# Tic-Tac-Toe OOAD Analysis

## Minimum Requirements

Place Symbols: As a player, I need to place X or O on the board to attempt to win. Implementation: The placeSymbol(row: int, col: int, symbol: char) method in the TicTacToeBoard class.

Detect Winning Condition: As a player, I need to detect when a winning condition is reached. Implementation: The isWinning(board: char[][], player: char) method in the TicTacToeBoard class.

Switch Players: As a player, I want the game to alternate turns between Player X and Player O. Implementation: The currentPlayer field in TicTacToeGame handles turn-switching logic.

Reset Game: As a player, I want an option to reset the board and start a new game. Implementation: The reset() method in the TicTacToeBoard class and startGame() method in TicTacToeGame.

Display Board: As a player, I want to view the current state of the board. Implementation: The show() method in the TicTacToeBoard class.

Detect Invalid Moves: As a player, I need to ensure I only place my symbol on empty cells. Implementation: The isEmpty(row: int, col: int) method in the TicTacToeBoard class.

## Main Nouns (Potential Classes)

Launcher (TicTacToeLauncher): Responsible for starting the game. Includes the main(args: String[]) method to initialize the application.

Game (TicTacToeGame): Manages the overall game logic, including turn-switching, game state, and tracking moves. Fields: board, playerX, playerO, currentPlayer, gameWon, and moves. Methods: startGame().

Board (TicTacToeBoard): Handles the representation and operations on the board. Fields: grid (a 2D array of characters). Methods: show(), isEmpty(), placeSymbol(), reset(), getGrid(), isWinning().

Player (TicTacToePlayer): Represents individual players. Fields: symbol (X or O), name. Methods: getSymbol(), getName().

## Pros & Cons of Object-Oriented Analysis and Design (OOAD)

### Pros

Modular Design: Classes (TicTacToeBoard, TicTacToePlayer, TicTacToeGame) clearly separate responsibilities, making the system easier to maintain and scale.

Clarity: The class-based structure provides a clear understanding of how data and operations are organized.

Scalability: New features (e.g., AI for a computer player) can be added without modifying existing classes significantly.

Testing: Each class can be tested individually, ensuring fewer errors when integrating.

### Cons

Upfront Design Overhead: Requires detailed planning, which may be excessive for simple games like Tic-Tac-Toe.

Complexity: Might overcomplicate small projects with a high number of classes and methods.

## MVP Approach

### Essentials

Board Display: Implement the TicTacToeBoard class to show the grid using the show() method.

Placing Symbols: Add functionality to place symbols using the placeSymbol(row, col, symbol) method, ensuring validity through isEmpty().

Win/Draw Detection: Add the isWinning() method to detect if a player has won.

Game Start and Reset: Enable game initialization and resetting using the startGame() and reset() methods in the TicTacToeGame class.

.