Student Name :  
Student ID :

Programming files as follow.

I have took the GUI window size is 400 width and 400 height. I have completed the task of TicTacToe programming based on user interaction. As show bellow code.

***TicTacToe.java:***

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class TicTacToe extends JFrame

{

public static void main(String [] args)

{

new TicTacToe();

}

private JButton buttonObjA1, buttonObjA2, buttonObjA3, buttonObjB1, buttonObjB2, buttonObjB3, buttonObjC1, buttonObjC2, buttonObjC3;

private TicTacToeBoardInfo tictactoeBoardObject;

public TicTacToe()

{

// Set up the grid

this.setSize(400,400);

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

this.setTitle("Tic-Tac-Toe");

JPanel panel1 = new JPanel();

panel1.setSize(300,300);

panel1.setLayout(new GridLayout(3,3));

buttonObjA1 = createButton("A1");

buttonObjA2 = createButton("A2");

buttonObjA3 = createButton("A3");

buttonObjB1 = createButton("B1");

buttonObjB2 = createButton("B2");

buttonObjB3 = createButton("B3");

buttonObjC1 = createButton("C1");

buttonObjC2 = createButton("C2");

buttonObjC3 = createButton("C3");

panel1.add(buttonObjA1);

panel1.add(buttonObjA2);

panel1.add(buttonObjA3);

panel1.add(buttonObjB1);

panel1.add(buttonObjB2);

panel1.add(buttonObjB3);

panel1.add(buttonObjC1);

panel1.add(buttonObjC2);

panel1.add(buttonObjC3);

this.add(panel1);

this.setVisible(true);

// Start the game

tictactoeBoardObject = new TicTacToeBoardInfo();

}

private JButton createButton(String square)

{

JButton button = new JButton();

button.setPreferredSize(new Dimension(100, 100));

Font f = new Font("Dialog", Font.PLAIN, 80);

button.setFont(f);

button.addActionListener(e -> buttonClick(e, square));

return button;

}

private void buttonClick(ActionEvent e, String square)

{

if (tictactoeBoardObject.getSquare(square) != 0)

return;

JButton button = (JButton)e.getSource();

button.setText("X");

tictactoeBoardObject.playAt(square, 1);

if (tictactoeBoardObject.isGameOver() == 3)

{

JOptionPane.showMessageDialog(null,"It's a draw!", "Game Over",JOptionPane.INFORMATION\_MESSAGE);

resetGame();

return;

}

if (tictactoeBoardObject.isGameOver() == 1)

{

JOptionPane.showMessageDialog(null,"You beat me!", "Game Over",JOptionPane.INFORMATION\_MESSAGE);

resetGame();

return;

}

String computerMove = tictactoeBoardObject.getNextMove();

tictactoeBoardObject.playAt(computerMove,2);

switch (computerMove)

{

case "A1":

buttonObjA1.setText("O");

break;

case "A2":

buttonObjA2.setText("O");

break;

case "A3":

buttonObjA3.setText("O");

break;

case "B1":

buttonObjB1.setText("O");

break;

case "B2":

buttonObjB2.setText("O");

break;

case "B3":

buttonObjB3.setText("O");

break;

case "C1":

buttonObjC1.setText("O");

break;

case "C2":

buttonObjC2.setText("O");

break;

case "C3":

buttonObjC3.setText("O");

break;

}

if (tictactoeBoardObject.isGameOver() == 2)

{

JOptionPane.showMessageDialog(null,"I beat you!", "Game Over",JOptionPane.INFORMATION\_MESSAGE);

resetGame();

return;

}

}

private void resetGame()

{

tictactoeBoardObject.reset();

buttonObjA1.setText("");

buttonObjA2.setText("");

buttonObjA3.setText("");

buttonObjB1.setText("");

buttonObjB2.setText("");

buttonObjB3.setText("");

buttonObjC1.setText("");

buttonObjC2.setText("");

buttonObjC3.setText("");

}

}

**TicTacToeBoardInfo.java**:

public class TicTacToeBoardInfo

{

private int board [];

private int vectors [] [] =

{

{0, 1, 2},

{3, 4, 5},

{6, 7, 8},

{0, 3, 6},

{1, 4, 7},

{2, 5, 8},

{0, 4, 8},

{2, 4, 6}

};

public TicTacToeBoardInfo()

{

this.reset();

}

public void reset()

{

board = new int[] {2, 2, 2, 2, 2, 2, 2, 2, 2};

}

private int getSquare(int indexVal)

{

if (indexVal < 0 | indexVal > 8)

throw new IllegalArgumentException("index must be 0-9");

return board[indexVal];

}

public int getSquare(String squareVal)

{

int indexVal = mapSquareToIndex(squareVal);

if (indexVal == -1)

throw new IllegalArgumentException("Invalid square");

switch (getSquare(indexVal))

{

case 3:

return 1;

case 5:

return 2;

default:

return 0;

}

}

private int mapSquareToIndex(String squareVal)

{

switch (squareVal)

{

case "A1":

return 0;

case "A2":

return 1;

case "A3":

return 2;

case "B1":

return 3;

case "B2":

return 4;

case "B3":

return 5;

case "C1":

return 6;

case "C2":

return 7;

case "C3":

return 8;

default:

return -1;

}

}

private String mapIndexToSquare(int indexValue)

{

switch (indexValue)

{

case 0:

return "A1";

case 1:

return "A2";

case 2:

return "A3";

case 3:

return "B1";

case 4:

return "B2";

case 5:

return "B3";

case 6:

return "C1";

case 7:

return "C2";

case 8:

return "C3";

default:

return "";

}

}

public void playAt(String squareValue, int playerValue)

{

int indexValue = mapSquareToIndex(squareValue);

if (indexValue == -1)

throw new IllegalArgumentException("Invalid square");

this.playAt(indexValue, playerValue);

}

private void playAt(int indexValue, int playerValue)

{

if (indexValue < 0 | indexValue > 8)

throw new IllegalArgumentException("Square must be 0-8");

if (playerValue <1 | playerValue > 2)

throw new IllegalArgumentException("Player must be 1 or 2");

if (board[indexValue] != 2)

throw new IllegalArgumentException("Square is not empty.");

if (playerValue == 1)

board[indexValue] = 3;

else

board[indexValue] = 5;

}

public int isGameOver()

{

for (int i = 0; i < 8; i++)

{

int k = getVectorProduct(i);

if (k == 27)

return 1;

if (k == 125)

return 2;

}

int blankCount = 0;

for (int j = 0; j < 9; j++)

if (board[j] == 2)

blankCount++;

if (blankCount == 0)

return 3; // Game is a draw

return 0; // Game is not over

}

public String canPlayerWin(int playerValue)

{

if (playerValue <1 | playerValue > 2)

throw new IllegalArgumentException("player must be 1 or 2");

boolean playerCanWin = false;

for (int v = 0; v < 8; v++)

{

int p = getVectorProduct(v);

if ( (playerValue == 1 & p == 18) | (playerValue == 2 & p == 50) )

{

if (board[vectors[v][0]] == 2)

return mapIndexToSquare(vectors[v][0]);

if (board[vectors[v][1]] == 2)

return mapIndexToSquare(vectors[v][1]);

if (board[vectors[v][2]] == 2)

return mapIndexToSquare(vectors[v][2]);

}

}

return "";

}

private int getVectorProduct(int vector)

{

return board[vectors[vector][0]] \*

board[vectors[vector][1]] \*

board[vectors[vector][2]];

}

public String getNextMove()

{

String myGameMove;

myGameMove = this.canPlayerWin(2);

if (myGameMove != "")

return myGameMove;

myGameMove = this.canPlayerWin(1);

if (myGameMove != "")

return myGameMove;

if (board[4] == 2)

return "B2";

if (board[0] == 2)

return "A1";

if (board[2] == 2)

return "A3";

if (board[6] == 2)

return "C1";

if (board[8] == 2)

return "C3";

if (board[1] == 2)

return "A2";

if (board[3] == 2)

return "B1";

if (board[5] == 2)

return "B3";

if (board[7] == 2)

return "C2";

return "";

}

public String toString()

{

return " " +

getMark(board[0]) + " | " +

getMark(board[1]) + " | " +

getMark(board[2]) +

"\n-----------\n" +

" " +

getMark(board[3]) + " | " +

getMark(board[4]) + " | " +

getMark(board[5]) +

"\n-----------\n" +

" " +

getMark(board[6]) + " | " +

getMark(board[7]) + " | " +

getMark(board[8]);

}

private String getMark(int markStatus)

{

if (markStatus == 3)

return "X";

if (markStatus == 5)

return "O";

return " ";

}

}

**Output:**









