

**PROJECT ON:**

**TIC-TAC-TOE**

**SUBMITTED BY:**

SHRIHARI SG21CSE137

SHUBHAM SG21CSE141

SHRINIDHI SG21CSE138

VIJAY BUDDHI SG21CSE170

**Guided By:**

PROF. MANJULA.BIRADAR

**TABLE OF CONTENTS:**

|  |  |  |
| --- | --- | --- |
| **SL.NO** | **PARTICULARS** | **PAGE NO.** |

**ABSTRACT**

1. **INTRODUCTION 4 - 5**
2. **IMPLEMENTATION 6 - 7**
3. **LANGUAGE SPEC 8**
4. **FLOW CHART 9**
5. **H/W & S/W REQ 10**
6. **ADVANTAGES & 11  
    DISADVANTAGES**
7. **OUTPUT 12-14**
8. **FUTURE WORK 15**
9. **CONCLUSION 16**

**REFERENCES**

|  |
| --- |
|  |

* **ABSTRACT:**

This report outlines the development of a simple Tic Tac Toe game using Java programming language. The objective of the project was to practice Java programming skills and develop a functional game. The game was developed using an object-oriented programming approach, and the report details the implementation process. The report concludes that the project was successful in creating a functional game that provides an enjoyable user experience. Finally, the report suggests potential future work or improvements that could be made to the game, such as adding AI functionality for single-player mode or improving the user interface.

* **INTRODUCTION:**

The Tic Tac Toe game is a classic, well-known game that has been enjoyed by people of all ages for generations. In this project, we present a Java implementation of Tic Tac Toe that allows two players to compete against each other. The game is played on a 3x3 game board, where players take turns placing their symbol (either 'X' or 'O') on the board until one of the players wins or the game ends in a draw.

This implementation of the game includes a basic algorithm to determine the winner by checking rows, columns, and diagonals for matching symbols. Additionally, the program features a print Board method to display the current state of the game board after each move, as well as error messages if an invalid move is attempted.

By exploring and understanding the code of this Tic Tac Toe game, one can gain valuable insights into programming logic, algorithm development, and game development. It provides a fun and interactive way to apply the fundamentals of Java programming and develop problem-solving skills.



The game board was represented as a 2D array of characters, with each player represented by either 'X' or 'O'. The game logic was contained within the Main class, with helper methods used to print the board and check for win conditions.

The game loop continually prompts the user to enter a row and column to place their marker and alternates between the players until a win condition is met or the board is filled.

The game utilizes the Scanner class to read input from the user, which allows for dynamic input of moves during gameplay. Overall, the implementation of the game is simple and straightforward, but it provides an excellent foundation for beginners to learn about programming concepts in Java and apply them to a simple game project.

* **IMPLEMENTATION:**
* import java.util.Scanner;  
    
  class tic {  
   public static void main(String[] args) {  
   char[][] board = new char[3][3];  
   for (int row = 0; row < board.length; row++) {  
   for (int col = 0; col < board[row].length; col++) {  
   board[row][col] = ' ';  
   }  
   }  
    
   char player = 'X';  
   boolean gameOver = false;  
   Scanner scanner = new Scanner(System.*in*);  
    
   while (!gameOver) {  
   *printBoard*(board);  
   System.*out*.print("Player " + player + " enter: ");  
   int row = scanner.nextInt();  
   int col = scanner.nextInt();  
   System.*out*.println();  
    
   if (board[row][col] == ' ') {  
   board[row][col] = player; // place the element  
   gameOver = *haveWon*(board, player);  
   if (gameOver) {  
   System.*out*.println("Player " + player + " has won: ");  
   } else {  
   // if (player == 'X') {  
   // player = 'O';  
   // } else {  
   // player = 'X';  
   // }  
   player = (player == 'X') ? 'O' : 'X';  
   }  
   } else {  
   System.*out*.println("Invalid move. Try again!");  
   }  
   }  
   *printBoard*(board);  
   }

public static boolean haveWon(char[][] board, char player) {  
 // check the rows  
 for (int row = 0; row < board.length; row++) {  
 if (board[row][0] == player && board[row][1] == player && board[row][2] == player) {  
 return true;  
 }  
 }  
  
 // check for col  
 for (int col = 0; col < board[0].length; col++) {  
 if (board[0][col] == player && board[1][col] == player && board[2][col] == player) {  
 return true;  
 }  
 }  
  
 // diagonal  
 if (board[0][0] == player && board[1][1] == player && board[2][2] == player) {  
 return true;  
 }  
  
 if (board[0][2] == player && board[1][1] == player && board[2][0] == player) {  
 return true;  
 }  
 return false;  
 }  
  
 public static void printBoard(char[][] board) {  
 for (int row = 0; row < board.length; row++) {  
 for (int col = 0; col < board[row].length; col++) {  
 System.*out*.print(board[row][col] + " | ");  
 }  
 System.*out*.println();  
 }  
 }  
}

This code implements a simple Tic Tac Toe game in Java. The game is played on a 3x3 board, with two players taking turns placing their symbol (X or O) on the board. The first player to get three in a row (horizontally, vertically, or diagonally) wins the game.

The implementation uses a 2D array of chars to represent the board. The board is initialized to contain all spaces at the beginning of the game. The game loop continues until one player has won the game or the board is completely filled.

The haveWon() method checks if a given player has won the game by iterating through each row, column, and diagonal of the board and checking if all elements in that row/column/diagonal are equal to the player's symbol.The printBoard() method prints the current state of the board to the console.

* **LANGUAGE SPECIFICATION:**

**Java** is a high-level programming language that is designed to be platform-independent, meaning that code written in Java can run on any platform that has a Java Virtual Machine (JVM) installed. It was first released by Sun Microsystems in 1995 and has since become one of the most widely used programming languages in the world.

Some key features of Java include:

**Object-oriented:** Java is a fully object-oriented language, which means that everything in Java is an object, including data types, variables, and functions.

**Automatic memory management:** Java has a built-in garbage collector that automatically manages memory allocation and deallocation, freeing developers from the burden of manually managing memory.

**Platform independence:** Java code can be compiled into bytecode, which can run on any platform that has a JVM installed, making Java programs highly portable.

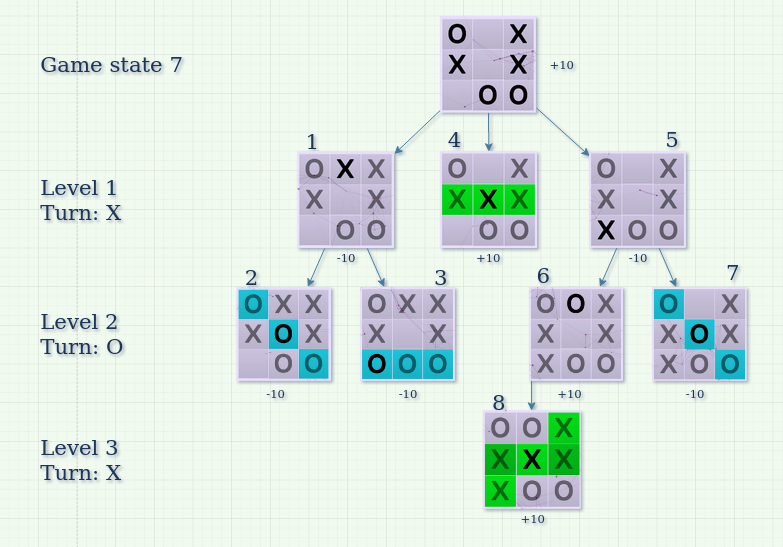
**Multi-threaded:** Java supports multi-threading, allowing developers to write programs that can execute multiple tasks simultaneously.

**Exception handling:** Java has a robust exception handling mechanism that allows developers to catch and handle errors at runtime. This helps to make Java programs more robust and less prone to crashing.

Libraries:Java comes with a large number of standard libraries that make it easy to perform common tasks, such as reading and writing files, working with databases, and creating graphical user interfaces.

Java is widely used for developing a wide range of applications, including web applications, mobile applications, desktop applications, and enterprise applications. It is also the primary language used for developing Android applications.

* **FLOWCHART:**



* **HARDWARE REQUIREMENTS:**

The hardware requirements for running a Java-based Tic Tac Toe game are minimal. Since it is a simple console-based game, it can be run on any computer that supports Java. The basic requirements are:

Processor: Any processor that supports Java

RAM: At least 512MB RAM

Storage: At least 1GB free storage space

Display: A monitor capable of displaying a resolution of 1024x768 or higher

Overall, the hardware requirements are very low, and almost any modern computer or laptop can easily run a Java-based Tic Tac Toe game without any issues.

* **SOFTWARE REQUIREMENTS:**

Java Development Kit (JDK): The Java Development Kit is required to compile and run Java programs. It includes the Java Runtime Environment (JRE), which is necessary to run Java applications.

Integrated Development Environment (IDE): An IDE is not strictly required, but it can make the development process easier. IDEs like Eclipse, NetBeans, and IntelliJ IDEA provide a variety of features such as syntax highlighting, auto-completion, and debugging tools that can simplify the process of writing, testing, and debugging Java code.

Command Prompt or Terminal: The command prompt or terminal is used to compile and run the Java program.

Text Editor: A text editor is necessary to write the Java code. Any text editor, such as Notepad, Sublime Text, or Atom can be used to write Java code.

* **ADVANTAGES:**
* It can be played without wasting paper
* It helps to improve your concentration, critical thinking skills in children.
* It helps the gamers by giving them small and simple game which can be played in leisure time and hence provides them entertainment.
* It teaches patience to children.
* It helps to improve strategic skills in children.
* **DISADVANTAGES:**
* Though it looks nice to use tic-tae-toe but it is very complex rather to use pen and paper
* The game is not very challenging to the player as the game can be completed under a minute.
* The game can`t be played solo and with more than two people.
* When you know how to win the game isn`t very challenging.
* **OUTPUT:**

| |

---------

| |

---------

| |

Player X enter: 1 1

| |

---------

| X |

---------

| |

Player O enter: 0 2

| | O

---------

| X |

---------

| |

Player X enter: 0 0

X | | O

---------

| X |

---------

| |

Player O enter: 1 2

X | | O

---------

| X | O

---------

| |

Player X enter: 2 1

X | | O

---------

| X | O

---------

| X |

Player X has won:

X | | O

---------

| X | O

---------

| X |

* **FUTURE WORK:**

**Implement a tie condition:** Currently, the game only ends when a player wins. You can modify the haveWon method to check for a tie condition, which happens when all the spaces on the board are filled and no player has won.

**Improve input validation:** The current implementation assumes that the user will always enter valid input (i.e., integer values between 0 and 2). You can add some error handling code to validate user input and handle invalid inputs gracefully.

**Implement a replay feature:** Once the game is over, the program ends. You can modify the main method to allow the user to choose whether they want to play again or not.

**Add a computer player:** Currently, the game can only be played by two human players. You can implement a computer player that plays against the human player.

**Add a graphical interface:** The current implementation uses the console to display the game board. You can create a graphical interface to make the game more visually appealing.

* **CONCLUSION:**

Through this project, we have explored some important language specifications of Java, such as object-oriented programming, platform independence, garbage collection, exception handling, multithreading, libraries, security, and IDEs. Java's robust features and libraries make it a powerful and popular language for developing software applications in various domains, including web development, mobile app development, game development, and enterprise applications.

Overall, this project has provided a solid foundation for beginners to understand the basic concepts of Java programming and apply them to create simple yet interesting programs. By continuing to build upon this foundation and exploring more advanced concepts and features, developers can create more complex and sophisticated application.

* **REFERENCES:**

**1.THE COMPLETE REFERENCE JAVA-HERBERT SCHILDT**

**2.** [**https://github.com/harry /DSA-Bootcamp-Java**](https://github.com/harry%20/DSA-Bootcamp-Java)