

# **TIC TAC TOE GAME**



## **A PROJECT REPORT**

*Submitted by*

**BAVADHARANI D ( 8115U23EC007 )**

*in partial fulfillment of requirements for the award of the course*

**EGB1201 - JAVA PROGRAMMING**

*in*

**ELECTRONICS AND COMMUNICATION  
ENGINEERING**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING**

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

**SAMAYAPURAM -621 112**

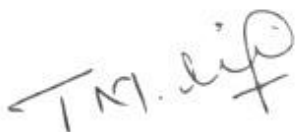
**DECEMBER - 2024**

**K. RAMAKRISHNAN COLLEGE OF ENGINEERING  
(AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “**TIC TAC TOE GAME**” is the bonafide work of **BAVADHARANI D ( 8115U23EC007)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



**SIGNATURE**

Dr. T. M. NITHYA, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

ASSOCIATE PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.



**SIGNATURE**

Mr.V.KUMARARAJA, M.E.,(Ph.D.),

**SUPERVISOR**

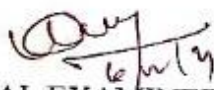
ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Engineering  
(Autonomous)

Samayapuram-621112.

Submitted for the viva-voce examination held on 06/12/24



INTERNAL EXAMINER



EXTERNAL EXAMINER



## DECLARATION

I declare that the project report on “**TIC TAC TOE GAME**” 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**



---

BAVADHARANI D

Place: Samayapuram

Date:6/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Engineering (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. D. SRINIVASAN, B.E, M.E., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. T. M. NITHYA, 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 supervisor **MR. V. KUMARARAJA, M.E., (Ph.D.)**, 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 achieve a prominent position among the top technical institutions.

## **MISSION OF THE INSTITUTION**

- M1: To bestow standard technical education par excellence through state of the art infrastructure, competent faculty and high ethical standards.
- M2: To nurture research and entrepreneurial skills among students in cutting edge technologies.
- M3: To provide education for developing high-quality professionals to transform the society.

## **VISION OF DEPARTMENT**

To create eminent professionals of Computer Science and Engineering by imparting quality education.

## **MISSION OF DEPARTMENT**

**M1:** To provide technical exposure in the field of Computer Science and Engineering through state of the art infrastructure and ethical standards.

**M2:** To engage the students in research and development activities in the field of Computer Science and Engineering.

**M3:** To empower the learners to involve in industrial and multi-disciplinary projects for addressing the societal needs.

## **PROGRAM EDUCATIONAL OBJECTIVES**

Our graduates shall

PEO1: Analyse, design and create innovative products for addressing social needs.

PEO2: Equip themselves for employability, higher studies and research.

PEO3: Nurture the leadership qualities and entrepreneurial skills for their successful career.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

- **PSO1:** Apply the basic and advanced knowledge in developing software, hardware and firm-ware solutions addressing real life problems.
- **PSO2:** Design, develop, test and implement product-based solutions for their career enhancement.

### **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 project focuses on creating a Tic-Tac-Toe game with a user-friendly graphical user interface (GUI) built using Java Swing, integrated with a MySQL database for tracking match results and player performance. The application is divided into three main components for clarity and modularity. The first component is the Game Logic, implemented in the `TicTacToeGame` class, which manages the mechanics of the game. It handles board setup, move validation, player switching, and winner detection. This ensures a clean separation of functionality from the user interface. The second component is the Graphical User Interface (GUI), built in the `TicTacToeGUI` class. It includes a visually intuitive 3x3 grid of buttons for gameplay, along with real-time updates on the current player, match progress, and results. The GUI supports multiple matches, ensuring seamless transitions and a smooth user experience. The final component is the Database Integration, handled by the `DatabaseHelper` class. The MySQL database contains two tables: `game_results` for individual match details and `overall_results` for aggregated performance statistics. Secure data handling is ensured using prepared statements, with methods to log and retrieve game data for historical analysis. This project demonstrates the integration of Java programming, GUI design, and SQL database management to create an engaging, scalable, and robust application. Players enjoy interactive gameplay while their performance data is recorded and accessible for review.

## ABSTRACT WITH POs AND PSOs MAPPING

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

ABSTRACT	POs MAPPED	PSOs MAPPED
<p>This project develops a Tic-Tac-Toe game with a graphical interface using Java Swing and MySQL database integration. The Game Logic, managed by the TicTacToeGame class, handles board setup, move validation, and player switching. The Graphical User Interface (GUI), implemented in the TicTacToeGUI class, offers a 3x3 grid with real-time updates, player status, and match tracking. The Database Integration by the DatabaseHelper class stores game results and overall statistics i, ensuring secure and reliable data management. Con MySQLm- bining Java programming, GUI design, and database handling, this project delivers a user-friendly and interactive gaming experience.</p>	<p><b>PO1 -3</b>  <b>PO2 -3</b>  <b>PO3 -3</b>  <b>PO4 -3</b>  <b>PO5 -3</b>  <b>PO6 -3</b>  <b>PO7 -3</b>  <b>PO8 -3</b>  <b>PO9 -3</b>  <b>PO10 -3</b>  <b>PO11-3</b>  <b>PO12 -3</b></p>	<p><b>PSO1 -3</b>  <b>PSO2 -3</b></p>

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

## **TABLE OF CONTENTS**

<b>CHAPTER NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	<b>ABSTRACT</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Objective	1
	1.2 Overview	1
	1.3 Java Programming concepts	2
<b>2</b>	<b>PROJECT METHODOLOGY</b>	<b>3</b>
	2.1 Proposed Work	3
	2.2 Block Diagram	4
	3.3 Graphical User Interface	6
	3.4 Match Results and Score Management	6
<b>3</b>	<b>MODULE DESCRIPTION</b>	<b>5</b>
	3.1 Player Input and Initialization	5
	3.2 Game Rules and Logic	5
	3.5 Database Integration	7
<b>4</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>8</b>
	4.1 Conclusion	8
	4.2 Future Scope	9
	<b>APPENDIX A (SOURCE CODE)</b>	<b>10</b>
	<b>APPENDIX B (SCREENSHOTS)</b>	<b>22</b>
	<b>REFERENCES</b>	<b>28</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1. Objective**

The primary objective of this project is to develop an interactive and user-friendly Tic-Tac-Toe Game using Java Swing for the graphical user interface and MySQL for data storage. The project aims to provide an engaging multiplayer experience where two players can compete, track their progress across multiple matches, and determine an overall winner. Additionally, the application ensures seamless integration with a database to store individual match results, player statistics, and overall game summaries, fostering robust data management and analysis capabilities.

### **1.1 Overview**

This project implements a Tic-Tac-Toe game using Java Swing for an interactive graphical interface and MySQL for persistent data storage. The game is designed for two players, allowing them to compete across multiple matches while maintaining a record of each game's outcome. Players' names, match results, and overall performance statistics are stored in a database for future reference. The application also provides features to handle invalid moves, track scores dynamically, and display the overall winner at the end of the series. By integrating a database, the project ensures scalability and easy retrieval of game data, making it a comprehensive and engaging application for players. The Graphical User Interface (GUI), implemented in the TicTacToeGUI class, offers a 3x3 grid with real-time updates, player status, and match tracking. The Database Integration by the DatabaseHelper class stores game results and overall statistics in MySQL, ensuring secure and reliable data management. Combining Java programming, GUI design, and database handling, this project delivers a user-friendly and interactive gaming experience.

### **1.3 Java Programming Concepts**

Object-Oriented Programming (OOP) is a foundational concept in programming that organizes code using objects and classes. Its primary principles include encapsulation, which bundles data and methods within a class to ensure data security and controlled access. Inheritance allows one class to derive properties and methods from another, promoting code reusability and hierarchical structuring. Polymorphism provides the ability to use a single interface for different implementations, offering flexibility and dynamic behavior through method overloading and overriding. Lastly, abstraction hides complex details while exposing only essential features, simplifying programming and enhancing focus on core functionality.

These principles form the backbone of modular, scalable, and maintainable software development. The Tic-Tac-Toe project extensively uses Java programming concepts to demonstrate practical applications. The GUI development utilizes Java Swing to create an interactive interface with components like buttons and labels, offering an engaging user experience. Event handling is implemented using ActionListener, enabling the game to respond dynamically to user actions. The core game logic, including move validation, player switching, and winner detection, is modularly structured to maintain code clarity and flexibility.

Additionally, the project integrates database connectivity using MySQL and JDBC, storing match results and player statistics for long-term analysis. To ensure data security, prepared statements are used for database operations, preventing SQL injection and maintaining data integrity. These elements showcase a comprehensive application of Java's features in real-world development scenarios.

## **CHAPTER 2**

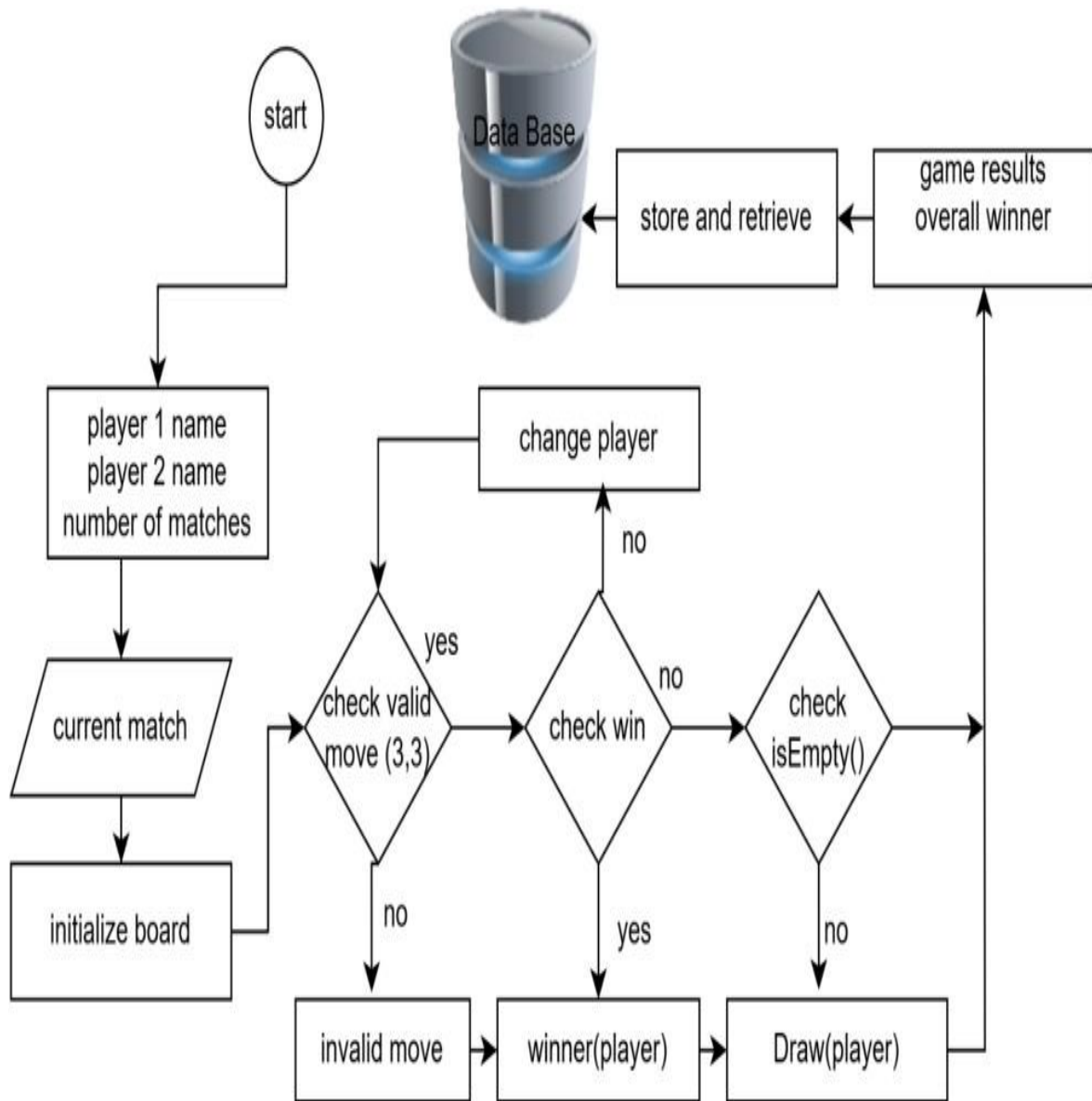
### **PROJECT METHODOLOGY**

#### **2.1 Proposed Work**

The flowchart outlines the detailed process of a Tic-Tac-Toe game with database integration, focusing on player interaction, game logic, and result storage. The game begins with the system prompting players to enter their names (Player 1 and Player 2) and specifying the number of matches to be played. The match count is initialized to track which game is currently being played out of the total number of matches. Once the game starts, the system initializes a 3x3 Tic-Tac-Toe board, which is set to be empty, preparing it for player moves. When a player selects a position to place their mark, the system checks if the move is valid by ensuring the selected spot is within the 3x3 grid and not already occupied. If the move is valid, the game continues; if invalid, the system notifies the player to try again. After each valid move, the system checks whether the current player has won by forming a complete row, column, or diagonal. If a winner is found, the current player is declared the winner, and the result is stored in the database. The game then advances to the next match, or ends if it was the last match.

If no winner is detected after a move, the system checks if the board is full, indicating a draw. If the board is full and no winner has been found, the game is declared a draw, and the result is also saved to the database. If neither a win nor a draw occurs, the system switches to the other player, and the game continues. Each match result is stored in the database, including the names of the players, the winner (if any), and whether the match ended in a draw. Once all matches are completed, the system retrieves the match results from the database and calculates the overall winner based on the number of matches each player has won. The overall winner and match statistics are then displayed to the players, marking the conclusion of the game.

## 2.2 Block Diagram



**Fig.2.1 Tic Tac Toe Game Play Logic**

## **CHAPTER 3**

### **MODULE DESCRIPTION**

#### **3.1 Player Input and Initialization**

The Player Input and Initialization step begins by prompting players to enter their names and the number of matches to be played. Player 1 and Player 2 provide their names, which will be used to identify them throughout the game. Additionally, the system asks for the number of matches, which sets the total rounds for the game. This information is crucial for managing game flow and tracking results. Once entered, the system initializes the match count to keep track of which round is being played. Following this, the game board is initialized as an empty 3x3 grid, ready for player moves. The player's input data is then used to set up the game, ensuring the proper structure for a series of matches. The initialization step sets the foundation for the entire game, organizing the flow and preparing the system for player interactions. This ensures that both players are ready to begin their first move in an organized and clear manner.

#### **3.2 Game Rules and Logic**

The Game Rules and Logic govern how the Tic-Tac-Toe game operates, ensuring a smooth and structured gameplay experience. The game follows basic rules: two players, Player 1 (X) and Player 2 (O), take turns marking spaces on a 3x3 grid. A player wins by placing three of their marks in a row, either horizontally, vertically, or diagonally. If the grid is filled and no player has won, the match ends in a draw. The game logic includes checks for valid moves to prevent players from selecting already occupied spaces. After each move, the system verifies if a player has won or if the board is full. If no winner is determined and the board isn't full, the turn switches to the other player.



### **3.3 Graphical User Interface (GUI)**

The Graphical User Interface (GUI) of the Tic-Tac-Toe game provides an intuitive and visually appealing way for players to interact with the game. Designed using Java Swing, the interface consists of a 3x3 grid of buttons, each representing a cell in the Tic-Tac-Toe board. Players can click on these buttons to make their moves, with the game updating the grid to show "X" or "O" depending on the current player. Alongside the board, the GUI displays essential information such as the current player's turn, match number, and overall game status. The interface provides real-time feedback, such as announcing the winner or if the match results in a draw. It also handles transitions between matches, ensuring a smooth and enjoyable user experience. The use of visual elements like buttons, labels, and message dialogs makes the game more engaging and accessible, even for users with minimal technical knowledge.

### **3.4 Match Results and Score Management**

Match Results and Score Management in the Tic-Tac-Toe game tracks and displays the outcomes of each match, providing an overview of player performance across multiple rounds. The system keeps a running tally of the number of wins, losses, and draws for each player. After every match, the game checks if there is a winner or if the game ends in a draw. Based on the result, the scores are updated for Player 1 and Player 2. The game then stores these results in a database for future reference. This allows for tracking overall performance across several matches. In addition to storing individual match outcomes, the system calculates the overall winner after all matches have been completed by comparing the total wins for each player.

### **3.5 Database Integration**

Database Integration in the Tic-Tac-Toe game is a crucial component that ensures match results and player performance are stored and managed efficiently. The game uses MySQL as the database to record and track the outcomes of each match, including details such as the players' names, the winner, the match number, and any draws. After each match, the system inserts the results into the database using prepared SQL statements to ensure security and prevent SQL injection. Two main tables are used: `game_results` for storing individual match details, and `overall_results` for keeping track of cumulative statistics, such as the total number of wins, losses, and draws for each player across multiple matches. This integration allows players to view historical results, analyze trends in their performance, and determine the overall winner after all matches are completed. By storing the data persistently, the game can easily retrieve and display past results, making the gaming experience more interactive and competitive. Database integration thus adds an important layer of functionality and user engagement to the game.

## **CHAPTER 4**

### **CONCLUSION & FUTURE SCOPE**

#### **4.1 CONCLUSION**

In conclusion, the Tic-Tac-Toe game project effectively combines core programming concepts such as object-oriented programming (OOP), graphical user interface (GUI) design, and database integration to create a fully functional and interactive gaming experience. The game logic, managed by the `TicTacToeGame` class, handles all critical operations like validating moves, detecting wins, and alternating turns between players. The GUI, developed using Java Swing, provides a user-friendly interface, allowing players to easily interact with the game on a 3x3 board.

Database integration enhances the game by storing match results and player statistics in MySQL, enabling players to track their performance over time. The use of secure SQL statements ensures data integrity and safety. The game's modular design allows for easy modifications and improvements, such as adding more features or refining the UI.

Moreover, the application offers a scalable platform for future enhancements, such as multiplayer options or advanced gameplay features. By integrating essential aspects of game development—logic, interface, and data management—the project not only demonstrates technical proficiency but also provides an engaging, competitive experience for users. The project successfully fulfills its objective of creating an interactive game with robust backend support, combining learning and fun in a seamless package.

## 4.2 FUTURE SCOPE

The future scope of this Tic-Tac-Toe game project is promising, with several opportunities for enhancements and new features. Firstly, the user interface can be upgraded with more modern design elements and animations to improve the overall player experience. Adding sound effects and visual cues for different game actions (like winning, losing, or drawing) could make the game more engaging.

A significant improvement would be to implement an AI opponent. Currently, the game requires two human players, but integrating a single-player mode with AI that uses algorithms like Minimax to make optimal moves would offer a more challenging experience for players.

Another area of improvement is expanding the game's multiplayer capabilities. Implementing online gameplay, where players can compete against each other over a network, would significantly increase the game's reach and user engagement.

Additionally, the database system can be expanded to track more detailed statistics such as player rankings, win/loss ratios, and player history over time. These can be integrated into a user account system, allowing players to sign in and track their progress.

To improve scalability and performance, the project could also be refactored to handle a larger number of concurrent users, especially in an online environment. This can be achieved through optimization techniques or by transitioning to more advanced frameworks and cloud databases.

Finally, incorporating features like a leaderboard, achievements, and social media integration can further increase the game's appeal and attract a larger user base. The future scope of this project is broad, with multiple avenues for technical advancements and user engagement.

## **APPENDIX A**

### **(SOURCE CODE)**

```
package javatictactoefinal;

import java.awt.*;
import java.awt.event.*;

public class TicTacToeAWT extends Frame implements ActionListener {

    private Button[] buttons = new Button[9]; // 9 buttons for the grid
    private char currentPlayer = 'X'; // Current player ('X' or 'O')
    private boolean gameWon = false; // To track if the game is won

    public TicTacToeAWT() {
        // Set up the Frame
        setTitle("Tic-Tac-Toe");
        setSize(400, 400);
        setLayout(new GridLayout(3, 3));

        // Initialize buttons
        for (int i = 0; i < 9; i++) {
            buttons[i] = new Button("");
            buttons[i].setFont(new Font("Arial", Font.BOLD, 50));
            buttons[i].addActionListener(this);
            add(buttons[i]);
        }

        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

```

    }
});

setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    if (gameWon) return; // Ignore clicks after the game is won

    Button clickedButton = (Button) e.getSource();

    // If the button is already marked, do nothing
    if (!clickedButton.getLabel().equals("")) {
        return;
    }

    // Mark the button with the current player's symbol
    clickedButton.setLabel(String.valueOf(currentPlayer));

    // Check if the game is won or drawn
    if (checkWin()) {
        gameWon = true;
        showResult("Player " + currentPlayer + " wins!");
    } else if (isDraw()) {
        showResult("It's a draw!");
    } else {
        // Switch player
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }
}

```

```

    }
}

private boolean checkWin() {
    // Check rows, columns, and diagonals
    int[][] winPatterns = {
        {0, 1, 2}, {3, 4, 5}, {6, 7, 8}, // Rows
        {0, 3, 6}, {1, 4, 7}, {2, 5, 8}, // Columns
        {0, 4, 8}, {2, 4, 6}           // Diagonals
    };

    for (int[] pattern : winPatterns) {
        if (!buttons[pattern[0]].getLabel().equals("") &&
            buttons[pattern[0]].getLabel().equals(buttons[pattern[1]].getLabel()) &&
            buttons[pattern[0]].getLabel().equals(buttons[pattern[2]].getLabel())) {
            return true;
        }
    }
    return false;
}

private boolean isDraw() {
    for (Button button : buttons) {
        if (button.getLabel().equals("")) {
            return false; // If any button is empty, the game is not a draw
        }
    }
    return true; // All buttons are filled and no one has won
}

```

```

private void showResult(String message) {
    // Display the result in a dialog box
    Dialog resultDialog = new Dialog(this, "Game Over", true);
    resultDialog.setLayout(new FlowLayout());
    resultDialog.setSize(300, 150);

    Label resultLabel = new Label(message);
    resultLabel.setFont(new Font("Arial", Font.BOLD, 16));
    resultDialog.add(resultLabel);

    Button okButton = new Button("OK");
    okButton.addActionListener(e -> System.exit(0));
    resultDialog.add(okButton);

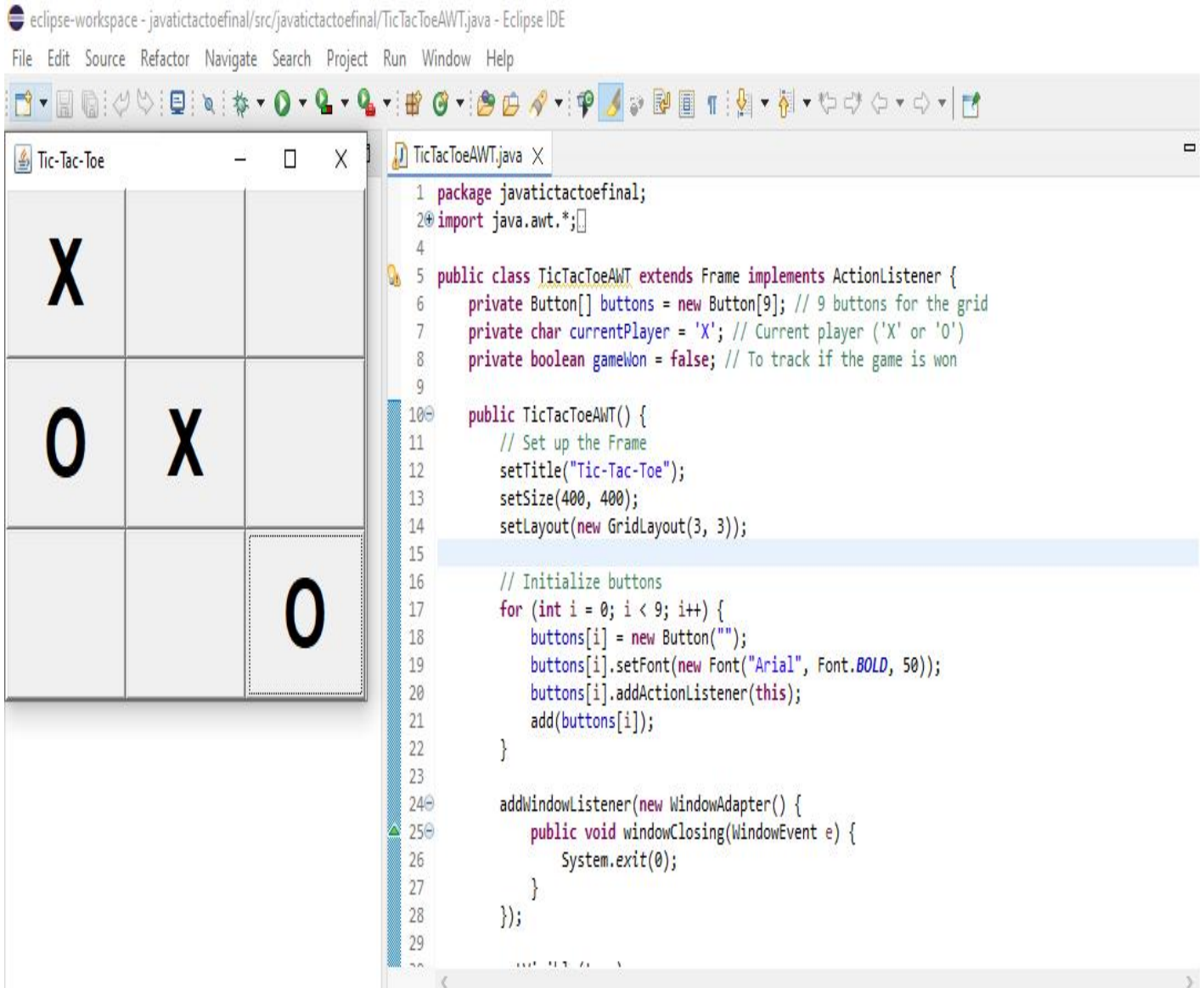
    resultDialog.setVisible(true);
}

public static void main(String[] args) {
    new TicTacToeAWT();
}
}

```



## APPENDIX B (SCREENSHOTS)



## REFERENCES

1. Herbert Schildt. “Java: The Complete Reference”, 11th Edition. McGraw Hill Education, 2018.
2. <https://docs.oracle.com/javase/tutorial/uiswing/>
3. <https://www.tutorialspoint.com/swing/index.htm>