```
<terminated> TicTacToe [Java Application] R:\Program
Welcome to Tic-Tac-Toe
==================
 | |
-----
 | |
-----
 | |
Player 1 turn 'X'
Which Row would you like?
0
Which Col would you like?
0
X| |
-----
 | |
-----
 | |
Player 2 turn 'O'
Which Row would you like?
1
Which Col would you like?
0
X| |
-----
O| |
-----
 | |
Player 1 turn 'X'
Which Row would you like?
1
Which Col would you like?
1
X| |
-----
O|X|
-----
 | |
Player 2 turn 'O'
Which Row would you like?
1
Which Col would you like?
1
Illegal Move
Which Row would you like?
0
Which Col would you like?
2
X| |O
-----
O|X|
-----
 | |
Player 1 turn 'X'
Which Row would you like?
2
Which Col would you like?
2
```

```
X Player Wins!
X| |O
-----
O|X|
-----
 | |X
```

<terminated> TicTacToe [Java Application] R:\Program

```
Welcome to Tic-Tac-Toe
==================
 | |
-----
 | |
-----
 | |
Player 1 turn 'X'
Which Row would you like?
0
Which Col would you like?
0
X| |
-----
 | |
-----
 | |
Player 2 turn 'O'
Which Row would you like?
1
Which Col would you like?
1
X| |
-----
 |O|
-----
 | |
Player 1 turn 'X'
Which Row would you like?
0
Which Col would you like?
1
X|X|
-----
 |O|
-----
 | |
Player 2 turn 'O'
Which Row would you like?
0
Which Col would you like?
2
X|X|O
-----
 |O|
-----
 | |
Player 1 turn 'X'
Which Row would you like?
2
Which Col would you like?
0
X|X|O
-----
 |O|
-----
X| |
Player 2 turn 'O'
Which Row would you like?
0
Which Col would you like?
1
```

```
Illegal Move
Which Row would you like?
1
Which Col would you like?
0
X|X|O
-----
O|O|
-----
X| |
Player 1 turn 'X'
Which Row would you like?
1
Which Col would you like?
2
X|X|O
-----
O|O|X
-----
X| |
Player 2 turn 'O'
Which Row would you like?
2
Which Col would you like?
2
X|X|O
-----
O|O|X
-----
X| |O
Player 1 turn 'X'
Which Row would you like?
1
Which Col would you like?
2
Illegal Move
Which Row would you like?
2
Which Col would you like?
1
Cat's Game!
X|X|O
-----
O|O|X
-----
X|X|O
```

```java
import java.util.Scanner;

public class TicTacToe {

    public static void main(String[] args) {
        int Turn = 1;
        boolean PlayerWin = false;
        PlayerSide activePlayer = PlayerSide.X;
        Game_Board board = new Game_Board();
        int inputCol;
        int inputRow;
        Scanner input = new Scanner(System.in);


        System.out.println("Welcome to Tic-Tac-Toe\n==================");

//      printBoard(CurrentMoves);
        while(Turn < 10 && !PlayerWin) {

            board.displayBoard();

            if(activePlayer == PlayerSide.X) {
                System.out.println("Player 1 turn 'X'");
            }
            else {
                System.out.println("Player 2 turn 'O'");
            }

            System.out.println("Which Row would you like?");
            inputRow = input.nextInt();
            input.nextLine();

            System.out.println("Which Col would you like?");
            inputCol = input.nextInt();
            input.nextLine();

            while(!board.updateBoard(inputCol, inputRow, activePlayer)) {
                System.out.println("Illegal Move");

                System.out.println("Which Row would you like?");
                inputRow = input.nextInt();
                input.nextLine();

                System.out.println("Which Col would you like?");
                inputCol = input.nextInt();
                input.nextLine();
            }


            PlayerWin = checkWin(board.getStateBoard(), activePlayer);

            if(!PlayerWin) {

                Turn ++;

                if(activePlayer == PlayerSide.X) {
                    activePlayer = PlayerSide.O;
                }
```

```
                else {
                    activePlayer = PlayerSide.X;
                }
            }

        }

        if(PlayerWin) {
            System.out.println(activePlayer.getSide() + " Player Wins!");
        }
        else {
            System.out.println("Cat's Game!");
        }

        board.displayBoard();


        input.close();
    }

    public static boolean checkWin(PlayerSide CurrentMovesf[][], PlayerSide CurrentPlayer) {
        //col 1
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][0], CurrentMovesf[1][0], CurrentMovesf[2][0])) {
            return true;
        }
        //col 2
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][1], CurrentMovesf[1][1], CurrentMovesf[2][1])) {
            return true;
        }
        //col 3
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][2], CurrentMovesf[1][2], CurrentMovesf[2][2])) {
            return true;
        }
        //row 1
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][0], CurrentMovesf[0][1], CurrentMovesf[0][2])) {
            return true;
        }
        //row 2
        if(AreEqual(CurrentPlayer, CurrentMovesf[1][0], CurrentMovesf[1][1], CurrentMovesf[1][2])) {
            return true;
        }
        //row 3
        if(AreEqual(CurrentPlayer, CurrentMovesf[2][0], CurrentMovesf[2][1], CurrentMovesf[2][2])) {
            return true;
        }
        //diag 1
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][0], CurrentMovesf[1][1], CurrentMovesf[2][2])) {
            return true;
        }
        //diag 2
        if(AreEqual(CurrentPlayer, CurrentMovesf[0][2], CurrentMovesf[1][1], CurrentMovesf[2][0])) {
            return true;
        }
        return false;
    }

    public static boolean AreEqual(PlayerSide x1, PlayerSide x2, PlayerSide x3, PlayerSide x4) {
        if(x1 == x2 && x3 == x4 && x1 == x3) {

            return true;
          }
          return false;
      }
}

```

```java

public enum PlayerSide {
    X('X'), O('O'), nan(' ');

    private char side;

    private PlayerSide(char sidef) {
        side = sidef;
    }

    public char getSide() {
        return side;
    }

}

```

```java

public class Game_Board {
    PlayerSide currentBoard[][];

    Game_Board(){

        currentBoard = new PlayerSide[3][3];

        for(int i = 0; i < 3; i++) {
            for(int j = 0; j < 3; j++) {
                currentBoard[i][j] = PlayerSide.nan;
            }
        }
    }

    boolean updateBoard(int col, int row, PlayerSide activePlayer) {
        if(currentBoard[col][row] == PlayerSide.nan) {
            currentBoard[col][row] = activePlayer;
            return true;
        }
        else {
            return false;
        }
    }

    void displayBoard() {
        System.out.println(currentBoard[0][0].getSide() + "|" + currentBoard[1][0].getSide() + "|"
                + currentBoard[2][0].getSide());
        System.out.println("-----");
        System.out.println(currentBoard[0][1].getSide() + "|" + currentBoard[1][1].getSide() + "|"
                + currentBoard[2][1].getSide());
        System.out.println("-----");
        System.out.println(currentBoard[0][2].getSide() + "|" + currentBoard[1][2].getSide() + "|"
                + currentBoard[2][2].getSide());
    }

    PlayerSide[][] getStateBoard(){
        return currentBoard;
    }
}

```