```
//Mario Diaz

import java.util.Scanner;



public class Main
{

      public static void main(String[] args)
      {

            TicTacToe game = new TicTacToe();

            game.play();

      }

}

public class TicTacToe
{
      // enums
      enum CellState {X, O, EMPTY}
      enum GameState {WIN, DRAW, CONTINUE}

      private CellState[][] board;
      private String winner = "";
      private CellState turn;
      private int col_choice;
      private int row_choice;
      private Scanner input;

      // constructor
      public TicTacToe()
      {
            board = new CellState[3][3];
            //Clears the board
            resetBoard();
            //X always makes first move
            turn = CellState.X;
            //Scanner is created
            input = new Scanner ( System.in );

            //Testing
            /*board [0][0] = CellState.X;
            board [0][1] = CellState.X;
            board [0][2] = CellState.X;*/
      }

      private String getCellText (CellState state)
      {
            if (state == CellState.X)
                  return " X |";
            else if (state == CellState.O)
                  return " O |";
            else
                  return "   |";
      }

      // printBoard() method
      public void printBoard()
      {
            String board_text = (" ___________\n");
            for(int i = 0; i < 3; i++)
            {
                  board_text += "|   |   |   |\n|";
                        for(int j = 0; j < 3; j++)
                        {
                              board_text += getCellText(board[i][j]);
                        }
                        board_text += "\n|___|___|___|\n";
            }
            System.out.println(board_text);
      }

      //Play() method loops until the game is finished
      public void play()
      {

            this.printBoard();

            GameState state = GameState.CONTINUE;
            while ( state == GameState.CONTINUE )
            {
                  this.getPlayerInput();
                  while( !this.validMove() )
                  {
                        System.out.println ( "Try again, that was not a
valid move.");
                        this.getPlayerInput();
                  }

                  board[ this.row_choice ][ this.col_choice ] = turn;

                  this.printBoard();

                  state = this.gameStatus();
                  if ( state == GameState.WIN )
                  {
                        System.out.println ("Player "+this.winner+"
wins.");
                  }
                  else if ( state == GameState.DRAW )
                  {
                        System.out.println("Game ends in a draw.");
                  }
```

```
				if ( turn == CellState.X )
				{
					turn = CellState.O;
				}
				else
				{
					turn = CellState.X;
				}

			}


		}


		public boolean validMove()
		{

			if(this.row_choice < 0 || this.row_choice > 2 ||
this.col_choice < 0 || this.col_choice > 2)
			{
				return false;
			}
			return board [this.row_choice] [this.col_choice] ==
CellState.EMPTY;

		}

		private void getPlayerInput()
		{
			String player = "X";
			if ( turn == CellState.O )
			{
				player = "O";
			}

			System.out.println ( "Player " +player+" 's turn." );
			System.out.print ( "Player " +player+ ":  Enter row (0, 1,
2):    ");
			this.row_choice = this.input.nextInt();
			System.out.print ( "Player " +player+ ":  Enter column (0, 1,
2):    ");
			this.col_choice = this.input.nextInt();

		}

		private void resetBoard()
		{
			for (int r=0; r<board.length; r++)
			{
				for (int c=0; c<board[r].length; c++)
				{
					board[r][c] = CellState.EMPTY;
				}
```

```
            }
      }


      public GameState gameStatus()
      {
            int x;
            GameState state = GameState.DRAW;
                  //Check Columns
            for ( x=0; x<3; x++ )
            {
                  if ( board[0][x] == CellState.X && board[1][x] ==
CellState.X && board[2][x] == CellState.X)
                  {
                        winner = "X";
                        return GameState.WIN;

                  }
                  else if ( board[0][x] == CellState.O && board[1][x] ==
CellState.O && board[2][x] == CellState.O)
                  {
                        winner = "O";
                        return GameState.WIN;

                  }
            }
                  //Check Rows
            for ( x=0; x<3; x++ )
            {
                  if ( board[x][0] == CellState.X && board[x][1] ==
CellState.X && board[x][2] == CellState.X)
                  {
                        winner = "X";
                        return GameState.WIN;

                  }
                  else if ( board[x][0] == CellState.O && board[x][1] ==
CellState.O && board[x][2] == CellState.O)
                  {
                        winner = "O";
                        return GameState.WIN;

                  }
            }

                  // Check Diagnols UpperLeft to LowerRight
            if ( board[0][0] == CellState.X && board[1][1] == CellState.X
&& board[2][2] == CellState.X)
            {
                  winner = "X";
                  return GameState.WIN;
            }
            else if ( board[0][0] == CellState.O && board[1][1] ==
CellState.O && board[2][2] == CellState.O)
            {
                  winner = "O";
                  return GameState.WIN;
            }

                  // Check Diagnols UpperRight to BottomLeft
            if ( board[0][2] == CellState.X && board[1][1] == CellState.X
&& board[2][0] == CellState.X)
            {
                  winner = "X";
                  return GameState.WIN;
            }
            else if ( board[0][0] == CellState.O && board[1][1] ==
CellState.O && board[2][2] == CellState.O)
            {
                  winner = "O";
                  return GameState.WIN;
            }


                  //Check if every square is filled
                  //    r=row c=column*
            for ( int r=0; r<3; r++ )
            {
                  for ( int c=0; c<3; c++ )
                  {
                        if ( board[r][c] == CellState.EMPTY )
                        {
                              return GameState.CONTINUE;
                        }
                  }

            }return state;
      }



}
```