**CODTECH IT SOLUTIONS TASK 4**

**For refactoring:**

Tic-Tac-Toe and the Need for Refactoring

What is Tic-Tac-Toe?

Tic-Tac-Toe is a classic two-player game where players take turns marking "X" or "O" in a 3×3 grid. The goal is to form a horizontal, vertical, or diagonal line with three of the same symbols to win the game. This game is widely used in programming as a beginner’s project to practice logic, user input handling, and graphical interface development.

Why Was Refactoring Needed?

The provided JavaFX-based Tic-Tac-Toe project was refactored to improve:

1. Code Readability – Improved naming conventions, modular structure, and reduced redundant code.

2. Performance Optimization – Removed unnecessary computations, optimized event handling, and improved UI responsiveness.

3. Maintainability – Separated game logic from UI, making future modifications easier.

4. Scalability – Made the game more adaptable for enhancements like AI opponents or additional features.

**Before (Original Code Example - handleButtonClick)**

public void handleButtonClick(ActionEvent event) {

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

if (clickedButton.getText().equals("")) {

clickedButton.setText(currentPlayer);

checkForWinner();

currentPlayer = (currentPlayer.equals("X")) ? "O" : "X";

}

}

**After (Refactored for Readability & Efficiency)**

public void handleButtonClick(ActionEvent event) {

Button button = (Button) event.getSource();

if (!button.getText().isEmpty()) return;

button.setText(currentPlayer);

if (checkForWinner()) {

displayWinner(currentPlayer);

} else {

togglePlayer();

}

}

private void togglePlayer() {

currentPlayer = (currentPlayer.equals("X")) ? "O" : "X";

}

Improvements:

Reduced redundant getText().equals("") check by using !isEmpty().

Extracted togglePlayer() method for cleaner logic.

checkForWinner() now returns a boolean for better readability.

**So we can do:**

**Step 1: Cloning the Project**

We can clone the repository using:

git clone https://github.com/akgarg0472/JavaFX\_TicTacToe.git

cd JavaFX\_TicTacToe

**Step 2: Identifying Areas for Refactoring**

From a quick review of the repository, we can optimize the following:

**1. Code Structure & Readability**

Improve class and method organization.

Use meaningful variable and function names.

Reduce redundant code.

**2. Performance Optimization**

Optimize event handling in JavaFX.

Use efficient loops and data structures.

Reduce unnecessary object creation

Etc.