

Java Project

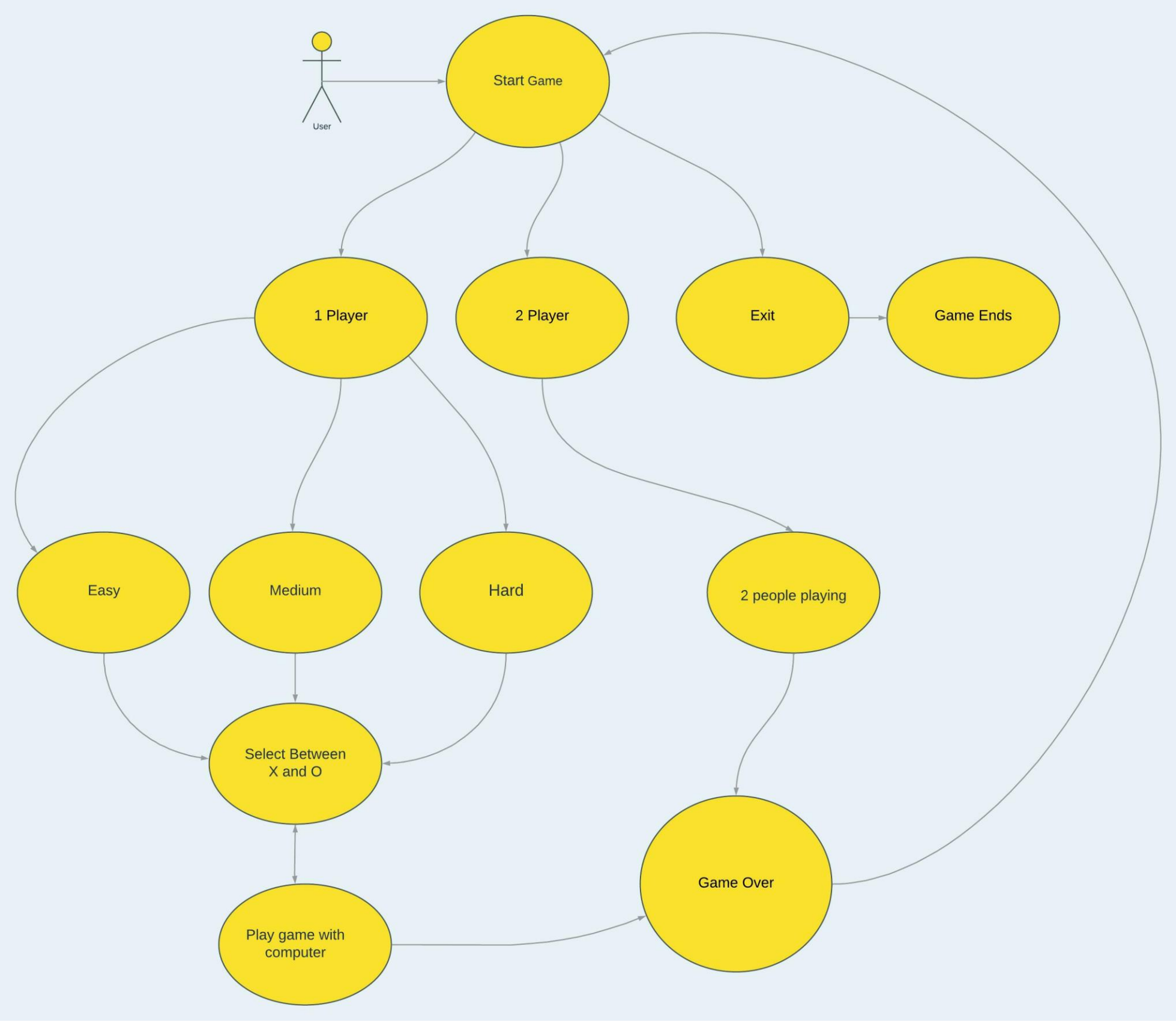
Tic Tac Toe with AI

Created by:

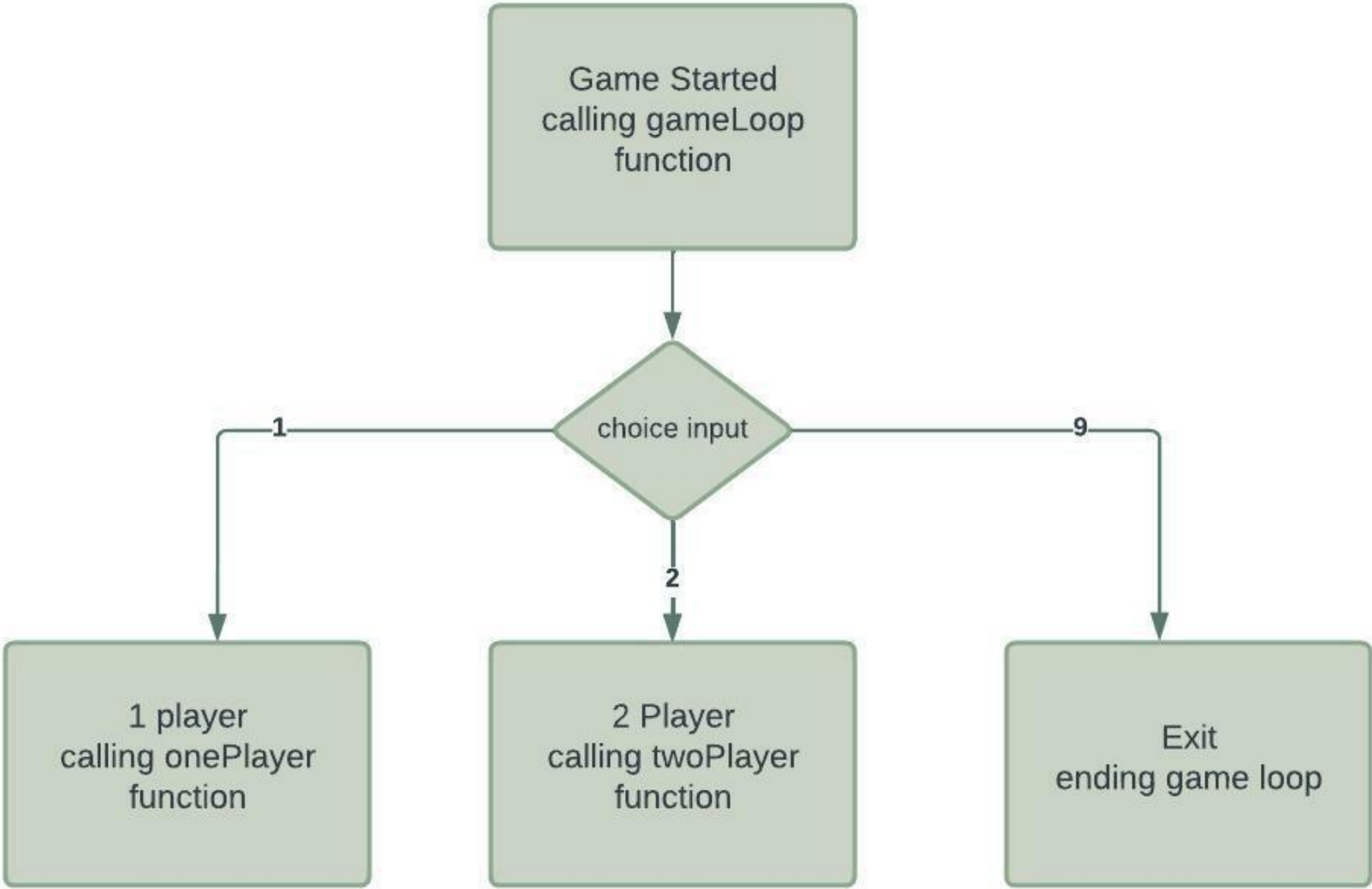
Pretty Verma and Tarun Chawla

Project Overview: Our project is on famous game Tic Tac Toe, which everyone has played in their childhood. This game can be a good time pass. You can play it with your friends and if you are alone you can play with computer, while playing with computer there are three levels and in hard level you are playing with AI, and at this level It is impossible to win, in this level your goal should be to tie the game, that’s the challenge in it. This game is developed in java 18.0.2.1 released on 2022-08-18 using VS code editor. I have only used utility classes of java such as random and scanner in my whole game. The AI part of the game use famous Minmax algorithm.

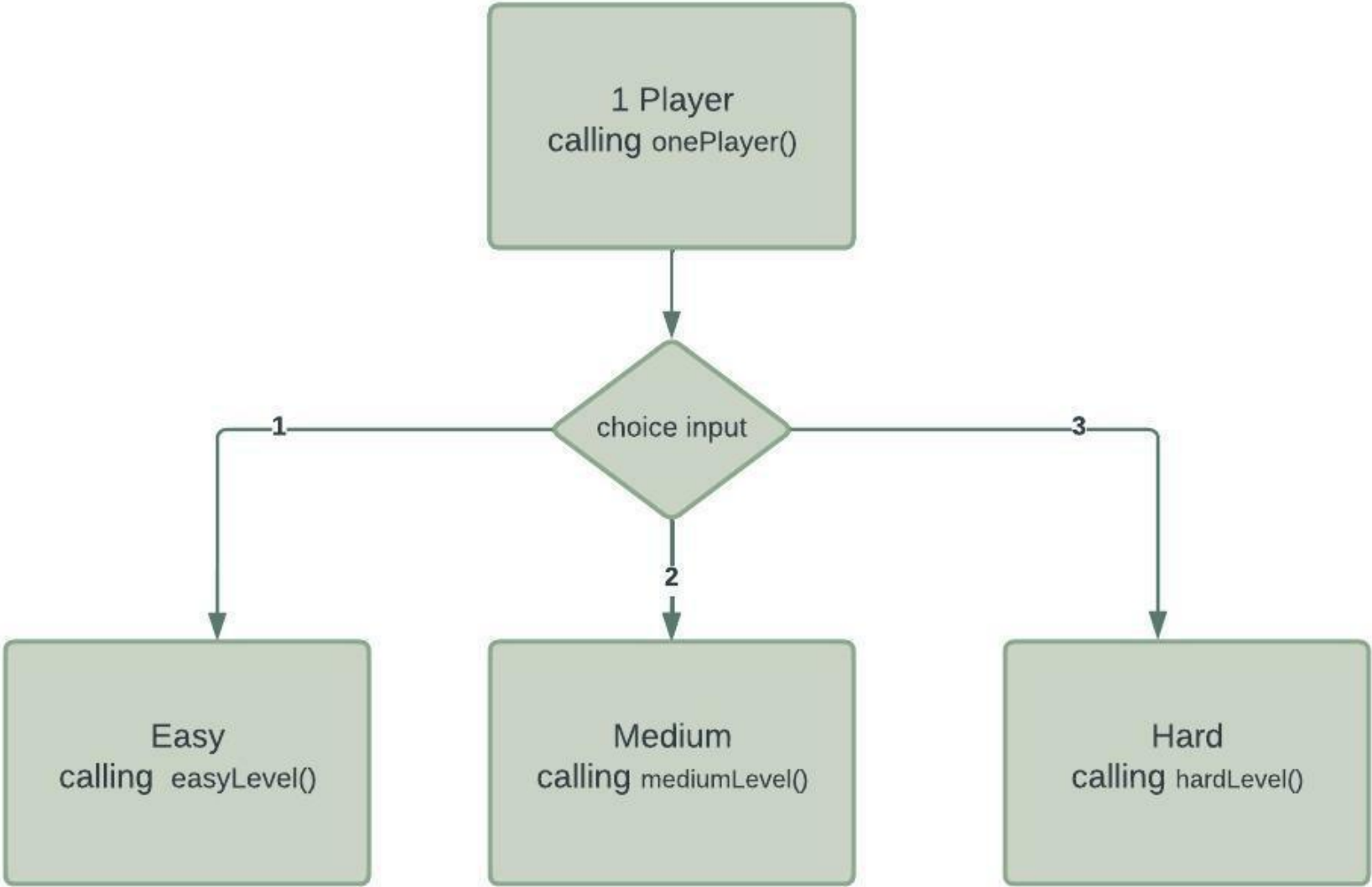
Game Diagram:



Block Diagrams:

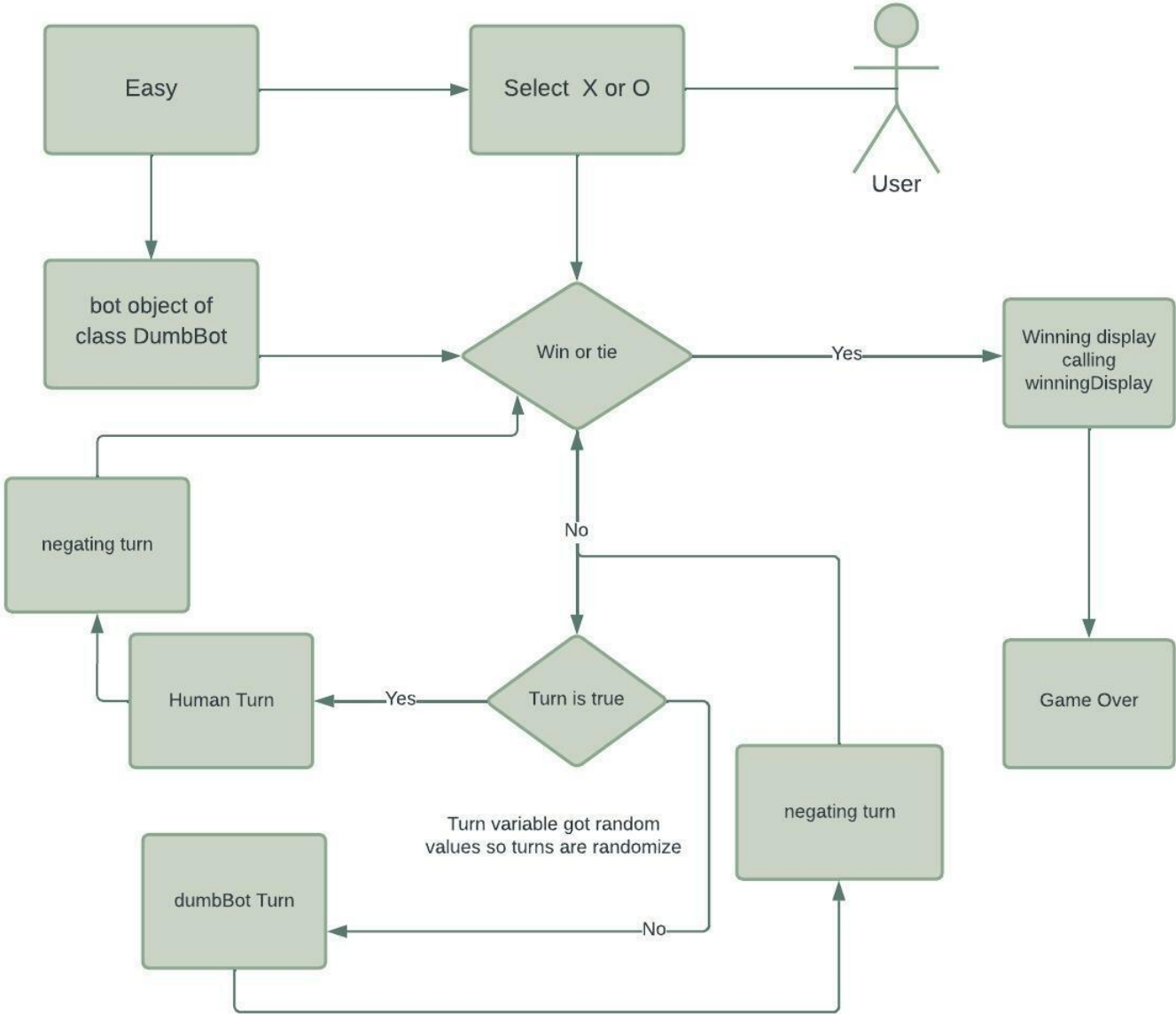


When user selected 1 Player:



Easy:

A board object of Board Class is created for generating game board and performing essential functions on it.

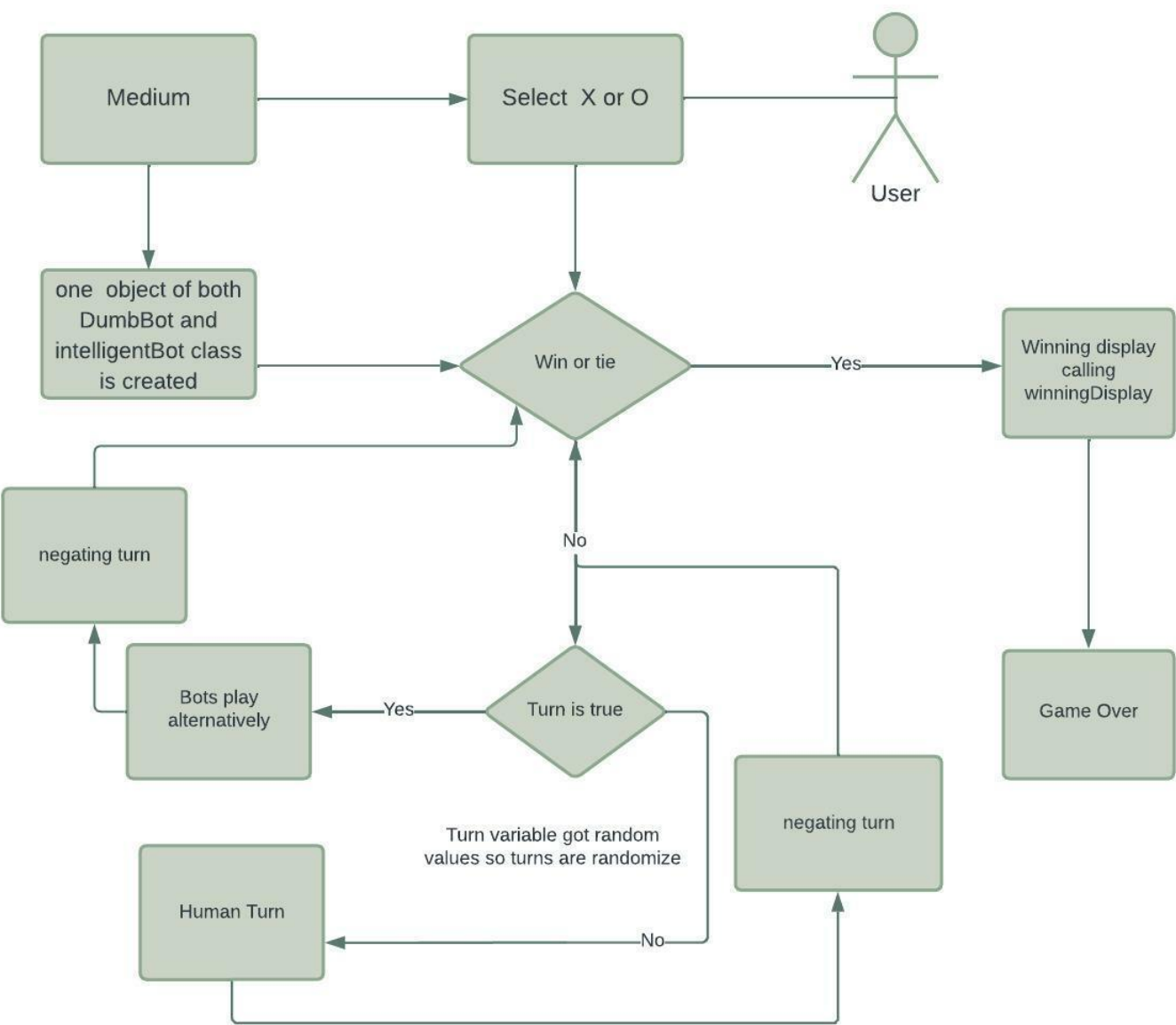


Medium:

A board object of Board Class is created for generating game board and performing essential functions on it.

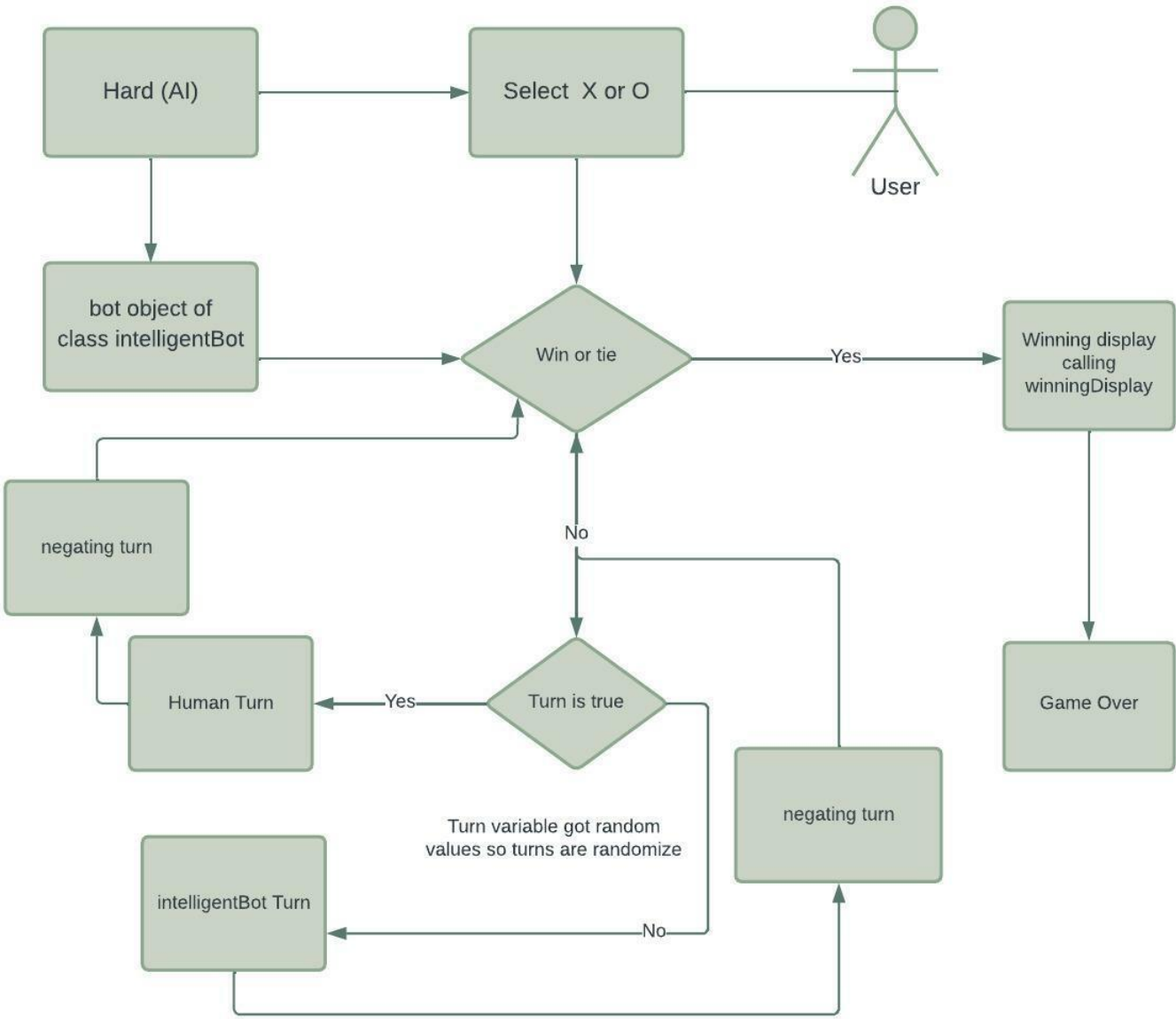
Dumbbot play its move randomly.

intelligentbot play its move using AI algorithm Minimax which is in Minimax class

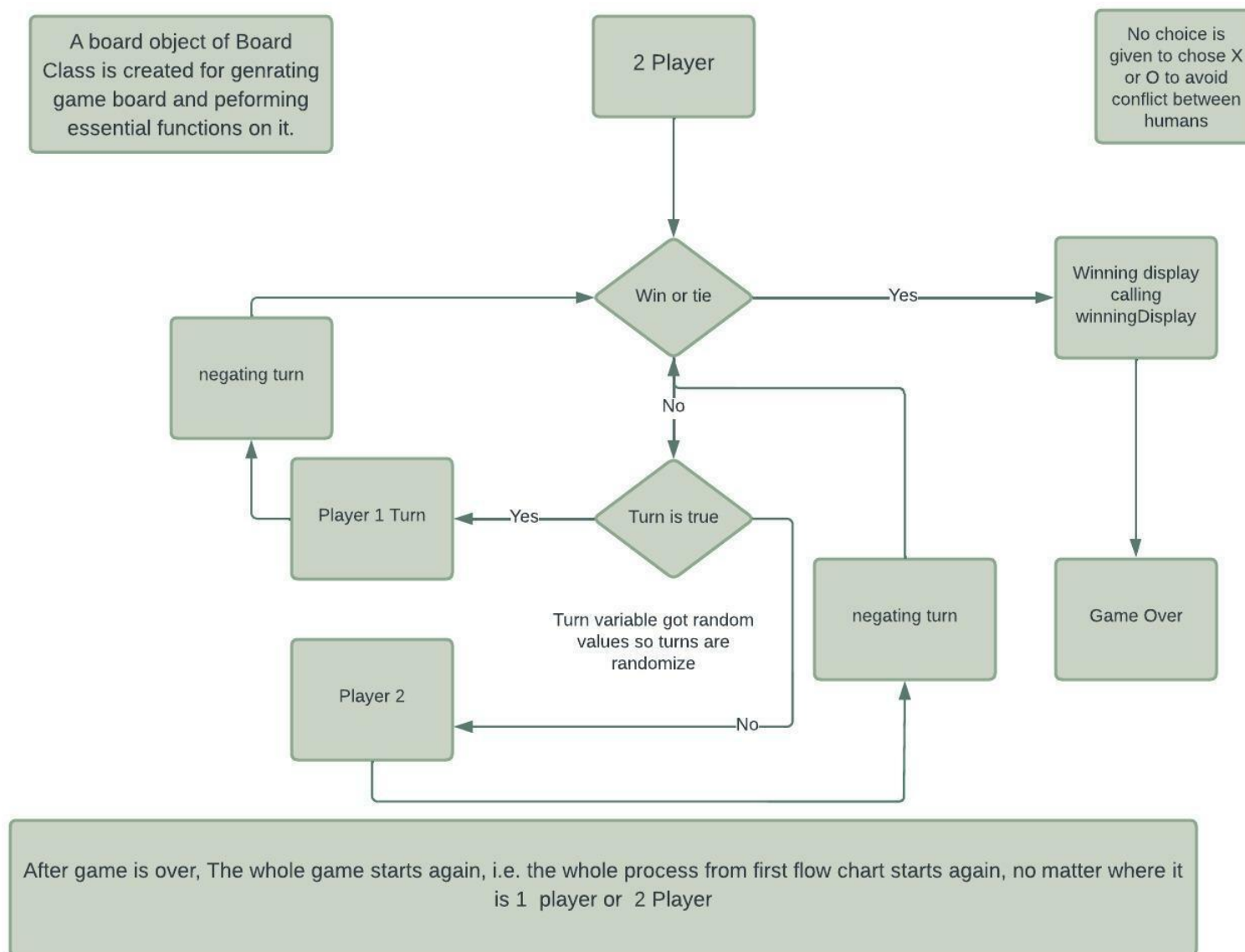


Hard:

A board object of Board Class is created for generating game board and performing essential functions on it.



When user selected 2 Player:



Work Contribution:

Pretty Verma:

- Created Board Class which contains all the essential methods such as marking and unmarking the board, displaying board and checking status of board (win or tie),
- Created DumbBot Class which has a method for making move and greeting.

Tarun Chawla:

- Created Game class which contains the main method and other methods such as gameloop, oneplayer, twoplayer, easylevel, mediumlevel, hardlevel, winningdisplay and checkinput.
- Created Intelligentbot class which has methods of making move on board and greeting. It uses the famous minimax algorithm to make move.
- Created Minimax class which has a static method which performs the minimax algorithm and returns a place on board where the computer should mark.

Code:

Board.java:

```
public class Board {  
    // 2D Array to store Making on the board;  
    private char board[][] = {  
        {' ', ' ', ' '},  
        {' ', ' ', ' '},  
        {' ', ' ', ' '}  
    };  
  
    // Function for print board on the screen.  
    public void displayBoard() {  
        // Printing two lines for Formatting  
        System.out.println();  
        System.out.println();  
        // for rows of board  
        for (int row = 0; row < board.length; row++) {  
            // for columns of board  
            // for spacing from left before printing first row of the board  
            System.out.print("      ");  
            for (int col = 0; col < board[row].length; col++) {  
                System.out.print(" " + board[row][col] + " ");  
                if (col < 2) {  
                    // Printing pipe symbol for separation of columns  
                    System.out.print(" | ");  
                }  
            }  
            System.out.println();  
            if (row < 2) {  
                System.out.println("      -----+-----+-----");  
            }  
        }  
    }  
}
```



```
// Printing two lines for Formatting
System.out.println();
System.out.println();
}
```

```
// marking the board with just number between 1 to 9
```

```
public boolean markBoard(int num, char markchar) {
    if (num >= 10 || num <= 0) {
        System.out.println("Wrong input!!\nPlease Enter value(1-9):");
        return false;
    }
    num--;
    int row = num / 3;
    int col = num % 3;
    if (board[row][col] != ' ') {
        System.out.println("Operation failed!!, Place is already occupied");
        return false;
    }

    board[row][col] = markchar;
    return true;
}
```

```
// as computer is going to mark according row and column, so overloaded marked
```

```
// function
```

```
// with functionality of making according to rows and columns instead of single
```

```
// number
```

```
public boolean markBoard(int row, int col, char markchar) {
    if (board[row][col] != ' ') {
        return false;
    }

    board[row][col] = markchar;
    return true;
}
```

```
}
```

```
// It will unmark the board at specific postion
```

```
public void unmarkBoard(int row, int col) {
```

```
    board[row][col] = ' ';
```

```
}
```

```
// It will return the character at given position.
```

```
public char getCharAt(int i, int j) {
```

```
    return board[i][j];
```

```
}
```

```
// This function is going to return the the status of board i.e. its check
```

```
// whather someone win or not or it is a tie
```

```
// It return 'X' if X player wins , 'O' if O player wins 't' if it is a tie and
```

```
// ' ' if nothing happened
```

```
public char boardStatus() {
```

```
    char winner = ' ';
```

```
    // checking Each Row for wining
```

```
    // for rows
```

```
    for (int i = 0; i < 3; i++) {
```

```
        if (board[i][0] != ' ' && board[i][0] == board[i][1]
```

```
            && board[i][1] == board[i][2]) {
```

```
            winner = board[i][0];
```

```
        }
```

```
    }
```

```
    for (int i = 0; i < 3; i++) {
```

```
        if (board[0][i] != ' ' && board[0][i] == board[1][i]
```

```
            && board[1][i] == board[2][i]) {
```

```
            winner = board[0][i];
```

```
        }
```

```
    }
```

```
    if (board[0][0] != ' ' && board[0][0] == board[1][1]
```

```
        && board[1][1] == board[2][2]) {
    winner = board[0][0];
} else if (board[2][0] != ' ' && board[0][2] == board[1][1]
        && board[1][1] == board[2][0]) {
    winner = board[2][0];
}
int openSpots = 0;
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        if (board[i][j] == ' ') {
            openSpots++;
        }
    }
}
if (winner == ' ' && openSpots == 0) {
    return 't';
}
return winner;
}
}
```

DumbBot.java:

```
import java.util.Random;

// Dumb bot is going to
public class DumbBot {
    private char playerChar;
    private String greeting = "\n\nHello, pal, I am dumb. \nMost likely you are going to win";

    public DumbBot(char playerChar) {
        this.playerChar = playerChar;
    }

    public void makeMove(Board board) {
        Random random = new Random();
        int num1 = random.nextInt(3);
        int num2 = random.nextInt(3);
        // marking untill the right spot got marked
        while (!board.markBoard(num1, num2, playerChar)) {
            num1 = random.nextInt(3);
            num2 = random.nextInt(3);
        }
        System.out.printf("I Played from my side at %d, it's your turn", num1 * 3 + num2 + 1);
    }

    // Function for intial greeting
    public void displayGreeting() {
        System.out.println(greeting);
    }
}
```

Minimax.java:

```
// Minimax class containing
public class Minimax {

    // the scores are arranged in the form of ai wins win, tie and human wins;

    static public int minimax(Board board, int depth, boolean maximizing, char human, char ai) {
        char winner = board.boardStatus();
        if (winner == ai) {
            return 1;
        }
        if (winner == human) {
            return -1;
        }
        if (winner == 't') {
            return 0;
        }
        // maximizing the score when computer is playing
        if (maximizing) {
            int bestScore = Integer.MIN_VALUE;
            for (int row = 0; row < 3; row++) {
                for (int col = 0; col < 3; col++) {
                    if (board.getCharAt(row, col) == ' ') {
                        board.markBoard(row, col, ai);
                        bestScore = Math.max(minimax(board, depth + 1, false, human, ai), bestScore);
                        board.unmarkBoard(row, col);
                    }
                }
            }
            return bestScore;
        } // else we have to minize the score when human is play
        else {
            int bestScore = Integer.MAX_VALUE;
            for (int row = 0; row < 3; row++) {
```

```
for (int col = 0; col < 3; col++) {  
    if (board.getCharAt(row, col) == ' ') {  
        board.markBoard(row, col, human);  
        bestScore = Math.min(minimax(board, depth + 1, true, human, ai), bestScore);  
        board.unmarkBoard(row, col);  
    }  
}  
return bestScore;  
}  
}
```

intelligentBot.java:

```
public class IntelligentBot {  
    private char playerChar;  
    private char humanChar;  
    private String greeting = "\n\nHello, pal, I my name is Smarty\nif you won you gonna get free  
Pizza.";  
  
    public IntelligentBot(char playerChar) {  
        this.playerChar = playerChar;  
        humanChar = (playerChar == 'X') ? 'O' : 'X';  
    }  
  
    // Function to make move aptimially using minimax algorithm.  
    public void makeMove(Board board) {  
        int move[] = new int[2];  
        int score = 0;  
        int bestScore = Integer.MIN_VALUE;  
        for (int row = 0; row < 3; row++) {  
            for (int col = 0; col < 3; col++) {  
                if (board.getCharAt(row, col) == ' ') {  
                    board.markBoard(row, col, playerChar);  
                    score = Minimax.minimax(board, 0, false, humanChar, playerChar);  
                    if (bestScore < score) {  
                        bestScore = score;  
                        move = new int[] { row, col };  
                    }  
                    board.unmarkBoard(row, col);  
                }  
            }  
        }  
  
        // Function for intial greeting  
        board.markBoard(move[0], move[1], playerChar);  
    }  
}
```

```
        System.out.printf("I Played from my side at %d, it's your turn\nhave a good one", move[0] * 3 +  
move[1] + 1);  
    }
```

```
    public void displayGreeting() {  
        System.out.println(greeting);  
    }  
}
```


game.java:

```
import java.util.Scanner;
import java.util.Random;

public class game {
    static Scanner input = new Scanner(System.in);
    static Random random = new Random();

    private static void gameloop() {
        boolean run = true;
        int choice = 0;
        while (run) {
            System.out.println("\n\nWelcome to Tic Tac Toe!!!\n");
            System.out.println("Choose the option:\n");
            System.out.println("1 - 1 Player");
            System.out.println("2 - 2 Player");
            System.out.println("9 - Exit");
            choice = input.nextInt();
            while (choice != 1 && choice != 2 && choice != 9) {
                System.out.println("Wrong Input, Please Try enter correct input!!!");
                choice = input.nextInt();
            }
            if (choice == 1) {
                onePlayer();

            } else if (choice == 2) {
                twoPlayer();

            } else if (choice == 9) {
                System.out.println("Bye, Have a good day :)");
            }
        }
    }
}
```

```
        run = false;
        return;
    }

}

}
```

```
static void twoPlayer() {
    char Player1 = 'X';
    char Player2 = 'O';
    System.out.println("\n\nPlayer 1: X");
    System.out.println("Player 2: O\n\n");
    char winner = ' ';
    boolean turn = random.nextBoolean();
    int num1;
    int num2;
    Board board = new Board();
    board.displayBoard();
    while (winner == ' ') {
        if (turn) {
            System.out.println("Player 1 turn, Mark in between(1-9) at empty spots:");
            try {
                num1 = input.nextInt();
            } catch (Exception e) {
                System.out.println("Wong Input!!");
                input.nextLine();
                continue;
            }
            if (board.markBoard(num1, Player1)) {
                turn = !turn;
            }
        }
    }
}
```

```

    }
} else {
    System.out.println("Player 2 turn, Mark in between(1-9) at empty spots:");
    try {
        num2 = input.nextInt();
    } catch (Exception e) {
        System.out.println("Wong Input!!");
        input.nextLine();
        continue;
    }
    if (board.markBoard(num2, Player2)) {
        turn = !turn;
    }
}
winner = board.boardStatus();
board.displayBoard();
}
System.out.println("Player " + winner + " Wins");
}

```

```

static void onePlayer() {
    System.out.println("\n\nPlease chose the level!!\n");
    System.out.println("1 - Easy");
    System.out.println("2 - Medium");
    System.out.println("3 - Hard");
    int choice = 0;
    choice = input.nextInt();
    while (choice != 1 && choice != 2 && choice != 3) {
        System.out.println("Enter the correct choice!!");
    }
}

```

```
        choice = input.nextInt();
    }
    if (choice == 1) {
        easyLevel();
    }
    if (choice == 2) {
        mediumLevel();
    }
    if (choice == 3) {
        hardLevel();
    }
}
```

```
static void hardLevel() {
    input.nextLine();
    System.out.println("Select from X or O: ");
    char Human = input.nextLine().toUpperCase().charAt(0);
    while (!checkInput(Human)) {
        System.out.println("Wrong Input, please select from X or O: ");
        Human = input.nextLine().toUpperCase().charAt(0);
    }
    char botchar = (Human == 'X') ? 'O' : 'X';
    IntelligentBot bot = new IntelligentBot(botchar);
    System.out.println("\n\nComputer: " + botchar + "\n");
    System.out.println("You: " + Human + "\n\n");
    bot.displayGreeting();
    Board board = new Board();
    char winner = ' ';
    boolean turn = random.nextBoolean();
}
```

```

int num;
board.displayBoard();
while (winner == ' ') {
    if (turn) {
        System.out.println("Human Player turn, Mark in between(1-9) at empty spots:");
        try {
            num = input.nextInt();
        } catch (Exception e) {
            System.out.println("Wong Input!!");
            input.nextLine();
            continue;
        }
        if (board.markBoard(num, Human)) {
            turn = !turn;
        }
    } else {
        bot.makeMove(board);
        turn = !turn;
    }
    winner = board.boardStatus();
    board.displayBoard();
}
winnerDisplay(winner);
}

```

```

static void mediumLevel() {
    input.nextLine();
    System.out.println("Select from X or O: ");
    char Human = input.nextLine().toUpperCase().charAt(0);
    while (!checkInput(Human)) {

```

```

        System.out.println("Wrong Input, please select from X or O: ");
        Human = input.nextLine().toUpperCase().charAt(0);
    }
    char botchar = (Human == 'X') ? 'O' : 'X';
    IntelligentBot ibot = new IntelligentBot(botchar);
    DumbBot dbot = new DumbBot(botchar);
    Board board = new Board();
    char winner = ' ';
    boolean turn = random.nextBoolean();
    boolean bturn = random.nextBoolean();
    int num;
    System.out.println("\n\nComputer: " + botchar + "\n");
    System.out.println("You: " + Human + "\n\n");
    System.out.println("Hello, All the best.");
    board.displayBoard();
    while (winner == ' ') {
        if (turn) {
            System.out.println("Human Player turn, Mark in between(1-9) at empty spots:");
            try {
                num = input.nextInt();
            } catch (Exception e) {
                System.out.println("Wong Input!!");
                input.nextLine();
                continue;
            }
            if (board.markBoard(num, Human)) {
                turn = !turn;
            }
        } else {
            if (bturn) {

```

```

        ibot.makeMove(board);
        bturn = !bturn;
    } else {
        dbot.makeMove(board);
        bturn = !bturn;
    }
    turn = !turn;
}
winner = board.boardStatus();
board.displayBoard();
}
winnerDisplay(winner);
}

```

```

static void easyLevel() {
    input.nextLine();
    System.out.println("Select from X or O: ");
    char Human = input.nextLine().toUpperCase().charAt(0);
    while (!checkInput(Human)) {
        System.out.println("Wrong Input, please select from X or O: ");
        Human = input.nextLine().toUpperCase().charAt(0);
    }
    char botchar = (Human == 'X') ? 'O' : 'X';
    DumbBot bot = new DumbBot(botchar);
    System.out.println("\n\nComputer: " + botchar + "\n");
    System.out.println("You: " + Human + "\n\n");
    bot.displayGreeting();
    Board board = new Board();
    char winner = ' ';
    boolean turn = random.nextBoolean();
}

```

```

int num;
board.displayBoard();
while (winner == ' ') {
    if (turn) {
        System.out.println("Human Player turn, Mark in between(1-9) at empty spots:");
        try {
            num = input.nextInt();
        } catch (Exception e) {
            System.out.println("Wong Input!!");
            input.nextLine();
            continue;
        }
        if (board.markBoard(num, Human)) {
            turn = !turn;
        }
    } else {
        bot.makeMove(board);
        turn = !turn;
    }
    winner = board.boardStatus();
    board.displayBoard();
}
winnerDisplay(winner);
}

```

```

public static void winnerDisplay(char win) {
    if (win == 'X') {
        System.out.println("\n----- :) Player X wins ----- \n");
    } else if (win == 'O') {
        System.out.println("\n----- :) Player O wins ----- \n");
    }
}

```



```

    } else {
        System.out.println("\n----- It's a tie ----- \n");
    }
}

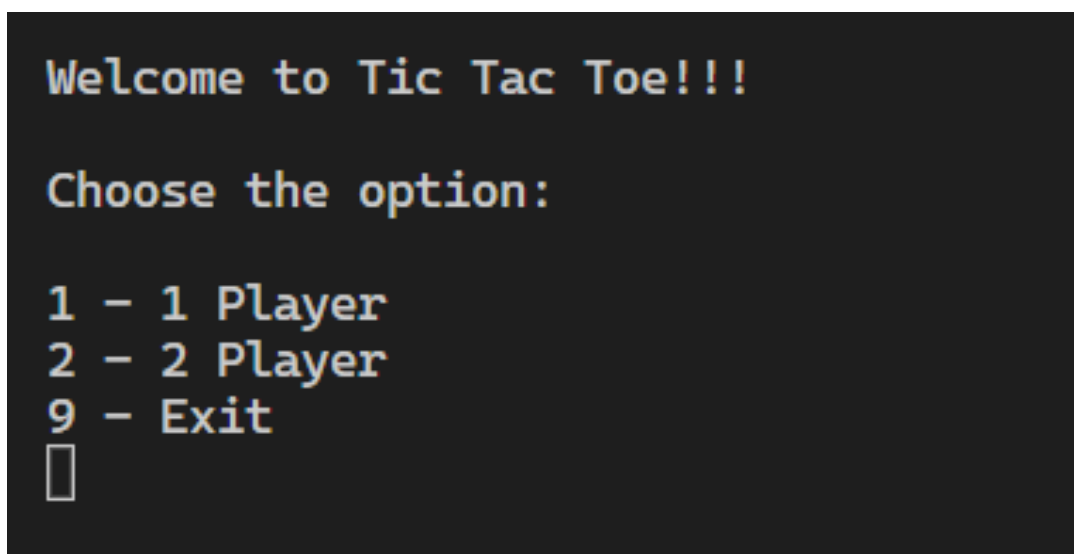
// utility function
static boolean checkInput(char chk) {
    if (chk != 'O' && chk != 'X') {
        return false;
    }
    return true;
}

public static void main(String[] args) {
    gameloop();
}

}

```

Initial Output: as It is not possible to attach each and every output, During presentation outputs will be more clear.



```

Welcome to Tic Tac Toe!!!

Choose the option:

1 - 1 Player
2 - 2 Player
9 - Exit
█

```