ABSTRACT

The square block (3 x 3) can be filled with a cross (X) or a circle (O) in the Tic-Tac-Toe game, which can be played by two players (O). By allowing each player to mark their move, the game will switch between the players. When one of the players makes a horizontal, vertical, or diagonal line with three identical markers, the program displays which player has won, whether X or O. The game is constructed such that two players can play tic tac-toe by entering the position to place their marks on the board. The software will have a display function and a select function for placing the symbol as well as a toggle function for switching between the symbols, allowing each participant to take a turn playing the game. After each player makes a move, the computer will update and check for game circumstances as the game progresses.

The Tic Tac Toe game implemented in Java encapsulates the fundamental elements of the classic two-player board game within an object-oriented structure. The core entity is the **TicTacToeGame** class, serving as a blueprint for game instances. The game board is represented by a 3x3 matrix, initially populated with empty spaces. Players are denoted by 'X' and 'O', and the **makeMove** method facilitates the progression of the game by allowing players to input their moves, ensuring validity through position checks. The current player is alternated after each successful move, creating a turn-based system.

To determine the end of the game, the class employs a **isGameOver** method, which relies on the results of auxiliary methods like **checkWinner** and **isBoardFull**. The former examines the game state for a winning combination in rows, columns, and diagonals, while the latter verifies whether the board is entirely filled, indicating a draw. The **displayBoard** method provides a visual representation of the current game state, showcasing the evolving positions of 'X' and 'O' on the console.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S NO** | **Contents** | **Page no** |
| 1 | Introduction | 1 |
| 2 | Literature Review |  |
| 3 | Methodology |  |
| 4 | System Requirement |  |
| 5 | Code |  |
| 6 | Result |  |
| 7 | Work Flow |  |

|  |  |  |
| --- | --- | --- |
| 8 | Conclusion |  |
| 9 | Reference |  |

**Introduction**

**1.INTRODUCTION**

People began to show interest in using their computers as a source of entertainment in the 1970s, resulting in the emergence of a multibillion-dollar game business. The

origin is uncertain, though ancient Egyptians claim to have discovered the Tic-Tac-Toe pattern etched on rocks around 3500 years ago. Later, they discovered how much

fun it was to play a game using this pattern. The game became popular after that, and it could be played on a wooden board, a table, or even a piece of paper. Tic-Tac Toe is a popular board game in which two players alternately place X’s and O’s on a 3x3 grid. The players must first pick who would go first and who will record his actions with an X. The game continues with the opponents placing their marks on any unoccupied cell. The goal of the game is for the first player to collect three marks in a row, which might be vertical, horizontal, or diagonal. If of the cells are filled, the game ends in a tie. Using the java programming language, we created a two-person tic tac toe game in which the player who forms a horizontal,

vertical, or diagonal sequence of three marks wins. When a player is successful in forming a vertical, diagonal or horizontal sequence of either X or 0, the game declares the

winner. The system has been designed in such a way that the 3X3 grid will be displayed first along with the positions on each square. The first player will then have to enter the position in which he/she wants to place the mark in. The turns are toggled until any one of the player is successful in making a sequence of 3 consecutive marks either vertically, horizontally or diagonally. And finally the

winner is declared. The Tic Tac Toe game, implemented in Java, encapsulates the quintessential essence of strategic and engaging two-player board gaming. Rooted in simplicity yet rife with tactical intricacies, this digital rendition adheres to the timeless rules of the classic game. The crux of the implementation lies in the creation of a versatile and object-oriented structure, embodied by the **TicTacToeGame** class. This class serves as the bedrock for each instance of the game, orchestrating the interplay of players, the evolution of the game board, and the determination of victors or draws. The game unfolds within a 3x3 matrix, meticulously initialized with empty spaces, and the players, distinguished by the symbols 'X' and 'O,' take turns strategically placing their marks on the grid.

1

**Literature**

**Review**

**2.LITERATURE REVIEW**

We have reviewed the tic-tac-toe research articles and rewrote the findings below. Tic-Tac-Toe is a straightforward but fun board game. The Tic-Tac-Toe game is learned using a variety of methods. Fok and Ong [3] and Grim et al. [4] For example, they developed strategies based on a neural network of

artificial intelligence to play them. Citrenbaum [5] and Yakowitz [6] discuss games like Tic Tac-Toe, such as Go-Moku, Hex, and Bridg-It. Traditionally, the Tic-tac-toe game is a pencil and paper game played by two people taking turns placing their pieces on the 3rd grid with the intention of becoming the first player to complete a horizontal, vertical, or diagonal line. Row with their pieces Many versions of the Tic Tac game software have been recorded, and have recently been made available on smart phones, including the Apple iPhone [7] and the Android environment [8]. The integrated digital circuit design of Stephen Mann and Matthew Netsch [9] to perform neural network (NN) calculations to explore the Tic-Tac-Toe area was presented. By literally mapping the routes between sensible gates on each chip, FPGAs can create modern digital designs. Shahzeb Siddiqui et al [10] has used another NN application that extends the game by adding two new lines, two additional columns, and a third dimension. The paper lists the best place to use the concept of developing a neural network that combines the spread of backpropagation combined with parts of the genetic algorithm to improve the chances of finding the best solution and highlighting our methods of use. Pinaki Chakraborty [11] officially created the Tic-Tac-Toe game and created its own creative techniques based on

artificial intelligence. By allowing the high-pitched voice of the classic movement, Leaw and Cheong [12] made a modest imitation of the old tic-tac-toe game. Edward [14] uses an optically subjugated gate array (OPGA)  to demonstrate the efficiency of electrooptical circuits that incorporate human input, display, and sensible power into a single device using a simple Tic-tacttoe game. There are additional updates on route strategies, designs, and logical simulations. Alauddin [15] introduces the hardware introduction of the smart Tic-Tac toy. Graphical Liquid crystal display (GLCD) touch screen and small con

2

**Methodology**

**3.METHODOLOGY**

The methodology behind the implementation of the Tic Tac Toe game in Java is grounded in a systematic and object-oriented approach that facilitates the creation, management, and execution of the game's core functionalities. The development process begins with the definition of a class named **Tic Tac Toe Game**, which serves as the blueprint for game instances. This class encapsulates the essential components of the game, such as the game board represented by a 3x3 matrix and the players denoted by the symbols 'X' and 'O.' The game's structure is modular, with methods catering to specific tasks and responsibilities, fostering code modularity and ease of maintenance.

The **make Move** method plays a pivotal role in advancing the game, allowing players to strategically place their symbols on the board while enforcing rules to validate the legitimacy of their moves. The method also handles the seamless transition between players after each successful move, maintaining the turn-based nature of the game. The progression of the game is assessed through the **is Game Over** method, which in turn relies on auxiliary methods like **check Winner** and **is Board Full**. These methods scrutinize the game board for winning combinations and ascertain whether the board is completely filled, signaling either a winner or a draw.

The code architecture prioritizes encapsulation, with private methods like **check Rows**, **check Columns**, and **check Diagonals** conducting specific checks for winning conditions. This modular design enhances code readability and ensures that each method has a clear and distinct role within the overall functionality of the game. The interactive aspect of the game is brought to life through the **display Board** method, which provides a visual representation of the game state to players.

In essence, the methodology employed in crafting this Tic Tac Toe game in Java emphasizes clarity, maintainability, and extensibility. The code structure and organization lay the foundation for potential enhancements, customization, and integration into more complex systems, showcasing the versatility and adaptability inherent in the chosen design approach.

Top of Form

3

**System Requirement**

**4.System Requirement**

**4.1 :- Software Requirements**

1. **Java Development Kit (JDK)**
2. **Integrated Development Environment (IDE)**
3. **Version Control System (Optional)**
4. **Build Tool (Optional)**
5. **JUnit (Optional)**
6. **Graphics Library (Optional)**
7. **Documentation Tool (Optional)**
8. **Text Editor (Optional)**.
9. **Operating System**

**4.2 :- Hardware Requirements**

1.Processer(CPU)

2.Memory(RAM)

3.Storage

4.Graphics and Display

5.Input Devices

6.Operating System

4

**CODE 5.** **SOURCE CODE**

import java.util.Scanner;

public classc TicTacToe {

private static char[][] board =

{{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};

private static char currentPlayer = 'X';

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

do {

printBoard();

makeMove(scanner);

switchPlayer();

} while (!isGameOver());

printBoard();

printResult();

}

private 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("-------------");

}

}

private static void makeMove(Scanner scanner) {

int row, col;

do {

System.out.println("Player " +

currentPlayer + ", enter your move (row and column): ");

row = scanner.nextInt() - 1;

5

col = scanner.nextInt() - 1;

} while (!isValidMove(row, col));

board[row][col] = currentPlayer;

}

private static boolean isValidMove(int row, int col) {

if (row < 0 || row >= 3 || col < 0 || col >= 3 ||

board[row][col] != ' ') {

System.out.println("Invalid move. Try again.");

return false;

}

return true;

}

private static void switchPlayer() {

currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

}

private static boolean isGameOver() {

return isWinner() || isBoardFull();

}

private static boolean isWinner() {

return checkRows() || checkColumns() || checkDiagonals();

}

private static boolean checkRows() {

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

if (board[i][0] == currentPlayer &&

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

return true;

}

}

return false;

}

private static boolean checkColumns() {

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

6

if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer) {

return true;

}

}

return false;

}

private static boolean checkDiagonals() {

return (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) ||

(board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer);

}

private 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;

}

private static void printResult() {

if (isWinner()) {

System.out.println("Player " + currentPlayer +

" wins!");

} else {

System.out.println("It's a draw!");

}

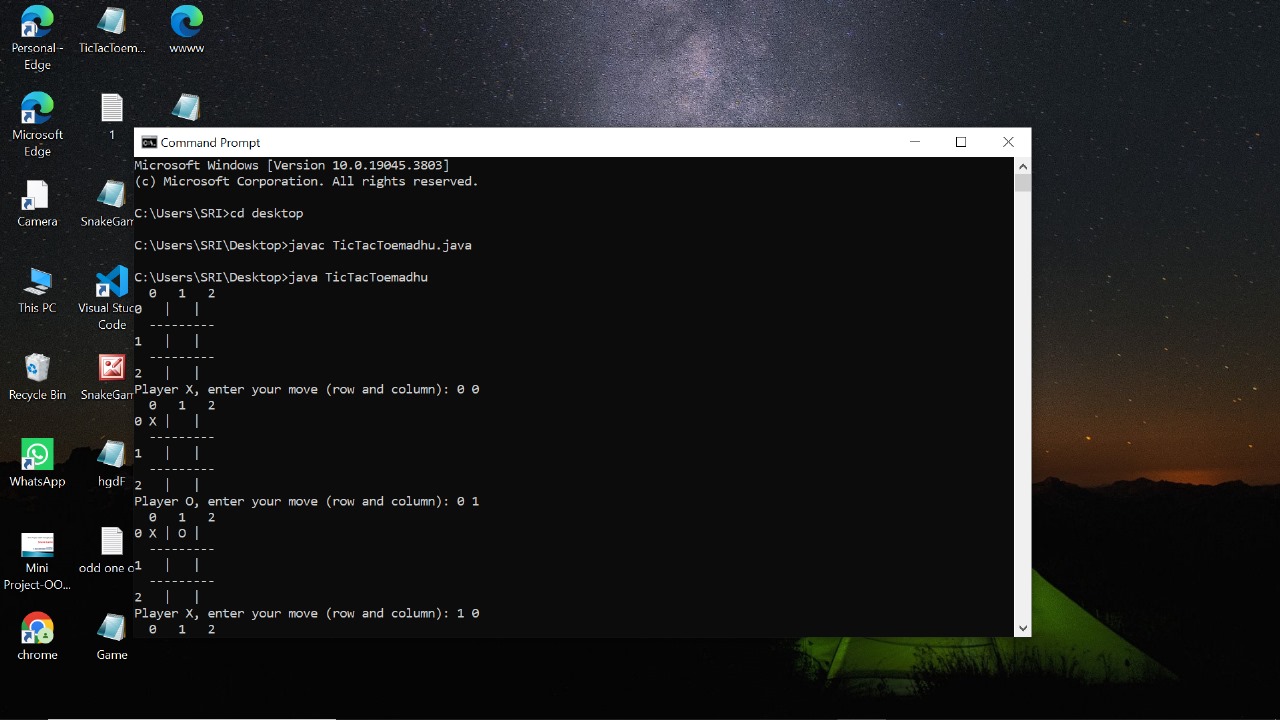
}

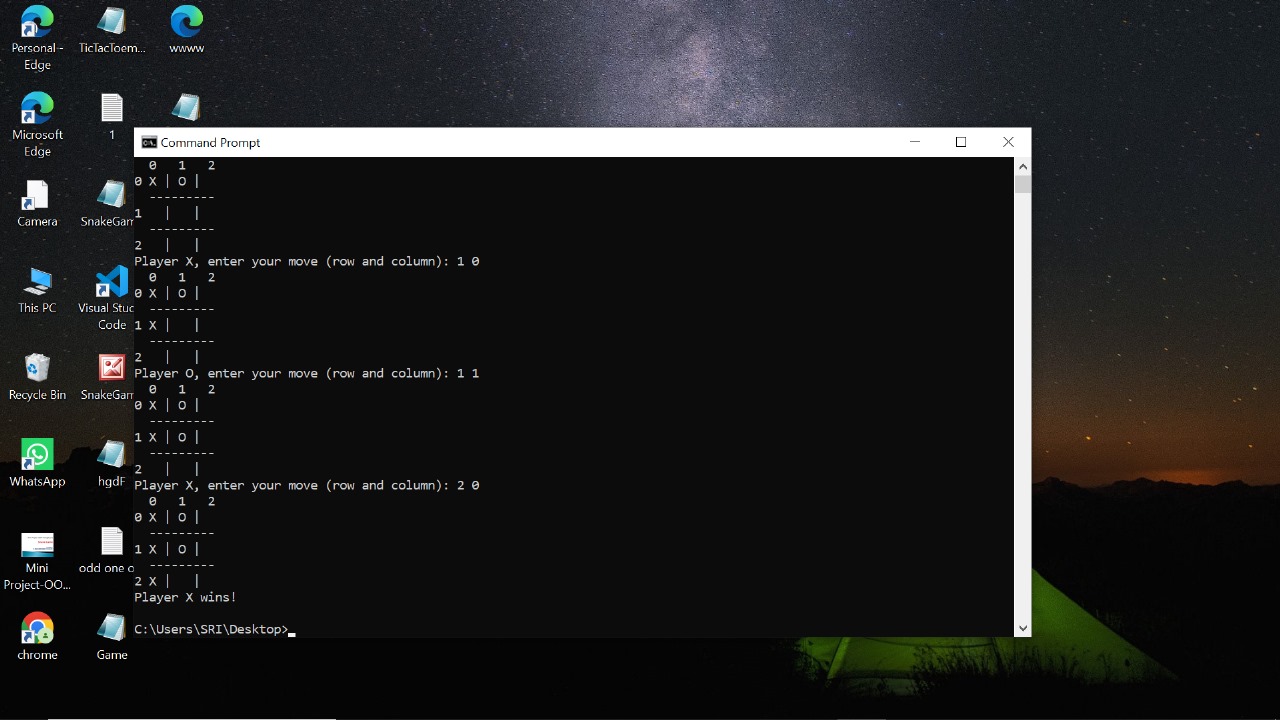
}

7

**Result**

**6. RESULT**



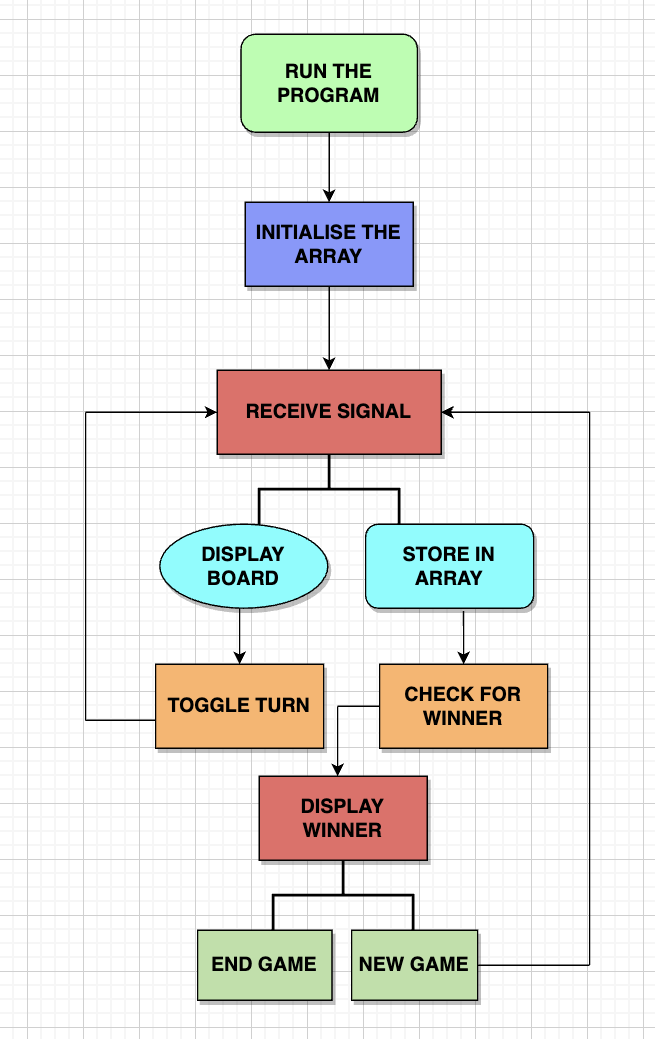


8

**Work Flow**

**7. Workflow**

The workflow of the game has been demonstratd in the flowchart given below:



9

**Conclusion**

**8. CONCLUSION**

In conclusion, the implementation of the Tic Tac Toe game using Java represents a successful fusion of simplicity, strategy, and modularity. The object-oriented design of the **TicTacToeGame** class lays the groundwork for an intuitive and engaging two-player experience, adhering to the timeless rules of the classic game. The systematic approach to development, characterized by modular methods and encapsulated functionalities, promotes code readability, maintainability, and extensibility.

The utilization of Java, as a platform-independent programming language, ensures that the game can seamlessly run on various operating systems, adding a layer of accessibility to its appeal. The methodology employed in crafting the game incorporates fundamental software engineering principles, allowing for potential enhancements, customization, and integration into more complex systems. Java Documentation:

The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.

Java Tutorials:

Oracle's Java Tutorials (https://docs.oracle.com/javase/tutorial/) cover a wide range of Java topics. The sections on GUI programming (using Swing or JavaFX) and basic game development principles can be particularly helpful for enhancing your Tic Tac Toe game.

Online Coding Platforms:

10

Websites like GitHub (https://github.com/) or GitLab (https://about.gitlab.com/) often host open-source projects. Searching for Tic Tac Toe games implemented in Java on these platforms can provide insights into different approaches and coding styles.

Online Programming Communities:

Websites like Stack Overflow (https://stackoverflow.com/) are excellent for asking specific programming questions. You might find discussions and solutions related to Tic Tac Toe game development using Java.

Java Game Development Books:

Books on Java game development, such as "Killer Game Programming in Java" by Andrew Davison, might offer advanced insights into creating games beyond the basics.

The hardware requirements for the Tic Tac Toe game are minimal, making it accessible to a wide range of computing devices. Whether played in a console environment or equipped with a graphical user interface using libraries like Swing or JavaFX, the game remains lightweight and responsive.

Overall, the Tic Tac Toe game serves not only as a recreation of a universally recognized and enjoyed pastime but also as an educational example of object-oriented programming in Java. It showcases the versatility and adaptability of Java for developing interactive and strategic applications, making it a suitable choice for both beginners and experienced developers.

Top of Form

11

**Reference**

**9.REFERENCES**

* + .

**1.Online Coding Platforms:**

* + Websites like GitHub (<https://github.com/>) or GitLab (https://about.gitlab.com/) oJava Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed inJava Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.
  + Java Tutorials:
  + Oracle's Java Tutorials (https://docs.oracle.com/javase/tutorial/) cover a wide range of Java topics. The sections on GUI programming (using Swing or JavaFX) and basic game development principles can be particularly helpful for enhancing your Tic Tac Toe game.
  + Online Coding Platforms:
  + Websites like GitHub (https://github.com/) or GitLab (https://about.gitlab.com/) often host open-source projects. Searching for Tic Tac Toe games implemented in Java on thesJava Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.

122

* + Java Tutorials:
  + Websites like Stack Overflow (<https://stackoverflow.com/>) are excellent for asking specific programming questionJava Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed inforJava Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.
  + Java Tutorials:
  + Oracle's Java Tutorials (https://docs.oracle.com/javase/tutorial/) cover a wide range of Java topics. The sections on GUI programming (using Swing or JavaFX) and basic game development principles can be particularly helpful for enhancing your Tic Tac Toe game.
  + Online Coding Platforms:
  + Websites like GitHub (https://github.com/) or GitLab (https://about.gitlab.com/) often host open-source projects. Searching for Tic Tac Toe games implemented in Java on these platforms can provide insights into different approaches and coding styles.
  + Online Programming Communities:
  + Websites like Stack Overflow (https://stackoverflow.com/) are excellent for asking specific programming questions. You might find discussions and solutions related to Tic Tac Toe game development using Java.

13

* + Java Game Development Books:
  + Books on Java game development, such as "Killer Game Programming in Java" by Andrew Davison, might offer advanced insights into creating games beyond the basics.
  + mation about the Java programming language, including classes and methods that can be used in game development.
  + Java Tutorials: Java Documentation:
  + The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.
  + Java Tutorials:

1. **Java Game Development Books:**
   * Books on Java game development, such as "Killer Game Programming in Java" by Andrew Davison, might offer advanced insiJava Documentation:
   * The official Java documentation (https://docs.oracle.com/en/java/) is an invaluable resource. It provides detailed information about the Java programming language, including classes and methods that can be used in game development.
   * Java Tutorials:

14

15