Mini Project Report

on

A.I TICTACTOE

Submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY

DEGREE

Session 2020-21
in

Information Technology

By
SAURABH UPPAL
1900320130146

Coordinated By: - Mr. JITENDRA KR. CHAUHAN
(Assistant Professor)

DEPARTMENT OF INFORMATION TECHNOLOGY
ABES ENGINEERING COLLEGE, GHAZIABAD

AFFILIATED TO
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, U.P., LUCKNOW
(Formerly UPTU)

Student's Declaration

I hereby declare that the work being presented in this report entitled **"A.I TIC TAC TOE"** is an authentic record of my own work carried out under the supervision of **Mr. Jitendra Kr. Chauhan, Assistant Professor, Information Technology.**

The matter embodied in this report has not been submitted by me for the award of any other degree.

Date:

Signature of student
Name: SAURABH UPPAL
Roll No. 1900320130146
Department: Information Technology

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

Signature of HOD
(Name: Dr Amit Sinha)

(Information Technology)

Signature of Coordinator
(Name: Mr. Jitendra Kr. Chauhan)

(Assistant Professor)
(Information Technology)

Date: .....

Acknowledgement

Presentation inspiration and motivation have always played a key role in the success of any venture. I express my sincere thanks to my project coordinator **Mr. Jitendra Kr. Chauhan** to encourage me to the highest peak and to provide me this opportunity to prepare this project. I extend my hearty thanks for giving me the proper guidance even in this time, when everything is continuing on just online platforms. I am highly indebted to **Mr. Jitendra Kr. Chauhan** for the constant supervision, for providing all the necessary information and support in completing the project. Finally, I am sincerely thankful to all those people who are directly or indirectly involved in the successful completion of this project work.

Signature of student
(Name: Saurabh Uppal)
(Roll No.1900320130146)

Table of Contents

LIST OF FIGURES AND TABLES

ABSTRACT

Tic-Tac-Toe Game is a very popular game played by two participants on the grid of 3 by 3. A special symbol (X or O) is assigned to each participant to indicate that the slot is covered by the respective participant. The winner of the game is the participant who first cover a horizontal, vertical or diagonal row of the board having only their symbols.

I chose tic-tac-toe (also known as noughts and crosses) since virtually everyone knows the game and the rules are simple enough that we don't need an elaborate analysis of game configurations. Despite being a simple game, the basic AI principles shown here can be applied to more complicated games such as checkers, Connect 4, go and even chess.

This report proposed a **Tic-Tac-Toe** game made using java which implements Java libraries and an AI Algorithm called MiniMaxAlgorithm.

This algorithm is designed for computer as a player in which computer act according to the intelligence of model to maximize the chances of success. The human player can make its own choices. Human Player starts first. The computation rules ensure selection of best slot for computer that will lead to win or prevent opponent to make a winning move.

The contribution of this work is to implement a algorithm to play **Tic-Tac-Toe** game in which **computer will never lose**.

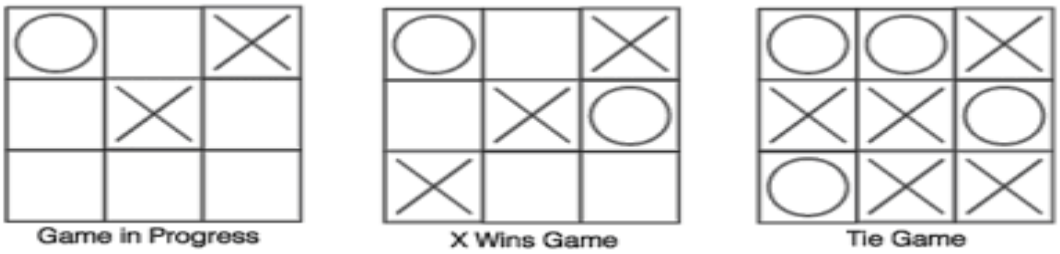
****

Saurabh Uppal

CHAPTER 1: INTRODUCTION

1.1 ABOUT TIC TAC TOE

Figure 1: -



Tic-tac-toe also known as " Noughts and crosses" is a paper and pencil game for two players, who take turns marking the spaces in a 3 x 3 grid traditionally. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row wins the game. It is solved game with a forced draw assuming best play from both players.

1.2 How many Tic-Tac-Toe games are possible?

A nice simple question which is perfectly possible to solve with a little bit of brute force. The number is clearly bounded above, since there are 9 possible ways of placing the first mark, 8 remaining ways of placing the second, 7 the third, ..., and 1 the ninth.

255,168 unique games of Tic Tac Toe are posible. Of these, 131,184 are won by the first player, 77,904 are won by the second player, and 46,080 are drawn. This supports the intuition that it is an advantage to begin the game. These numbers do not take similar board positions into account – rotating the board, mirroring it and so on. It does not matter which corner you place the first piece in, but this is not taken into account here. If neither player makes a mistake, the game is drawn (but we knew that already.)

CHAPTER 2: PROBLEM STATEMENT

Problem: Given a 3x3 grid, the players has to find the optimal cell to fill with respective marks.

Goals: To find the optimal cell to fill with respective marks and in order to win the game, the cell must be filled such that one of the following criteria is satisfied:

1. A row is completely filled by a mark ‘X’ or ‘O’.
2. A diagonal is completely filled by a mark ‘X’ or ‘O’.
3. A column is completely filled by a mark ‘X’ or ‘O’. If these criteria are not satisfied by both the agents, the game is terminated with a tie situation.

Constraints: -

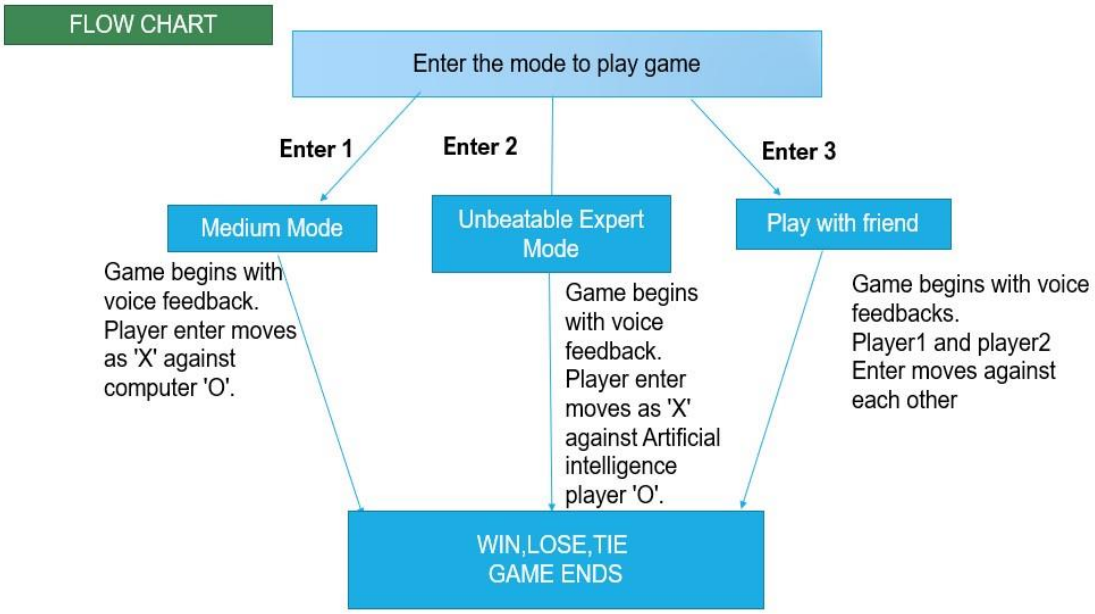
1. Once the cell is occupied by a mark, it cannot be reused.
2. Players place the mark alternatively. So, consecutive moves from any player is not allowed.

CHAPTER 3: PROJECT OBJECTIVE

- The aim of this project is to develop a **A.I Tic-Tac-Toe game with voice feedback**.
- It has three parts - 1. Human Vs Computer (Medium difficulty mode)
  2. Human Vs Computer (Unbeatable Expert mode)
  3.PlayerVsPlayerMode
- Tic Tac Toe is a great way to pass your free time. The friendliness of Tic Tac Toe makes it ideal as a pedagogical tool for teaching and learning the concepts of **Object Oriented Programming (OOP) in JAVA,** implementing **java libraries** and implementing the branch of **Artificial intelligence** that deals with the searching of game trees.
- To implement **AI** based **MINIMAX ALGORITHM** in java to make a **Unbeatable Tic Tac Toe AI**.
- To create a separate program for playing the game over local network by using java sockets.
- To eliminate the use of paper for playing **TIC-TAC-TOE**.
- Another purpose for developing this program is to make this traditional game famous among today's exclusively tech loving kids.

CHAPTER 4: PROJECT METHODOLOGY

Figure2: -



Process –

1.User enters the mode in which they want to play the game.

- Enter 1 for medium mode.
- Enter 2 for Unbeatable expert mode.
- Enter 3 for player vs player.

2.The game starts with voice feedback, Welcoming you to game.

3.Enter the position where you want to put your mark in the option pane.

4.The Computer put its mark automatically by using its algorithm and speak to put your move.

5.If you enter a position already taken, the game tells you to enter another move.

5.Each time mark is put program check if someone **lost, win or game tied**.

6.If it happens game stops.

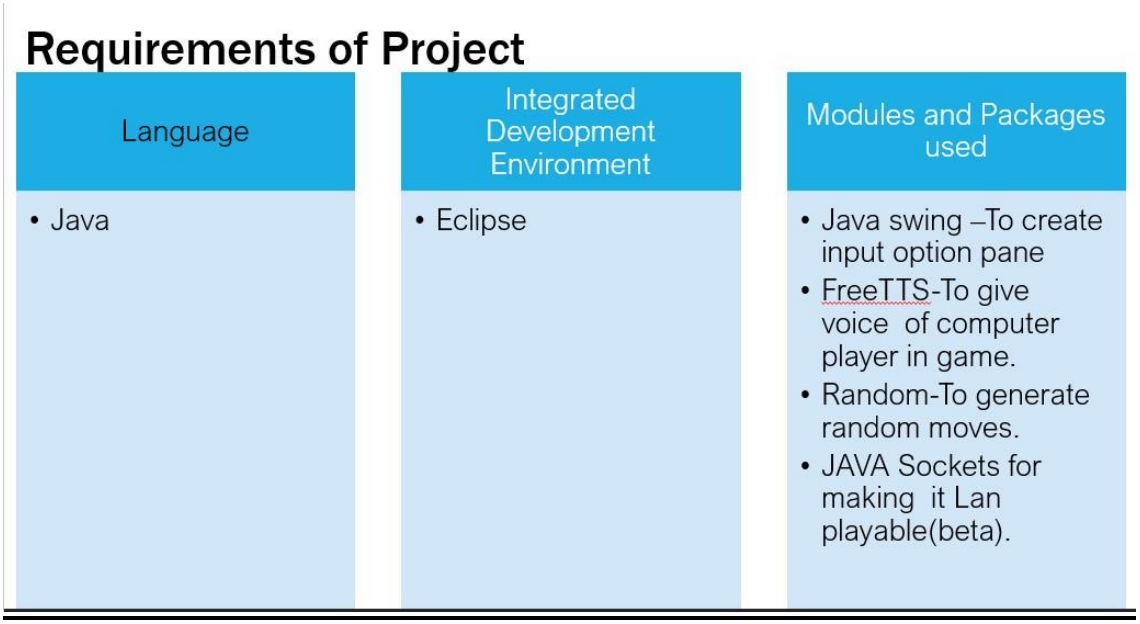
METHODS USED IN PROGRAM AND THEIR USE

- **displayBoard() -**This method is used to display the current board.
- **getComputerMove() -**This method is used to get computer moves based on different algorithms.
- **speak() -**This method is used to call text to speech synthesizer for computer voice.
- **checkForWinner() -**This method is used to check if someone has won or the game has tied.
- **getUserMove() -**This method is used to take moves from user

CHAPTER 5: DETAILS OF PROJECT WORK

JAVA PROJECT ON AI TICTACTOE

Figure 3: -



1.In this project I have made a Tic Tac Toe game in which you can play against computer or your friend.

2.The Tic Tac Toe has voice feedback.

3.It has three modes medium, unbeatable expert and play with friend.

4.There are three core logics each for each mode.

❖ 5.1 MEDIUM MODE AI

The logic used for medium mode AI is as follows : -

1. **First Move-**If center is available take center else randomly take any position.
2. **Second Move-**
    - Check if computer is winning. If winning put computer move there.

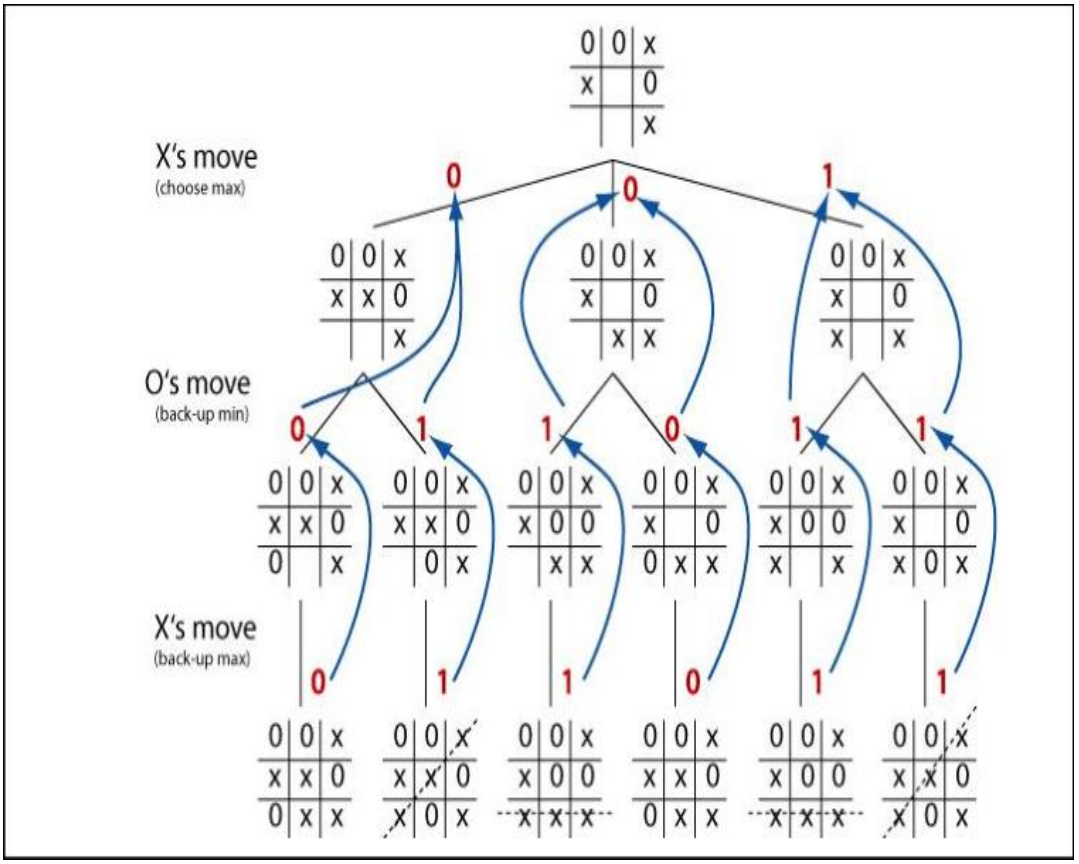
- If computer not winning check if player is winning. If player winning then block player from winning by putting move.
- Else randomly put move anywhere.

❖ 5.2 UNBEATABLE EXPERT MODE AI

Expert mode is implemented by a **MINIMAX ALGORITHM**

ABOUT MINIMAX ALGORITHM

Figure 4: -



Mini-Max Algorithm in Artificial Intelligence

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.

- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm performs a depth-first search algorithm for the exploration of the complete game tree.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

Pseudo-code for MiniMax Algorithm:

```
1. function minimax(node, depth, maximizingPlayer) is
2. if depth ==0 or node is a terminal node then
3. return static evaluation of node
4.
5. if MaximizingPlayer then      // for Maximizer Player
6. maxEva= -infinity
7.  for each child of node do
8.  eva= minimax(child, depth-1, false)
9. maxEva= max(maxEva,eva)       //gives Maximum of the values
10. return maxEva
```

```
11.
12. else                    // for Minimizer player
13. minEva= +infinity
14. for each child of node do
15. eva= minimax(child, depth-1, true)
16. minEva= min(minEva, eva)        //gives minimum of the values
17. return minEva
```

Initial call:

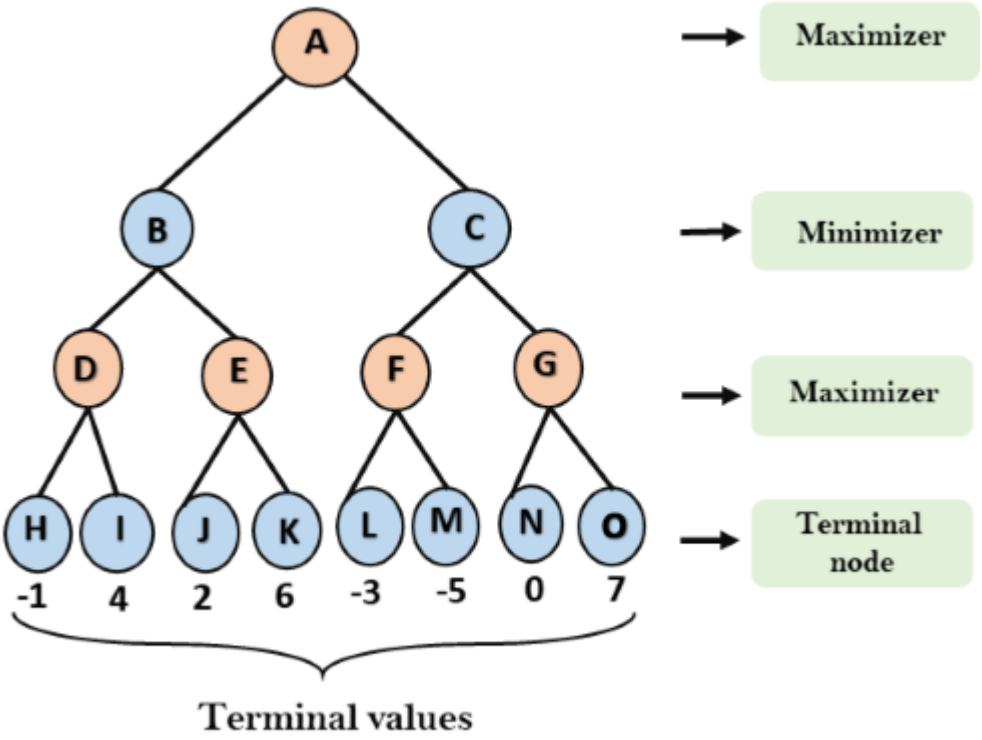
Minimax(node, 3, true)

Working of Min-Max Algorithm:

- The working of the minimax algorithm can be easily described using an example. Below we have taken an example of game-tree which is representing the two-player game.
- In this example, there are two players one is called Maximizer and other is called Minimizer.
- Maximizer will try to get the Maximum possible score, and Minimizer will try to get the minimum possible score.
- This algorithm applies DFS, so in this game-tree, we have to go all the way through the leaves to reach the terminal nodes.
- At the terminal node, the terminal values are given so we will compare those value and backtrack the tree until the initial state occurs. Following are the main steps involved in solving the two-player game tree:

Step-1: In the first step, the algorithm generates the entire game-tree and apply the utility function to get the utility values for the terminal states. In the below tree diagram, let's take A is the initial state of the tree. Suppose maximizer takes first turn which has worst-case initial value =- infinity, and minimizer will take next turn which has worst-case initial value = +infinity.

Figure 5: -

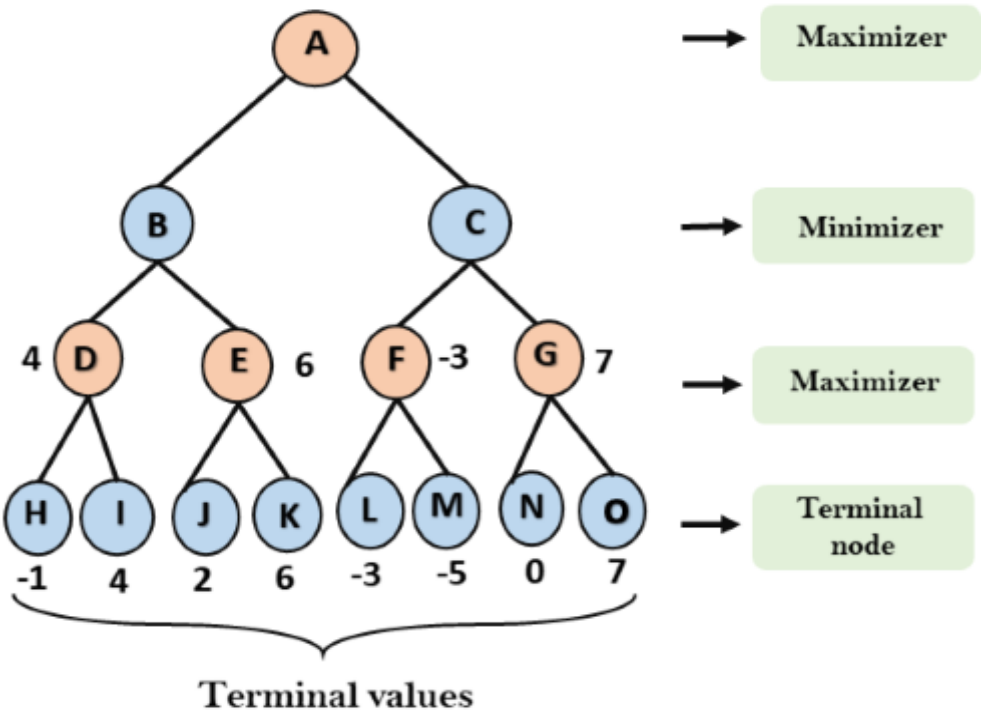


Step 2: Now, first we find the utilities value for the Maximizer, its initial value is -∞, so we will compare each value in terminal state with initial value of Maximizer and determines the higher nodes values. It will find the maximum among the all.

- For node D max(-1,- -∞) => max(-1,4)= 4

- For Node E max(2, -∞) => max(2, 6)= 6
- For Node F max(-3, -∞) => max(-3,-5) = -3
- For node G max(0, -∞) = max(0, 7) = 7

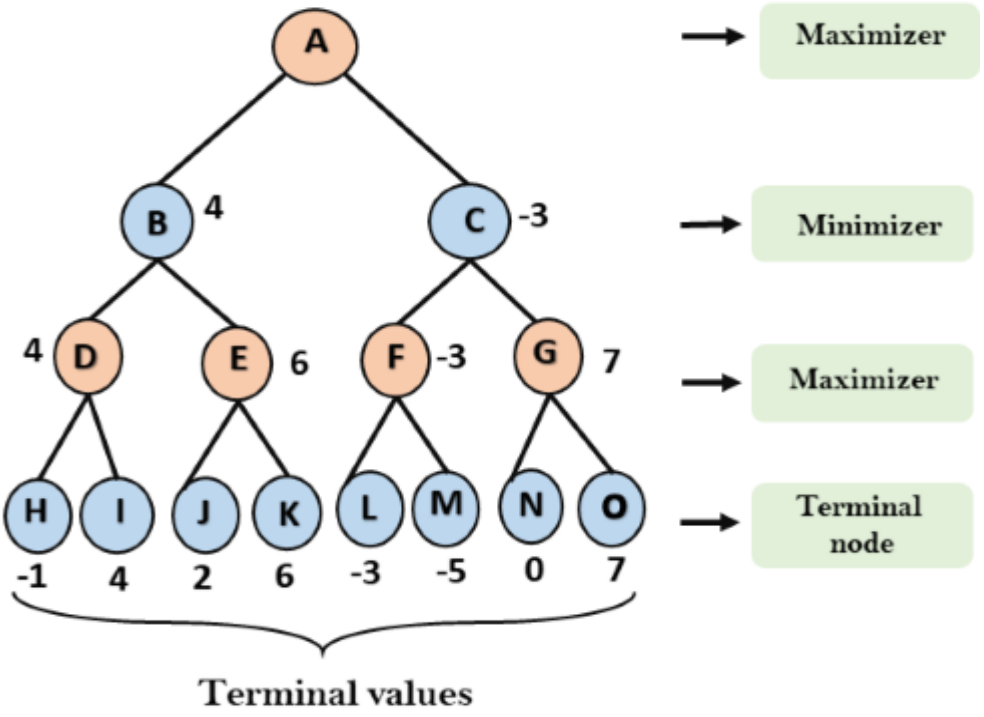
Figure 6: -



Step 3: In the next step, it's a turn for minimizer, so it will compare all nodes value with +∞, and will find the 3rd layer node values.

- For node B= min(4,6) = 4
- For node C= min (-3, 7) = -3

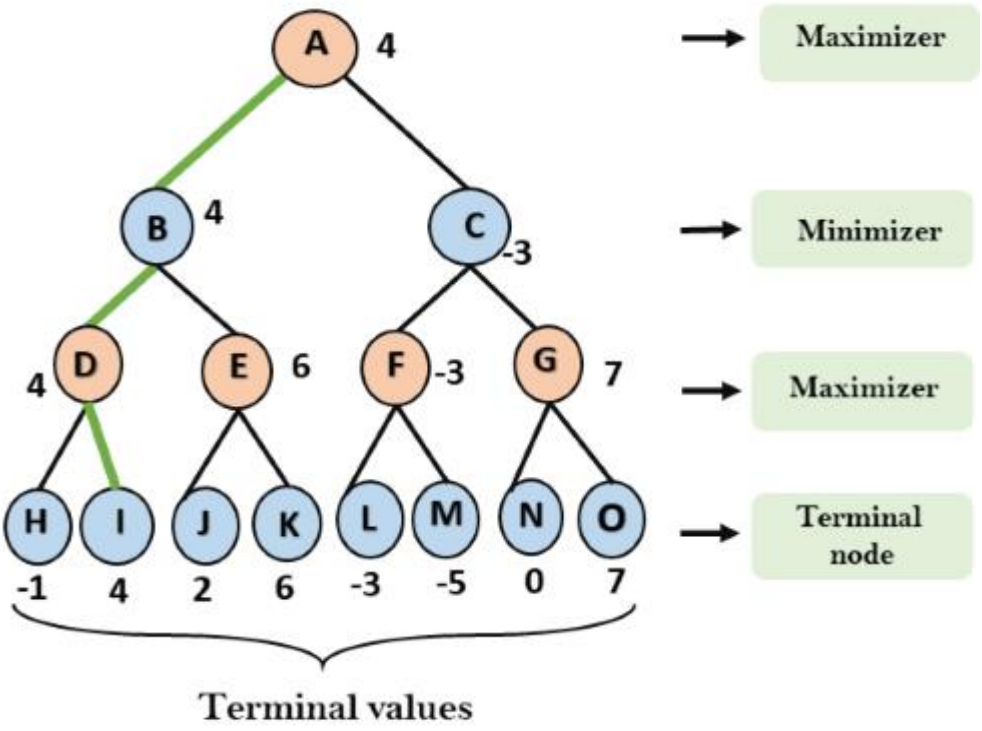
Figure 7: -



Step 4: Now it's a turn for Maximizer, and it will again choose the maximum of all nodes value and find the maximum value for the root node. In this game tree, there are only 4 layers, hence we reach immediately to the root node, but in real games, there will be more than 4 layers.

- For node A max(4, -3)= 4

Figure 8: -



That was the complete workflow of the minimax two player game.

Properties of Mini-Max algorithm:

- **Complete-** Min-Max algorithm is Complete. It will definitely find a solution (if exist), in the finite search tree.
- **Optimal-** Min-Max algorithm is optimal if both opponents are playing optimally.
- **Time complexity-** As it performs DFS for the game-tree, so the time complexity of Min-Max algorithm is **$O(b^m)$**, where b is branching factor of the game-tree, and m is the maximum depth of the tree.
- **Space Complexity-** Space complexity of Mini-max algorithm is also similar to DFS which is **O(bm)**.

Limitation of the minimax Algorithm:

The main drawback of the minimax algorithm is that it gets really slow for complex games such as Chess, go, etc. This type of games has a huge branching factor, and the player has lots of choices to decide. This limitation of the minimax algorithm can be improved from **alpha-beta pruning**.

5.3 API's AND LIBRARIES USED

❖ FREE TTS FOR TEXT TO SPEECH

What is Free TTS?

- Free TTS is entirely written in Java programming language, which is nothing but an open-source Speech Synthesis system by which we can make our computer speak.
- In simple words, we can say that it is an artificial production of human speech which converts normal language text into speech. So in this tutorial, We will learn about how to convert text to speech in Java using the Eclipse IDE.

Converting Text to Speech in Java

Java Speech API: The Java Speech API allows Java applications to incorporate speech technology into their user interfaces. It defines a cross-platform API to support command and control recognizers, dictation systems and speech synthesizers.

Java Speech supports speech synthesis which means the process of generating spoken the language by machine on the basis of written input.

It is important to keep in mind that Java Speech is only a specification i.e. no implementation is included. Thus third-parties provide the implementations. The **javax.speech package** defines the common functionality of recognizers, synthesizers, and other speech engines. The package **javax.speech.synthesis** extends this basic functionality for synthesizers.

We will understand that what is required for java API to convert text to speech

1. **Engine:** The Engine interface is available inside the speech package.”Speech engine” is the generic term for a system designed to deal with either speech input or speech output.

   import javax.speech.Engine;

2. **Central:** Central provides the ability to locate, select and create speech recognizers and speech synthesizers.

   import javax.speech.Central;

3. **SynthesizerModeDesc:** SynthesizerModeDesc extends the EngineModeDesc with the properties that are specific to speech synthesizers.

   import javax.speech.synthesis.SynthesizerModeDesc;

4. **Synthesizer:** The Synthesizer interface provides primary access to speech synthesis capabilities.SynthesizerModeDesc adds two properties: List of voices provided by the synthesizer Voice to be loaded when the synthesizer is started.

   import javax.speech.synthesis.Synthesizer;

❖ JAVA JOPTION PANE FOR CREATING INPUT PANEL

The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box. These dialog boxes are used to display information or get input from the user. The JOptionPane class inherits JComponent class.

Common Constructors of JOptionPane class

Table 1: -

Constructor	Description
JOptionPane()	It is used to create a JOptionPane with a test message.
JOptionPane(Object message)	It is used to create an instance of JOptionPane to display a message.
JOptionPane(Object message, int messageType	It is used to create an instance of JOptionPane to display a message with specified message type and default options.

Common Methods of JOptionPane class

Table 2: -

Methods	Description
JDialog createDialog(String title)	It is used to create and return a new parentless JDialog with the specified title.

static void showMessageDialog(Component parentComponent, Object message)	It is used to create an information-message dialog titled "Message".
static void showMessageDialog(Component parentComponent, Object message, String title, int messageType)	It is used to create a message dialog with given title and messageType.
static int showConfirmDialog(Component parentComponent, Object message)	It is used to create a dialog with the options Yes, No and Cancel; with the title, Select an Option.
static String showInputDialog(Component parentComponent, Object message)	It is used to show a question-message dialog requesting input from the user parented to parentComponent.
void setInputValue(Object newValue)	It is used to set the input value that was selected or input by the user.

❖ RANDOM NUMBER GENERATOR

- The **java.lang.Math.random()** returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.
- Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range. When this method is first called, it creates a single new pseudorandom-number generator, exactly as if by the expression new java.util.Random.

❖ Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

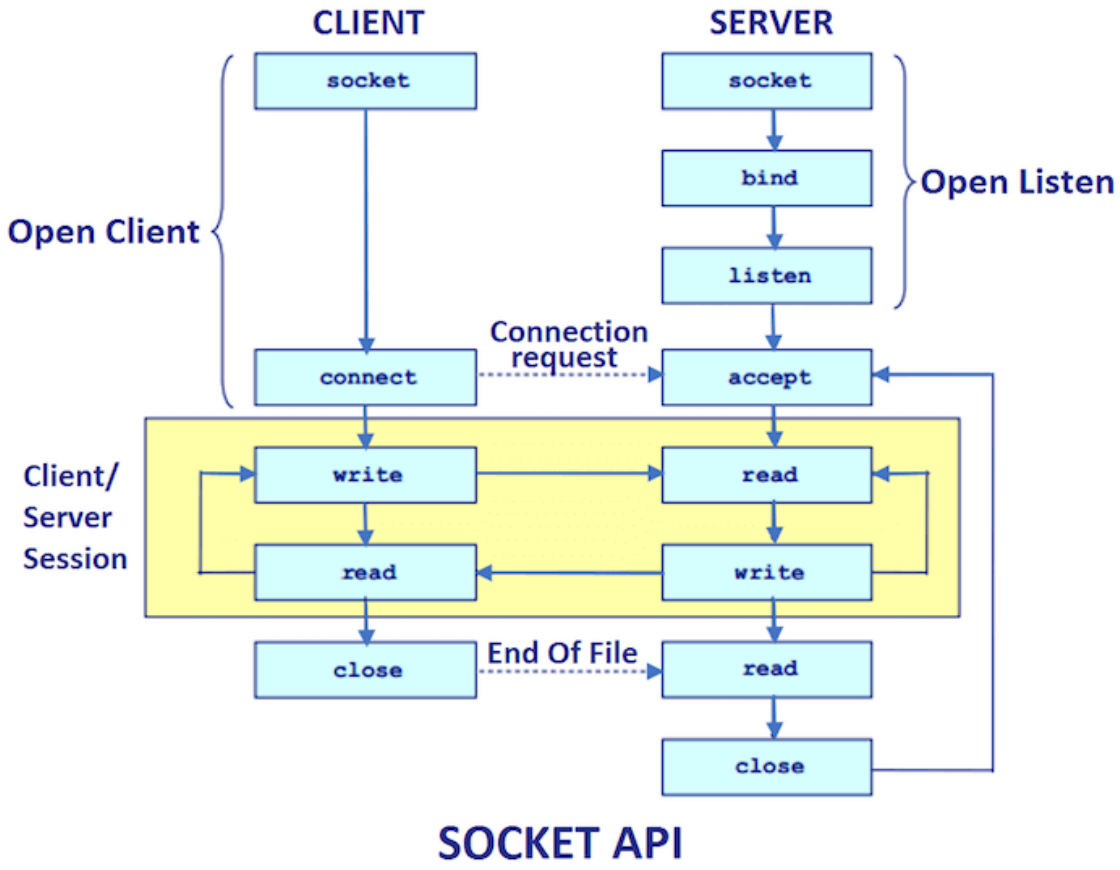
Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server, and
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.



Creating Server: To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
1. ServerSocket ss=new ServerSocket(6666);
2. Socket s=ss.accept();//establishes connection and waits for the client
```

Creating Client: To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
1.Socket s=new Socket("localhost",6666);
```

CHAPTER 6: RESULT AND DISCUSSIONS

Results that I got after running code written for this project are quite similar to those that I expected and there is not traceback or any other kind of errors.

The game is playable and all its functions are working fine. Here are snippets: -

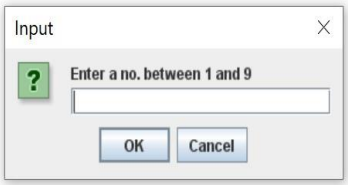
```
TicTacToeConsole (6) [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe  (Jan 27, 2021, 11:47:10 PM)
xxxxxxxxxxxxTIC TAC TOE CONSOLExxxxxxxxxxxxxxx
Enter the mode in which you want to play the game
Enter 1 for Medium difficulty Mode AI opponent
Enter 2 for Unbeatable Expert Mode AI opponent
Enter 3 to play with a friend
Enter any other no. to exit
Instructions-Player marker is X
            Enter the position where you want to move your mark
2
UNBEATABLE MODE

1 | 2 | 3
-----------
4 | 5 | 6
-----------
7 | 8 | 9


1 | 2 | X
-----------
4 | 5 | 6
-----------
7 | 8 | 9

Computer is moving to 5

1 | 2 | X
-----------
4 | O | 6
-----------
7 | 8 | 9
```




```
TicTacToeConsole (6) [Java Application] C:\Program Files\Java\jre1.8.0_271\bin\javaw.exe  (Jan 27, 2021, 11:47:10 PM)
7 | 8 | O

X | 2 | X
-----------
4 | O | X
-----------
7 | 8 | O

Computer is moving to 2

X | O | X
-----------
4 | O | X
-----------
7 | 8 | O

X | O | X
-----------
4 | O | X
-----------
X | 8 | O

Computer is moving to 8

X | O | X
-----------
4 | O | X
-----------
X | O | O

O wins!
```

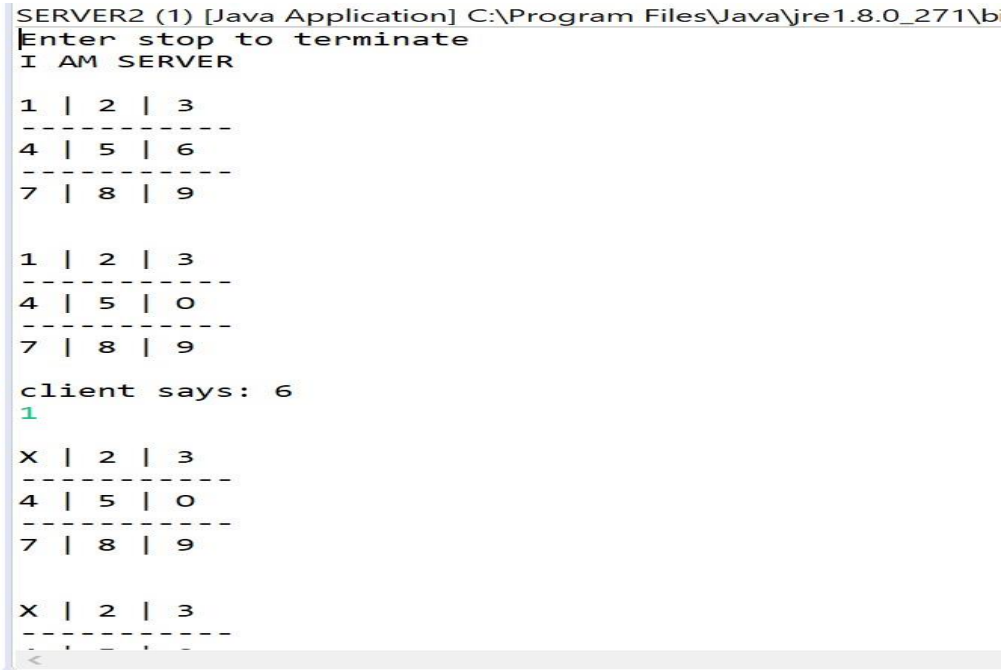
EXTENDED FUNCTIONALITIES: -

A **LOCAL AREA NETWORK** playable program is available separately.

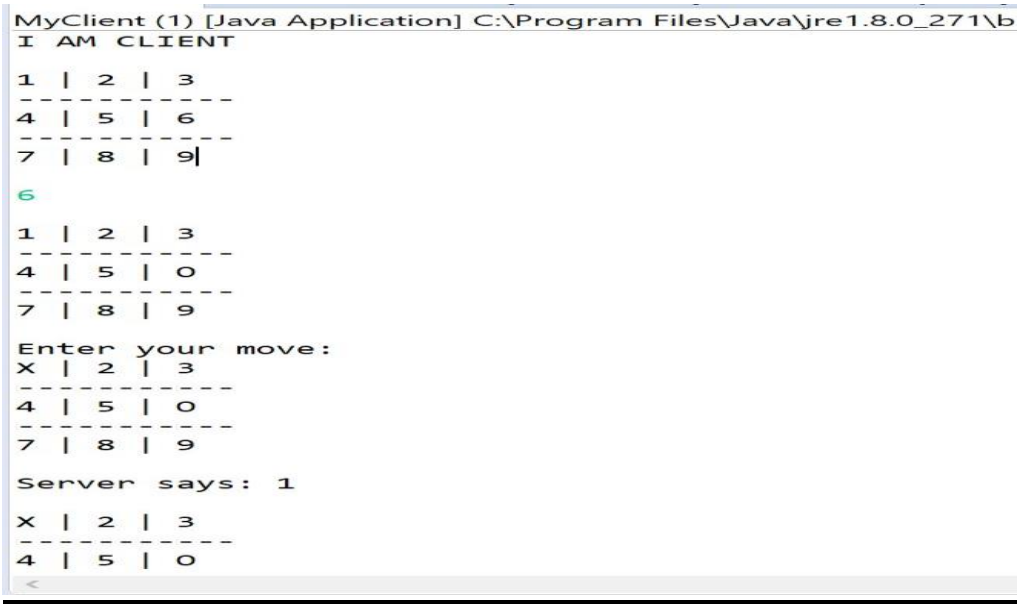
It has two parts **server side** and **client side.**

It is implemented using **JAVA SOCKETS.**

1.SERVER SIDE



2.CLIENT SIDE



CHAPTER 7: CONCLUSION AND FUTURE SCOPE

CONCLUSION : -

The project was completed successfully. Java is an easy and fun to work with Object Oriented Programming language. While developing this project I got to work with many concept of java, its libraries and algorithms.

The games is working perfectly fine without any known bugs .

The unbeatable mode is truly unbeatable!

FUTURE WORK : -

1.The speed of minimax algorithm and be improved using the concepts of alpha beta pruning.

2.The LAN playable mode can be extended to make it playable over internet.

3.Better GUI can be given.

4.More functions and modes can be added

REFERENCES

Minimax Algorithm in Game Theory | Set 1 (Introduction) - GeeksforGeeks

Java - Networking - Tutorialspoint

Converting Text to Speech in Java - GeeksforGeeks

Java JOptionPane - javatpoint

Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI - Finding optimal move) - GeeksforGeeks

```java
package presentation;
import java.util.Random;
import javax.swing.JOptionPane;
import java.util.Locale;
import javax.speech.Central;
import javax.speech.synthesis.Synthesizer;
import javax.speech.synthesis.SynthesizerModeDesc;
import java.util.Scanner;

public class TicTacToeConsole {
    char turn;
    int flag=0;
    char mBoard[] = {'1','2','3','4','5','6','7','8','9'};
    int BOARD_SIZE = 9;
    static char HUMAN_PLAYER = 'X';
    static char COMPUTER_PLAYER = 'O';
    Random mRand=new Random();

    public TicTacToeConsole() {
        System.out.println("MEDIUM MODE");
        speak("welcome to tictac toe");
        turn = HUMAN_PLAYER; // Human starts first
        int  win = 0;
        while (win == 0)     // Keep looping until someone wins or a tie
        {
            displayBoard();
            if (turn == HUMAN_PLAYER)
            {
                speak("your move");
                getUserMove();
                turn = COMPUTER_PLAYER;
            }
            else
            {
                getComputerMove();
                speak("moved");
                turn = HUMAN_PLAYER;
            }
        win = checkForWinner();}
        displayBoard();
        System.out.println();
        if (win == 1)
            {System.out.println("It's a tie.");
             speak("Its a tie");
             flag=1;
                }
        else if (win == 2) {
            System.out.println(HUMAN_PLAYER + " wins!");
            speak("Congratulations you have won the game");
                            }
        else if (win == 3)
            {System.out.println(COMPUTER_PLAYER + " wins!");
             speak("Computer won");}
        else
            System.out.println("There is a logic problem!");
        }
    void displayBoard() {
```

```
58          System.out.println();
59          System.out.println(mBoard[0] + " | " + mBoard[1] + " | " + mBoard[2]);
60          System.out.println("-----------");
61          System.out.println(mBoard[3] + " | " + mBoard[4] + " | " + mBoard[5]);
62          System.out.println("-----------");
63          System.out.println(mBoard[6] + " | " + mBoard[7] + " | " + mBoard[8]);
64          System.out.println();}
65       int checkForWinner() {
66          for (int i = 0; i <= 6; i += 3) {// Check horizontal wins
67              if (mBoard[i] == HUMAN_PLAYER &&
68                  mBoard[i+1] == HUMAN_PLAYER &&
69                  mBoard[i+2]== HUMAN_PLAYER)
70                  return 2;
71              if (mBoard[i] == COMPUTER_PLAYER &&
72                  mBoard[i+1]== COMPUTER_PLAYER &&
73                  mBoard[i+2] == COMPUTER_PLAYER)
74                  return 3;
75          }
76          for (int i = 0; i <= 2; i++) {// Check vertical wins
77              if (mBoard[i] == HUMAN_PLAYER &&
78                  mBoard[i+3] == HUMAN_PLAYER &&
79                  mBoard[i+6]== HUMAN_PLAYER)
80                  return 2;
81              if (mBoard[i] == COMPUTER_PLAYER &&
82                  mBoard[i+3] == COMPUTER_PLAYER &&
83                  mBoard[i+6]== COMPUTER_PLAYER)
84                  return 3;
85          }
86          if ((mBoard[0] == HUMAN_PLAYER &&// Check for diagonal wins
87               mBoard[4] == HUMAN_PLAYER &&
88               mBoard[8] == HUMAN_PLAYER) ||
89              (mBoard[2] == HUMAN_PLAYER &&
90               mBoard[4] == HUMAN_PLAYER &&
91               mBoard[6] == HUMAN_PLAYER))
92              return 2;
93          if ((mBoard[0] == COMPUTER_PLAYER &&
94               mBoard[4] == COMPUTER_PLAYER &&
95               mBoard[8] == COMPUTER_PLAYER) ||
96              (mBoard[2] == COMPUTER_PLAYER &&
97               mBoard[4] == COMPUTER_PLAYER &&
98               mBoard[6] == COMPUTER_PLAYER))
99              return 3;
100
101         for (int i = 0; i < BOARD_SIZE; i++) {// Check for tie
102             if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
103                 return 0;
104         }
105         return 1;}// If we make it through the previous loop, all places are taken, so it's a
    tie
106
107     void getUserMove()
108     {
109         int move = -1;
110         while ( move == -1)
111         try{ {
112             String response = JOptionPane.showInputDialog(null,"Enter a no. between 1 and
    9");
```

```
113                 move = Integer.parseInt(response);
114
115             while (move < 1 || move > BOARD_SIZE || mBoard[move-1] == HUMAN_PLAYER ||
    mBoard[move-1] == COMPUTER_PLAYER) {
116
117                     if (move < 1 || move > BOARD_SIZE)
118                         System.out.println("Please enter a move between 1 and " + BOARD_SIZE +
    ".");
119                     else
120                         System.out.println("That space is occupied.  Please choose another
    space.");
121
122                     System.out.print("Enter your move: ");
123
124                      response = JOptionPane.showInputDialog(null,
125                          "Enter a no. between 1 and 9");
126                       move = Integer.parseInt(response);}
127             }}catch (NumberFormatException ex) {
128                 System.out.println("exiting");
129                 System.exit(0);}
130         mBoard[move-1] = HUMAN_PLAYER;
131     }
132     void getComputerMove()
133     {
134         int move;
135
136         // First see if there's a move O can make to win
137         for (int i = 0; i < BOARD_SIZE; i++) {
138             if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER) {
139                 char current = mBoard[i];
140                 mBoard[i] = COMPUTER_PLAYER;
141                 if (checkForWinner() == 3) {
142                     System.out.println("Computer is moving to " + (i + 1));
143                     return;
144                 }
145                 else
146                     mBoard[i] = current;
147             }
148         }
149
150         // See if there's a move O can make to block X from winning
151         for (int i = 0; i < BOARD_SIZE; i++) {
152             if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER) {
153                 char curr = mBoard[i];
154                 mBoard[i] = HUMAN_PLAYER;
155                 if (checkForWinner() == 2) {
156                     mBoard[i] = COMPUTER_PLAYER;
157                     System.out.println("Computer is moving to " + (i + 1));
158                     return;
159                 }
160                 else
161                     mBoard[i] = curr;
162             }
163         }
164         do// Generate random move
165         {
166             move = mRand.nextInt(BOARD_SIZE);
```

```
167         } while (mBoard[move] == HUMAN_PLAYER || mBoard[move] == COMPUTER_PLAYER);
168
169         System.out.println("Computer is moving to " + (move + 1));
170         mBoard[move] = COMPUTER_PLAYER;}
171     void speak(String k){
172      try{
173         // Set property as Kevin Dictionary
174         System.setProperty(
175             "freetts.voices", "com.sun.speech.freetts.en.us"+
    ".cmu_us_kal.KevinVoiceDirectory");
176         // Register Engine
177         Central.registerEngineCentral(
    "com.sun.speech.freetts"+".jsapi.FreeTTSEngineCentral");
178         // Create a Synthesizer
179         Synthesizer synthesizer = Central.createSynthesizer(new
    SynthesizerModeDesc(Locale.US));
180         // Allocate synthesizer
181          synthesizer.allocate();
182         // Resume Synthesizer
183          synthesizer.resume();
184         // Speaks the given text  until the queue is empty.
185         synthesizer.speak(k, null);
186        }
187        catch(Exception e) { e.printStackTrace();  }}
188
189 public static void main(String[] args) throws Exception{
190       System.out.println("xxxxxxxxxxxxTIC TAC TOE CONSOLExxxxxxxxxxxxxxx ");
191       System.out.println("Enter the mode in which you want to play the game");
192       System.out.println("Enter 1 for Medium difficulty Mode AI opponent");
193       System.out.println("Enter 2 for Unbeatable Expert Mode AI opponent ");
194       System.out.println("Enter 3 to play with a friend");
195       System.out.println("Enter any other no. to exit");
196       System.out.println("Instructions-Player marker is X");
197       System.out.println("              Enter the position where you want to move your mark");
198        Scanner sc= new Scanner(System.in);
199        int n;
200        n= sc.nextInt();
201        sc.close();
202        switch (n) {
203        case 1:
204            new TicTacToeConsole();
205            break;
206        case 2:
207            new Tic();
208            break;
209        case 3:
210            new Turnbyturn();
211            break;
212        default:
213            System.exit(0);}}}
```

```java
package presentation;
import javax.swing.JOptionPane;
import java.util.Locale;
import javax.speech.Central;
import javax.speech.synthesis.Synthesizer;
import javax.speech.synthesis.SynthesizerModeDesc;
public class Tic{
    char mBoard[] = {'1','2','3','4','5','6','7','8','9'};
    int BOARD_SIZE = 9;
    static char HUMAN_PLAYER = 'X';
    static char COMPUTER_PLAYER = 'O';
        Tic() {
        System.out.println("UNBEATABLE MODE");
        speak("Welcome to UNBEATABLE tictac toe");
        char turn = HUMAN_PLAYER;  // Human starts first
        int  win = 0;              // Set to 100,-100, or0 when game is over
        while (win == 0)           // Keep looping until someone wins or a tie
        {
            displayBoard();
            if (turn == HUMAN_PLAYER)
            {   speak("your move");
                getUserMove();
                turn = COMPUTER_PLAYER;
            }
            else
            {
                getComputerMove();
                speak("moved");
                turn = HUMAN_PLAYER;
            }
            win = checkForWinner();}
        displayBoard();
        System.out.println();
        if (win == 2){                          // Report the winner
            System.out.println("It's a tie.");
            speak("Its a tie");           }
        else if (win == -100) {
            System.out.println(HUMAN_PLAYER + " wins!");
              speak("Congratulations you won the game");}
        else if (win == 100) {
            System.out.println(COMPUTER_PLAYER + " wins!");
               speak("Computer player won");
                           }
        else
            System.out.println("There is a logic problem!");
           }
    void displayBoard() {
        System.out.println();
        System.out.println(mBoard[0] + " | " + mBoard[1] + " | " + mBoard[2]);
        System.out.println("-----------");
        System.out.println(mBoard[3] + " | " + mBoard[4] + " | " + mBoard[5]);
        System.out.println("-----------");
        System.out.println(mBoard[6] + " | " + mBoard[7] + " | " + mBoard[8]);
        System.out.println();
    }
     int checkForWinner() {
        for (int i = 0; i <= 6; i += 3) {       // Check horizontal wins
```

```
58            if (mBoard[i] == HUMAN_PLAYER &&
59                mBoard[i+1] == HUMAN_PLAYER &&
60                mBoard[i+2]== HUMAN_PLAYER)
61                return -100;
62            if (mBoard[i] == COMPUTER_PLAYER &&
63                mBoard[i+1]== COMPUTER_PLAYER &&
64                mBoard[i+2] == COMPUTER_PLAYER)
65                return 100;
66        }
67        for (int i = 0; i <= 2; i++) {          // Check horizontal wins
68            if (mBoard[i] == HUMAN_PLAYER &&
69                mBoard[i+3] == HUMAN_PLAYER &&
70                mBoard[i+6]== HUMAN_PLAYER)
71                return -100;
72            if (mBoard[i] == COMPUTER_PLAYER &&
73                mBoard[i+3] == COMPUTER_PLAYER &&
74                mBoard[i+6]== COMPUTER_PLAYER)
75                return 100;
76        }
77        if ((mBoard[0] == HUMAN_PLAYER &&      // Check for diagonal wins
78             mBoard[4] == HUMAN_PLAYER &&
79             mBoard[8] == HUMAN_PLAYER) ||
80            (mBoard[2] == HUMAN_PLAYER &&
81             mBoard[4] == HUMAN_PLAYER &&
82             mBoard[6] == HUMAN_PLAYER))
83            return -100;
84        if ((mBoard[0] == COMPUTER_PLAYER &&
85             mBoard[4] == COMPUTER_PLAYER &&
86             mBoard[8] == COMPUTER_PLAYER) ||
87            (mBoard[2] == COMPUTER_PLAYER &&
88             mBoard[4] == COMPUTER_PLAYER &&
89             mBoard[6] == COMPUTER_PLAYER))
90            return 100;
91        for (int i = 0; i < BOARD_SIZE; i++) {  // Check for tie
92            if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
93                return 0;
94        }
95          return 2;// If we make it through the previous loop, all places are taken, so it's a
  tie
96     }
97    void getUserMove()
98    {
99        int move = -1;
100       while ( move == -1)
101       try{ {
102            String response = JOptionPane.showInputDialog(null,"Enter a no. between 1 and
  9");
103               move = Integer.parseInt(response);
104
105           while (move < 1 || move > BOARD_SIZE || mBoard[move-1] == HUMAN_PLAYER ||
  mBoard[move-1] == COMPUTER_PLAYER) {
106
107                   if (move < 1 || move > BOARD_SIZE)
108                       System.out.println("Please enter a move between 1 and " + BOARD_SIZE +
  ".");
109                   else
110                       System.out.println("That space is occupied.  Please choose another
```

```
   space.");
111
112                     System.out.print("Enter your move: ");
113
114                      response = JOptionPane.showInputDialog(null,
115                          "Enter a no. between 1 and 9");
116                       move = Integer.parseInt(response);}
117             }}catch (NumberFormatException ex) {
118                 System.out.println("exiting");
119                 System.exit(0);
120             }
121         mBoard[move-1] = HUMAN_PLAYER;
122     }
123 int minimax(int depth,Boolean isMax)
124 {
125 int score = checkForWinner() ;
126
127 if (score==100)
128 return score;
129
130 if (score==-100)
131 return score;
132
133 if(score==2)
134 return 0;
135 if (isMax)// If this maximizer's move
136 {
137 int best =-10000;
138 // Traverse all cells
139     for (int j = 0; j<9; j++)
140     {
141         if (mBoard[j] != HUMAN_PLAYER && mBoard[j] != COMPUTER_PLAYER)
142         {
143             // Make the move
144             char curr = mBoard[j];
145             mBoard[j] = COMPUTER_PLAYER;
146             // Call minimax recursively and choose  the maximum value
147             best = Math.max(best,minimax(depth+1,false));
148             // Undo the move
149             mBoard[j]=curr;
150         }}
151 return best-depth;
152 }
153 else
154 {
155 int best = 10000;
156     for (int j = 0; j <9; j++)
157     {
158         // Call minimax recursively and choose
159         // the minimum value
160         if (mBoard[j] != HUMAN_PLAYER && mBoard[j] != COMPUTER_PLAYER)
161         {
162             char curr = mBoard[j];  // Make the move
163             mBoard[j] = HUMAN_PLAYER;
164             best = Math.min(best, minimax(depth+1,true));
165             mBoard[j]=curr;         // Undo the move
166         }
```

```
167     }
168 return best+depth; }
169 }
170     void getComputerMove()
171     {
172         int bestVal = -10000;
173         int j=-20;
174         char curr;
175         for (int i = 0; i <9; i++) {
176             if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
177             {
178               curr=mBoard[i];
179               mBoard[i]=COMPUTER_PLAYER;
180                 int moveVal= minimax(0,false);
181                 mBoard[i]=curr;
182                 if (moveVal > bestVal)
183                 {
184                         j=i;
185                         bestVal = moveVal;}
186             }}System.out.println("Computer is moving to " + (j+ 1));
187         mBoard[j]=COMPUTER_PLAYER;}
188     void speak(String k){
189     try{
190         // Set property as Kevin Dictionary
191         System.setProperty(
192             "freetts.voices", "com.sun.speech.freetts.en.us"+
    ".cmu_us_kal.KevinVoiceDirectory");
193         // Register Engine
194         Central.registerEngineCentral(
    "com.sun.speech.freetts"+".jsapi.FreeTTSEngineCentral");
195         // Create a Synthesizer
196         Synthesizer synthesizer = Central.createSynthesizer(new
    SynthesizerModeDesc(Locale.US));
197         // Allocate synthesizer
198         synthesizer.allocate();
199         // Resume Synthesizer
200         synthesizer.resume();
201         // Speaks the given text
202         // until the queue is empty.
203         synthesizer.speak(k, null);
204          }
205       catch(Exception e) {
206             e.printStackTrace();
207             }}}
```

```java
package presentation;
import javax.swing.JOptionPane;
public class Turnbyturn{
    char mBoard[] = {'1','2','3','4','5','6','7','8','9'};
    int BOARD_SIZE = 9;
     static char HUMAN_PLAYER = 'X';
     static char COMPUTER_PLAYER = 'O';
        Turnbyturn() {
        speak("Welcome to tictac toe");
        char turn = HUMAN_PLAYER;          // Human starts first
        int  win = 0;                      // Set to 100,-100, or0 when game is over
        while (win == 0)
        {
            displayBoard();
            if (turn == HUMAN_PLAYER)
            {   speak("player1 move");
                getUserMove();
                turn = COMPUTER_PLAYER;
            }
            else
            {   speak("player 2 move");
                getComputerMove();
                turn = HUMAN_PLAYER;
            }
            win = checkForWinner();
        }
        displayBoard();
        System.out.println();
        if (win == 2){
            System.out.println("It's a tie.");
            speak("Its a tie"); }
        else if (win == -100) {
            System.out.println(HUMAN_PLAYER + " wins!");
            speak("Congratulations player 1 won the game");
                                }
        else if (win == 100) {
            System.out.println(COMPUTER_PLAYER + " wins!");
            speak("player 2 won the game");
                                }
        else
            System.out.println("There is a logic problem!");
        }
    void displayBoard() {
        System.out.println();
        System.out.println(mBoard[0] + " | " + mBoard[1] + " | " + mBoard[2]);
        System.out.println("-----------");
        System.out.println(mBoard[3] + " | " + mBoard[4] + " | " + mBoard[5]);
        System.out.println("-----------");
        System.out.println(mBoard[6] + " | " + mBoard[7] + " | " + mBoard[8]);
        System.out.println();
    }
     int checkForWinner() {
        for (int i = 0; i <= 6; i += 3) {          // Check horizontal wins
            if (mBoard[i] == HUMAN_PLAYER &&
                mBoard[i+1] == HUMAN_PLAYER &&
                mBoard[i+2]== HUMAN_PLAYER)
                return -100;
```

```
62              if (mBoard[i] == COMPUTER_PLAYER &&
63                  mBoard[i+1]== COMPUTER_PLAYER &&
64                  mBoard[i+2] == COMPUTER_PLAYER)
65                  return 100;
66          }
67          for (int i = 0; i <= 2; i++) {           // Check vertical wins
68              if (mBoard[i] == HUMAN_PLAYER &&
69                  mBoard[i+3] == HUMAN_PLAYER &&
70                  mBoard[i+6]== HUMAN_PLAYER)
71                  return -100;
72              if (mBoard[i] == COMPUTER_PLAYER &&
73                  mBoard[i+3] == COMPUTER_PLAYER &&
74                  mBoard[i+6]== COMPUTER_PLAYER)
75                  return 100;
76          }
77          if ((mBoard[0] == HUMAN_PLAYER &&       // Check for diagonal wins
78               mBoard[4] == HUMAN_PLAYER &&
79               mBoard[8] == HUMAN_PLAYER) ||
80              (mBoard[2] == HUMAN_PLAYER &&
81               mBoard[4] == HUMAN_PLAYER &&
82               mBoard[6] == HUMAN_PLAYER))
83              return -100;
84          if ((mBoard[0] == COMPUTER_PLAYER &&
85               mBoard[4] == COMPUTER_PLAYER &&
86               mBoard[8] == COMPUTER_PLAYER) ||
87              (mBoard[2] == COMPUTER_PLAYER &&
88               mBoard[4] == COMPUTER_PLAYER &&
89               mBoard[6] == COMPUTER_PLAYER))
90              return 100;
91          for (int i = 0; i < BOARD_SIZE; i++) {  // Check for tie
92              if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
93                  return 0;
94          }
95            return 2;                             // If we make it through the previous loop, all
    places are taken, so it's a tie
96       }
97      void getUserMove()
98      { int move = -1;
99      while ( move == -1)
100     try{ {
101          String response = JOptionPane.showInputDialog(null,"Enter a no. between 1 and 9");
102             move = Integer.parseInt(response);
103
104         while (move < 1 || move > BOARD_SIZE || mBoard[move-1] == HUMAN_PLAYER ||
    mBoard[move-1] == COMPUTER_PLAYER) {
105
106                 if (move < 1 || move > BOARD_SIZE)
107                     System.out.println("Please enter a move between 1 and " + BOARD_SIZE +
    ".");
108                 else
109                     System.out.println("That space is occupied.  Please choose another
    space.");
110
111                 System.out.print("Enter your move: ");
112
113                  response = JOptionPane.showInputDialog(null,
114                      "Enter a no. between 1 and 9");
```

```
115                 move = Integer.parseInt(response);}
116         }}catch (NumberFormatException ex) {
117             System.out.println("exiting");
118             System.exit(0); }
119         mBoard[move-1] = HUMAN_PLAYER;
120     }
121     void getComputerMove(){
122     int move = -1;
123     while ( move == -1)
124     try{ {
125          String response = JOptionPane.showInputDialog(null,"Enter a no. between 1 and 9");
126             move = Integer.parseInt(response);
127 
128         while (move < 1 || move > BOARD_SIZE || mBoard[move-1] == HUMAN_PLAYER ||
    mBoard[move-1] == COMPUTER_PLAYER) {
129 
130                 if (move < 1 || move > BOARD_SIZE)
131                     System.out.println("Please enter a move between 1 and " + BOARD_SIZE +
    ".");
132                 else
133                     System.out.println("That space is occupied.  Please choose another
    space.");
134 
135                 System.out.print("Enter your move: ");
136 
137                  response = JOptionPane.showInputDialog(null,
138                      "Enter a no. between 1 and 9");
139                   move = Integer.parseInt(response);}
140         }}catch (NumberFormatException ex) {
141             System.out.println("exiting");
142             System.exit(0);
143         }
144             mBoard[move-1] = COMPUTER_PLAYER;
145     }
146     void speak(String k){
147     try{// Set property as Kevin Dictionary
148         System.setProperty("freetts.voices", "com.sun.speech.freetts.en.us"+
    ".cmu_us_kal.KevinVoiceDirectory");
149         // Register Engine
150         Central.registerEngineCentral(
    "com.sun.speech.freetts"+".jsapi.FreeTTSEngineCentral");
151         // Create a Synthesizer
152         Synthesizer synthesizer = Central.createSynthesizer(new
    SynthesizerModeDesc(Locale.US));
153         // Allocate synthesizer
154         synthesizer.allocate();
155         // Resume Synthesizer
156         synthesizer.resume();
157         // Speaks the given text  until the queue is empty.
158         synthesizer.speak(k, null);
159          }
160       catch(Exception e) {
161             e.printStackTrace();
162             }}}
163 
```

```java
package presentation;
import java.util.Scanner;
import java.net.*;
import java.io.*;
public class SERVER2{
static String stop;
static char mBoard[] = {'1','2','3','4','5','6','7','8','9'};
static int BOARD_SIZE = 9;
 final static char HUMAN_PLAYER = 'X';
 final static char COMPUTER_PLAYER = 'O';
public static void main(String args[])throws Exception{
    System.out.println("Enter stop to terminate");
    System.out.println("I AM SERVER");
    Scanner sc= new Scanner(System.in);
    ServerSocket ss=new ServerSocket(5000);
    Socket s=ss.accept();
    DataInputStream din=new DataInputStream(s.getInputStream());
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    String str="",str3="";
    while(!str.equals("stop"))
    {displayBoard();
    str=din.readUTF();
    getComputerMove(str);
    displayBoard();
    checkForWinner();
    System.out.println("client says: "+str);
    str3=sc.nextLine();
    dout.writeUTF(str3);
    getUserMove(str3);
    displayBoard();
    checkForWinner();
    dout.flush();}
    din.close();
    s.close();
    ss.close();
    sc.close();
    }
static void displayBoard()  {
System.out.println();
System.out.println(mBoard[0] + " | " + mBoard[1] + " | " + mBoard[2]);
System.out.println("-----------");
System.out.println(mBoard[3] + " | " + mBoard[4] + " | " + mBoard[5]);
System.out.println("-----------");
System.out.println(mBoard[6] + " | " + mBoard[7] + " | " + mBoard[8]);
System.out.println();
}
static void checkForWinner() {
for (int i = 0; i <= 6; i += 3) {// Check horizontal wins
if (mBoard[i] == HUMAN_PLAYER &&
    mBoard[i+1] == HUMAN_PLAYER &&
    mBoard[i+2]== HUMAN_PLAYER)
    {System.out.println("Host won");
    }
if (mBoard[i] == COMPUTER_PLAYER &&
    mBoard[i+1]== COMPUTER_PLAYER &&
    mBoard[i+2] == COMPUTER_PLAYER)
    {System.out.println("client won");
```

```
58    }
59}
60for (int i = 0; i <= 2; i++) {// Check vertical wins
61if (mBoard[i] == HUMAN_PLAYER &&
62    mBoard[i+3] == HUMAN_PLAYER &&
63    mBoard[i+6]== HUMAN_PLAYER)
64    {System.out.println("Host won");}
65if (mBoard[i] == COMPUTER_PLAYER &&
66    mBoard[i+3] == COMPUTER_PLAYER &&
67    mBoard[i+6]== COMPUTER_PLAYER)
68    System.out.println("client won");}
69// Check for diagonal wins
70if ((mBoard[0] == HUMAN_PLAYER &&
71 mBoard[4] == HUMAN_PLAYER &&
72 mBoard[8] == HUMAN_PLAYER) ||
73(mBoard[2] == HUMAN_PLAYER &&
74 mBoard[4] == HUMAN_PLAYER &&
75 mBoard[6] == HUMAN_PLAYER))
76    {System.out.println("Host won");}
77if ((mBoard[0] == COMPUTER_PLAYER &&
78 mBoard[4] == COMPUTER_PLAYER &&
79 mBoard[8] == COMPUTER_PLAYER) ||
80(mBoard[2] == COMPUTER_PLAYER &&
81 mBoard[4] == COMPUTER_PLAYER &&
82 mBoard[6] == COMPUTER_PLAYER))
83 {System.out.println("client won");}// Check for tie
84for (int i = 0; i < BOARD_SIZE; i++) {
85// If we find a number, then no one has won yet
86if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
87     {break;}
88     }}
89static void getUserMove(String str6)
90{     int move=Integer.parseInt(str6);
91      mBoard[move-1]=HUMAN_PLAYER;  }
92static void getComputerMove(String str2){
93int j = Integer.parseInt(str2);
94mBoard[j-1]=COMPUTER_PLAYER;}
95}
```

```java
package presentation;
import java.util.Scanner;
import java.net.*;
import java.io.*;
public class MyClient{
int win=0;
static char mBoard[] = {'1','2','3','4','5','6','7','8','9'};
static int BOARD_SIZE = 9;
 final static char HUMAN_PLAYER = 'X';
 final static char COMPUTER_PLAYER = 'O';
public static void main(String args[])throws Exception{
System.out.println("I AM CLIENT");
Scanner sc= new Scanner(System.in);
Socket s=new Socket("localhost",5000);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
String str="",str2="";
while(!str.equals("stop"))
{
displayBoard();
str=sc.nextLine();
getComputerMove(str);
dout.writeUTF(str);
dout.flush();
displayBoard();
str2=din.readUTF();
getUserMove(str2);
displayBoard();
System.out.println("Server says: "+str2);
}
dout.close();
s.close();
sc.close();
}
static void displayBoard()  {
System.out.println();
System.out.println(mBoard[0] + " | " + mBoard[1] + " | " + mBoard[2]);
System.out.println("-----------");
System.out.println(mBoard[3] + " | " + mBoard[4] + " | " + mBoard[5]);
System.out.println("-----------");
System.out.println(mBoard[6] + " | " + mBoard[7] + " | " + mBoard[8]);
System.out.println();
}
int checkForWinner() {
for (int i = 0; i <= 6; i += 3) {
if (mBoard[i] == HUMAN_PLAYER &&
    mBoard[i+1] == HUMAN_PLAYER &&
    mBoard[i+2]== HUMAN_PLAYER)
    return -100;
if (mBoard[i] == COMPUTER_PLAYER &&
    mBoard[i+1]== COMPUTER_PLAYER &&
    mBoard[i+2] == COMPUTER_PLAYER)
    return 100;
}
for (int i = 0; i <= 2; i++) {
if (mBoard[i] == HUMAN_PLAYER &&
    mBoard[i+3] == HUMAN_PLAYER &&
```

```
58    mBoard[i+6]== HUMAN_PLAYER)
59    return -100;
60if (mBoard[i] == COMPUTER_PLAYER &&
61    mBoard[i+3] == COMPUTER_PLAYER &&
62    mBoard[i+6]== COMPUTER_PLAYER)
63    return 100;
64}
65if ((mBoard[0] == HUMAN_PLAYER &&
66 mBoard[4] == HUMAN_PLAYER &&
67 mBoard[8] == HUMAN_PLAYER) ||
68(mBoard[2] == HUMAN_PLAYER &&
69 mBoard[4] == HUMAN_PLAYER &&
70 mBoard[6] == HUMAN_PLAYER))
71return -100;
72if ((mBoard[0] == COMPUTER_PLAYER &&
73 mBoard[4] == COMPUTER_PLAYER &&
74 mBoard[8] == COMPUTER_PLAYER) ||
75(mBoard[2] == COMPUTER_PLAYER &&
76 mBoard[4] == COMPUTER_PLAYER &&
77 mBoard[6] == COMPUTER_PLAYER))
78return 100;
79// Check for tie
80for (int i = 0; i < BOARD_SIZE; i++) {
81// If we find a number, then no one has won yet
82if (mBoard[i] != HUMAN_PLAYER && mBoard[i] != COMPUTER_PLAYER)
83    return 0;
84}
85// If we make it through the previous loop, all places are taken, so it's a tie
86return 2;
87}
88static void getUserMove(String str)
89{
90        System.out.print("Enter your move: ");
91         int move=Integer.parseInt(str);
92      mBoard[move-1]=HUMAN_PLAYER;
93}
94static void getComputerMove(String str2){
95int j = Integer.parseInt(str2);
96mBoard[j-1]=COMPUTER_PLAYER;}}
```