

**NAME : SANKALP MUKIM**

**REG NO:** **20BDS0128**

**NAME : HARDIK JAIN S**

**REG NO: 20BDS0063**

**SUBMITTED TO : Dr.VARALAKSHMI M**

**SUBJECT: Java On-spot Project**

**Demonstration Video:** [**https://youtu.be/OG\_CzEoG9fw**](https://youtu.be/OG_CzEoG9fw)

Q] Design and develop a game application using JavaFX that includes the necessary graphical components, UI controls and event handling mechanism. Incorporate as many (relevant) features of JavaFX as possible.

**Source code:**

import java.util.ArrayList;

import java.util.List;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Pos;

import javafx.stage.Stage;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.paint.Color;

import javafx.scene.shape.Rectangle;

import javafx.scene.text.Text;

public class Project extends Application {

    String currMove;

    String[][] grid = { { "", "", "" }, { "", "", "" }, { "", "", "" } };

    Text wonState = new Text("Click to start game");

    Text currPlayer = new Text("Chance of: X");

    Boolean playable = true;

    List<Tile> tiles = new ArrayList<>();

    public static void main(String[] args) {

        launch();

    }

    private String checkWin() {

        for (int i = 0; i < grid.length; i++) {

            if ((grid[i][0] == "X") && (grid[i][0] == grid[i][1]) && (grid[i][0] == grid[i][2])) {

                return "X";

            }

            if ((grid[0][i] == "X") && (grid[0][i] == grid[1][i]) && (grid[0][i] == grid[2][i])) {

                return "X";

            }

            if ((grid[i][0] == "O") && (grid[i][0] == grid[i][1]) && (grid[i][0] == grid[i][2])) {

                return "O";

            }

            if ((grid[0][i] == "O") && (grid[0][i] == grid[1][i]) && (grid[0][i] == grid[2][i])) {

                return "O";

            }

        }

        if ((grid[0][0] == "X") && (grid[0][0] == grid[1][1]) && (grid[0][0] == grid[2][2])) {

            return "X";

        } else if ((grid[0][0] == "O") && (grid[0][0] == grid[1][1]) && (grid[0][0] == grid[2][2])) {

            return "O";

        }

        return "";

    }

    void changeMove() {

        if (currMove == "X") {

            currMove = "O";

            currPlayer.setText("Chance of: O");

        } else {

            currMove = "X";

            currPlayer.setText("Chance of: X");

        }

        String winState = checkWin();

        if (winState == "X") {

            wonState.setText("Game Won by X!");

            currPlayer.setText("Reset board to continue playing.");

            playable = false;

        } else if (winState == "O") {

            wonState.setText("Game Won by O!");

            currPlayer.setText("Reset board to continue playing.");

            playable = false;

        } else {

            wonState.setText("Not won!");

        }

    }

    private Parent createContent() {

        wonState.setFill(Color.BLACK);

        wonState.setX(175);

        wonState.setY(50);

        currPlayer.setFill(Color.BLACK);

        currPlayer.setX(325);

        currPlayer.setY(50);

        Button resetButton = new Button("Reset board");

        resetButton.setLayoutX(560);

        resetButton.setLayoutY(35);

        resetButton.setCancelButton(true);

        resetButton.setAlignment(Pos.CENTER);

        currMove = "X";

        Pane root = new Pane();

        root.setPrefSize(800, 750);

        Pane grid = new Pane();

        grid.setPrefSize(600, 600);

        grid.setLayoutX(100);

        grid.setLayoutY(125);

        root.getChildren().addAll(wonState, currPlayer, resetButton, grid);

        for (int i = 0; i < 3; i++) {

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

                Tile tile = new Tile(i, j);

                tile.setTranslateX(j \* 200);

                tile.setTranslateY(i \* 200);

                tiles.add(tile);

                grid.getChildren().add(tile);

            }

        }

        resetButton.setOnAction(new EventHandler<ActionEvent>() {

            @Override

            public void handle(ActionEvent ev) {

                playable = true;

                makeGridEmpty(grid);

                wonState.setText("Not won!");

                currPlayer.setText("Chance of: " + currMove);

            }

        });

        return root;

    }

    private void makeGridEmpty(Pane grid) {

        for (int i = 0; i < this.grid.length; i++) {

            for (int j = 0; j < this.grid.length; j++) {

                this.grid[i][j] = "";

            }

        }

        for (Tile tile : this.tiles) {

            tile.setEmpty();

        }

    }

    @Override

    public void start(Stage stage) {

        stage.setScene(new Scene(createContent()));

        stage.setTitle("Tic Tac Toe game by Sankalp and Hardik");

        stage.setResizable(false);

        stage.show();

    }

    private class Tile extends StackPane {

        private Text text = new Text();

        private int i, j;

        public Tile(int I, int J) {

            this.i = I;

            this.j = J;

            Rectangle border = new Rectangle(200, 200);

            border.setFill(null);

            border.setStroke(Color.BLACK);

            setAlignment(Pos.CENTER);

            getChildren().addAll(border, text);

            setOnMouseClicked(event -> {

                if (playable) {

                    if (currMove == "X") {

                        drawX();

                    } else {

                        drawO();

                    }

                    changeMove();

                }

            });

        }

        private void drawX() {

            text.setText("X");

            grid[i][j] = "X";

        }

        private void drawO() {

            text.setText("O");

            grid[i][j] = "O";

        }

        private void setEmpty() {

            text.setText("");

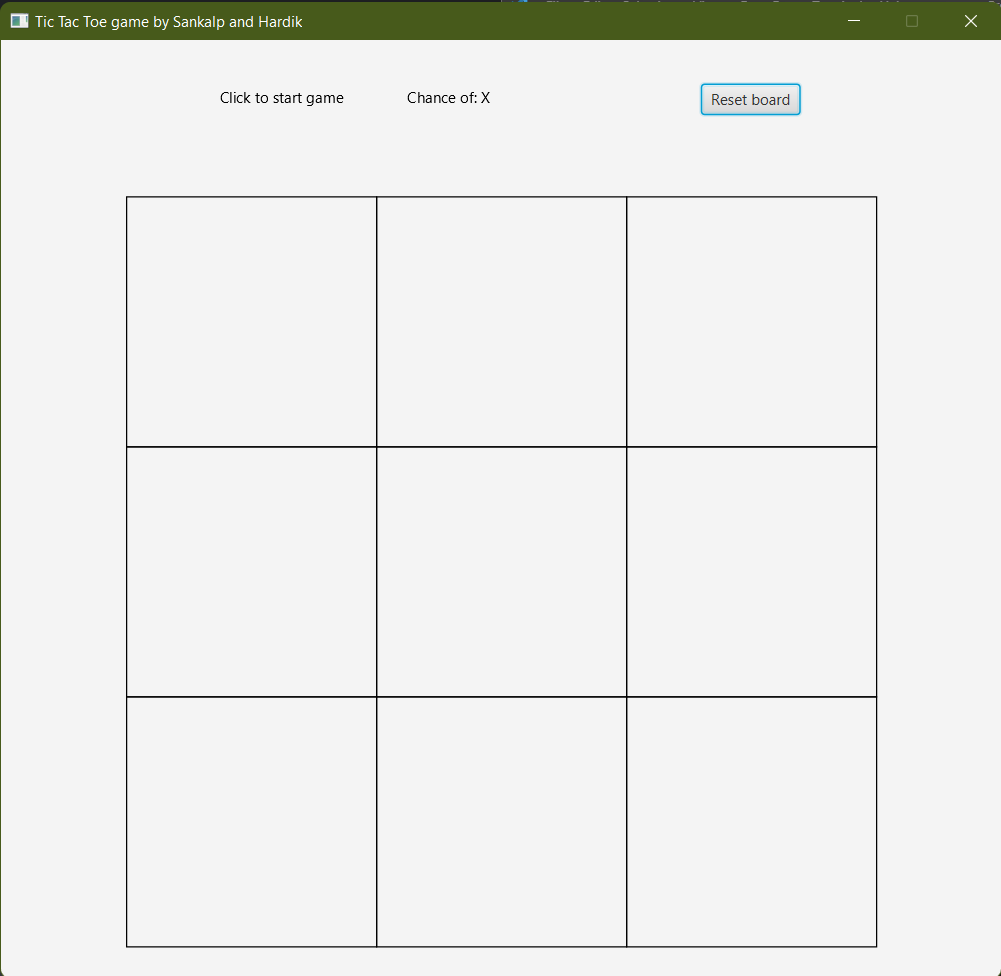
        }

    }

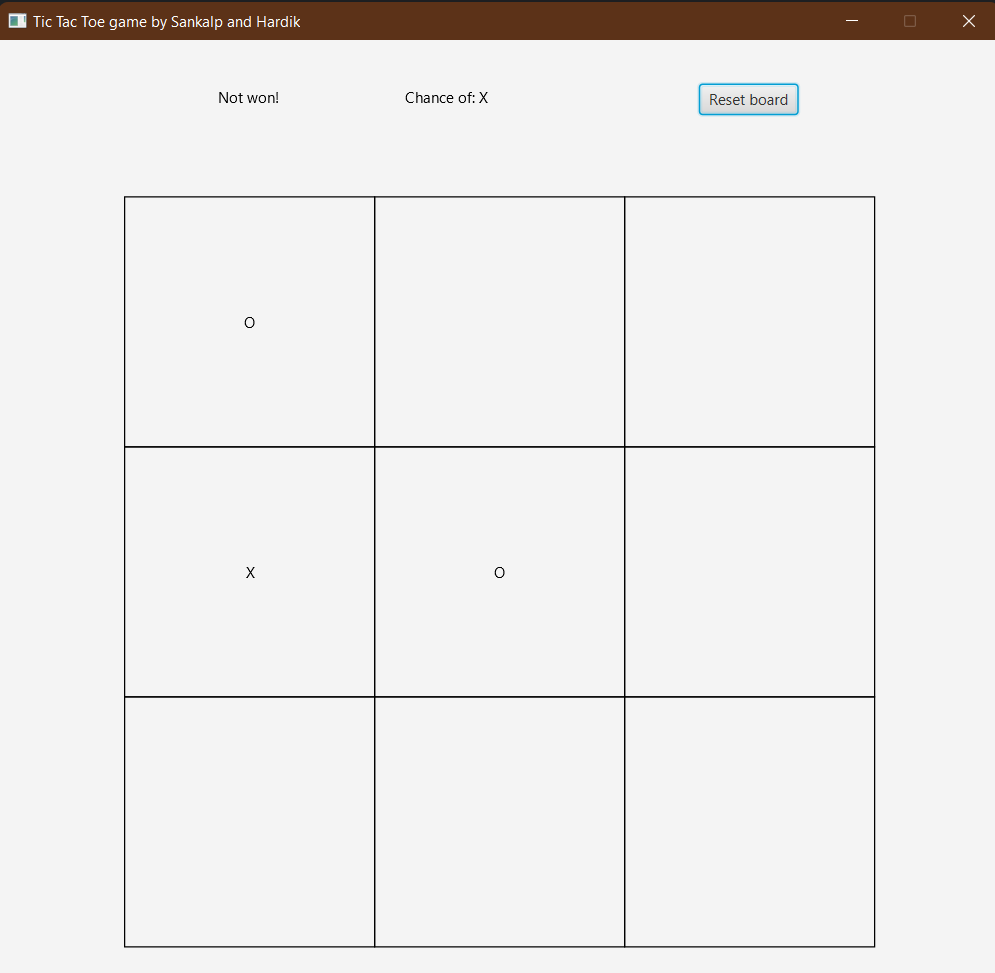
}

**SNAP SHOTS OF THE OUTPUT:**

**Beginning state: Empty 3x3 grid**



**Intermediate state: Game dynamically updates and tells which player's turn is it.**



**Won state: Board freezes until player resets the board.**

