TIC-TAC-TOE GAME

A PROJECT REPORT

Submitted by

KOMAL
UID: (23BCS11220)

in partial fulfillment for the award of the
degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE ENGINEERING

Submitted to:

PRAVINDRA KUMAR GOLE(T2225)

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1. Identification of Client/ Need/ Relevant Contemporary Issue

In the field of computer science education, beginner-level programming projects are crucial for learning core concepts such as GUI design, event handling, and logic implementation. Many students struggle to apply these concepts practically. A simple yet engaging project like a Tic Tac Toe game using Java Swing offers a great handson learning experience. The primary beneficiaries or "clients" of this project are students, educators, and developers.

1.2. Identification of Problem

While Tic Tac Toe is a basic game, implementing it with graphical interfaces and responsive logic in Java can be challenging for beginners. Existing examples are often either too complex or lack proper documentation.

1.3. Identification of Tasks

Analyze the requirements for a basic 3x3 Tic Tac Toe game.
Design the user interface using Java Swing.
Implement alternating player logic, win/draw conditions, and reset functionality.
Test the application across multiple scenarios.
Document the complete development and testing process.

1.4. Timeline

Week 1: Research on Java Swing and basic GUI frameworks.
Week 2: Design of the game layout and main classes.
Week 3: Implementation of core logic and win/draw validations. Week 4: Testing, debugging, and final documentation.

\

1.5. Organization of the Report

Chapter 1 outlines the problem context, goals, and project scope.
Chapter 2 reviews similar existing projects and defines the objectives.
Chapter 3 explains the design process, including logic flow and UI elements.
Chapter 4 covers the implementation results and validation of the application.
Chapter 5 concludes the project and discusses future work.

CHAPTER 2
LITERATURE REVIEW/BACKGROUND STUDY

2.1. Timeline of the Reported Problem

The need for simple, interactive programming examples has been ongoing for many years. Beginners often find it hard to implement GUI-based applications that involve real-time decision-making. Games like Tic Tac Toe offer a fun and effective way to understand logic building and GUI handling.

2.2. Existing Solutions

- Console-based Tic Tac Toe games in C, Python, etc.
- Basic Java GUI versions without modular structure
- Online multiplayer Tic Tac Toe using web technologies

2.3. Bibliometric Analysis

Various tutorials and textbooks highlight the importance of mini-games for teaching programming. Java Swing is frequently chosen due to its native support in Java and ability to build simple interfaces without external libraries.

2.4. Review Summary

Existing versions of the game often lack proper documentation and design structure. This project improves upon those by integrating a clear layout, modular code, and clean UI.

2.5. Problem Definition

Design and implement a 2-player Tic Tac Toe game in Java using the Swing framework with proper logic, interactive buttons, and win/draw/reset functionality.

2.6. Goals/Objectives

- Provide a simple and interactive GUI for Tic Tac Toe
- Implement win/draw condition detection

- Use modular and well-documented Java code
- Provide reset functionality for replay

CHAPTER 3
DESIGN FLOW / PROCESS

3.1 Evaluation & Selection of Specifications/Features

- Grid Layout: A 3x3 matrix of buttons to represent the game board.
- Player Symbols: “X” and “O” to denote player turns.
- Functionality: Detect win/draw conditions and display result.
- User Interface: Interactive GUI built using Java Swing.
- Reset Button: Allows restarting the game without relaunching the program.

3.2 Design Constraints

- No external libraries: Only standard Java and Swing packages to be used.
- Platform compatibility: Should run on any system with a Java runtime.
- Beginner-friendly code: Designed for learning and demonstration purposes.

3.3 Analysis of Features and Finalization Subject to Constraints

- The finalized features based on simplicity and effectiveness include:
- A clean and interactive 3x3 grid UI.
- Logical implementation of alternating turns.
- Real-time win/draw checking after every move.
- Reset bu on to replay without restarting the application.

3.4. Design Flow

- GUI Setup: Create main window using JFrame and set layout.
- Button Grid Creation: Initialize 9 buttons in a 3x3 arrangement.
- Event Handling: Attach ActionListener to each button for move detection.
- Logic Processing: Check for win/draw condition after every move.
- Output Display: Show winner or draw message using JLabel.
- Reset Game: Clear the board and start a new game on button click.

3.5. **Design Selection**

Java Swing was selected due to its: Simplicity and native support in JavaSuitability for desktop GUI-based educational applications.

CHAPTER 4
RESULTS ANALYSIS AND VALIDATION

4.1. Implementation of Solution

The Tic Tac Toe game was successfully developed using Java Swing, providing a practical and engaging user interface. The game features a 3x3 grid of interactive buttons, each representing an empty cell in the game board. Players alternate between "X" and "O," making their moves by clicking on the respective buttons. The application ensures smooth gameplay through the following core functionalities:

Turn-based Logic: The game alternates between two players, with Player 1 represented by "X" and Player 2 represented by "O." Each player's turn is managed through a simple event listener, ensuring that no player can make consecutive moves. After each move, the current player's symbol is placed on the button corresponding to their selection, and the game switches to the next player.

Win Detection: The game incorporates logic to check for winning conditions after every move. A win occurs when a player's symbols (either "X" or "O") are aligned across any row, column, or diagonal. The program checks these conditions by analyzing the state of the buttons on the grid. As soon as a player's winning combination is detected, the game halts further input and declares the winner.

Draw Condition: If all cells are filled and no winner is found, the game correctly identifies the draw condition. This is handled by verifying that all cells contain either "X" or "O" and ensuring no player has a winning combination. The user is then informed that the game has ended in a draw.

Reset Feature: After a game is concluded, players can choose to start a new round without needing to restart the application. This is facilitated by a reset button that clears the grid, allowing both players to play again. The reset function does not require any re-initialization of the entire game, providing a smooth user experience.

Testing and Validation: Testing was a critical part of ensuring the correctness and reliability of the game. The following test cases were considered to validate the implementation

Turn Alternation: Multiple tests were run to ensure that the game alternates correctly between players. If Player 1 starts with "X," Player 2 should automatically play "O" in the next turn, and so on.

Win Detection: The win detection logic was validated by testing various winning combinations (rows, columns, and diagonals). Scenarios like "X" winning horizontally, "O" winning vertically, and diagonal wins were tested extensively. The system consistently detected and declared winners as expected.

Draw Detection: The draw condition was tested by filling the grid with alternating moves that resulted in no winner. The system successfully identified these cases and declared the game a draw, ensuring that no false positives or negatives occurred.

Reset Functionality: The reset button was tested to ensure that it properly clears the game board and allows the players to start a new game without any issues. After resetting, the board should be empty, and the game should resume from Player 1's turn.

User Interface Interaction: The GUI was tested for responsiveness and ease of use. All buttons were accessible, and players could interact with the grid without delay. The interface was designed to be intuitive, allowing players to easily identify their moves and game status (win, draw, or ongoing).

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Conclusion

The successful development and implementation of the **Tic Tac Toe** game using **Java Swing** highlights the practicality of combining foundational programming logic with graphical user interface (GUI) development. The project meets all specified requirements and objectives, including:

- A visually intuitive and interactive 3x3 grid-based layout.
- Turn-based gameplay logic alternating correctly between two players.
- Accurate detection of win and draw conditions.
- A reset feature for restarting the game without closing the application.

This project not only replicates the classic Tic Tac Toe game effectively but also serves as an **educational tool** for beginners to understand GUI programming, event-driven logic, and modular coding principles in Java. The use of **Swing components** like JFrame, JPanel, and JButton, along with **action listeners**, demonstrates how real-time user interactions can be managed in a structured and efficient way.

Overall, the game strikes a balance between simplicity and functionality, making it an ideal learning model for students and aspiring developers venturing into Java GUI programming.

5.2. Future Work

While the current implementation meets all core objectives, there are several areas for potential improvement and enhancement. These future developments can add both educational value and user engagement:

- **Single-Player Mode with AI:**
  Introducing a basic **artificial intelligence (AI)** opponent will allow users to play solo. The AI could use simple decision-making algorithms like Minimax or Randomized Moves to challenge the user, making the game more dynamic and interactive.

- **Sound Effects and Visual Themes:**
  Enhancing the **user experience** with sound cues for actions (e.g., move played, win/draw announcements) and customizable **themes or color palettes** would improve engagement and usability, especially for younger audiences or casual users.
- **Scoreboard / Leaderboard Integration:**
  Implementing a **scoreboard** to keep track of wins, losses, and draws across multiple sessions would make the game more competitive and replayable. Persistent storage using file I/O or databases could be introduced for saving scores.
- **Online Multiplayer Functionality:**
  Extending the application to support **network-based multiplayer** would allow users to connect and play remotely. This would involve integrating **Java networking APIs** (like Socket, ServerSocket) and basic server-client communication protocols.
- **Mobile or Web-Based Version:**
  Rewriting the game using **JavaFX** for modern UI or transitioning to **web technologies** (using JavaScript/HTML5) could expand accessibility and compatibility across devices like smartphones and tablets.
- **Accessibility Features:**
  Adding features such as keyboard navigation, screen reader support, or customizable font sizes could make the application more inclusive to users with disabilities

USER MANUAL

Setup:

- Ensure Java is installed on your system.
- Open the project in an IDE like NetBeans or Eclipse. Compile the Java files.

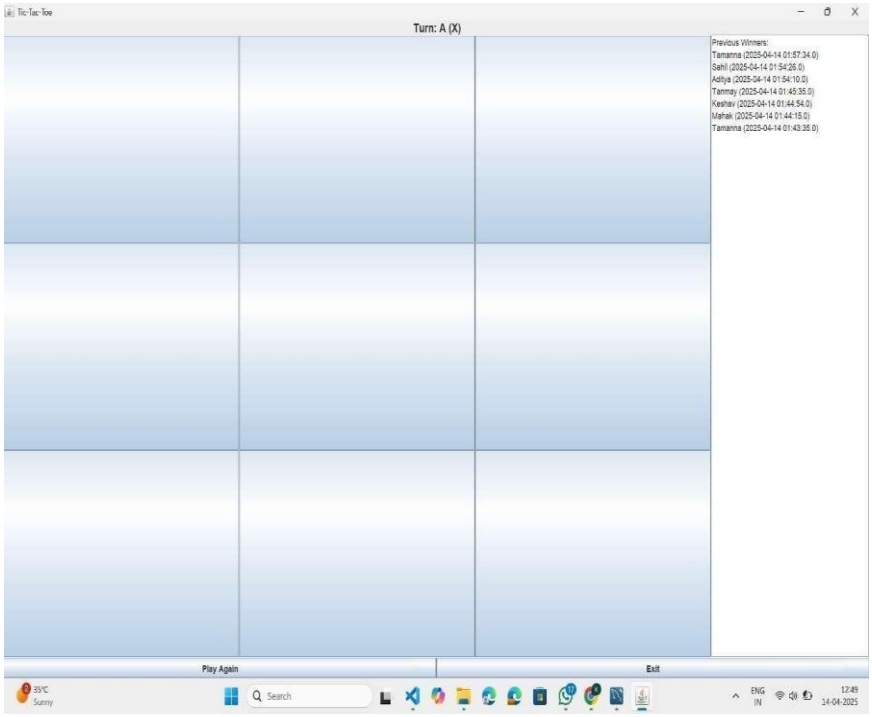
Operation:

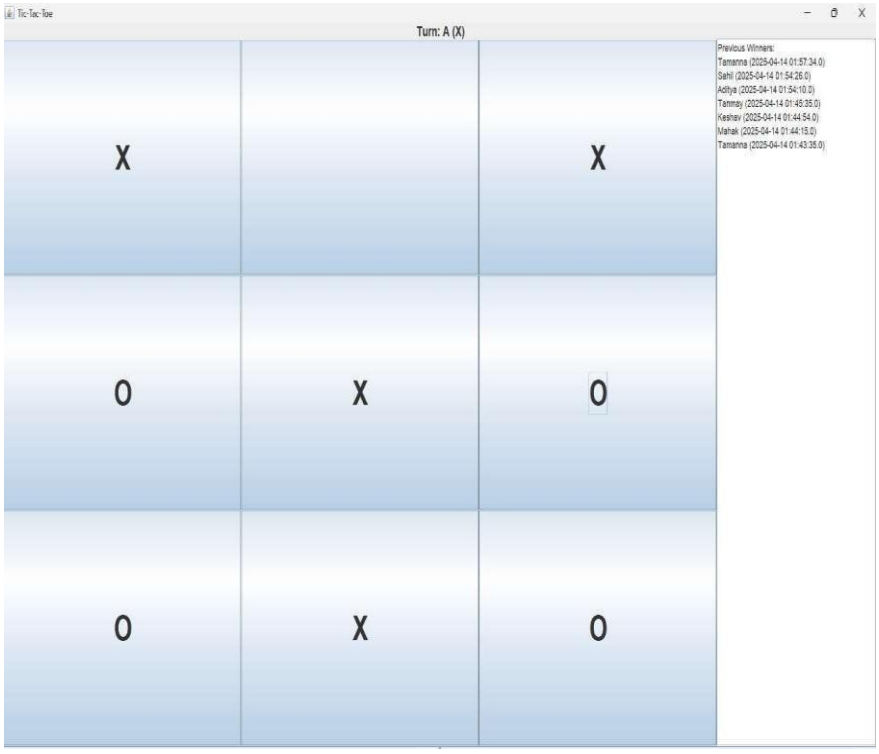
- Run the main class to launch the Tic Tac Toe game.
- Click on the grid buttons to play as Player X and O alternately.
- The result will be shown after a win or draw. Click the “Reset” button to restart the game.

Maintenance:

- Check code for logical errors before modifications.
- To add features like single-player mode or themes, update the code.

Final Output :





- In the final version of the Tic Tac Toe game, the following features and functionalities were successfully implemented:
- Player Interface: A user-friendly interface that allows two players to take turns marking X or O on the 3x3 grid.
- Game Logic: The game correctly identifies the winner or a draw after each turn, ensuring that the game stops when a player wins or the grid is filled.
  Error Handling: The program properly handles invalid moves (such as selecting an already marked cell or attempting to play after the game is over).
- Replay Option: Once a game is finished, players have the option to restart the game, allowing for multiple rounds of play.

REFERENCES

- Oracle Java Documentation – https://docs.oracle.com/en/java/
- Java Swing Tutorials – Javatpoint. https://www.javatpoint.com/java-swing
- Java: The Complete Reference by Herbert Schildt.
- GeeksforGeeks Java GUI – https://www.geeksforgeeks.org/java-awtandswing/
- Stack Overflow discussions on Java game logic and GUI design – https://stackoverflow.com/
- Java Swing Tutorial – TutorialsPoint. https://www.tutorialspoint.com/java/java_swing.htm
- Java Programming: Solving Problems with Software by Stuart Reges & Marty Stepp.
- Java AWT and Swing Tutorial – GeeksforGeeks. https://www.geeksforgeeks.org/java-awtandswing/
- Java Programming for Beginners – JavaCodeGeeks. https://www.javacodegeeks.com/
- Java GUI Programming with Swing by James Edward Keogh.