Untitled-5

```java
import java.util.Scanner;

public class TicTacToe {
    private static char[][] board = new char[3][3];
    private static char currentPlayer = 'X';

    public static void main(String[] args) {
        initializeBoard();
        printBoard();

        while (true) {
            playerMove();
            printBoard();
            if (checkForWinner()) {
                System.out.println("Player " + currentPlayer + " wins!");
                break;
            }
            if (isBoardFull()) {
                System.out.println("The game is a draw!");
                break;
            }
            switchPlayer();
        }
    }

    // Initialize the game board with empty spaces
    private static void initializeBoard() {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                board[i][j] = '-';
            }
        }
    }

    // Print the current state of the game board
    private static void printBoard() {
        System.out.println("Current board:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(board[i][j] + " ");
            }
            System.out.println();
        }
    }

    // Handle a player's move
    private static void playerMove() {
        Scanner scanner = new Scanner(System.in);
```

```
        int row, col;

        while (true) {
            System.out.println("Player " + currentPlayer + ", enter your move (row and column): ");
            row = scanner.nextInt() - 1;
            col = scanner.nextInt() - 1;

            if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == '-') {
                board[row][col] = currentPlayer;
                break;
            } else {
                System.out.println("This move is not valid");
            }
        }
    }

    // Switch the current player
    private static void switchPlayer() {
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }

    // Check if the current player has won
    private static boolean checkForWinner() {
        return (checkRows() || checkColumns() || checkDiagonals());
    }

    // Check rows for a win
    private static boolean checkRows() {
        for (int i = 0; i < 3; i++) {
            if (board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) {
                return true;
            }
        }
        return false;
    }

    // Check columns for a win
    private static boolean checkColumns() {
        for (int i = 0; i < 3; i++) {
            if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer) {
                return true;
            }
        }
        return false;
    }

    // Check diagonals for a win
    private static boolean checkDiagonals() {
```

```
97          if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] ==
    currentPlayer) {
98              return true;
99          }
100         if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] ==
    currentPlayer) {
101             return true;
102         }
103         return false;
104     }
105
106     // Check if the board is full (for a draw)
107     private static boolean isBoardFull() {
108         for (int i = 0; i < 3; i++) {
109             for (int j = 0; j < 3; j++) {
110                 if (board[i][j] == '-') {
111                     return false;
112                 }
113             }
114         }
115         return true;
116     }
117 }
118
```