TIC-TAC-TOE GAME

A PROJECT REPORT

Submitted by

CROSINI INFANTEENA R (2303811710422021)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on **"TIC-TAC-TOE"** is the bonafide work of **CROSINI INFANTEENA R (2303811710422021)** who carriedout the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

[illegible]

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

CGB1201-JAVA PROGRAMMING
Mr. M. SARAVANAN, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR

[illegible]

SIGNATURE

Mr. M. Saravanan, M.E.,

SUPERVISOR

ASSISTANTP ROFESSOR

Department of CSE

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on 02.12.2024

CGB1201-JAVA PROGRAMMING
Mr.MALARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

[illegible]

INTERNAL EXAMINER

CGB1201-JAVA PROGRAMMING
Dr.E.SENTHAMILSELVI, M.E., Ph.D.,
EXTERNAL EXAMINER
PROFESSOR
8138-SCE, TRICHY.

[illegible]

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on **“TIC-TAC-TOE”** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **“ANNA UNIVERSITY CHENNAI”** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201- JAVA PROGRAMMING.**

.

Signature

[illegible]

CROSINI INFANTEENA R

Place: Samayapuram

Date: 02.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr A. DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project guide **Mr. M. SARAVANAN, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This Tic-Tac-Toe game implementation in Java provides a simple, interactive console-based experience for two players. The program uses a 3x3 game board, where players alternate between 'X' and 'O' to place their marks in an empty cell. The board is displayed after each move, and the program checks after every turn whether a player has won or if the game has ended in a draw. The game continues until one of the players wins by getting three of their marks in a row, column, or diagonal, or if all spaces on the board are filled without a winner, resulting in a draw.The code features methods for initializing the board, displaying the current game state, validating player moves, checking for a winner, and determining if the game is over. It also includes functionality to switch between players after each move. The game logic is designed to handle invalid inputs, ensuring that players cannot make moves in already occupied or out-of-bound positions. This implementation serves as an educational example of basic game development, demonstrating how to manage game state, user input, and simple game rules within a console application. Additionally, this Tic-Tac-Toe game provides an opportunity to understand fundamental programming concepts such as loops, conditional statements, and array manipulation in Java. By using a two-dimensional array to represent the game board, the program efficiently stores and updates the state of the game after each player's move. The implementation also demonstrates the use of functions to organize code, promoting modularity and reusability. While this game is simple, it lays the foundation for more complex applications by showcasing how to structure a turn-based game, handle user input, and manage the flow of a program. Future enhancements could include features such as an AI opponent, a graphical user interface (GUI), or the ability to track multiple rounds of play.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
This console-based Tic-Tac-Toe game for two players using Java. The game allows players to take turns entering their moves on a 3x3 grid, alternating between 'X' and 'O'. The board is initialized with empty spaces, and players are prompted to input their moves by specifying the row and column indices (ranging from 0 to 2). After each move, the program checks for a winning condition or a draw. The game ends when one player wins by aligning three of their symbols in a row, column, or diagonal, or when the board is full, resulting in a draw	**PO1 -3** **PO2 -3** **PO3 -3** **PO4 -3** **PO5 -3** **PO6 -3** **PO7 -3** **PO8 -3** **PO9 -3** **PO10 -3** **PO11-3** **PO12 -3**	**PSO1 -3** **PSO2 -3** **PSO3 -3**

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**viii**
1	**INTRODUCTION**	**1**
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming Concepts	2
2	**PROJECT METHODOLOGY**	**3**
	2.1 Proposed Work	3
	2.2 Block Diagram	3
3	**MODULE DESCRIPTION**	**4**
	3.1 Calculate Net Salary Module	4
	3.2 Print Salary Report Module	4
	3.3 Add Employee Module	4
	3.4 Display Menu Module	4
	3.5 Player Switching Module	4
4	**CONCLUSION & FUTURE SCOPE**	**5**
	4.1 Conclusion	5
	4.2 Future Scope	5
	REFERENCES	**6**
	APPENDIX A(SOURCE CODE)	**7**
	APPENDIX B(SCREENSHOTS)	**10**

CHAPTER 1

INTRODUCTION

1.1 Objective

The objective of this Tic-Tac-Toe game implementation in Java is to provide a simple and interactive console-based game that allows two players to take turns in a game of Tic-Tac-Toe. The program aims to demonstrate key programming concepts, such as handling user input, using arrays to represent a game board, implementing conditional logic for win and draw conditions, and managing the flow of a game. The game enables players to engage in a competitive environment while allowing for basic error handling and validation of user moves.Additionally, the project aims to develop problem-solving skills in game design by structuring the game logic into modular methods for easier management and readability

1.2 Overview

The Tic-Tac-Toe game in Java is a simple, interactive console application where two players take turns placing 'X' and 'O' on a 3x3 grid. The program checks for winning conditions after each move, ensuring that a player who aligns three of their marks in a row, column, or diagonal wins the game. If the board is full and no winner emerges, the game ends in a draw. Key methods in the program include initializing the board, printing the game state, handling player input, and switching turns between players. Error handling ensures only valid moves are accepted. This project serves as an introduction to basic programming concepts like arrays, loops, and conditionals in Java, providing a foundation for future enhancements such as AI opponents or a graphical user interface.

1.3 Java Programming Concepts

The basic concepts of Object-Oriented Programming (OOP) are:

- ✓ **Encapsulation**: Bundling data and methods that operate on the data into a single unit, usually a class, and restricting access to some components for data protection.
- ✓ **Abstraction**: Hiding the complex details of an object and showing only the essential features, allowing users to focus on high-level operations.
- ✓ **Inheritance**: Allowing one class to inherit properties and methods from another, promoting code reuse and creating hierarchical relationships.
- ✓ **Polymorphism**: Enabling objects of different classes to be treated as objects of a common superclass, allowing for flexibility in method usage through method overriding or overloading.

Project related Concepts:

In the Tic-Tac-Toe project, while the implementation is primarily focused on procedural programming, some Object-Oriented Programming (OOP) concepts can still be applied or incorporated. Here are the key OOP concepts that have been used or can be used in this project:

- ✓ **Encapsulation:** Although the code doesn't define custom classes for the game (which would be a more typical use of encapsulation), the game state (board, current player, etc.) is managed within the main class, acting as a form of encapsulation. The game logic is separated into methods, which help to organize and manage the game's data and behavior.
- ✓ **Abstraction:** The board and game rules are abstracted from the user. Players interact with the game by providing inputs for moves, while the implementation details of how the game checks for wins, draws, and updates the board are hidden within methods. This allows players to focus on the gameplay without worrying about the internal workings.
- ✓ While the project could benefit from a more object-oriented design (for example, creating a Game class and separate Player class), the basic OOP principles could be applied if the game were expanded or refactored.

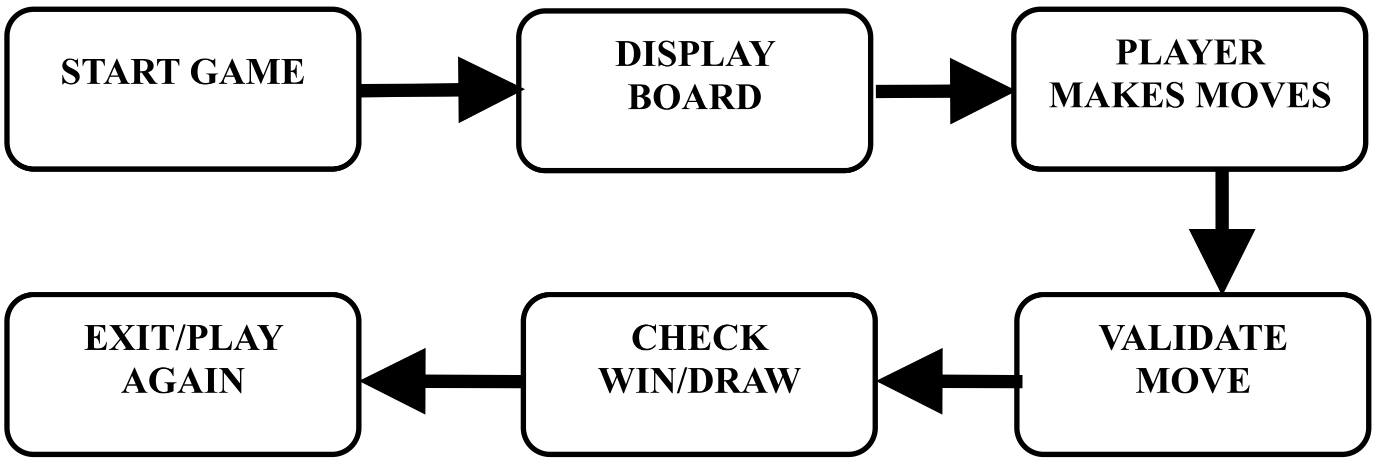
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for the Tic-Tac-Toe game in Java aims to enhance the functionality and user experience while incorporating object-oriented programming principles. First, the game can be refactored into multiple classes, such as a Game class to manage the game flow, a Board class to handle the board logic, and a Player class to manage player-specific data. Additionally, an AI opponent could be implemented, allowing one player to compete against the computer using basic strategies like the Minimax algorithm. For a more user-friendly experience, a graphical user interface (GUI) could be developed using JavaFX or Swing, replacing the console-based interaction. Multiplayer support could also be introduced to allow players to compete online using socket programming.

2.2 Block Diagram



CHAPTER 3
MODULE DESCRIPTION

3.1 Board Initialization Module:

This module initializes the 3x3 game board, which is represented as a 2D character array. The board is populated with empty spaces (' ') to set up the game for play. The initializeBoard() method handles this setup.

3.2 Display Board Module:

The printBoard() method is responsible for displaying the current state of the game board. It prints the grid in a readable format, showing the positions of 'X' and 'O' as well as the empty spaces, allowing players to see the board after each move.

3.3 Player Input Module:

The playerMove() method handles player input, prompting the current player to enter a row and column to place their mark. It includes input validation to ensure that the player selects a valid empty space on the board

3.4 Game Over Check:

This module checks the state of the game after every move. The checkGameOver() method evaluates if a player has won (via the checkWin() method) or if the board is full, leading to a draw. It updates the gameOver flag accordingly.

3.5 Player Switching Module:

The switchPlayer() method changes the active player after every move. It alternates between 'X' and 'O', ensuring that players take turns during the game.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

In conclusion, this **Tic-Tac-Toe game** project offers a solid foundation for understanding basic programming concepts in Java. It utilizes a 3x3 array to represent the game board and provides methods for initializing the board, printing the current state, and handling player moves. The game alternates turns between two players, ensuring that players take turns to place their marks ('X' or 'O') on the grid, while enforcing the rules of valid moves. The use of loops and conditionals makes the game logic simple and easy to follow, while the implementation of basic error checking ensures smooth gameplay. The program also includes essential functionality for determining the game's outcome. It checks for winning conditions across rows, columns, and diagonals, as well as detecting a draw when the board is full. By separating the logic into methods like checkWin(), isBoardFull(), and checkGameOver(), the code is modular and easy to maintain. Additionally, the alternating player feature is efficiently handled through the switchPlayer() method, which toggles between players 'X' and 'O'.

4.2 FUTURE SCOPE

The future scope for this Tic-Tac-Toe game includes enhancements like implementing a graphical user interface (GUI) using libraries such as JavaFX or Swing to make it visually appealing and user-friendly. Adding an AI opponent with varying difficulty levels using algorithms like Minimax can make it challenging for single players. It can also be extended to support online multiplayer functionality using network programming, allowing players to compete remotely. Additionally, expanding the game board size (e.g., 4x4 or larger) or introducing variations like timed moves can add complexity and excitement, making it more engaging for users.

REFERENCES

Java Books:

1."Head First Java" by Kathy Sierra and Bert Bates

This book is a great resource for beginners learning Java, with a focus on object-oriented programming concepts and real-world application development.

2. "Effective Java" by Joshua Bloch

A deeper dive into best practices for writing clean, maintainable Java code. It covers advanced topics like Java collections, concurrency, and design patterns that could be applied to more complex payroll systems.

Websites:

1. GeeksforGeeks - Java Tutorials

- URL: https://www.geeksforgeeks.org/java/
- A comprehensive collection of tutorials on Java, covering topics like classes, objects, inheritance, encapsulation, and more. Great for learning the core concepts of Java and applying them in projects like EPMS.

2. W3Schools - Java Tutorial

- URL: https://www.w3schools.com/java/
- A beginner-friendly resource that offers tutorials on Java programming, including object-oriented principles and core Java concepts.

YouTube Links:

1. Java for Beginners - Java Brains

- URL: https://www.youtube.com/user/koushks
- Offers Java tutorials from the basics to advanced concepts. The channel provides detailed guides on Java programming, including working with objects and classes, which are crucial for building an EPMS.

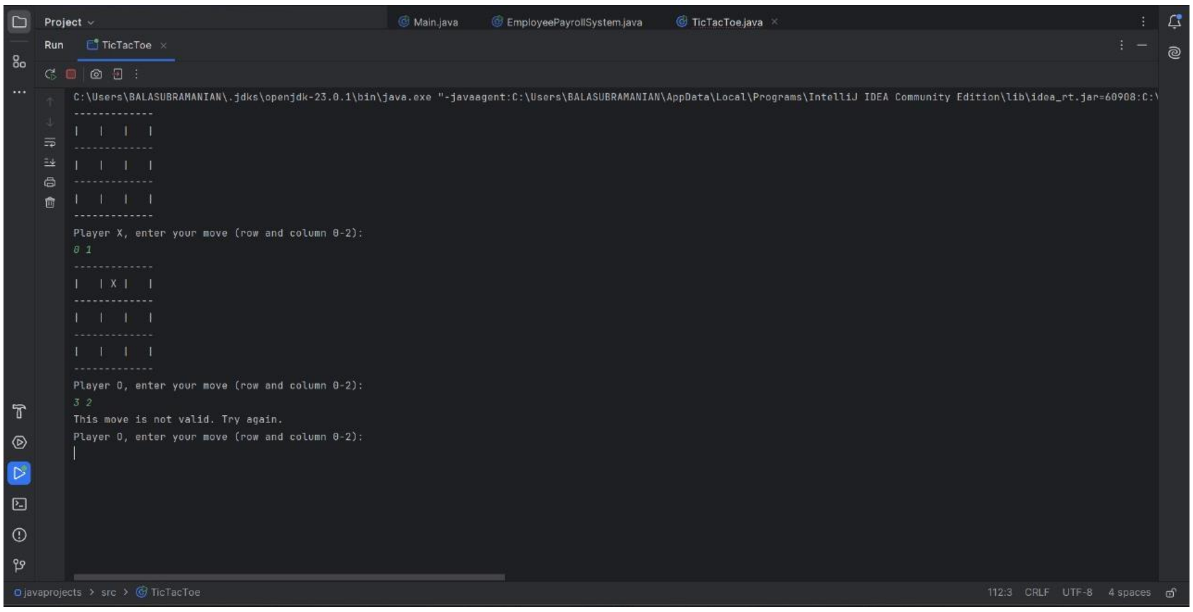
APPENDIX A (SOURCE CODE)

```
import java.util.Scanner;

public class TicTacToe {
    static char[][] board = new char[3][3];
    static char currentPlayer = 'X';
    static boolean gameOver = false;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        initializeBoard();

        while (!gameOver) {
            printBoard();
            playerMove(scanner);
            checkGameOver();
            switchPlayer();
        }

        printBoard();
        if (gameOver) {
            System.out.println("Game Over!");
        }
    }

    // Initialize the game board with empty spaces
    public static void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = ' ';
            }
        }
    }
    // Print the current board
    public static void printBoard() {
        System.out.println("-------------");
        for (int i = 0; i < 3; i++) {
            System.out.print("| ");
            for (int j = 0; j < 3; j++) {
                System.out.print(board[i][j] + " | ");
            }
            System.out.println();
            System.out.println("-------------");
        }
    }
        // Handle the player's move
    public static void playerMove(Scanner scanner) {
        int row, col;
        while (true) {
            System.out.println("Player " + currentPlayer + ", enter your move (row and column 0-2): ");
            row = scanner.nextInt();
```

```
   col = scanner.nextInt();
 if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ') {
                board[row][col] = currentPlayer;
                break;
            } else {
                System.out.println("This move is not valid. Try again.");
            }
        }
    }
    // Check if the game is over (win or draw)
    public static void checkGameOver() {
        if (checkWin()) {
            gameOver = true;
            System.out.println("Player " + currentPlayer + " wins!");
        } else if (isBoardFull()) {
            gameOver = true;
            System.out.println("It's a draw!");
        }
    }
    // Check if any player has won
    public static boolean checkWin() {
        // Check rows, columns, and diagonals
        for (int i = 0; i < 3; i++) {
            if (board[i][0] == currentPlayer && board[i][1] == currentPlayer &&
board[i][2] == currentPlayer) {
                return true;
            }
            if (board[0][i] == currentPlayer && board[1][i] == currentPlayer &&
board[2][i] == currentPlayer) {
                return true;
            }
        }
        if (board[0][0] == currentPlayer && board[1][1] == currentPlayer &&
board[2][2] == currentPlayer) {
            return true;
        }
        if (board[0][2] == currentPlayer && board[1][1] == currentPlayer &&
board[2][0] == currentPlayer) {
            return true;
        }
        return false;
    }
    // Switch the current player between X and O
    public static void switchPlayer() {
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }
}
```

```
 // Check if the board is full (i.e., the game is a draw)
public static boolean isBoardFull() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == ' ') {
                return false;
            }
        }
    }
    return true;
}
```

APPENDIX B(SCREENSHOTS)


Project
Main.java
EmployeePayrollSystem.java
TicTacToe.java
Run
TicTacToe
javaprojects > src > TicTacToe
112:3 CRLF UTF-8 4 spaces

```
C:\Users\BALASUBRAMANIAN\.jdks\openjdk-23.0.1\bin\java.exe "-javaagent:C:\Users\BALASUBRAMANIAN\AppData\Local\Programs\IntelliJ IDEA Community Edition\lib\idea_rt.jar=60908:C:\
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Player X, enter your move (row and column 0-2):
0 1
-------------
|   | X |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Player O, enter your move (row and column 0-2):
3 2
This move is not valid. Try again.
Player O, enter your move (row and column 0-2):
```