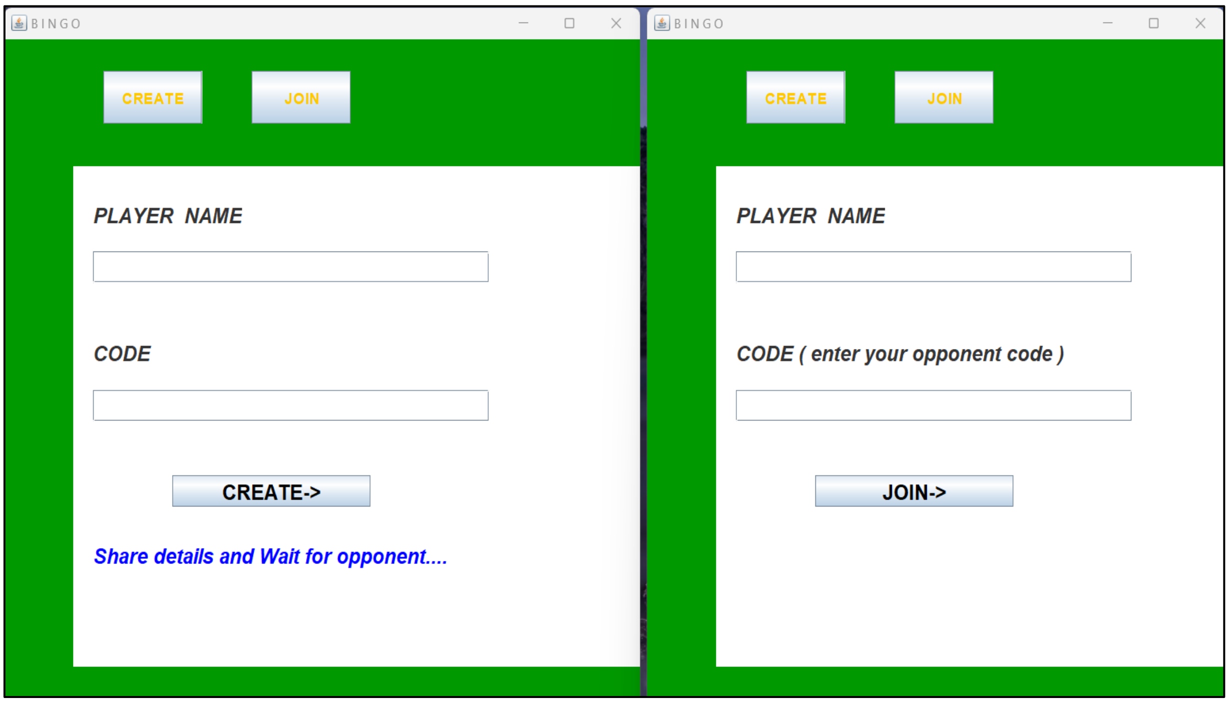
Fig 3.17 Player 1 won the game and player 2 lost the game


BINGO
Numbers:(1 to 25), with no repetition
BINGO
OK
Select a number from grid to POP
---wait---
POP
BINGO
Numbers:(1 to 25), with no repetition
BINGO
OK
Select a number from grid to POP
---wait---
POP

BINGO!!!
YOU WON

YOU LOST

Fig 3.18 Again moves to first page



Share details and Wait for opponent....

CHAPTER 4
CONCLUSION

We have created BINGO game TCP/IP socket programming that connects two players from different devices, uses internet to run. Any numbers of players can be connected to server. Players are able chat while playing. Data transfer occurs only within the room, IP address and port number of the server are known to the players.

This application uses java socket programming, TCP and IP protocol.

This BINGO game is implemented and run successfully.

Features that can be added to improve the game:

- Number of players in each room can be increased
- This game can be made available via web application
- We can have a login and password details storing in database and tracing the win streak of every player
- Quick chat can be added
- Using the database, player can directly send a request to another player using their username to play

CHAPTER 5

REFERENCES

Kenneth L. Calvert and Michael J. Donahoo TCP/IP Sockets in Java: Practical Guide for Programmers, Second Edition

Herbert Schildt, *The Complete Reference JAVA*, Tata McGraw Hill, Eleventh Edition, 2019.

Multi-threading in java: https://www.geeksforgeeks.org/multithreading-in-java/

Socket programming in java: https://www.geeksforgeeks.org/socket-programming-in-java/