TIC-TAC-TOE USING TCP/IP SOCKET PROGRAMMING IN JAVA

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

CSE 302: COMPUTER NETWORKS

Submitted by

DORNADULA VENKATA SAI HEMANTH

(Reg.No.:124003415 , B.Tech.Computer Science and Engineering)

December 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA - 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

Bonafide Certificate

This is to certify that the report titled **"Tic-Tac-Toe Using TCP/IP 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 **Shri. DORNADULA VENKATA SAI HEMANTH (Reg. No.: 124003415, B.Tech. Computer Science and Engineering)** during the academic year 2022-23, in the School of Computing.

Project Based Work *Viva voice* held on ___________________

Examiner 1 **Examiner 2**

TABLE OF CONTENTS

LIST OF FIGURES

ABREVATIONS

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

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 providing me an opportunity to do this project and for his guidance and support to successfully complete the project and also for his constant support, motivation and academic help extended for the past three years of my life in School of Computing.

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 and for successful completion of all courses during my academic life at SASTRA Deemed University.

Finally, I thank my parents and all others who help me acquire this interest in project and aided me in completing it within the deadline without much struggle.

ABSTRACT

socket programming is a approach of connecting 2 nodes on a network to communicate with one another.one socket(node) listens on a selected port at associated ip whereas the opposite socket reaches to other to make a connection. The server forms the listener socket while client reaches out to server.

The TCP/IP stands for Transmission control protocol/Internet protocol. It is a concise version of OSI model. It is a protocol that specifies how data is exchanged over internet that specifies how it should be broken as packets,segments,transmitted and recieved at destination.

online multiplayer games become very favourite mostly for the kids and teenagers in these days. In the free time they play these games for their relaxation. Also during the covid time these gained higher importance.

Normally these multiplayer games are played used LAN Connection. when playing multiplayer game you access a client , your software translates your control to central server via your internet then it runs and returns outcome to respective player. Socket Programming is used in interconnection which used TCP/IP transport protocol.

TIC-TAC_TOE is turn based game played by two players. It contains 3x3 grid in which players have to fill it with 'O' or 'X'. The one who fills their marks consecutively in a row or column or diagonal is the winner. So For the communication between server and 2 clients(players) we use socket programming where 2 players are in distant places.

KEYWORDS : TCP/IP , Socket Programming , Client and Server , transport protocol

CHAPTER 1

INTRODUCTION

1.1 COMPUTER NETWORKING

A set of interconnected computer, also called as hosts that uses network such as internet for exchanging resource is called computer network .

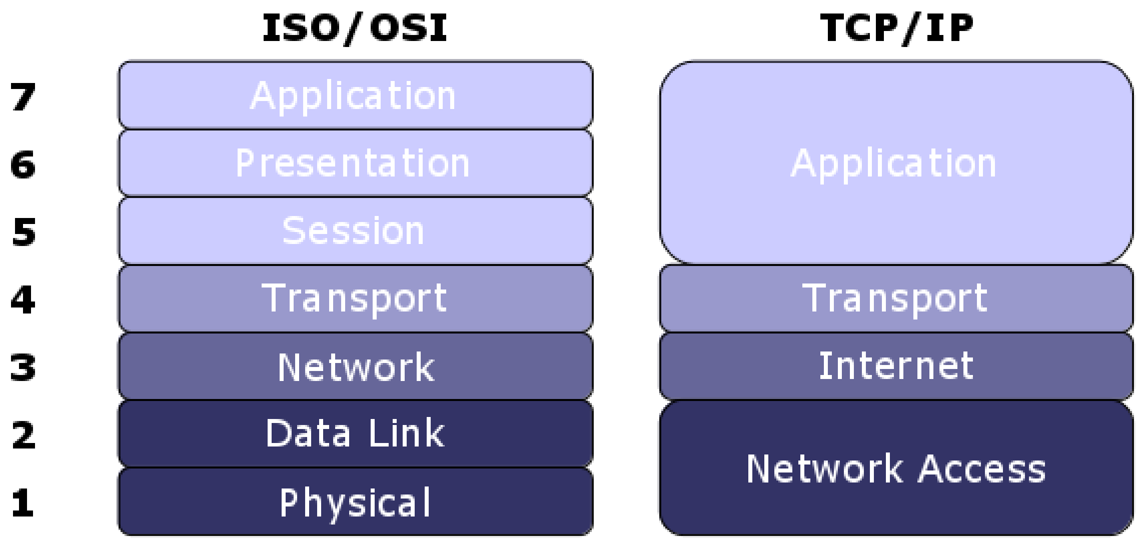
1.2 OSI MODEL

OSI stands for open systems interconnection. It has been developed by ISO. It is a framework that is used to describe the functions of networking system. It describes how data is passed from application in one computer to another computer.

The OSI model consists 7 layers. They are Application , Presentation ,session , Transport , Network ,Data link and Physical layer.

When data is going from application layer to lower layers each layer adds it services to the services provided by higher layer to control communication. All these layers has a specific task and each layer is independent.

The OSI model is a reference Model on how system talk to each other. But actually we don't use this in real life, but we use a similar model called as TCP/IP model. In TCP/IP model the 7 layers of OSI model is converted to 4 layers. 1,2 layer joined as Network access layer, layer 3 known as Internet layer,4 remaining as it is, layer 5,6,7 joined as one layer named Application layer.

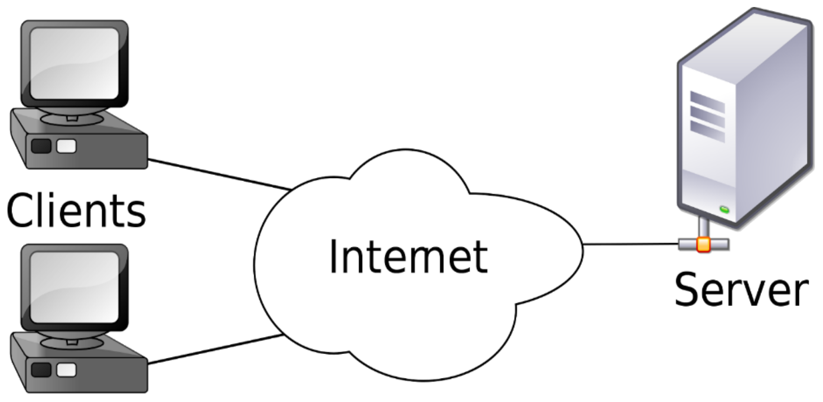


1.3 TRANSMISSION CONTROL PROTOCOL (TCP)

Tcp is a Transport layer protocol. It is connection oriented protocol i.e. it establishes connection prior to communication between the devices. It organizes data that is transmitted between source and Destination. It takes the messages from application layer, segregates them in to packets and provides numbering for sending them to destination.

1.4 TCP CLIENT AND SERVER

Client server architecture is a computing model in which Server is responsible for handling the requests from several clients and it also provides response messages with respect to requests for each client. The server act as central point for many clients. The client sends request for data through internet / local IP address and server accepts it. Then it sends packets of data that are requested.



In this project the two players act as clients and server is locally hosted on computer. Any two players can play the game from any region can connect using IP address and can play the game.

1.5 SOCKET PROGRAMMING

socket programming is a approach of connecting 2 nodes on a network to communicate with one another.one socket(node) listens on a selected port at associated IP whereas the opposite socket reaches to other to make a connection . The server forms the listener socket while client reaches out to server.

It shows how to use socket APIs for communication between remote and local process. Socket is first created, related to a kind of transport layer(ex : TCP) and OSI model is employed with IP address and port number .Then they are used to send or receive data.

The client sends request for connection to the server and server accepts connection if there are no errors.

CHAPTER 2

PROPOSED FRAMEWORK

Tic-Tac-Toe is a multiplayer game where TCP/IP protocol is used to establish connection between client and server and uses Socket programming. This is a turn based 2 player game where 2 players are clients interact through server that act as a medium between them. The 2 players may be on the same computer or different computers that are connected using IP address.

Intially TCP/IP protocol is used for connection between 2 players (i.e., clients) and we use sockets to connect them. When one player makes a move that move is sent to server which is then sent to other player as a opponent move who is already connected. As Each player makes a move that move is recorded and reflected back , this process until either of them wins or is a draw.

A player is declared as a winner if his mark is present in all bocks of appropriate row/ column / cross diagonals. We also included a timer i.e each player gets 10 seconds of time to make their move .If a player fails to make his move then other player is declared as winner and the game exits. The game ends in a draw if all the blocks are filled and no one wins. Untill both players decided to exit the score of each of them is updated. Here we use a function to block if a player tries to make his move more than once consecutively.

Client

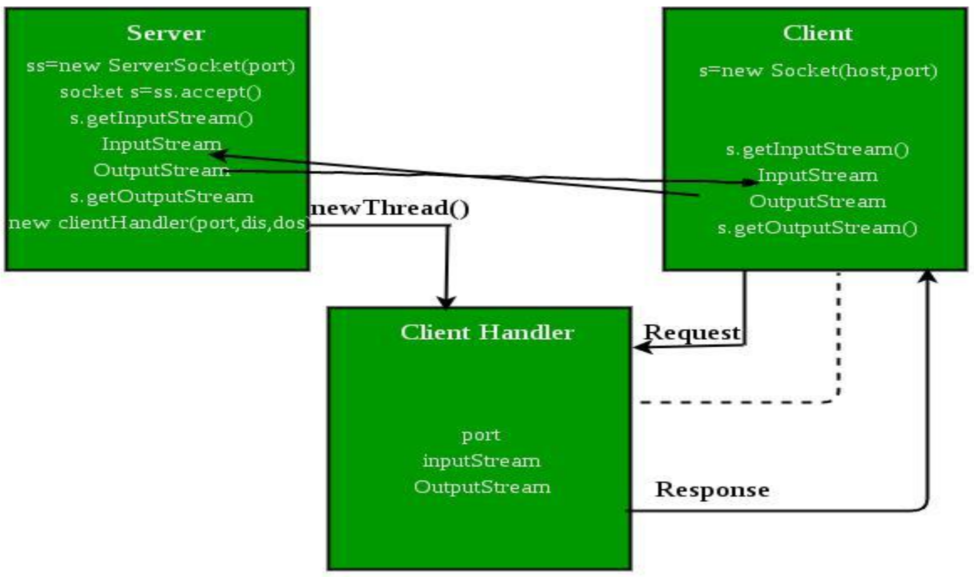
The player i.e. the client connects to server using TCP and connection is implemented and secure if server accepts the connection. #player_1 is named to the one who connected first to server and the other player is named as #player_2. No more clients are accepted after both players are connected. “X” mark is given to player 1 and “O” is given to player 2. For one player blocks the colour is Green and for other it is red. whenever a player completes his move his timer starts and waits for other player to make his turn . If other player fails to make his move then current player awarded as winner and exits from game. After 10 sec if other player wants to make his move then he will be informed as timeout using pop up window and the game exits.

For restarting the game or exit, a window pop up is used. The game terminates if any player does not restart / exit. Then server also shuts down.

Server

TCP protocol is used for server which is written in java. TCP is most used protocol and uses client server architecture. For both client and server port number is fixed. Here the server does only one job that is transmitting data(move) of player 1 to player 2   and vice versa. In this way data is sent through a loop. When any client got disconnected or if IO Exception occurred then loop breaks.

Since Tic-Tac-Toe is 2 player game, Code allows only 2 clients to connect to the server at once to play the game. Both the players can play either using localhost or IP address. We can get the IPv4 , IPv6 addresses using IPCONFIG command which can be used to connect to server and play.



In Java we use the Socket class by importing java.net package.

Some important methods in Socket class are :

public socket accept() : returns socket and establishes connection between server and client.

Public synchronized void close() : closes server socket

Creating server socket :

Serversocket ss = new ServerSocket(9696);

Socket s = ss.accept(); //waits for the client

Creating Client Socket :

Socket s = new Socket(“localhost”,9696);

Tic-Tac-Toe rules

1. Game is played on 3 X 3 grid. One player is assigned to “X” and other with “O”.Both have to mark the empty squares in turns.

2. Winner is the one who has 3 of their marks in entire row or column or diagonally. The game is completed when all Squares i.e. all 9 squares are filled. The game is tied if no player has who has 3 of their marks in entire row or column or diagonally.

3. Each player is given 10 seconds of time to make their move. If he fails then other player is declared as a winner and game completes.

CHAPTER 3

SOURCE CODE

4.1 server source code:

```
import java.util.*;

import java.net.*;

import java.io.*;


public class Server

{

  private int playerCount;

  private int player1val;

  private int player2val;

  private clienthandler player1;

  private clienthandler player2;

  private ServerSocket ss;


  Scanner sc = new Scanner(System.in);

  public Server()

  {

    playerCount = 0;

    try

    {

      ss  = new ServerSocket(26771);

        //creates Server

      }

      catch(IOException ex)

      {

         System.out.println("IOException in server Constructor");

    }

  }

private void start()
```

```
{
  try
  {
   System.out.println("Waiting for clients to connect with the
server.......");
   while (playerCount < 2)  //accepts 2 clients
   {
     Socket s = ss.accept();
     playerCount++;
     System.out.println("player #"+ playerCount + " is connected." );
     clienthandler cc = new clienthandler(s,playerCount);
     if(playerCount == 1)
     {
       player1 = cc;
          //if 1st client then player 1
     }
     else
     {
       player2 = cc;
          //if 2nd client then player 2
     }
     Thread t = new Thread(cc);
     t.start();
        //thread starts
   }
   System.out.println("No more connections accepted.(both the players have
joined)");
   //No more than 2 clients(players) accepted
 }
 catch (IOException ex)
 {
  System.out.println("IOException while initiating connection, please try
again.");
 }
```

```
}

private class clienthandler implements Runnable
{
  private Socket socket;
  private DataInputStream Din;
  private DataOutputStream Dout;
  private int playerID;
  clienthandler (Socket s, int id)
  {
    socket = s;
      //Socket created
    playerID = id;
      //player ID is assigned
    try
      {
      Din = new DataInputStream (socket.getInputStream());
        Dout = new DataOutputStream(socket.getOutputStream());
    }
    catch(IOException ex)
      {
      System.out.println("IOException in clienthandler");
    }
  }
  public void run()
  {
    try
      {
      Dout.writeInt(playerID);
        Dout.flush();
      while(true)
        {
```

```
        if(playerID == 1)
            {
          player1val = Din.readInt();
          System.out.println("player1 clicked "+ player1val);
               //displayed in command prompt as to which button player 1
clicked
          player2.sendButtonVal(player1val);
               //player 1 button sent to player 2 and it's reflected in player
2 GUI also


        }
        else if(player1.socket.isClosed() || player2.socket.isClosed())
            {
            System.out.println("Connection lost !! Please try again.");
                   //connection lost
            break;
        }
        else
            {
           player2val = Din.readInt();
           System.out.println("player2 clicked "+ player2val);
           player1.sendButtonVal(player2val);
        }
      }
    }
    catch(Exception ex)
      {
      player1.closeConnection();
      player2.closeConnection();
    }
  }
  //used to send value to Client
  public void sendButtonVal(int n)
  {
     try
       {
      Dout.writeInt(n);
        Dout.flush();
      System.out.println("value succesfully written back to client");
        //to verify that value successfully written in GUI for both players
     }
       catch(IOException ex)
       {
      System.out.println("IOException in sending value to opponent");
     }
  }
  //closes Server side connection
  public void closeConnection()
  {
    try
      {
     socket.close();
     System.out.println("Connection successfully closed.");
    }
    catch(IOException ex)
      {
     System.out.println("IOException while closing the connection");
    }
  }
}
  public static void main(String args[])
  {
    Server server = new Server();
    server.start();
  }}
```

4.2 Client side code:

```
// Game Client Side

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

public class Client extends JPanel
{
  private char playerMark;
  private int playerID;
  private int otherPlayer;
  private int x;
  private int currentplayerscore;
  private int OppplayerScore;
  private boolean setButtons;
  private JButton[] buttons = new JButton[9];
  private clientConnection CC;
  private JFrame window;
  private Timer timer;
  Scanner sc = new Scanner(System.in);
  public Client()
  {
      //intializing values
    setButtons = false;
      x = 0;
    playerID = 0;
    otherPlayer = 0;
    currentplayerscore = 0;
    OppplayerScore = 0;
  }
  public void buildGui()
  {
      //cretes GUI for client
   window = new JFrame("player #" + playerID);
   window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   window.getContentPane().add(this);
   window.setBounds(450,450,450,450);
   window.setLocationRelativeTo(null);
   setLayout(new GridLayout(3,3));
   //setting values of Player ID for both players
   if(playerID == 1)
   {
     otherPlayer = 2;
     setButtons = true;
     toggleButtons();
   }
   else
   {
     otherPlayer = 1 ;
     setButtons = false;
     toggleButtons();
       Thread t2 = new Thread(new Runnable()
       {
       public void run()
         {
          updateTurn();
       }
     });
     t2.start();
   }
   window.setVisible(true);
 }
public void initializeButtons()
{
   for(int i = 0; i <= 8; i++)
   {
     buttons[i] = new JButton();
     buttons[i].setText(" ");
     buttons[i].setBackground(Color.WHITE);
     add(buttons[i]);
   }
}
//button clicked then this function will invoke
public void keyPress()
{
   ActionListener al = new ActionListener()
   {
      public void actionPerformed(ActionEvent ae)
        {
         JButton buttonClicked = (JButton) ae.getSource();
         buttonClicked.setText(String.valueOf(playerMark));
         buttonClicked.setBackground(Color.GREEN);
         for(x = 0 ; x<9 ; x++)
         {
                   //to find out which button clicked
            if(buttonClicked == buttons[x])
              {
                break;
            }
         }
```

```
         CC.sendButtonVal(x);//sending that button value to server
         System.out.println("value sent to server");
         setButtons = false;
         toggleButtons();
         displaywinner();
         System.out.println("You Clicked button no "+ x+" Wait for Player #" +
otherPlayer);
         Thread t = new Thread(new Runnable()
           {
            public void run()
              {
                updateTurn();
            }
         });
         t.start();
        }
    };
    for(int i=0;i<9;i++)
      {
       buttons[i].addActionListener(al);
    }
}
//after sending button value waits for Server to recieve opp move
public void updateTurn()
{
      //as soon as move is done timer starts
      timer = new Timer();
    TimerTask task = new TimerTask()
       {
          public void run()
            {
              System.out.println("Time out !!");
            JOptionPane.showMessageDialog(window,"Other player left You won
!!");
            CC.closeConnection();
          System.exit(0);
          }
        };
    timer.schedule(task,15000);
    int n = CC.recButtonVal();
    oppMove(n);
      //recieves opponent move and reflected in GUI and Command Prompt
    System.out.println("Opponent feedback has been written");
    setButtons = true;
    toggleButtons();
      timer.cancel();
      timer.purge();
      //within 10 sec if other player makes his move then timer stops
```

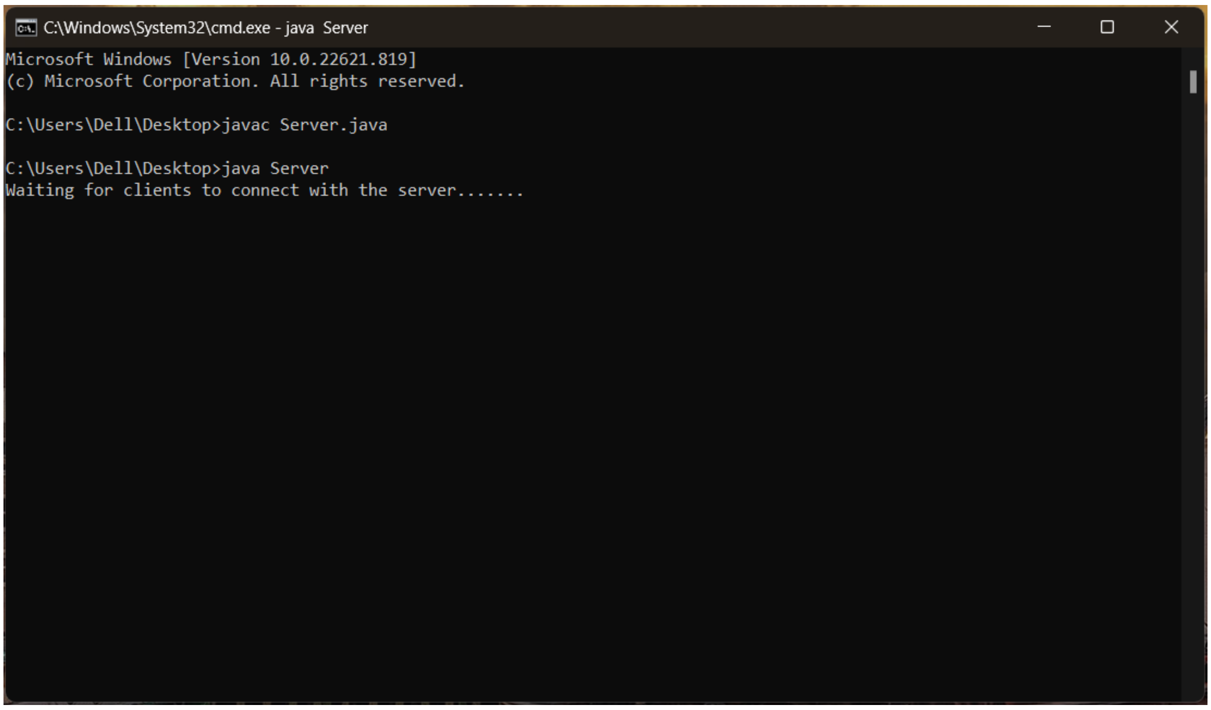
```
}
public void toggleButtons()
{

   if(setButtons == false)
   {
     for (int i=0;i<9;i++)
       {
        buttons[i].setEnabled(setButtons);
     }
   }
   else
   {
      for(int i =0;i<9;i++)
        {
        if(buttons[i].getText().charAt(0) == ' ')
             {
           buttons[i].setEnabled(setButtons);
        }
      }
    }
}
// To write opponent move
public void oppMove(int oppval)
{
   char oppPlayerMark;
   if (playerMark == 'X')
   {
      oppPlayerMark = 'O';
   }
   else
   {
      oppPlayerMark ='X';
   }
   buttons[oppval].setText(String.valueOf(oppPlayerMark));
   buttons[oppval].setBackground(Color.RED);
   buttons[oppval].setEnabled(false);
   displaywinner();
}
public void playerMarkDecider()
{
   if (playerID == 1)
   {
      playerMark = 'X';
   }
   else
   {
      playerMark = 'O';
   }
}
public void UpdatePoints(char x)
{
  if(x == playerMark)
  {
    currentplayerscore++;
  }
  else
  {
    OppplayerScore++;
  }
}
//checks and displays winner
public void displaywinner()
{
  if(checkwinner('X') == true)
  {
      timer.cancel();
      timer.purge();
    int dialogResult = JOptionPane.showConfirmDialog(window, "X wins. \n
you(#1):"+currentplayerscore+" \t opponent(#2):"+OppplayerScore+" \n Would you
like to play again?","Game over.",JOptionPane.YES_NO_OPTION);
    if(dialogResult == JOptionPane.YES_OPTION)
    {
            resetbuttons();
      }
    else
      {
      CC.closeConnection();
      System.exit(0);
      }
  }
  else if(checkwinner('O') == true)
  {
      timer.cancel();
      timer.purge();
    int dialogResult = JOptionPane.showConfirmDialog(window, "O wins. \n
you(#1):"+currentplayerscore+" \t opponent(#2):"+OppplayerScore+" \n Would you
like to play again?","Game over.",JOptionPane.YES_NO_OPTION);
     if(dialogResult == JOptionPane.YES_OPTION)
       {
            resetbuttons();
       }
     else
       {
        CC.closeConnection();
        System.exit(0);
      }
   }
   else if(checkDraw())
   {
        timer.cancel();
        timer.purge();
     int dialogResult = JOptionPane.showConfirmDialog(window,"Draw \n
you(#1):"+currentplayerscore+" \t opponent(#2):"+OppplayerScore+" \n Would you
like to play again?","Game over.", JOptionPane.YES_NO_OPTION);
     if(dialogResult == JOptionPane.YES_OPTION)
     {
             resetbuttons();
     }
     else
       {
        CC.closeConnection();
        System.exit(0);
       }
   }
}
private void resetbuttons()
{
   if(playerMark == 'O')
   {
     playerMark = 'O';
     setButtons = true;
   }
   else
   {
     playerMark = 'X' ;
     setButtons = false;
   }
   toggleButtons();
   for(int i =0;i<9;i++)
   {
      buttons[i].setText(" ");
      buttons[i].setBackground(Color.WHITE);
   }
}
public boolean checkDraw()
{
    boolean full = true;
    for(int i = 0 ; i<9;i++)
    {
      if(buttons[i].getText().charAt(0) == ' ')
        {
       full = false;
      }
   }
   return full;
}
//checks for winner
public boolean checkwinner(char x)
{
  if(checkRows(x) == true || checkColumns(x) == true || checkDiagonals(x)
==true)
  {
     return true;
  }
  else
  {
       return false;
  }
}
//checks rows for a win
public boolean checkRows(char x)
{
   int i = 0;
   for(int j = 0;j<3;j++)
   {
     if( buttons[i].getText().equals(buttons[i+1].getText()) &&
      buttons[i].getText().equals(buttons[i+2].getText()) &&
buttons[i].getText().charAt(0) != ' ' &&
      buttons[i].getText().charAt(0) == x)
        {
        UpdatePoints(x);
        return true;
        }
      i = i+3;
   }
   return false;
}
//checks columns for a win
public boolean checkColumns(char x)
{
   int i = 0;
   for(int j = 0;j<3;j++)
   {
     if( buttons[i].getText().equals(buttons[i+3].getText()) &&
        buttons[i].getText().equals(buttons[i+6].getText())&&
buttons[i].getText().charAt(0) != ' ' &&
        buttons[i].getText().charAt(0) == x)
      {
        UpdatePoints(x);
        return true;
        }
     i++;
   }
   return false;
}
//checks diagonals for win
public boolean checkDiagonals(char x)
{
   if(buttons[0].getText().equals(buttons[4].getText()) &&
     buttons[0].getText().equals(buttons[8].getText())&&
buttons[0].getText().charAt(0) !=' ' && buttons[0].getText().charAt(0) == x)
    {
      UpdatePoints(x);
      return true;
    }
   else if(buttons[2].getText().equals(buttons[4].getText()) &&
buttons[2].getText().equals(buttons[6].getText())&&
buttons[2].getText().charAt(0) !=' ' && buttons[2].getText().charAt(0) == x)
   {
     UpdatePoints(x);
     return true;
   }
   else return false;
}
public void connectToserver()
{
    CC = new clientConnection();
}
private class clientConnection
{
   private Socket socket;
   private DataInputStream Din;
   private DataOutputStream Dout;
   private String ip;
   public clientConnection()
   {
     try
       {
       System.out.println("Client side.");
         System.out.println("Enter IP address:");
       ip = sc.nextLine();
       socket = new Socket(ip,26771);
       Din = new DataInputStream(socket.getInputStream());
         Dout = new DataOutputStream(socket.getOutputStream());
         playerID = Din.readInt();
       System.out.println("player # " + playerID + "is connected to server");
       }
       catch (IOException ex)
       {
```

```
        System.out.println("IOException in client constructor");
             System.exit(0);
     }
  }
  public void sendButtonVal(int a)
  {
    try
      {
       Dout.writeInt(a);
       Dout.flush();
       System.out.println("value sent to server !");
    }
      catch(IOException ex)
      {
          //Timeout happened and player tries to make move after timeout then
this window will be displayed
       JOptionPane.showMessageDialog(window,"Timeout You Lost
!!","Alert",JOptionPane.WARNING_MESSAGE);
       System.out.println("Timeout !!");
       CC.closeConnection();
       System.exit(0);
    }
  }
  //used to recieve value from server
  public int recButtonVal()
  {
    int n=-1;
    try
      {
      n = Din.readInt();
        System.out.println("Value received from server");
       }
       catch (IOException ex)
        {
          System.out.println("error in recButton()");
        }
      return n;
  }
  //closes client side connection
  public void closeConnection()
  {
   try
   {
      socket.close();
      System.out.println("Socket closed successfully");
   }
   catch(IOException ex)
   {
     System.out.println("IOException while closing");
   }
  }
}
  public static void main(String[] args)
  {
    Client cl = new Client();
    cl.connectToserver();
    cl.playerMarkDecider();
    cl.initializeButtons();
    cl.buildGui();
    cl.keyPress();
  }
}
```

CHAPTER 4

RESULTS

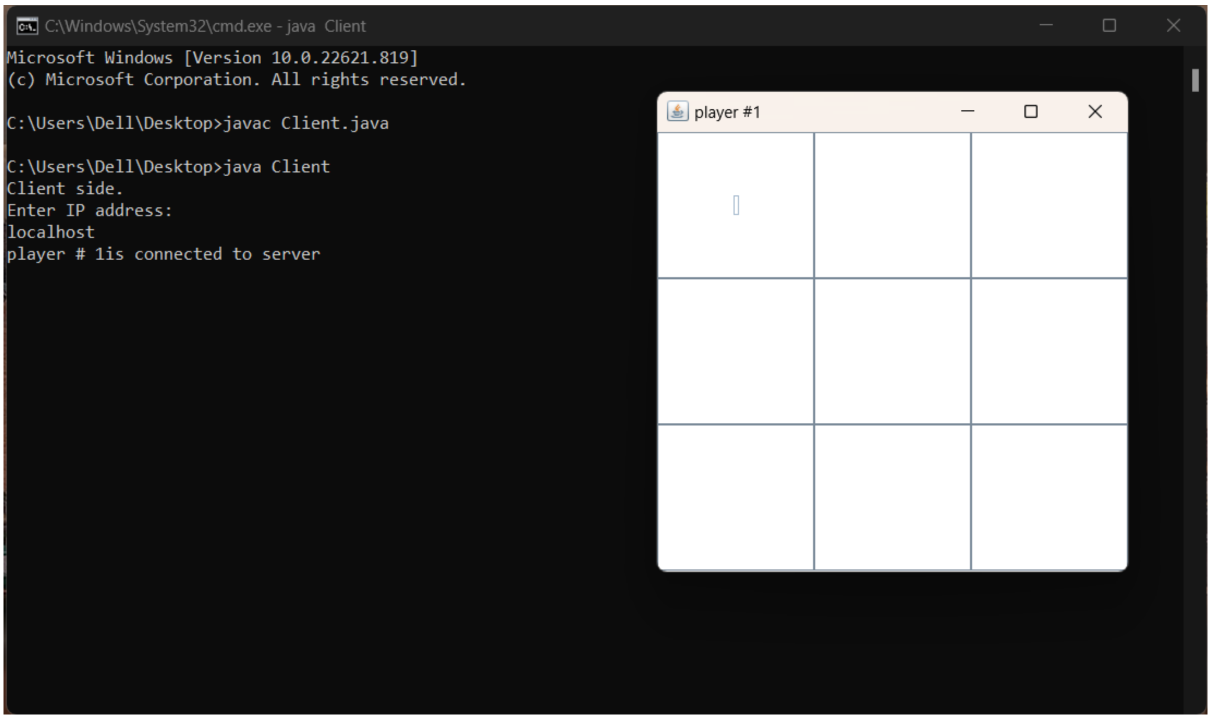
Fig 4.1 Server started



C:\Windows\System32\cmd.exe - java Server

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Server.java

C:\Users\Dell\Desktop>java Server
Waiting for clients to connect with the server.......
```

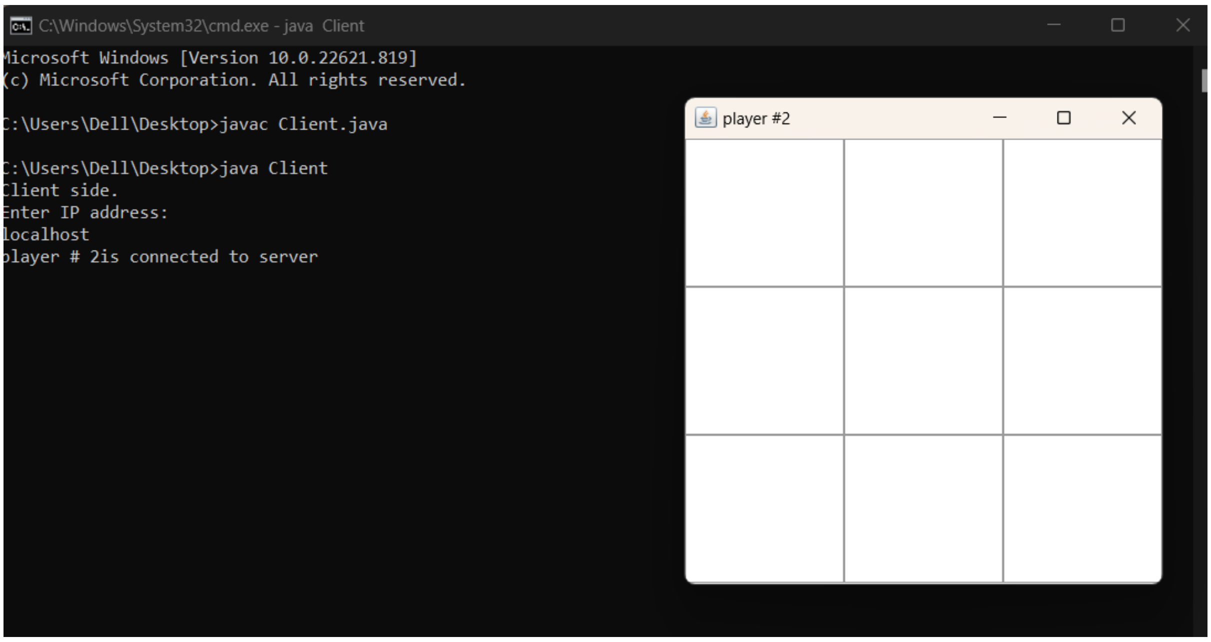
Fig 4.2 Client 1 joined using localhost



C:\Windows\System32\cmd.exe - java Client

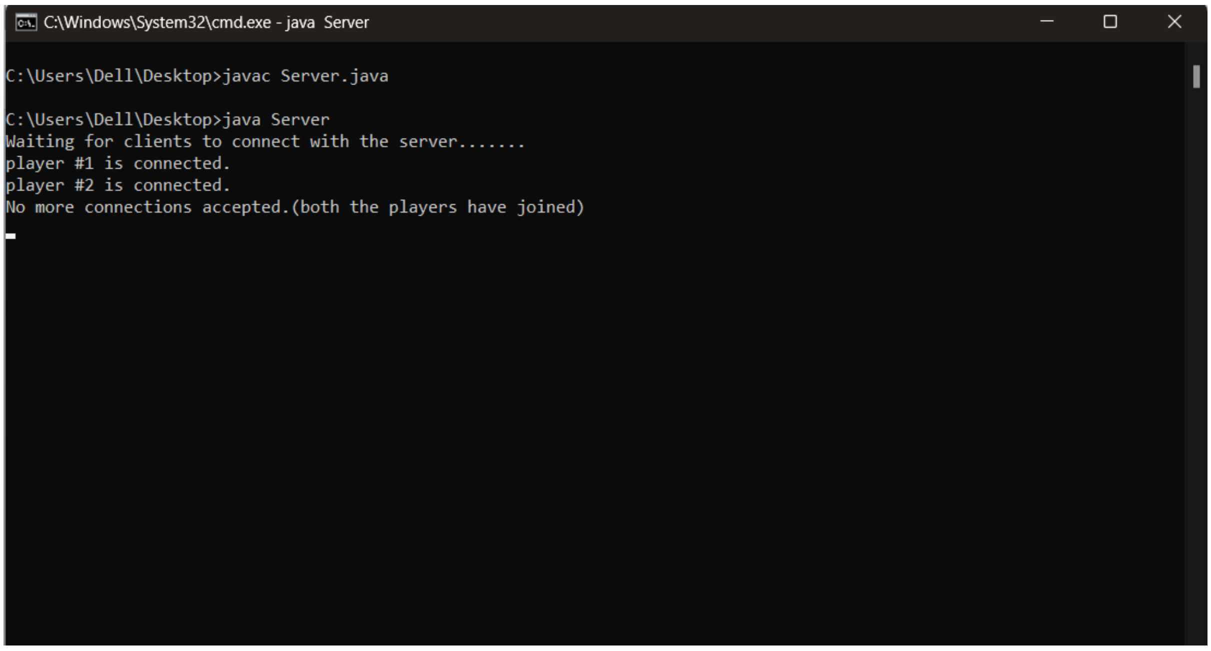
```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
```

Fig 4.3 Client 2 joined using localhost



```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
```

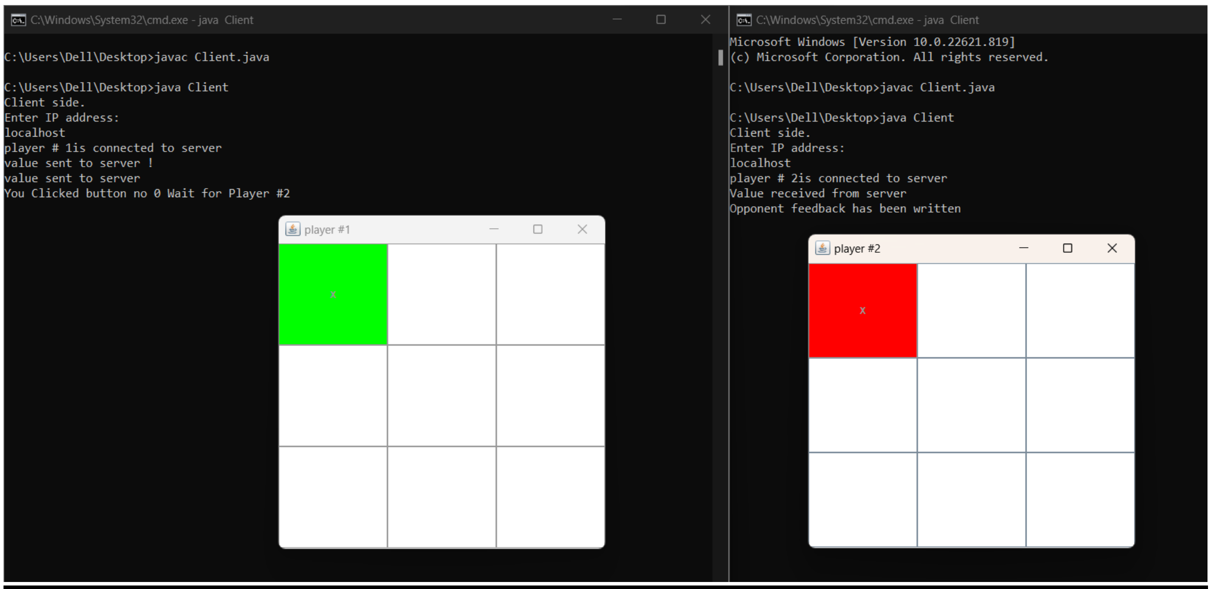
Fig 4.4 Server stops accepting anyomore clients (2 joined)



```
C:\Users\Dell\Desktop>javac Server.java

C:\Users\Dell\Desktop>java Server
Waiting for clients to connect with the server.......
player #1 is connected.
player #2 is connected.
No more connections accepted.(both the players have joined)
```

Player 1 wins("X" wins):

Fig 4.5: Input 1 from #player1

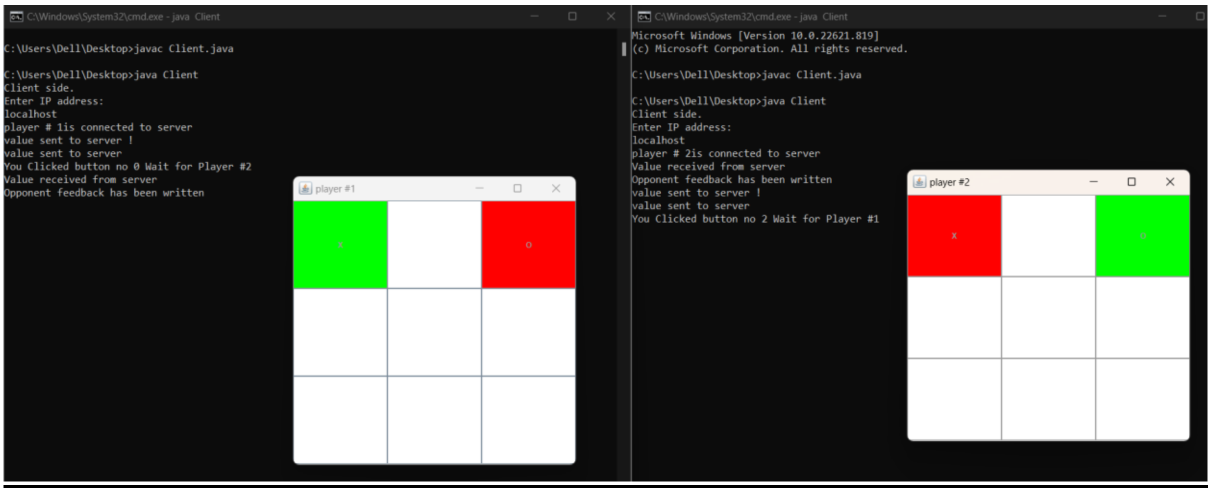


C:\Windows\System32\cmd.exe - java Client
player #1
x
C:\Windows\System32\cmd.exe - java Client
player #2
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
```

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

Fig 4.6 1st input from #player 2:

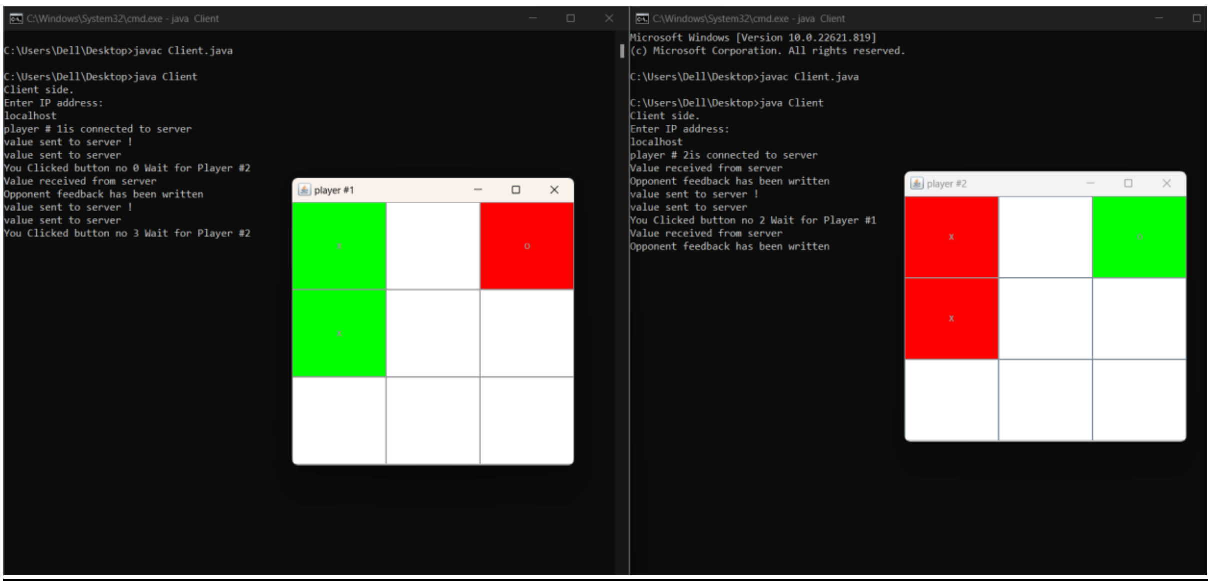


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
o

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 2 Wait for Player #1
```

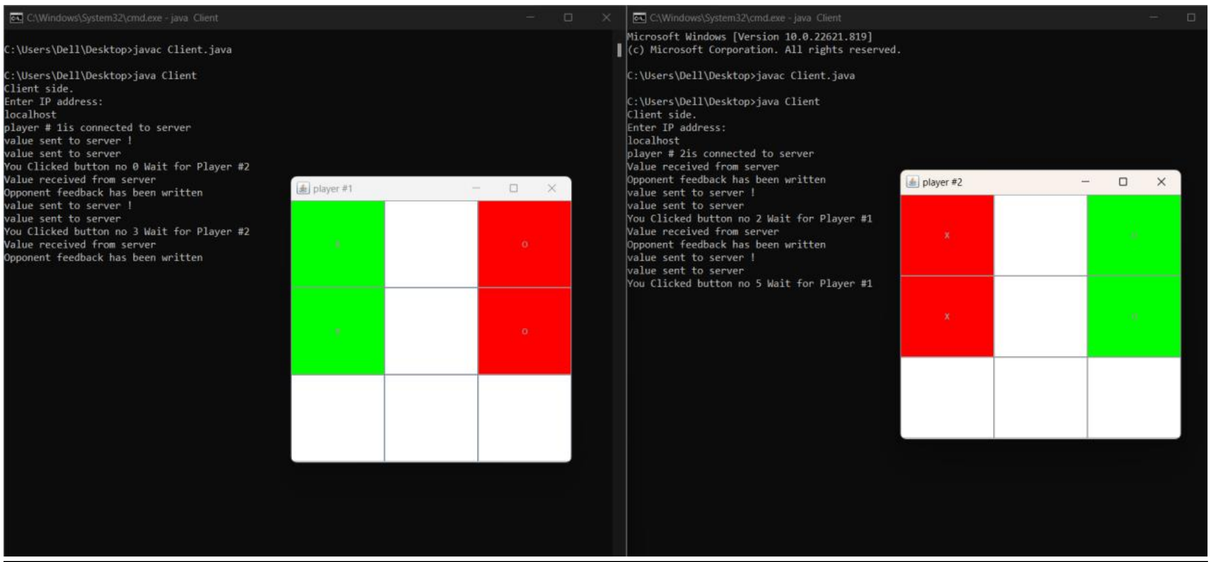
Fig 4.7 2nd input from #player 1:


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
x
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #2
```

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 2 Wait for Player #1
Value received from server
Opponent feedback has been written
```

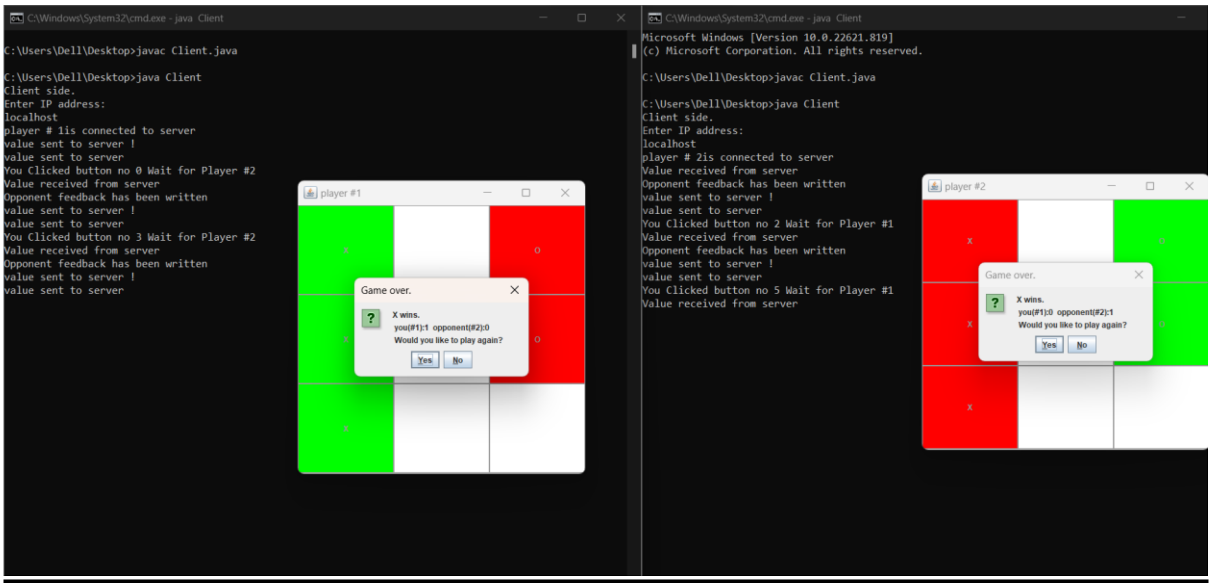
Fig 4.8: 2nd input from #player 2


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
x
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
x
o

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 2 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #1
```

Fig 4.9: 3rd input from #player 1



```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
```

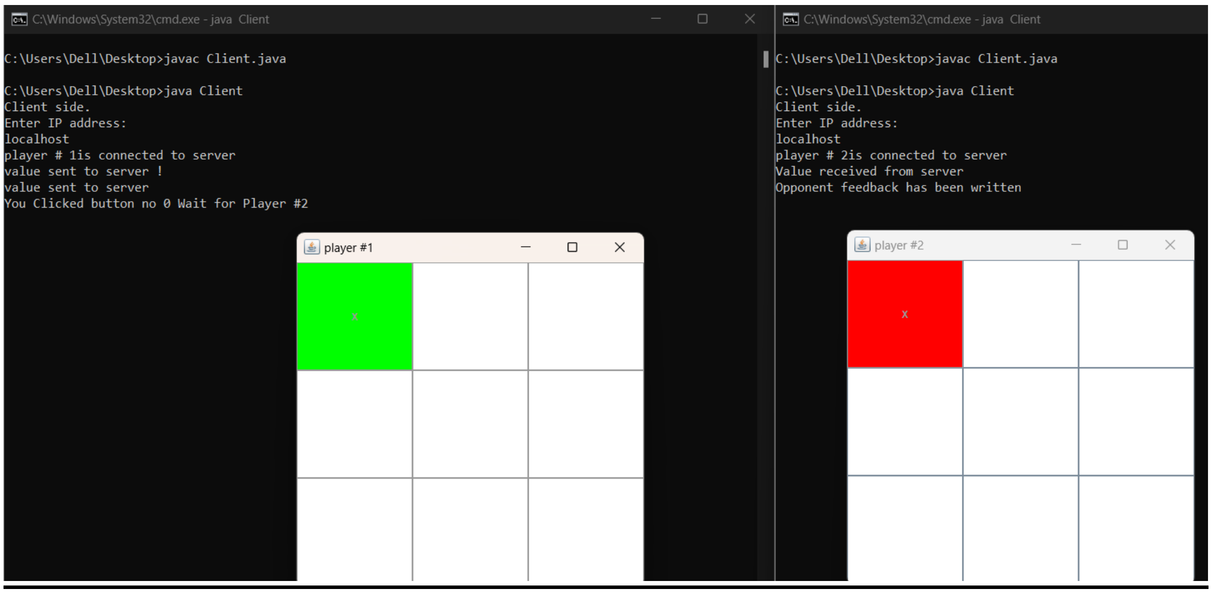
```
Game over.
X wins.
you(#1):1  opponent(#2):0
Would you like to play again?
Yes  No
```

```
Microsoft Windows [Version 10.0.22621.819]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 2 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #1
Value received from server
```

```
Game over.
X wins.
you(#1):0  opponent(#2):1
Would you like to play again?
Yes  No
```

Player 2 wins("O" wins):

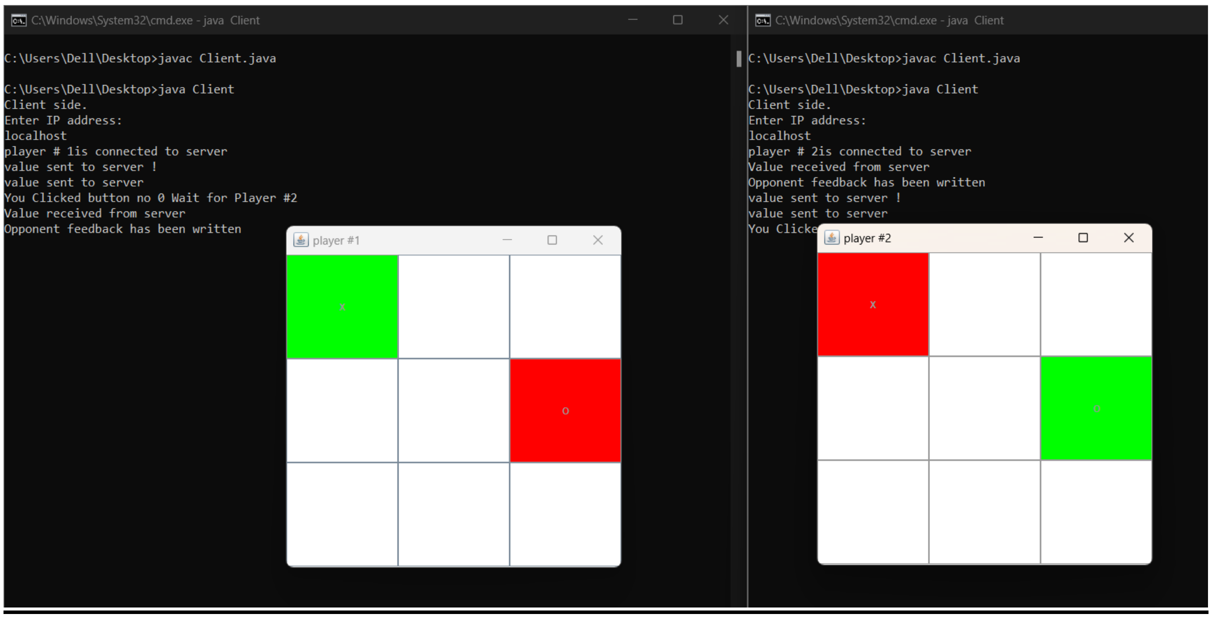
Fig 4.10 1st input from #player1



```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

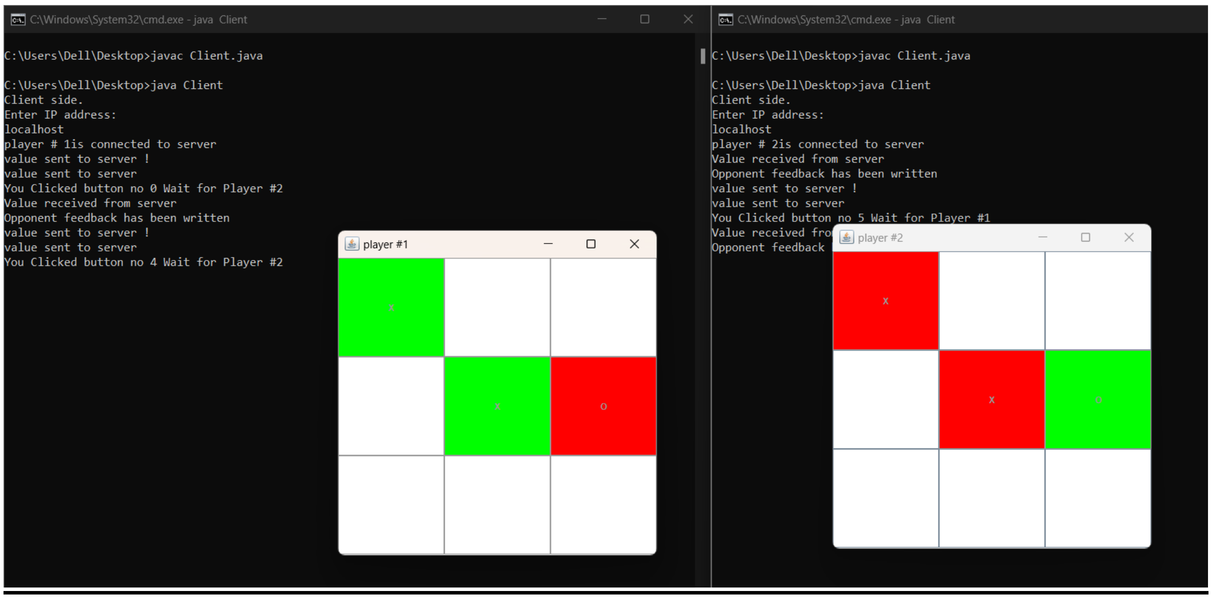
Fig 4.11:1st input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
o
C:\Windows\System32\cmd.exe - java Client
player #2
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicke
```

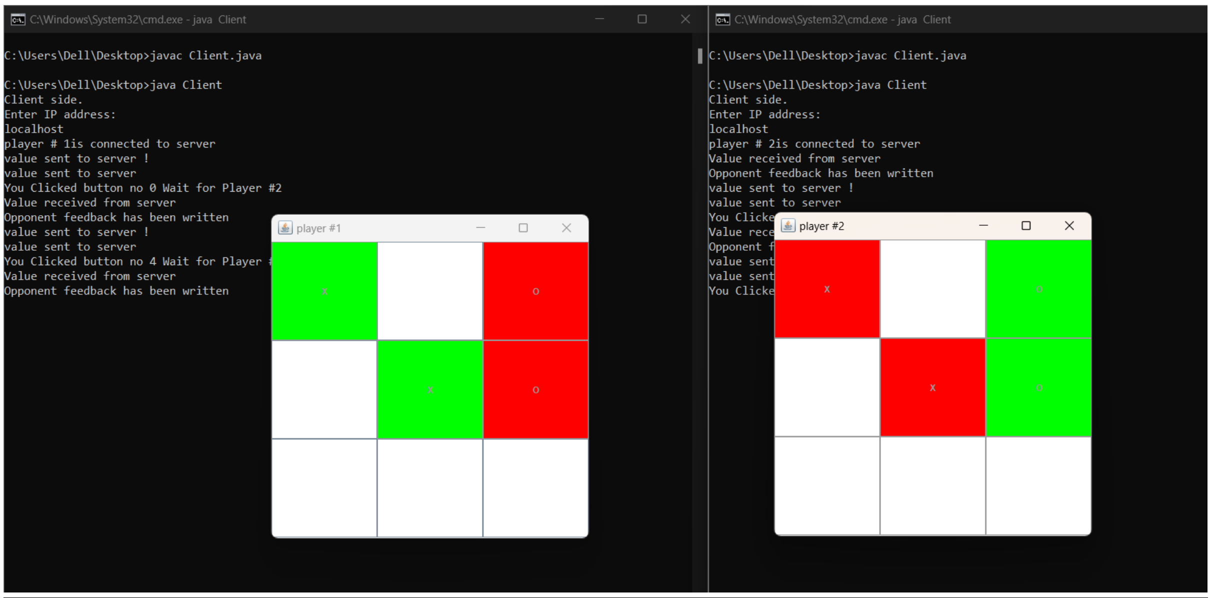
Fig 4.12: 2nd input from #player1


C:\Windows\System32\cmd.exe - java Client
player #1
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #1
Value received fro
Opponent feedback
```

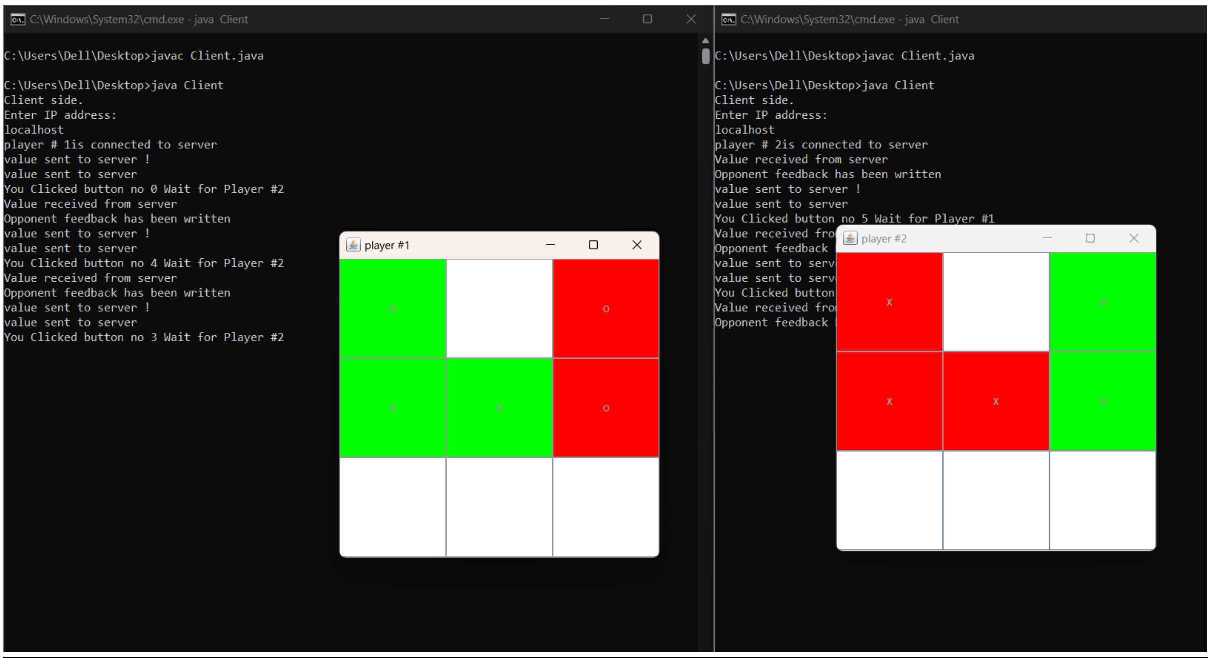
Fig 4.13: 2nd input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
x
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
x
o

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicke
Value rece
Opponent f
value sent
value sent
You Clicke
```

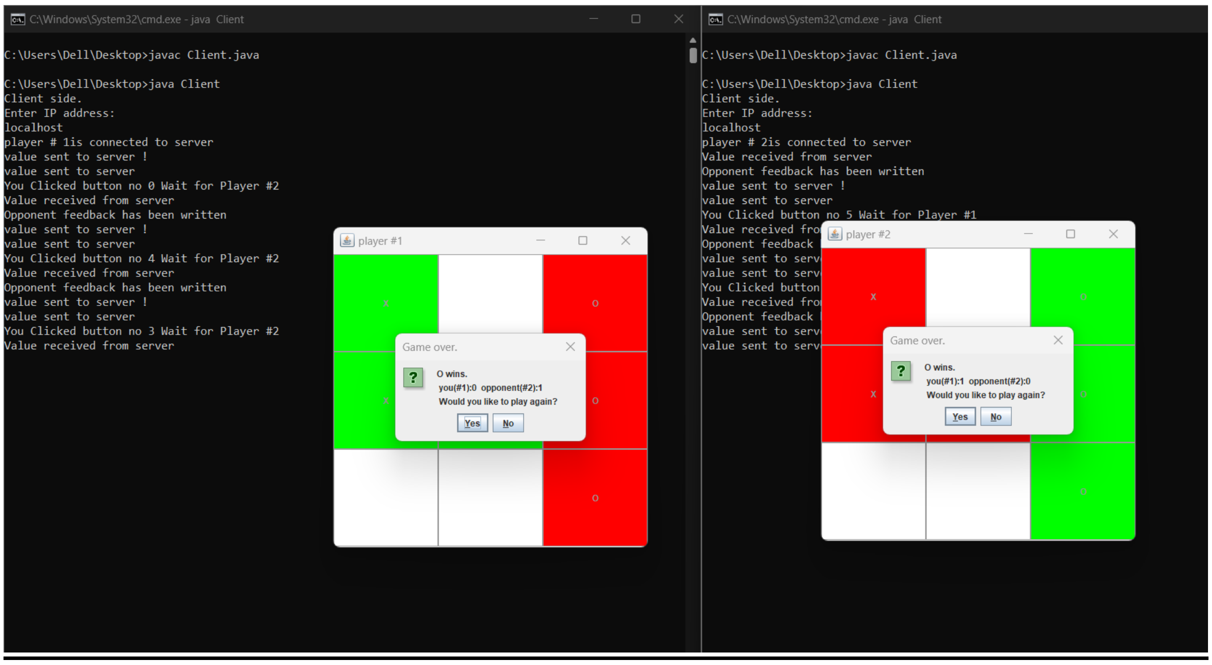
Fig 4.14: 3rd input from #player1:


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
x
x
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
x
x
o

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #1
Value received fro
Opponent feedback
value sent to serv
value sent to serv
You Clicked button
Value received fro
Opponent feedback
```

Fig 4.15: player 2 wins



```
C:\Windows\System32\cmd.exe - java  Client

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #2
Value received from server
```

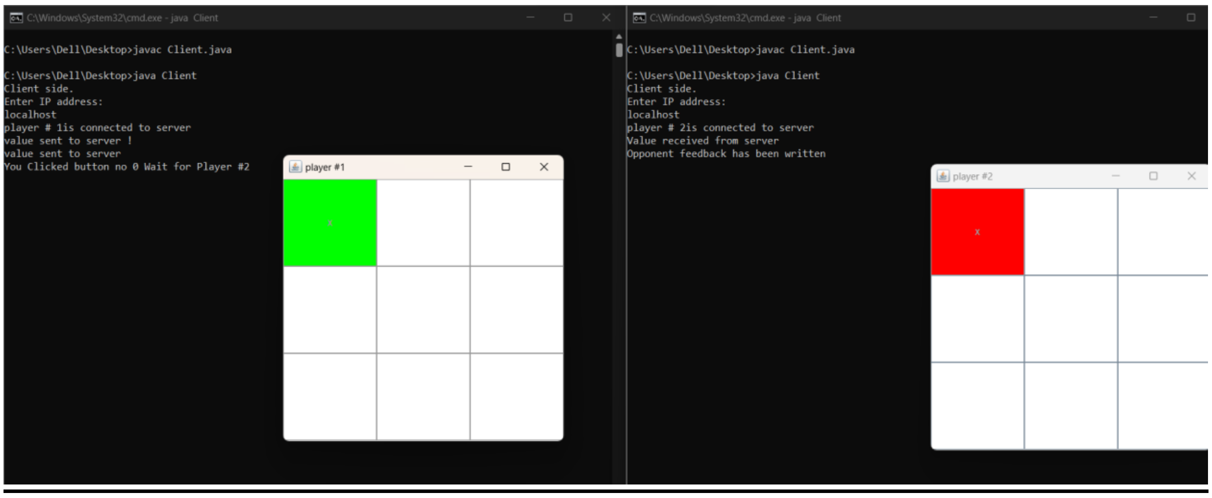
```
Game over.
O wins.
 you(#1):0  opponent(#2):1
 Would you like to play again?
```

```
C:\Windows\System32\cmd.exe - java  Client

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #1
Value received fro
Opponent feedback
value sent to serv
value sent to serv
You Clicked button
Value received fro
Opponent feedback
value sent to serv
value sent to serv
```

```
Game over.
O wins.
 you(#1):1  opponent(#2):0
 Would you like to play again?
```

Draw match:

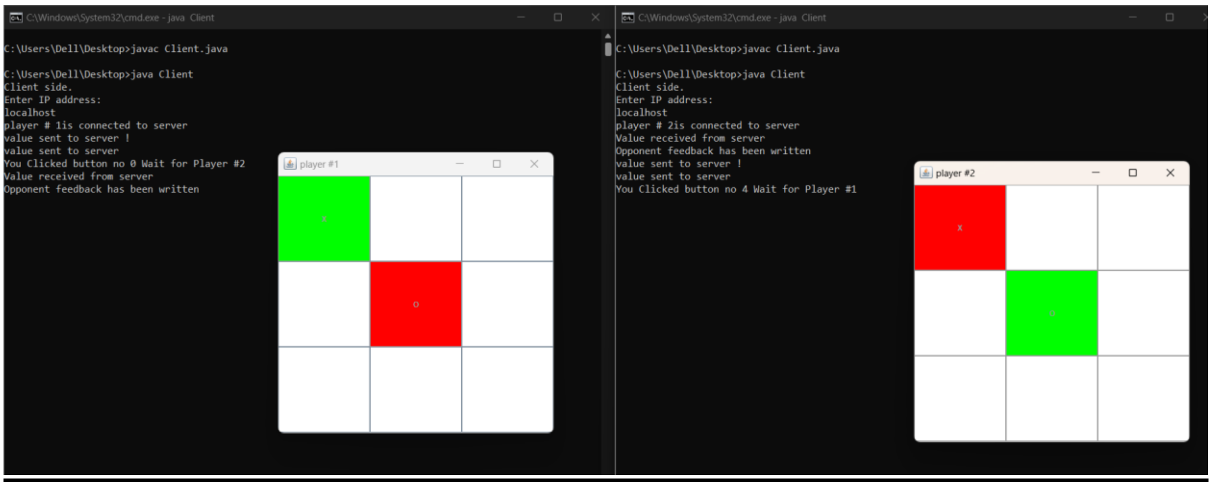
Fig 4.16: 1st input from #player1



```
C:\Windows\System32\cmd.exe - java  Client

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
```

```
C:\Windows\System32\cmd.exe - java  Client

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

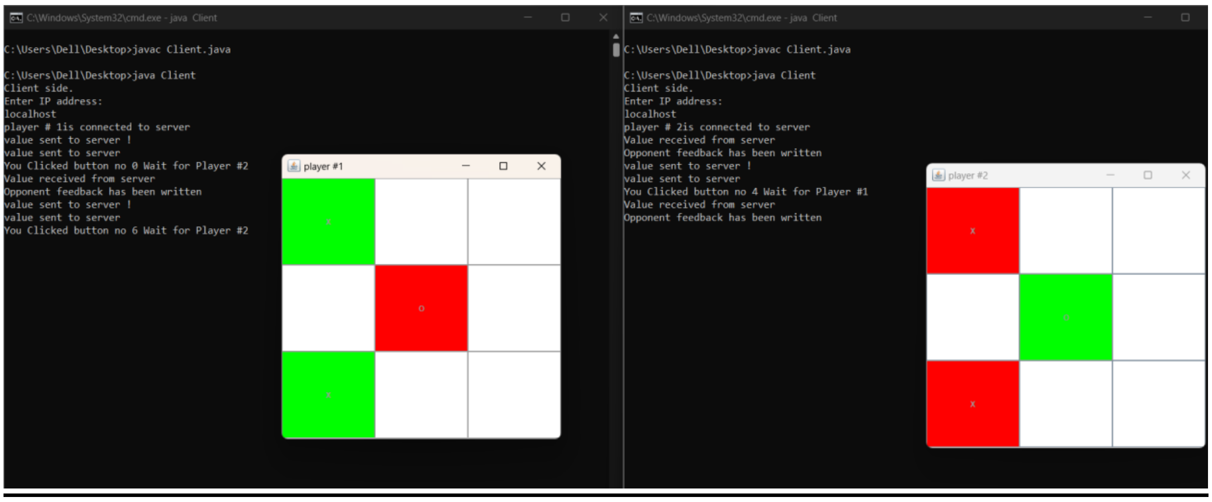
Fig 4.17: 1st input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
o

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
```

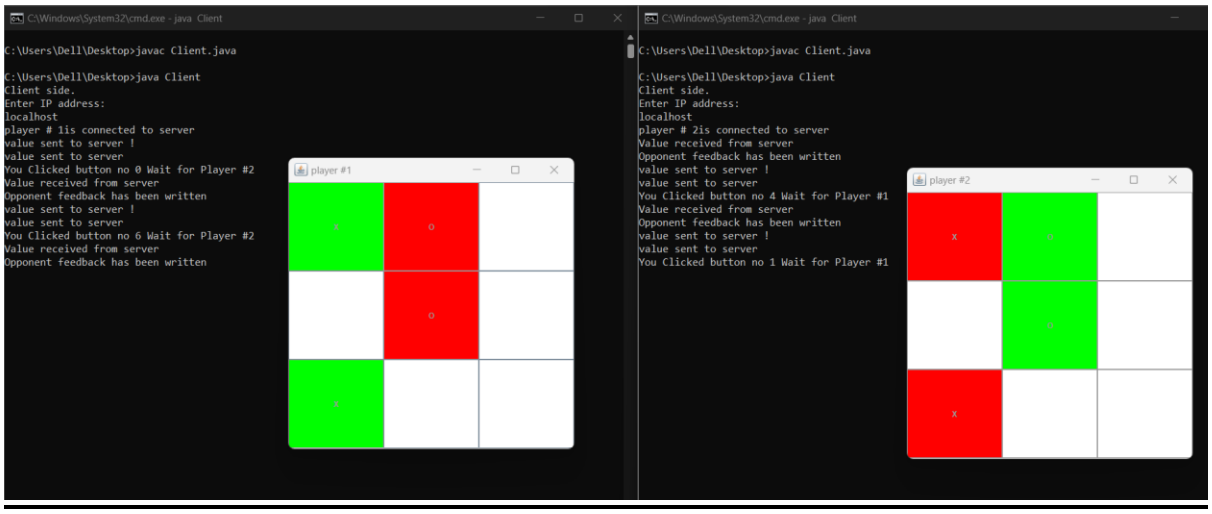
Fig 4.18: 2nd input from #player1


C:\Windows\System32\cmd.exe - java Client
player #1
o
C:\Windows\System32\cmd.exe - java Client
player #2
x
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
```

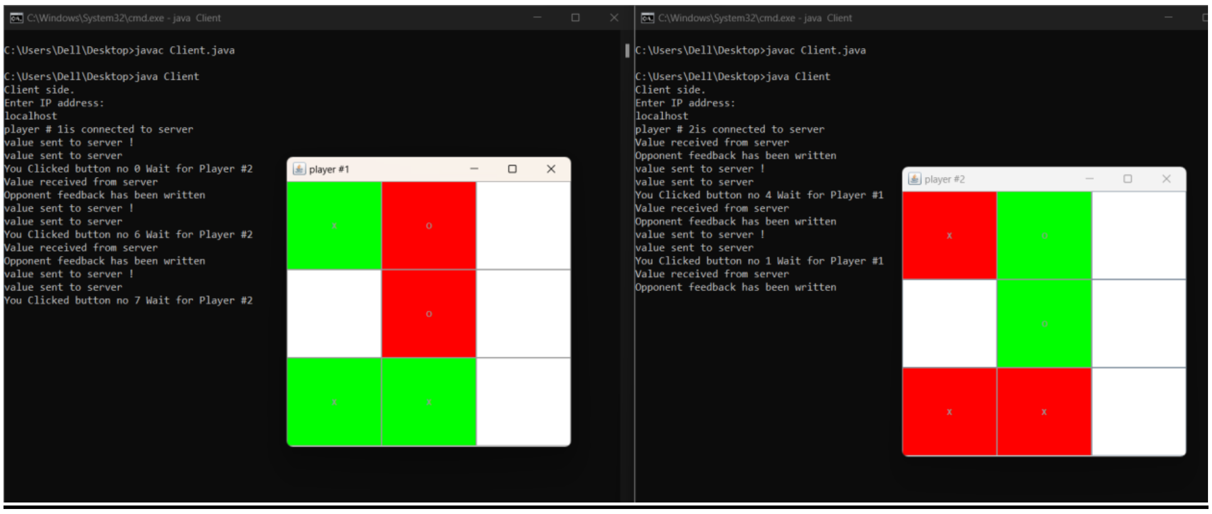
Fig 4.19: 2nd input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
o
x
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
o
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
```

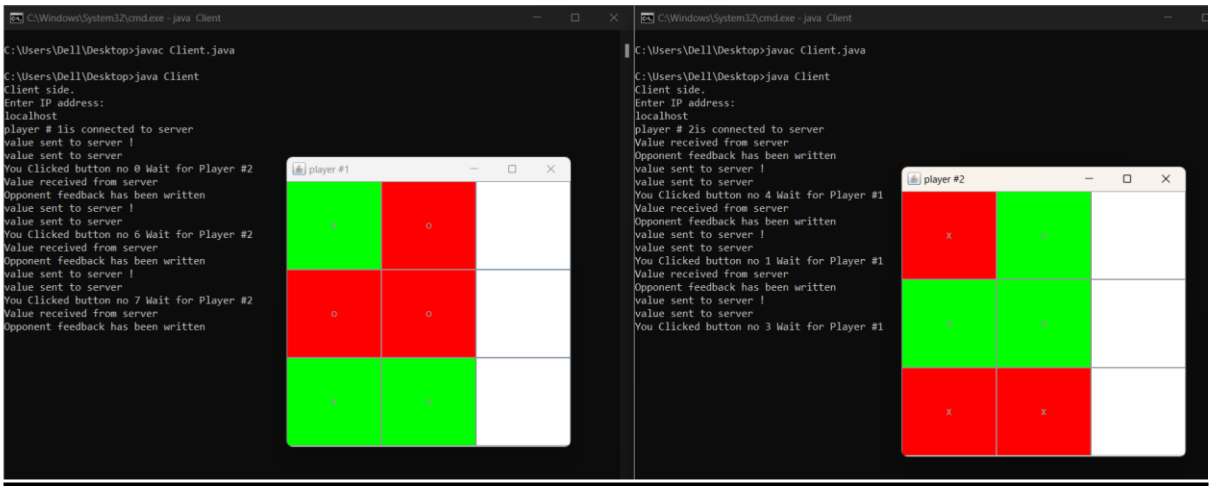
Fig 4.20: 3RD input from #player1


C:\Windows\System32\cmd.exe - java Client
player #1
x
o
o
x
x
C:\Windows\System32\cmd.exe - java Client
player #2
x
o
o
x
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 7 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
Value received from server
Opponent feedback has been written
```

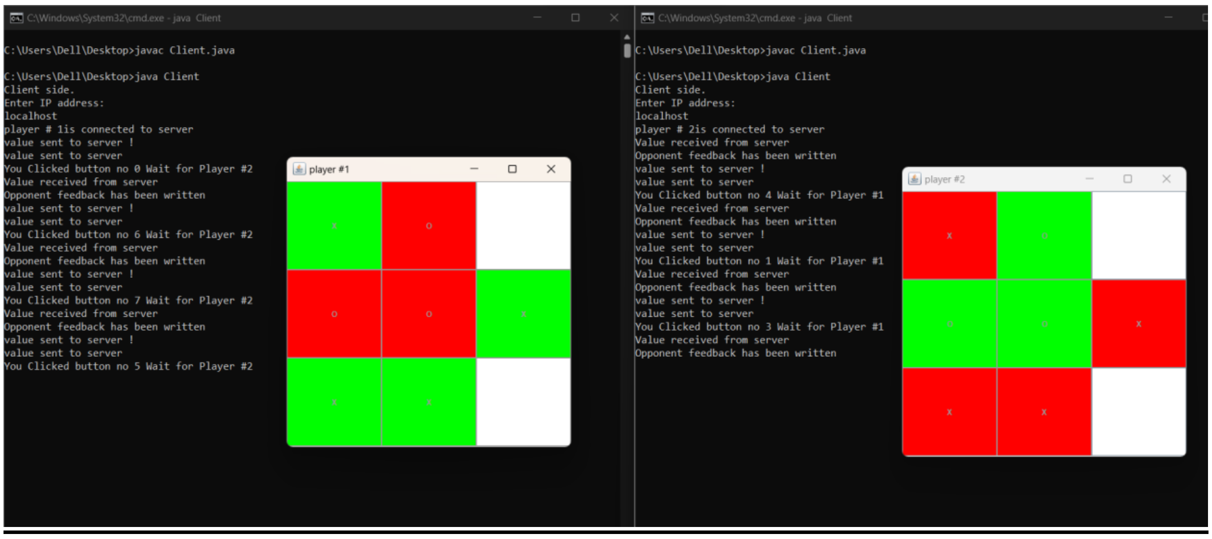
<u>Fig 4.21: 3rd input from #player2</u>


C:\Windows\System32\cmd.exe - java Client
player #1
C:\Windows\System32\cmd.exe - java Client
player #2

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 7 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #1
```

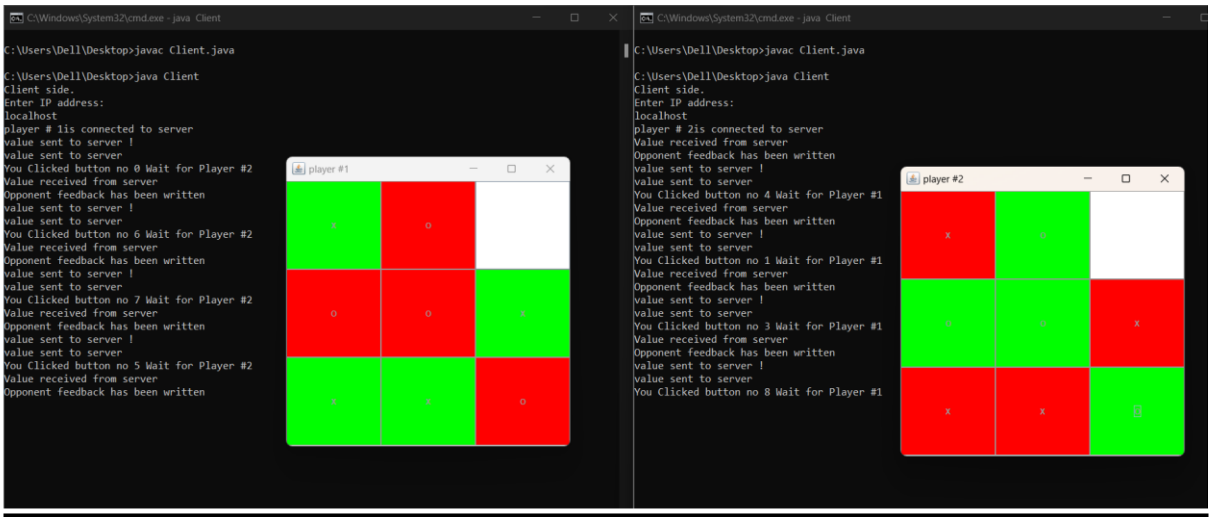
<u>Fig 4.22: 4th input from #player1</u>


C:\Windows\System32\cmd.exe - java Client
player #1
C:\Windows\System32\cmd.exe - java Client
player #2

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 7 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #1
Value received from server
Opponent feedback has been written
```

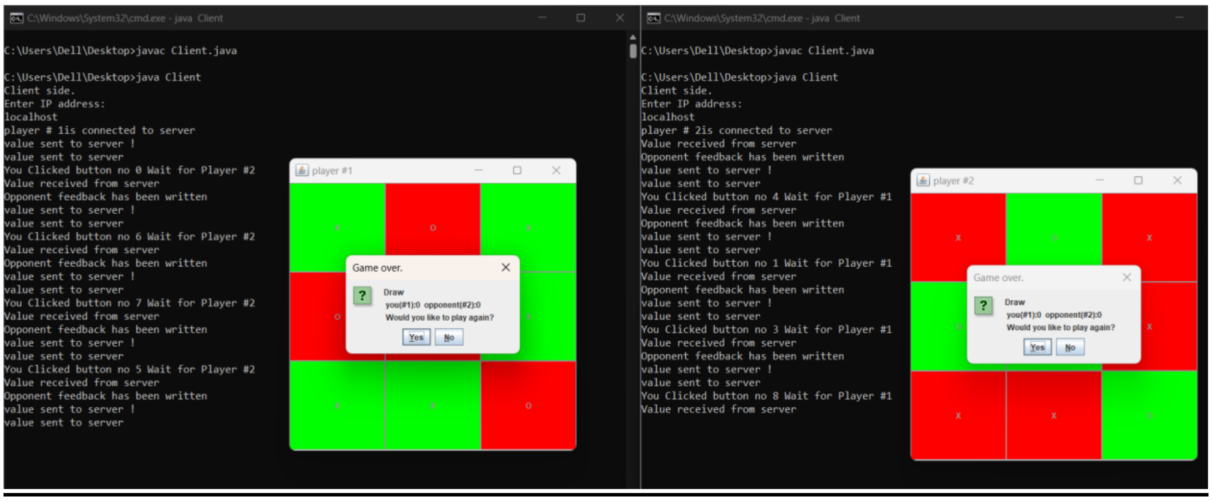
Fig 4.23: 4th input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
C:\Windows\System32\cmd.exe - java Client
player #2

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 7 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 8 Wait for Player #1
```

Fig 4.24: Draw match so no one gets a point


C:\Windows\System32\cmd.exe - java Client
player #1
Yes
No
C:\Windows\System32\cmd.exe - java Client
player #2
Yes
No

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 0 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 6 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 7 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 5 Wait for Player #2
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
```

Game over.

Draw
you(#1):0 opponent(#2):0
Would you like to play again?

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 3 Wait for Player #1
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 8 Wait for Player #1
Value received from server
```

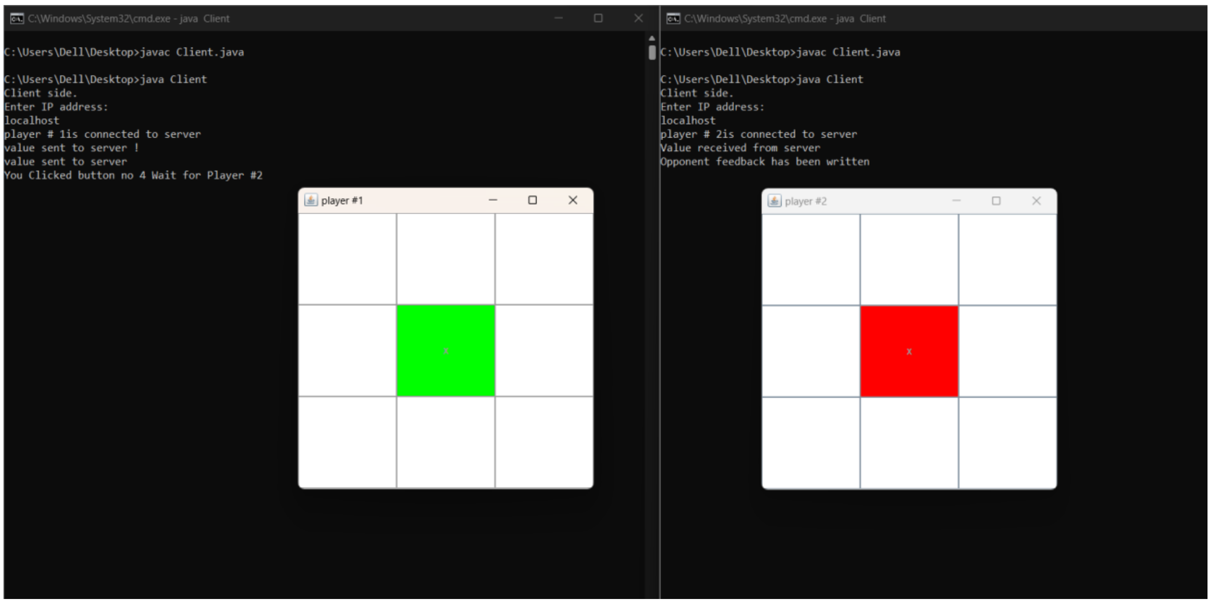
Game over.

Draw
you(#1):0 opponent(#2):0
Would you like to play again?

Timer:

#player1 wins due to timer:

Fig 4.25: 1st input from #player1:

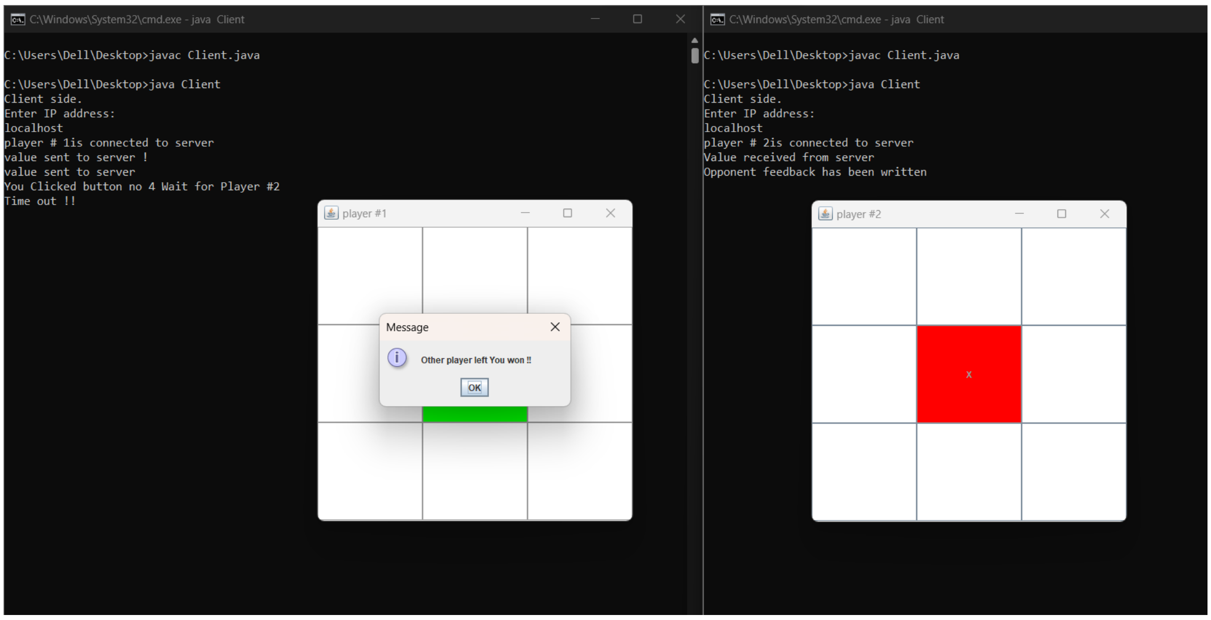


```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

C:\Windows\System32\cmd.exe - java Client
player #1
x
C:\Windows\System32\cmd.exe - java Client
player #2
x

Fig 4.26: #player2 left or fails to make move in 10 sec Timer is scheduled and #player1 wins:



```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Time out !!
```

Message

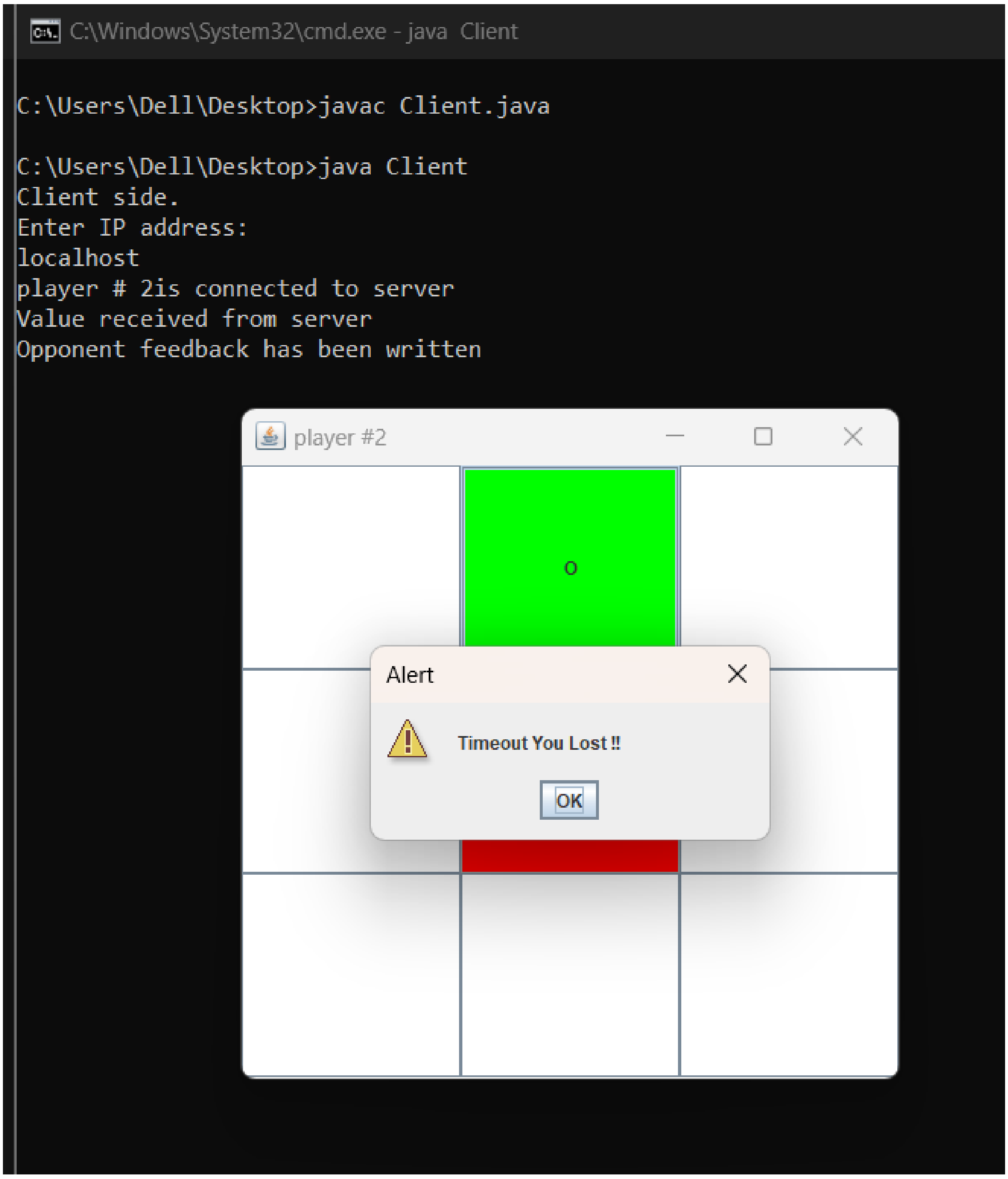
Other player left You won !!

OK

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

C:\Windows\System32\cmd.exe - java Client
player #1
C:\Windows\System32\cmd.exe - java Client
player #2
x

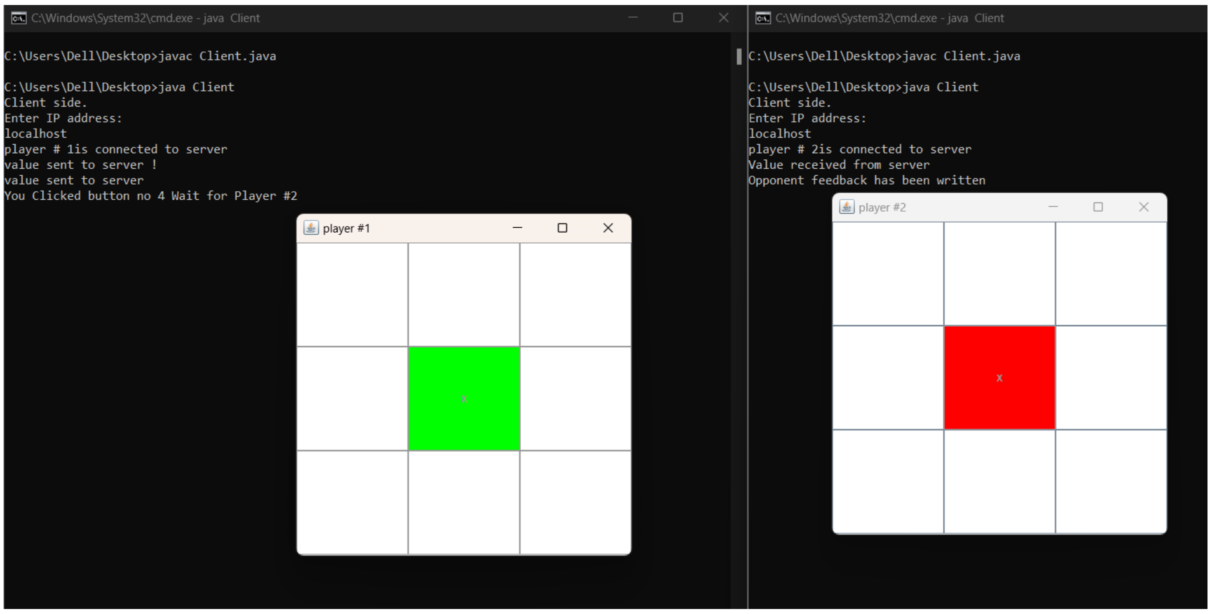
Fig 4.27: after time 10 sec if #player2 tries to make move then:



```
C:\Windows\System32\cmd.exe - java  Client

C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

#player2 wins due to timer:

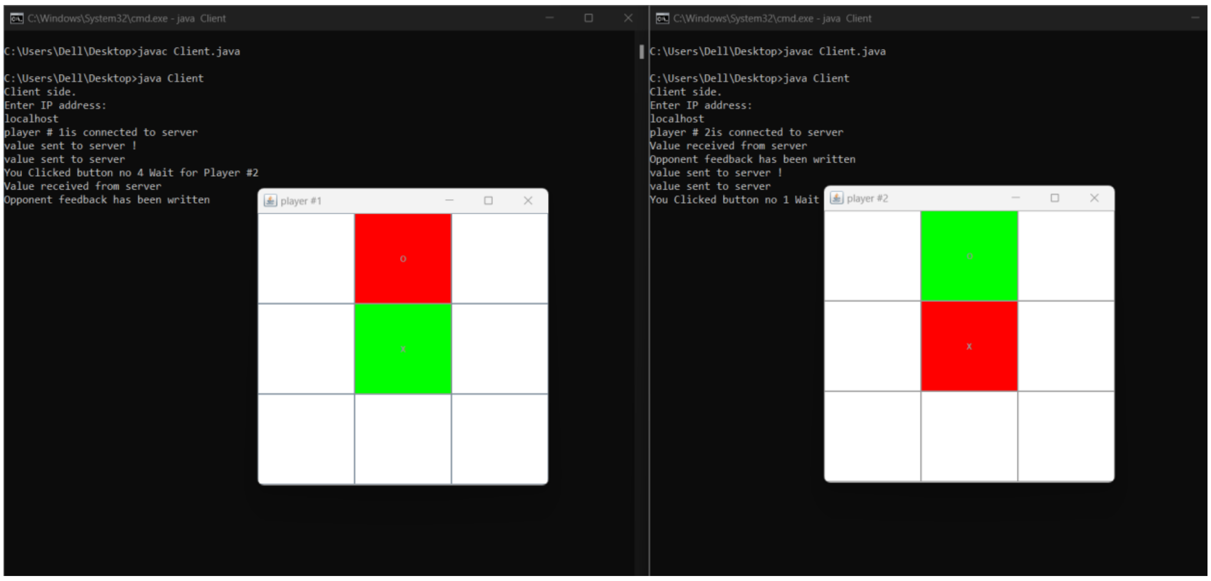
Fig 4.28: 1st input from #player1:


C:\Windows\System32\cmd.exe - java Client
player #1
x
C:\Windows\System32\cmd.exe - java Client
player #2
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
```

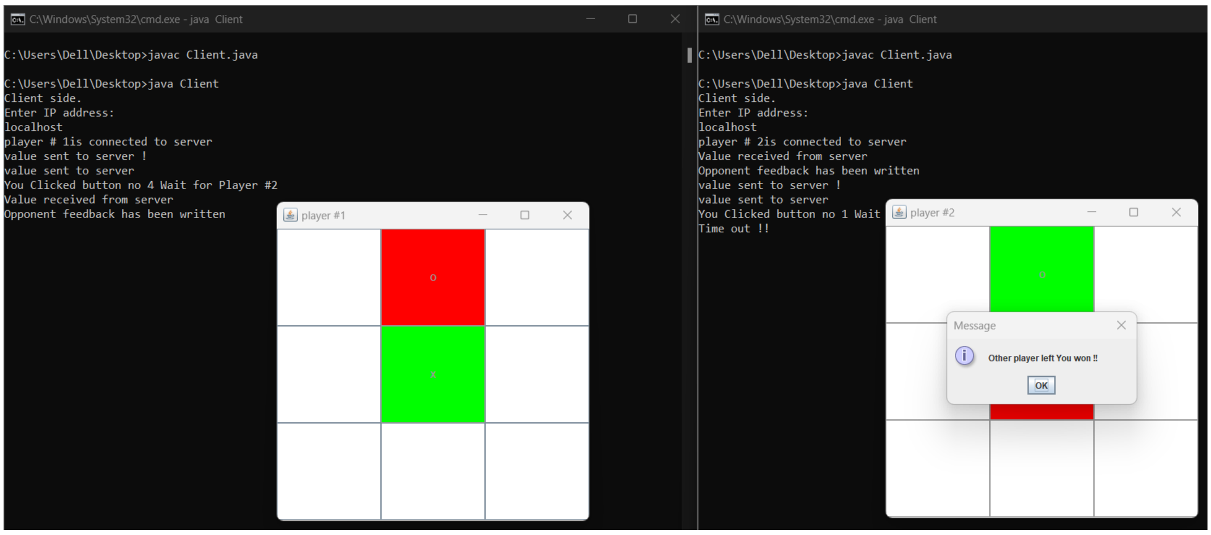
Fig 4.29: 1st input from #player2


C:\Windows\System32\cmd.exe - java Client
player #1
o
x
C:\Windows\System32\cmd.exe - java Client
player #2
o
x

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait
```

<u>Fig 4.30: #player1 left or fails to make move in 10 sec Timer is scheduled and #player2 wins:</u>

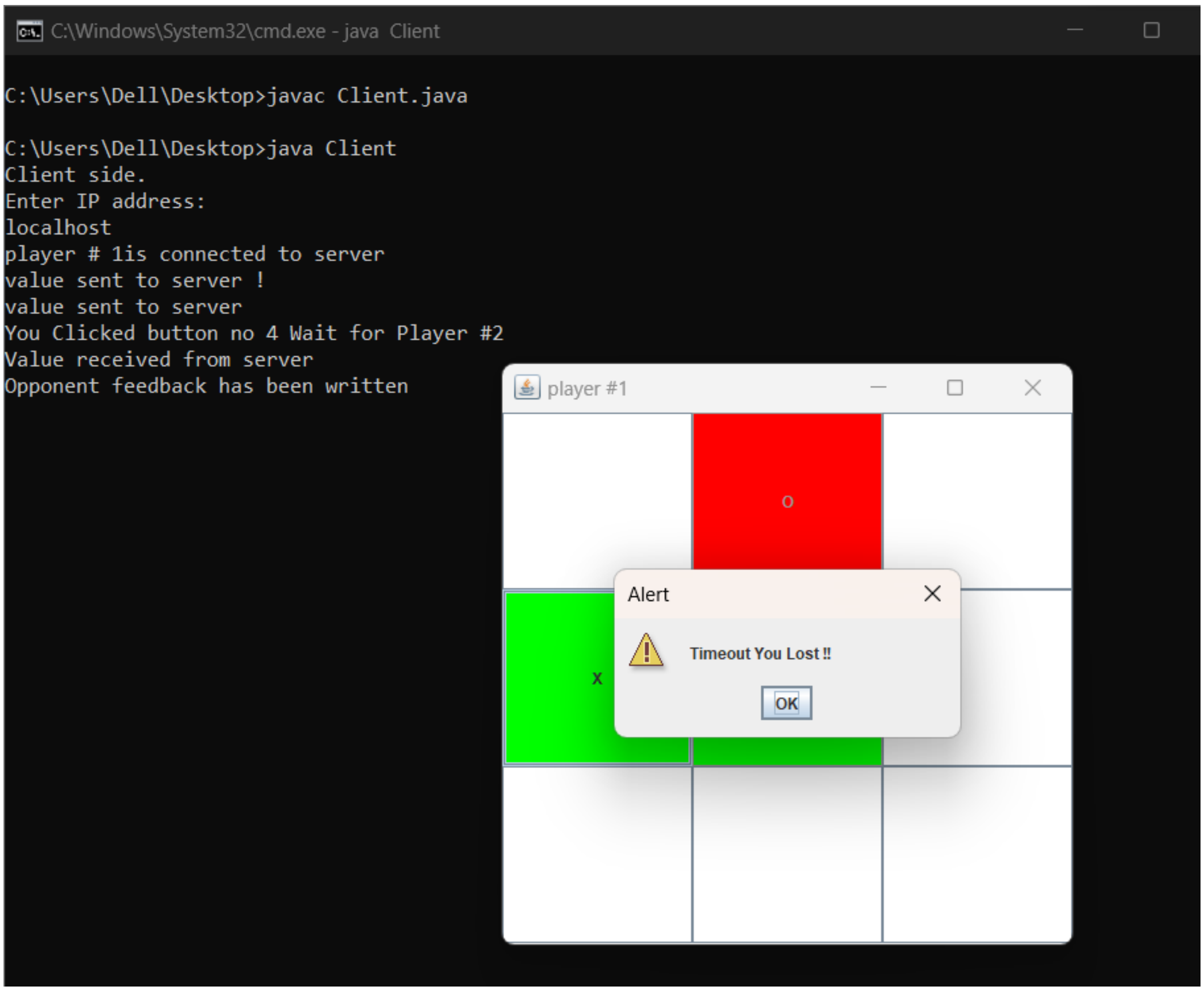


```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Value received from server
Opponent feedback has been written
```

```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 2is connected to server
Value received from server
Opponent feedback has been written
value sent to server !
value sent to server
You Clicked button no 1 Wait
Time out !!
```

Other player left You won !!

<u>Fig 4.31: after time 10 sec if #player1 tries to make move then:</u>



```
C:\Users\Dell\Desktop>javac Client.java

C:\Users\Dell\Desktop>java Client
Client side.
Enter IP address:
localhost
player # 1is connected to server
value sent to server !
value sent to server
You Clicked button no 4 Wait for Player #2
Value received from server
Opponent feedback has been written
```

Timeout You Lost !!

CHAPTER 5

CONCLUSION

Thus we have developed Tic-Tac-Toe Game where we can play across any 2 computers connecting with IP address.

Client Server architecture , Java Socket programming , TCP protocol has been implemented successfully. We also implemented Timer which is working good. Moves of players transmitted fast and also there is no delay.

We can also optimize this game some ideas are:

1. Making the Server handling multiple pairs of clients. So that multiple players can play with single server. Connecting players may be randomized or based on order or based on specific codes.
2. Creating Mobile app with flutter and playing across multi platform.
3. Maintaining player information by creating accounts.

CHAPTER 6

REFERENCES

https://en.wikipedia.org/wiki/Network_socket

https://en.wikipedia.org/wiki/Client%E2%80%93server_model

https://www.geeksforgeeks.org/introducing-threads-socket-programming-java/

https://www.javatpoint.com/how-to-set-timer-in-java

https://www.geeksforgeeks.org/java-util-timer-class-java/

https://www.geeksforgeeks.org/client-server-model/