//TICTACTOE GAME BY JAVA//

|  |
| --- |
|  |
| import java.util.Scanner;  class TicTacToe {  // Declaring Board  static char [] [] board; //Tic Tac Toe Board is nothing but a 2D array.  public TicTacToe () {  board = new char [3][3]; // default value == /u0000 == null character  initBoard ();  }  // Initialized Board  void initBoard () {  for (int i = 0; i<board.length; i++) { //row  for (int j=0; j<board[i].length; j++) { //column  board[i][j] = ' '; //filling the cells with space.  }  }  }  //Display Board  //Making the structure of the board.  static void displayBoard () {  System.out.println("-------------");  for (int i = 0; i<board.length; i++) {  System.out.print("| ");  for (int j=0; j<board[i].length; j++) {  System.out.print(board[i][j] + " | ");  }  System.out.println();  System.out.println("-------------");  }  }  //placing Mark  static void placemark (int row, int col, char mark) {  if(row>=0 && row<=2 && col>=0 && col<=2) { //before placing the mark, make sure that the position is valid or not.  board[row][col] = mark;  }  else {  System.out.println("Invalid Position"); //when the position is out of the array.  }  }  // Checking the win condition through columns.  static boolean checkColWin () {  for (int j=0; j<=2; j++) {  if(board[0][j] != ' ' && board[0][j]==board[1][j] && board[1][j]==board[2][j])  {  return true;  }  }  return false;  }    // Checking the win condition through Rows.  static boolean checkRowWin () {  for (int i=0; i<=2; i++) {  if(board[i][0] != ' ' && board[i][0]==board[i][1] && board[i][1]==board[i][2])  {  return true;  }  }  return false;  }    // Checking the win condition through Diagonal.  static boolean checkDiagWin () {  for (int i=0; i<=2; i++) {  if (board [0][0] != ' ' && board[0][0]==board[1][1] && board[1][1]==board[2][2] ||  board [0][2] != ' ' && board[0][2]==board[1][1] && board[1][1]==board[2][2])  {  return true;  }  }  return false;  }  }  // Creating a class for human players.  class HumanPlayer {  String name;  char mark;  HumanPlayer (String name, char mark) {  this.name=name;  this.mark=mark;  }    //Take row and column position from players.  void makeMove () {  Scanner sc = new Scanner (System.in);  int row;  int col;  do {  System.out.println("Enter the position..");  row = sc.nextInt();  col = sc.nextInt();  } while (! isValidMove(row,col)); // if user make an invalid move, he or she will get another chance until he put a valid move.  TicTacToe.placeMark(row,col,mark);  }  // valid move  boolean isValidMove(int row, int col) { // to check that the cell is empty or not and within the board.  if(row>=0 && row<=2 && col>=0 && col<=2)  {  If (TicTacToe.board[row][col] == ' ') //we bring the board by using TicTacToe class, because the board is static in nature.  {  return true;  }  }  return false;  }  }  //Main class.  public class Main  {  public static void main (String [] args) {  TicTacToe t = new TicTacToe ();  System.out.println("Game Started..!");  TicTacToe.displayBoard();    HumanPlayer p1 = new HumanPlayer("Joy", 'X');  HumanPlayer p2 = new HumanPlayer("Priya", '0');    HumanPlayer cp; //cp==current player.  cp=p1;    while(true)  {  System.out.println(cp.name + "'s turn..");  cp.makeMove();  TicTacToe.displayBoard();  If (TicTacToe.checkRowWin() || TicTacToe.checkColWin() || TicTacToe.checkDiagWin())  {  System.out.println("Game Over..!");  System.out.println(cp.name + " is the winner..!!");  break;  }  else  {  if(cp==p1)  {  cp = p2;  }  else  {  cp = p1;  }  }  }    }  } |