Abstract :

This project focuses on the design and implementation of a simple yet interactive Tic Tac Toe game using Java. The primary objective of the project is to create a functional two-player game that can be played on a console or with a graphical user interface (GUI). The game adheres to the classic 3x3 grid format and enables two players to take turns placing their marks ('X' and 'O') on the grid. The game logic checks for win conditions, such as three identical marks in a row, column, or diagonal, and announces the winner or a draw when appropriate.

The project demonstrates various Java concepts, including object-oriented programming (OOP) principles, such as encapsulation and polymorphism, as well as data structures for grid representation and control structures for managing game state and user input. Advanced features such as input validation, a basic AI opponent, and a GUI implemented with Java Swing may also be included to enhance the gameplay experience. This project aims to provide an engaging platform for learning Java fundamentals, user interaction handling, and efficient algorithm design for game logic.

Introduction :

This project involves the development of a Tic Tac Toe game using Java, providing a platform to apply fundamental programming concepts in an engaging and interactive manner. Tic Tac Toe is a classic two-player game where players alternate marking spaces on a 3x3 grid with 'X' or 'O' symbols. The objective is simple yet strategic: to align three of one's own marks in a horizontal, vertical, or diagonal line while preventing the opponent from doing so.

The project serves as an opportunity to deepen understanding of Java by incorporating core programming constructs such as loops, conditional statements, arrays, and functions. Object-oriented programming (OOP) principles, including class design, encapsulation, and data abstraction, are utilized to structure the game's components and logic effectively. The game checks for valid moves, determines win or draw conditions, and provides a user-friendly interface, either through console-based input or an optional graphical user interface (GUI) built using Java Swing.

Overall, this project aims to strengthen problem-solving skills, foster a solid grasp of Java programming, and demonstrate real-world application of software design concepts through the creation of a simple, yet enjoyable, game.

Project Requirements:

1. Development Environment:

Java Development Kit (JDK) installed (minimum version 8+ recommended).

Integrated Development Environment (IDE) such as IntelliJ IDEA, Eclipse, or NetBeans for writing and running Java code.

2. Core Components: Grid Representation: Use a 3x3 matrix (2D array or list structure) to represent the game board.

Game Logic: Methods to handle game functions such as placing player symbols, checking for a win or a draw, and switching between players.

Player Interaction:

Human vs. Human Mode: Allow two users to take turns interacting with the game.

Optional: Human vs. AI Mode: An AI player that can make decisions using basic or advanced strategies.

Input Validation: Ensure that user inputs are valid (e.g., selecting empty cells, staying within bounds).

3. Features and Functionality:

User Interface:

Console-Based Interface: Prompt user inputs and display the game board using text output.

Optional: Graphical User Interface (GUI): Use Java Swing (or similar) to create a more interactive graphical display.

Game Rules Enforcement: Proper handling of game rules such

Turn management (alternating moves between two players).

Detection of winning states (three identical marks in a row, column, or diagonal).

Recognition of a draw when all cells are filled without a winner.

Game Reset Option: Ability to restart or exit the game at any point.

4. Optional Enhancements:

Basic AI Opponent: An AI component that makes moves based on basic rules or strategies.

Difficulty Levels: Implement different AI difficulty settings (easy, medium, hard).

Score Tracking: Maintain a score tally for multiple rounds of play.

Sound Effects or Animations (if using a GUI).

5. Project Documentation:

Code Documentation: Properly commented code for easy understanding and maintenance.

User Guide: Instructions on how to play the game, installation steps, and other relevant details.

Technical Report (if applicable): Explanation of project structure, main algorithms, and design choices.

These requirements provide a structured path for creating a Tic Tac Toe game using Java, allowing for flexibility in the level of complexity based on project goals.

Source Code:

```
import java.util.Scanner;

class Main {
  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();
    }
  }
}
```

Explanation of Code:

Your Java implementation for the Tic Tac Toe game is solid and demonstrates a clear approach to building a functional two-player game. Here is a summary and analysis of your code:

Key Features

1. Board Initialization:

The board is represented by a 3x3 char array initialized to spaces (' ').

2. Player Input Handling:

The game continuously prompts players for input until there is a winner (gameOver is set to true).

User input is taken for row and column positions via a Scanner.

3. Game Logic:

It checks for valid moves, ensuring players cannot overwrite occupied cells.

The current player alternates between 'X' and 'O' using a ternary operator.

4. Winning Conditions:

The haveWon method checks rows, columns, and diagonals for winning conditions

5. Display Function:

The printBoard method displays the current state of the game board with a separator for clarity.

Possible Improvements

1.Input Validation

Currently, the program assumes valid inputs for row and column indices. You could add validation to ensure inputs are within bounds (0-2).

2. Draw Condition:

Add a check for a draw when all cells are filled, and there is no winner.

3. User-Friendly Output:

Improve the display output for better formatting (e.g., avoiding trailing | at the end of rows).

4. Loop Structure:

- Consider separating the input and game logic functions for a cleaner main loop.

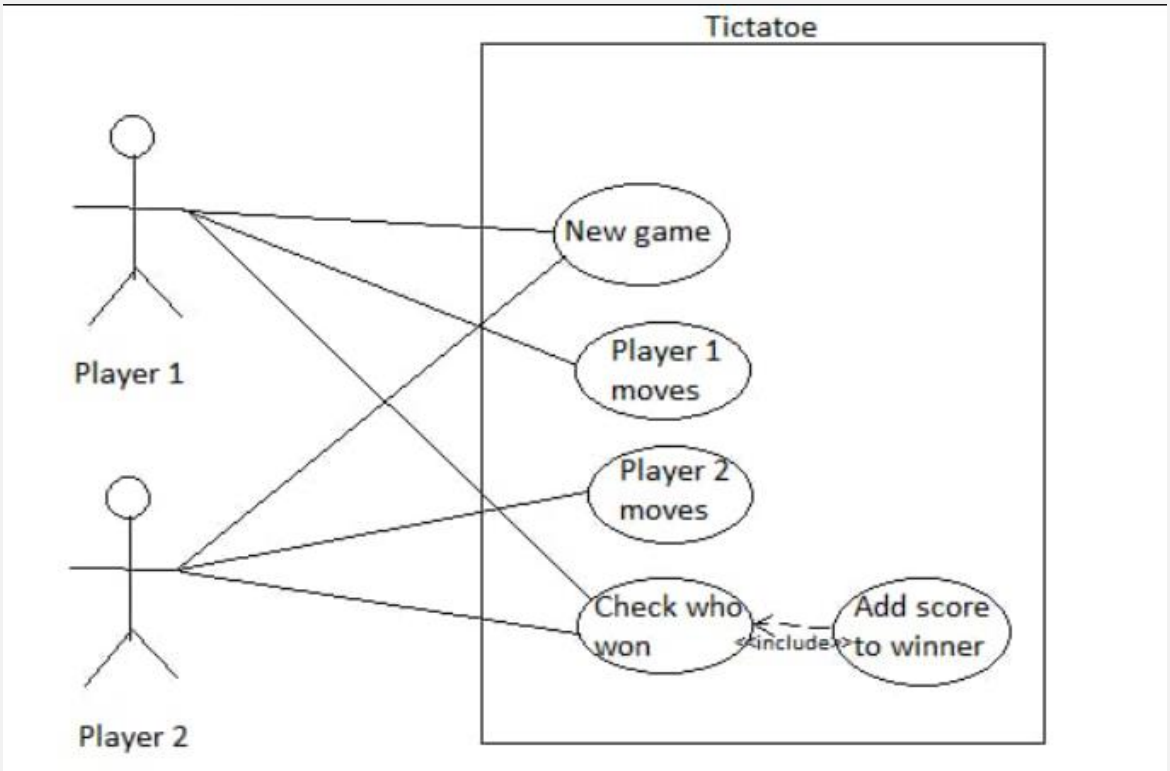
5. Additional Features:

Implement a reset option to allow another round of play. Allow configurable board sizes (e.g., NxN grid) and add flexible win conditions.

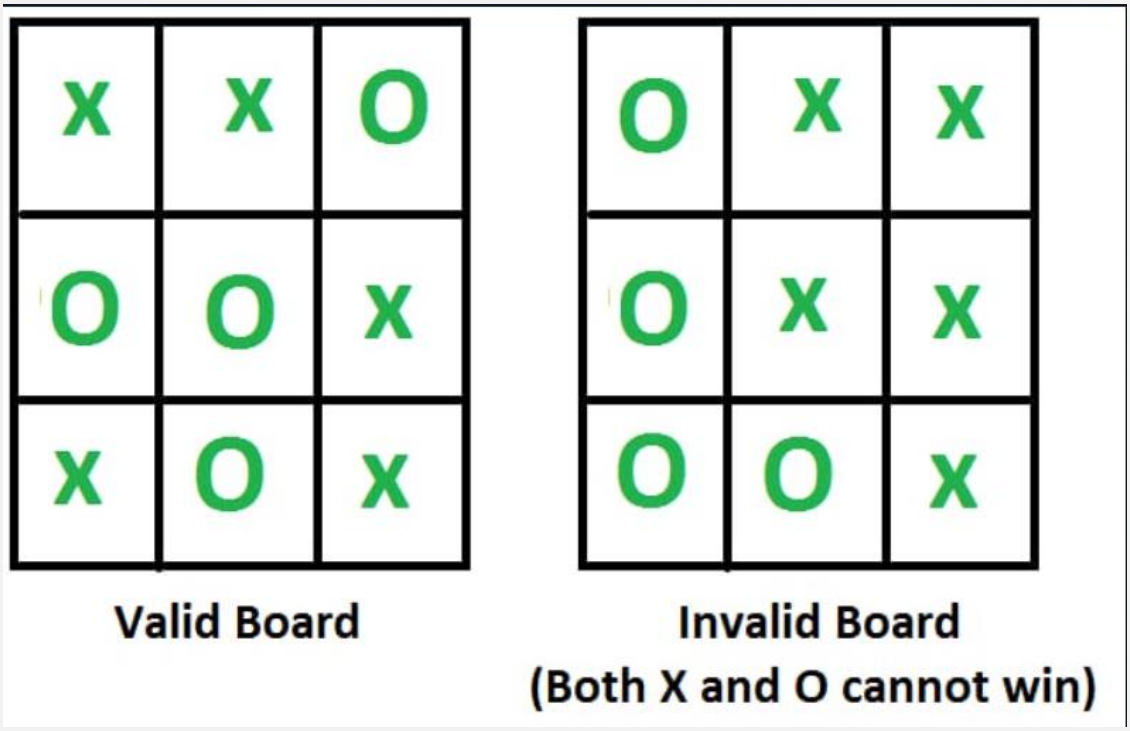
Overall Remarks

This code provides a great foundational example of a Tic Tac Toe game. It showcases essential Java concepts and offers room for enhancement and learning through added complexity.

UML Diagrams:



Condition for valid board:



Example of Console Output:

```
|  |
-----------
  |  |
-----------
  |  |
Player X enter: 0 0

X |  |
-----------
  |  |
-----------
  |  |
Player O enter: 1 1

X |  |
-----------
  | O |
-----------
  |  |
Player X enter: 0 1
```

X | X |

  | O |

  | |

Player O enter: 2 2

X | X |

  | O |

  | | O

Player X enter: 0 2

X | X | X

  | O |

  | | O

Player X has won!

Each move will prompt for input in the format row column (e.g., 0 0 for the top-left corner), and the game will continue until there's a winner or the board is full.

If you need further assistance with running or modifying the code, feel free to ask!

Conclusion

The Tic Tac Toe game project implemented in Java is a valuable exercise for understanding basic programming concepts and object-oriented principles. Through the design and implementation of the game, we have demonstrated several key aspects of Java, such as:

1. Array Management : Using a 2D array to represent the Tic Tac Toe board and managing user input for placing 'X' and 'O' on the board.

2. Control Structures : Employing loops and conditionals to validate user input, check win conditions, and handle the game flow.

3. Game Logic: Implementing the core mechanics of Tic Tac Toe, including switching between players, checking for a win or draw, and managing turn-based gameplay.

4. User Interaction : Handling input/output through the console for a simple and effective user interface.

This project also encourages problem-solving, as the logic for detecting winning conditions (across rows, columns, and diagonals) and ensuring valid moves must be carefully thought out. Additionally, enhancing the game with features like AI opponents, graphical interfaces, or game state persistence could further increase its complexity and educational value.

Overall, the Tic Tac Toe game project serves as a solid foundation for understanding how to structure simple games in Java and provides a platform for future enhancements, such as incorporating advanced algorithms or graphical user interfaces (GUIs).