Java Mini Project On

TICTACTOE GAME

Batch no:#2

Submitted by:

17121A1215	D SASI KUMAR
17121A1229	G SREEKANTH
17121A1219	E BHUPAL VIKRAM
17121A1255	M SATHISH KUMAR

SREE VIDYANIKETHAN ENGINEERING COLLEGE

SREE SAINATH NAGAR

CHANDRAGIRI(MANDAL)TIRUPATHI

CHITTOOR DIST. AP.

TEL:0877-2236711

CONTENTS

Abstract:

This program is a applet of TicTacToe.java. Play alternates between the user, who plays O, and the computer, which plays X. After the game ends the program displays a dialog box
Announcing the winner and the total number of wins, losses, and draws by the user, and asks if the user wants to play again. If the user clicks YES then the screen is cleared and a new game started. Otherwise the program exits. The computer moves first on alternate games.
At the top of the screen is a slider that allows the user to change the line thickness of the O's, X's and the 3x3 grid.There are also 2 buttons allowing the user to change the colors of the O's and X's. Initially lines are 4 pixels thick, O's are blue and X's are red. The computer's strategy is first to complete 3 X's in a row, or if that is not possible, to block a row of 3 O's, or if that is not possible, to move randomly.

Introduction:

- The game is to be played between two people (in this program between HUMAN and COMPUTER).
- One of the player chooses ‘O’ and the other ‘X’ to mark their respective cells.
- The game starts with one of the players and the game ends when one of the players has one whole row/ column/ diagonal filled with his/her respective character (‘O’ or ‘X’).
- If no one wins, then the game is said to be draw.

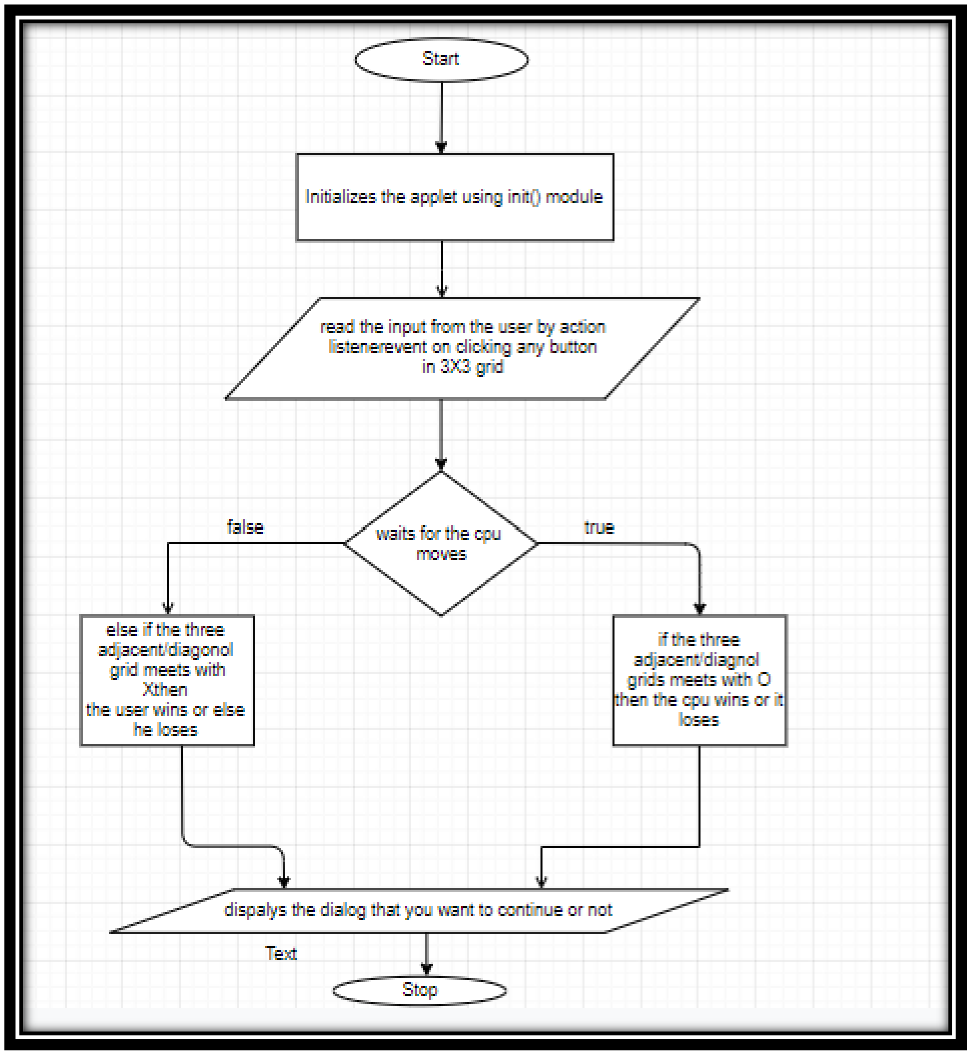
The class code below is divided into three sections.

- The first part is the board functions. These include initializing the board (which can be called to also reset the board state), printing a representation of the board state (so you can see what the game looks like at that point in time) and a check to see if the board is full (for detecting a draw scenario).
- The second part is the win conditions. Here we have a simple function which will tell us if there is a win or not on the board. This function will call other functions to check the rows, the columns and the diagonals for a win condition.
- The last part is the player functions. These control which player mark is active and where a user can place their mark on the board.

Related work:

- We study several research papers on tic-tac-toe game and summarize the findings below. Tic-Tac-Toe is a simple and yet an interesting board game. Researchers have used various approaches to study the Tic-Tac-Toe game.
- Many software implementations of Tic Tac game had been reported and recently it became available for smart phone.
- Traditionally, the game of Tic-tac-toe is a pencil and paper game played by two people who take turn to place their pieces on a 3 times 3 grid with the objective of being the first player to fill a horizontal, vertical, or diagonal row with their pieces.
- What if instead of having one person playing against another, one person plays against a team of nine players, each of whom is responsible for one cell in the 3 times 3 grid?
- The hardware implementation of intelligent Tic-Tac toe. The implementation uses Graphical LCD (GLCD) touch screen and microcontroller. The microcontroller receives the player move from GLCD (displayed as X) and uses intelligent algorithm to analyze the move and choose the best counter move. The microcontroller displays the counter move on the screen as circle (O).
- The algorithm decides the winner when game is finished according to the Tic-Tac playing rule. The system is implemented using cheap available offthe-shelf electronic components and tested and proved to be working fast and efficiently.

Methodology:


Start
Initializes the applet using init() module
read the input from the user by action listenerevent on clicking any button in 3X3 grid
waits for the cpu moves
false
true
else if the three adjacent/diagonol grid meets with Xthen the user wins or else he loses
if the three adjacent/diagnol grids meets with O then the cpu wins or it loses
dispalys the dialog that you want to continue or not
Text
Stop

Implementation:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.Random;
/*<applet code="JTicTacToe.class" width=300 height=300>
</applet>*/
/** tic tac toe applet class */
public class JTicTacToe extends JApplet implements ActionListener
{
   // Variables declaration
        private JLabeltitleLbl;
        private JLabeldirectionLbl;
        private JLabelresultLbl;
        private JButtonbn[][] = new JButton[9][9];
    private JPanelcontentPane;
        private Random rand = new Random();
        private final String X = "X";
        private final String O = "O";
        private boolean turn = true;
 // End of variables declaration
 /** initialize components */
 public void init()
 {
  initializeComponent();
 }
      /** method to initilize components */
 private void initializeComponent()
 {
  titleLbl = new JLabel();
  directionLbl = new JLabel();

                //bn[][]
                for(int x = 0; x < 9; x++)
                    for(int y = 0; y < 9; y++){
                        bn[x][y] = new JButton();
                        bn[x][y].setFont(new Font("Garrmond", Font.BOLD, 24));
                        bn[x][y].addActionListener(this);
                    }
  contentPane = (JPanel)this.getContentPane();
  //
  // titleLbl
  //
  titleLbl.setText("Tic Tac Toe");
                titleLbl.setFont(new Font("Garmmond",Font.BOLD,30));
```

```
//
// directionLbl
//
directionLbl.setText("Choose One Button");
            directionLbl.setFont(new Font("Garmmond", Font.BOLD, 18));
//
// resultLbl
//
                resultLbl = new JLabel();
//
// contentPane
//
contentPane.setLayout(null);
addComponent(contentPane, titleLbl, 59,13,346,32);
addComponent(contentPane, directionLbl, 55,69,354,28);
addComponent(contentPane, bn[0][0], 31,105,70,35);
addComponent(contentPane, bn[0][1], 112,105,70,34);
addComponent(contentPane, bn[0][2], 193,105,70,35);
addComponent(contentPane, bn[1][0], 31,146,70,35);
addComponent(contentPane, bn[1][1], 112,146,70,35);
addComponent(contentPane, bn[1][2], 193,146,70,35);
addComponent(contentPane, bn[2][0], 31,189,70,35);
addComponent(contentPane, bn[2][1], 112,189,70,35);
            addComponent(contentPane, bn[2][2], 193,189, 70,35);
            addComponent(contentPane, resultLbl, 94, 272,251,28);
                 }
                  /** Add Component Without a Layout Manager (Absolute Positioning) */
                  private void addComponent(Container
                 container,Componentc,intx,inty,intwidth,int height)
                  {
                   c.setBounds(x,y,width,height);
                   container.add(c);
                  }
                        /** checks and balances on game plays */
                        public void actionPerformed(ActionEvent e){
                            Object source = e.getSource();
                            if (source == bn[0][0]){
                                if( (turn == true) && (bn[0][0].getText().equals("")) ){
                                    bn[0][0].setText(X);
                                    turn = false;
                                    gameStatus();
                                    cpuMove();
                                }

                            }
                            if (source == bn[0][1]){
```

```
    if( (turn == true) && (bn[0][1].getText().equals("")) ){
        bn[0][1].setText(X);
        turn = false;
        gameStatus();
        cpuMove();
    }

}
if (source == bn[0][2]){
    if( (turn == true) && (bn[0][2].getText().equals("")) ){
        bn[0][2].setText(X);
        turn = false;
        gameStatus();
        cpuMove();
    }

}
if (source == bn[1][0]){
    if( (turn == true) && (bn[1][0].getText().equals("")) ){
        bn[1][0].setText(X);
        turn = false;
        gameStatus();
        cpuMove();
    }

}
if (source == bn[1][1]){
    if( (turn == true) && (bn[1][1].getText().equals("")) ){
        bn[1][1].setText(X);
        turn = false;
        gameStatus();
        cpuMove();
    }

}
if (source == bn[1][2]){
    if( (turn == true) && (bn[1][2].getText().equals("")) ){
        bn[1][2].setText(X);
        turn = false;
        gameStatus();
        cpuMove();
    }

}
if (source == bn[2][0]){
    if( (turn == true) && (bn[2][0].getText().equals("")) ){
```

```
                bn[2][0].setText(X);
                turn = false;
                gameStatus();
                cpuMove();
            }

        }
        if (source == bn[2][1]){
            if( (turn == true) && (bn[2][1].getText().equals("")) ){
                bn[2][1].setText(X);
                turn = false;
                gameStatus();
                cpuMove();
            }

        }
        if (source == bn[2][2]){
            if( (turn == true) && (bn[2][2].getText().equals("")) ){
                bn[2][2].setText(X);
                turn = false;
                gameStatus();
                cpuMove();
            }

        }
    }

    /** check for wins or tie */
    public void gameStatus(){
        //horizontal wins
        if( (bn[0][0].getText() == X) && (bn[0][1].getText() == X) &&
(bn[0][2].getText() ==                  X) ){
            resultLbl.setText("Game over! You Win!");
            disableGame();
            return;
        }
         if( (bn[0][0].getText() == O) && (bn[0][1].getText() == O) &&
(bn[0][2].getText() == O) ){
            resultLbl.setText("Game over! You Lose!");
            disableGame();
            return;
        }
         if( (bn[1][0].getText() == O) && (bn[1][1].getText() == O) &&
(bn[1][2].getText() == O) ){
            resultLbl.setText("Game over! You Lose!");
            disableGame();
```

```
                return;
            }
             if( (bn[1][0].getText() == X) && (bn[1][1].getText() == X) &&
(bn[1][2].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
             if( (bn[2][0].getText() == X) && (bn[2][1].getText() == X) &&
(bn[2][2].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
             if( (bn[2][0].getText() == O) && (bn[2][1].getText() == O) &&
(bn[2][2].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
            //virtical wins
             if( (bn[0][0].getText() == X) && (bn[1][0].getText() == X) &&
(bn[2][0].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
             if( (bn[0][0].getText() == O) && (bn[1][0].getText() == O) &&
(bn[2][0].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
             if( (bn[0][1].getText() == O) && (bn[1][1].getText() == O) &&
(bn[2][1].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
             if( (bn[0][1].getText() == X) && (bn[1][1].getText() == X) &&
(bn[2][1].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
             if( (bn[0][2].getText() == O) && (bn[1][2].getText() == O)
&&(bn[2][2].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
             if( (bn[0][2].getText() == X) && (bn[1][2].getText() == X)
&&(bn[2][2].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
            //diangle wins
             if( (bn[0][0].getText() == O) && (bn[1][1].getText() == O) &&
(bn[2][2].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
             if( (bn[0][0].getText() == X) && (bn[1][1].getText() == X) &&
(bn[2][2].getText()
== X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
             if( (bn[0][2].getText() == O) && (bn[1][1].getText() == O) &&
(bn[2][0].getText() == O) ){
                resultLbl.setText("Game over! You Lose!");
                disableGame();
                return;
            }
             if( (bn[0][2].getText() == X) && (bn[1][1].getText() == X) &&
(bn[2][0].getText() == X) ){
                resultLbl.setText("Game over! You Win!");
                disableGame();
                return;
            }
            //check if a block is still open if there are no open blocks the game
is disabled and tie is declared
            for(int x = 0; x < 3; x++)
                for(int y = 0; y < 3; y++)
                    if(bn[x][y].getText().equals(""))
                        return;
            resultLbl.setText("Game over! Tie");
            disableGame();
        }
```

```
            /** disable gameboard */
            public void disableGame(){
                 for(int x = 0; x < 3; x++)
                    for(int y = 0; y < 3; y++)
                        bn[x][y].setEnabled(false);
                 //turn is set true to prevent infinite loop at the end of the game if
      tied
                 turn = true;
                 reset();
            }

            /** cpu random move */
            public void cpuMove(){
                int x = 0;
                int y = 0;
                do{
                    x = rand.nextInt(3);
                    y = rand.nextInt(3);
                    if(bn[x][y].getText().equals("")){
                        bn[x][y].setText(O);
                        turn = true;
                        gameStatus();
                    }
                }while(turn == false);

            }

            /** reset game; If you win computer goes first */
            public void reset(){
                int answer = JOptionPane.showConfirmDialog(this, "Do you want
      to play again?", "reset", JOptionPane.YES_NO_OPTION);
                if(answer == JOptionPane.YES_OPTION){
                    for(int x = 0; x < 3; x++)
                        for(int y = 0; y < 3; y++){
                            bn[x][y].setEnabled(true);
                            bn[x][y].setText("");
                        }
                    turn = true;
                    resultLbl.setText("");
                }
            }
}
```

Results:

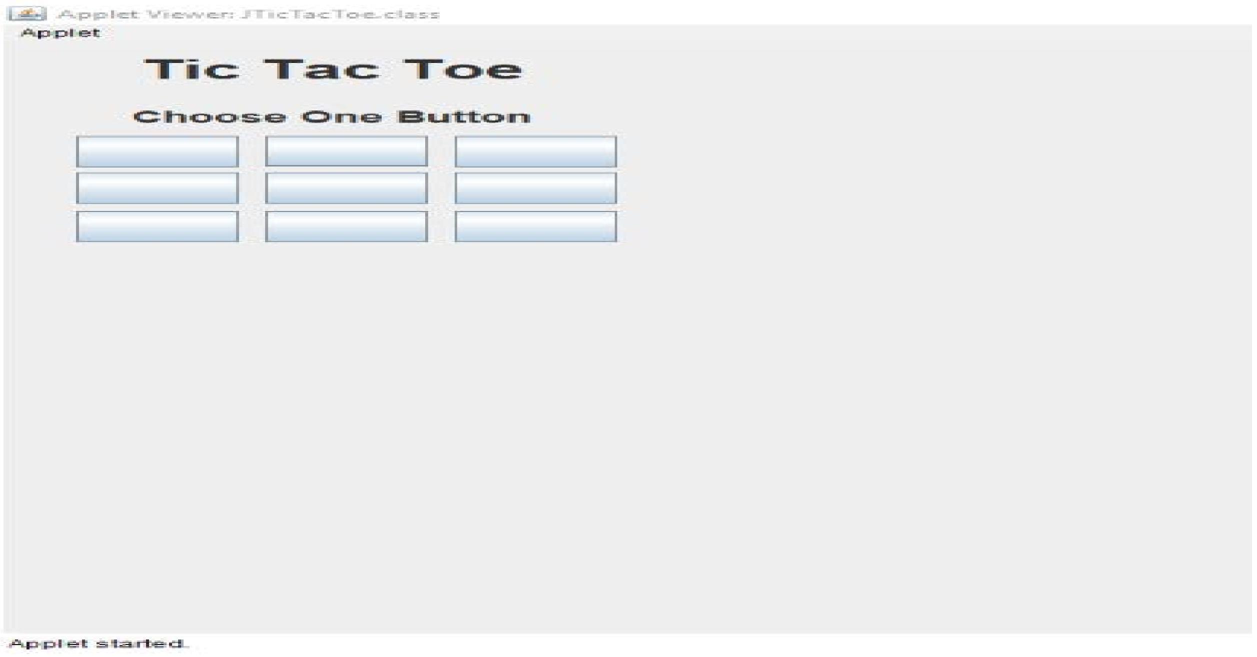
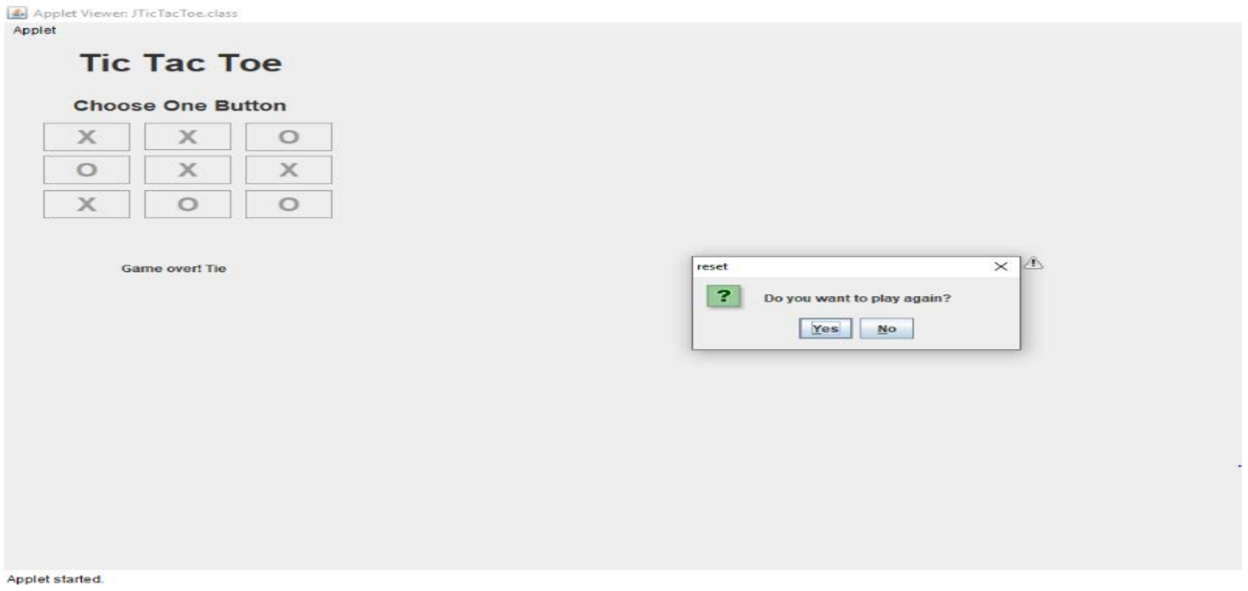


Fig-1:Initialization of applet window



Game over! Tie

Do you want to play again?

Fig-2:screenshot of tie of match between the cpu and player

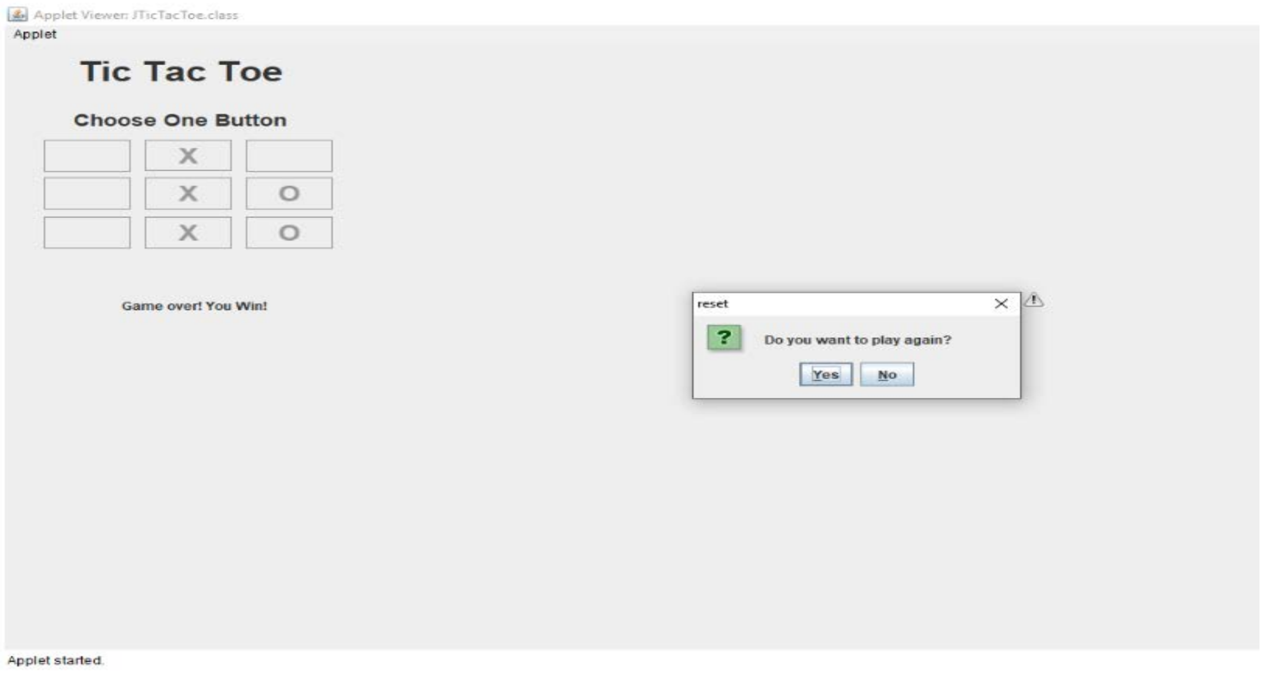


Fig-3:screenshot of winning of match by the player

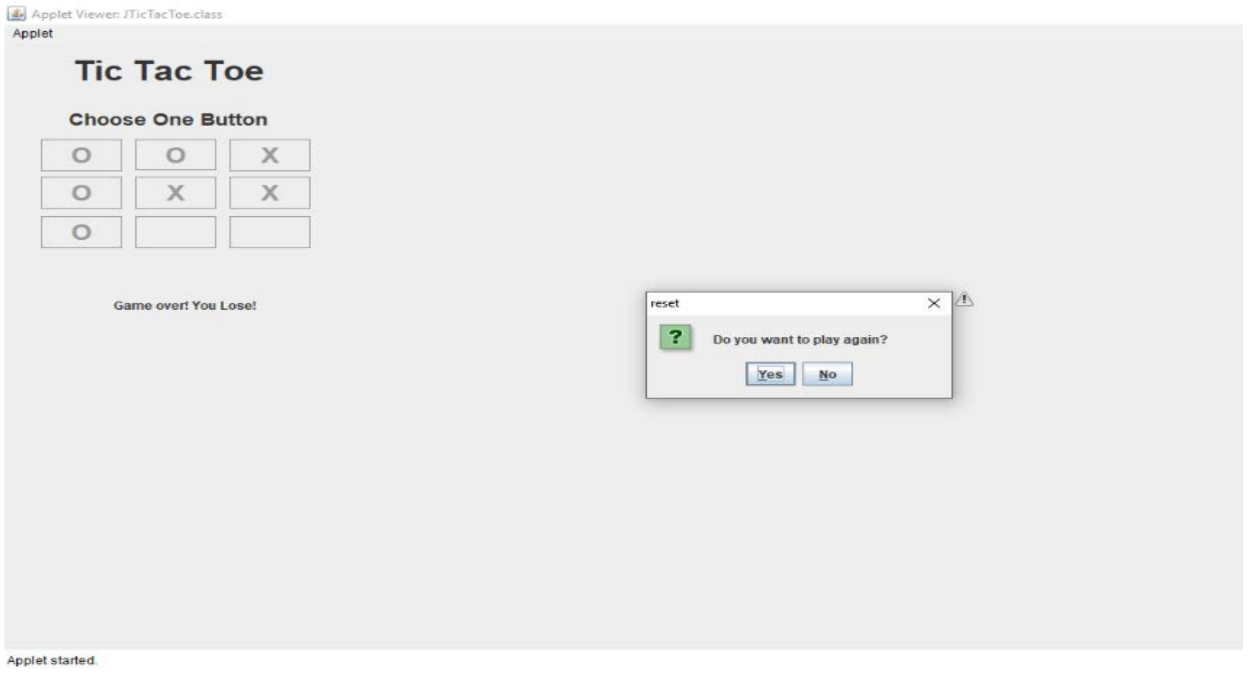


Fig-4:screenshot of losing of match by the player

Conclusion:

From this we can conclude that initially the game was pen and paper based.But now by using the concepts of applets we are able to bring the GUI look to our project.this can be achievedby using the extended classes from actionlistener which in turn handles the events of selecting the buttons by generating on click events .so the project is more user friendly and any body can easily play the game with simple logics.