import java.util.Scanner;

public class TicTacToeChallenge {

    private static char[][] board = {{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};

    private static char currentPlayer = 'X';

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        boolean gameOngoing = true;

        System.out.println("Welcome to Tic Tac Toe!");

        printBoard();

        while (gameOngoing) {

            playerMove(scanner);

            printBoard();

            if (checkWinner()) {

                System.out.println("Player " + currentPlayer + " wins!");

                gameOngoing = false;

            } else if (isBoardFull()) {

                System.out.println("It's a draw!");

                gameOngoing = false;

            } else {

                // Switch player

                currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

            }

        }

        scanner.close();

    }

    // Prints the current state of the board

    private static void printBoard() {

        System.out.println("  0 1 2");

        for (int i = 0; i < 3; i++) {

            System.out.print(i + " ");

            for (int j = 0; j < 3; j++) {

                System.out.print(board[i][j]);

                if (j < 2) System.out.print("|");

            }

            System.out.println();

            if (i < 2) System.out.println("  -----");

        }

    }

    // Handles the player's move

    private static void playerMove(Scanner scanner) {

        int row, col;

        while (true) {

            System.out.println("Player " + currentPlayer + ", enter your move (row and column): ");

            row = scanner.nextInt();

            col = scanner.nextInt();

            if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ') {

                board[row][col] = currentPlayer;

                break;

            } else {

                System.out.println("This move is invalid. Try again.");

            }

        }

    }

    // Checks if the current player has won

    private static boolean checkWinner() {

        // Check rows and columns

        for (int i = 0; i < 3; i++) {

            if (board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) {

                return true;

            }

            if (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer) {

                return true;

            }

        }

        // Check diagonals

        if (board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) {

            return true;

        }

        if (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer) {

            return true;

        }

        return false;

    }

    // Checks if the board is full

    private static boolean isBoardFull() {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                if (board[i][j] == ' ') {

                    return false;

                }

            }

        }

        return true;

    }

}

OUTPUT:

