TicTacToe  
**Analyzing the Problem**

Tic-Tac-Toe is a classic two-player game played on a 3x3 grid. The goal is for one player to place three of their marks (either 'X' or 'O') in a horizontal, vertical, or diagonal row. The game ends when one player wins or the board is full (a draw).

**Inputs:**

* Player moves: Two players take turns inputting their moves by specifying the row and column where they want to place their mark ('X' or 'O').
* Board state: The 3x3 grid is updated after each move.

**Outputs:**

* Updated board state after each move.
* A message declaring the winner (if any) or a draw when the board is full.

**Algorithm:**

* Use a 2D array to represent the 3x3 board.
* Implement a loop to alternate turns between players.
* After each move, check for a win condition (three marks in a row, column, or diagonal).
* If no win condition is met and the board is full, declare a draw.

**Problem Criteria:**

* The game must handle two players.
* The board must be displayed after each move.
* The game must detect wins or draws.

**Approaches to Solution**

1. **Brute Force Approach:**
   * Check all possible winning combinations after each move.
   * This is simple and efficient for a 3x3 board.
2. **Optimization:**
   * Instead of checking all 8 winning combinations, stop checking once a win is detected.
   * Use a counter to track the number of moves to detect a draw without checking the entire board.
3. **Choice:**
   * I chose the brute force approach because it is straightforward and efficient enough for a small 3x3 board. The simplicity of the algorithm makes it easy to implement and debug.

**Test Cases**

1. **Player X Wins (Row):**
   * Moves: (0,0), (1,0), (0,1), (1,1), (0,2)
   * Expected: Player X wins.
2. **Player O Wins (Column):**
   * Moves: (0,0), (0,1), (1,0), (1,1), (2,2), (2,1)
   * Expected: Player O wins.
3. **Draw:**
   * Moves: (0,0), (0,1), (0,2), (1,1), (1,0), (1,2), (2,1), (2,0), (2,2)
   * Expected: Draw.
4. **Invalid Move:**
   * Moves: (0,0), (0,0)
   * Expected: "Invalid move! Try again."
5. **Player X Wins (Diagonal):**
   * Moves: (0,0), (0,1), (1,1), (1,2), (2,2)
   * Expected: Player X wins.

**Screenshots of Results**

1. **Player X Wins (Row):**

A screenshot of a computer screen

AI-generated content may be incorrect.

1. Player O Wins (Column):

A screenshot of a computer

AI-generated content may be incorrect.

1. Draw:

A black background with white text

AI-generated content may be incorrect.

1. Invalid Move:

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

1. Player X Wins (Diagonal):

A black background with white text

AI-generated content may be incorrect.

Conclusion

The Tic-Tac-Toe game was successfully implemented in Java. The program handles two players, validates moves, checks for win conditions, and detects draws. The test cases confirm that the program works as expected for all scenarios. The brute force approach was sufficient for this small-scale problem, and the code is well-commented for readability and maintainability.