CHAPTER 1

INTRODUCTION

Games provide a real source of enjoyment in daily life. Games also are helpful in improving the physical and mental health of human. Apart from daily life physical games, people also play computer games. These games are different than those of physical games in a sense that they do not involve much physical activity rather mental and emotional activities. Getting games to react back to the user of a game has always been long hard question for game programmers. Because, let's just face it, a good game that doesn't challenge the user's ability to play the game doesn't keep the user around very long. This idea can be applied to any form of game that is out there. Board games are never fun when the opponent that he or she is playing doesn't learn or catches on. With today's computers always advancing, programmers are always looking for new ways to make a video game more interesting and challenging for the user. Tic-Tac-Toe game can be played by two players where the square block (3 x 3) can be filled with a cross (X) or a circle (O). The game will toggle between the players by giving the chance for each player to mark their move. When one of the players make a combination of 3 same markers in a horizontal, vertical or diagonal line the program will display which player has won, whether X or O. The Tic-Tac-Toe game is most familiar among all the age groups. The friendliness of Tic-tac-toe games makes them ideal as a pedagogical tool for teaching the concepts of good sportsmanship. The game is a very good brain exercise. It involves looking ahead and trying to figure out what the person playing against you might do next.

CHAPTER 2
PROBLEM STATEMENT

2.1 EXISTING SYSTEM

TIC TAC TOE (NOUGHTS and CROSSES, Xs or Os) is a pencil and paper game for two players O and X, who take turns marking the spaces in a 3*3 grid and it is time consuming as it requires drawing the grids each and every time.

2.2 PROPOSED GAMING SYSTEM

- The tic tac toe game is a game of two players, called ‘X’ or ‘O’, who take turns marking the spaces in a 3*3 grid.
- The tic tac toe is great way to pass your free time.
- This game can implement both the opponent and computer based gaming.
- Can restart the game at same interface without compiling again.

2.3 IMPLEMENTATION

Packages Used

java.awt.*;

- The graphics programming in Java is supported by the AWT package. The AWT stands for Abstract Window Toolkit.
- The AWT contains a large number of classes which help to include various graphical components in the Java program. These graphics include text boxes,buttons, frames and so on.

javax.swing.*;

- Swing is another approach to graphical programming in Java. Swing creates highly interactive GUI applications. It is the most flexible and robust approach.
- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

java.awt.Font;

- This class represents fonts. The capabilities of this class have been extended over the java.awt.Font class in JDK and earlier releases to provide developers the ability to utilize more sophisticated typographic features.
- An instance of the Font class represents a specific font to the system. Within AWT, a font is specified by its name, style, and point size. Each platform that supports Java provides a basic set of fonts.

java.awt.event;

Provides interfaces and classes for dealing with different types of events fired by AWT components. See the java.awt.AWTEvent class for details on the AWT event model. Events are fired by event sources. An event listener registers with an event source to receive notifications about the events of a particular type. This package defines events and event listeners, as well as event listener adapters, which are convenience classes to make the process of writing event listeners.

2.4 COMPONENTS

- Setting button for 9 boxes to print symbol of two players on alternative move this button will take pictures as result.
- We have created an interface where it shows radio buttons to select whether the player want to play with computer or friend.
- Used Joptionpanel to print message after the match end as user won or computer or draw.
- Also use button to represent restart for playing again and again without running commmand prompt again.
- Also created ok button in Joptionpanel message.

CHAPTER 3
FLOWCHART

CHAPTER 4
ALGORITHM

MINIMAX():

- Mini-max algorithm is a recursive or backtracking algorithm which is used in decision-making and game theory. It provides an optimal move for the player assuming that opponent is also playing optimally.
- Mini-Max algorithm uses recursion to search through the game-tree.
- Min-Max algorithm is mostly used for game playing in AI. Such as Chess, Checkers, tic-tac-toe, go, and various tow-players game. This Algorithm computes the minimax decision for the current state.
- In this algorithm two players play the game, one is called MAX and other is called MIN.
- Both the players fight it as the opponent player gets the minimum benefit while they get the maximum benefit.
- Both Players of the game are opponent of each other, where MAX will select the maximized value and MIN will select the minimized value.
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

CHAPTER 5

METHODOLOGY

- **public void showbutton():**

This is to show button to show x and o.

- **public void check():**

This is used to find whether the player won or lose.

- **public void complogic(int num):**

This method is used for setting computer logic and operation to display its turn automatically.

- **public void itemstatechanged():**

Invoked when an item has been selected or deselected by the user. the code written for this method performs the operations that need to occur when an item is selected.

- **public void actionperformed():**

Called just after the user performs an action.

CHAPTER 6
PROCEDURE

- **STEP 1**: Start the program
- **STEP 2**: Setting an interface for asking user to select his opponent as friend or computer by using radio button.
- **STEP 3**: Creating a method to make 3*3 box for playing tic tac toe.
- **STEP 4**: Then setting the box to represent clickable to show the symbol x and o on alternative clicks.
- **STEP 5**: A result message to print whether the player x or o has won otherwise the match is draw using joptionpanel .
- **STEP 6**: Creating a min-max or check function to find whether row/diagonal/column is equal for any one symbol. if yes goto step 7 otherwise step 8.
- **STEP 7**: If yes then it will check which symbol is satisfying the condition and print that person as the winner.
- **STEP 8**: Otherwise print draw.
- **STEP 9**: Then create a button to restart the game.
- **STEP 10**: Stop

CHAPTER 7
SOURCE CODE

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TTT1 extends JFrame implements ItemListener, ActionListener {
   int i, j, ii, jj, x, y, yesnull;
   int a[][] = {{10, 1, 2, 3, 11}, {10, 1, 4, 7, 11}, {10, 1, 5, 9, 11},
         {10, 2, 5, 8, 11}, {10, 3, 5, 7, 11},
         {10, 3, 6, 9, 11}, {10, 4, 5, 6, 11}, {10, 7, 8, 9, 11}};
   int a1[][] = {{10, 1, 2, 3, 11}, {10, 1, 4, 7, 11}, {10, 1, 5, 9, 11},
         {10, 2, 5, 8, 11}, {10, 3, 5, 7, 11},

         {10, 3, 6, 9, 11}, {10, 4, 5, 6, 11}, {10, 7, 8, 9, 11}};

boolean state, type, set;

   Icon ic1, ic2, icon, ic11, ic22;
   Checkbox c1, c2;
JLabel l1, l2;
JButtonb[] = new JButton[9];
JButton reset;

   public void showButton() {
```

```
        x = 10;
        y = 10;
        j = 0;
        for (i = 0; i<= 8; i++, x += 100, j++) {
            b[i] = new JButton();
            if (j == 3) {
                j = 0;
                y += 100;
                x = 10;
            }
            b[i].setBounds(x, y, 100, 100);
            add(b[i]);
            b[i].addActionListener(this);
        }//eof for

        reset = new JButton("RESET");
reset.setBounds(100, 350, 100, 50);
        add(reset);
reset.addActionListener(this);

    }//eofshowButton

    /*********************************************************/
    public void check(int num1) {
        for (ii = 0; ii <= 7; ii++) {
            for (jj = 1; jj<= 3; jj++) {
```

```
            if (a[ii][jj] == num1) {
                a[ii][4] = 11;
            }


        }//eof for jj

    }//eof for ii

}//eof check

public void complogic(int num) {

    for (i = 0; i<= 7; i++) {
        for (j = 1; j <= 3; j++) {
            if (a[i][j] == num) {
                a[i][0] = 11;
                a[i][4] = 10;
            }
        }
    }
    for (i = 0; i<= 7; i++) {                    // for 1
        set = true;
        if (a[i][4] == 10) {                     //if 1
            int count = 0;
            for (j = 1; j <= 3; j++) {           //for 2
                if (b[(a[i][j] - 1)].getIcon() != null) {        //if 2
```

```
                    count++;
}               //eof if 2
                  else {
yesnull = a[i][j];
                  }
}                //eof for 2
               if (count == 2) {       //if 2
b[yesnull - 1].setIcon(ic2);
this.check(yesnull);
                  set = false;
                  break;
}             //eof if 2
}               //eof if 1
            else if (a[i][0] == 10) {
               for (j = 1; j <= 3; j++) {         //for2
                  if (b[(a[i][j] - 1)].getIcon() == null) {
                     b[(a[i][j] - 1)].setIcon(ic2);
this.check(a[i][j]);
                     set = false;
                     break;
}                                          //eof if1
}                                            //eof for 2
               if (set == false)
                  break;
            }//eof elseif

            if (set == false)
```

```
            break;
        }//eof for 1


    }//eofcomplogic


    /*********************************************************/

    TTT1() {
super("NOUGHTS AND CROSSES");

CheckboxGroupcbg = new CheckboxGroup();
        c1 = new Checkbox("VS COMPUTER", cbg, false);
        c2 = new Checkbox("VS OPPONENT", cbg, false);
        c1.setBounds(120, 80, 100, 40);
        c2.setBounds(120, 150, 100, 40);
        add(c1);
        add(c2);
        c1.addItemListener(this);
        c2.addItemListener(this);


        state = true;
        type = true;
        set = true;
        ic1 = new ImageIcon("icc1.jpg");
```

```
        ic2 = new ImageIcon("icc2.jpg");
        ic11 = new ImageIcon("icc1.jpg");
        ic22 = new ImageIcon("icc2.jpg");

setLayout(null);
setSize(330, 450);
setVisible(true);
setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    }//eof constructor

    /*************************************************************/
    public void itemStateChanged(ItemEvent e) {
        if (c1.getState()) {
            type = false;
        } else if (c2.getState()) {
            type = true;
        }
        remove(c1);
        remove(c2);
repaint(0, 0, 330, 450);
showButton();
    }//eofitemstate

    /*************************************************************/

    public void actionPerformed(ActionEvent e) {
/*******************************/
```

```
        if (type == true) {    //logicfriend
            if (e.getSource() == reset) {
                for (i = 0; i<= 8; i++) {
                    b[i].setIcon(null);
                }//eof for
            } else {
                for (i = 0; i<= 8; i++) {
                    if (e.getSource() == b[i]) {

                        if (b[i].getIcon() == null) {
                            if (state == true) {
                                icon = ic2;
                                state = false;
                            } else {
                                icon = ic1;
                                state = true;
                            }
                            b[i].setIcon(icon);
                        }
                    }
                }//eof for
            }//eof else

            // Loop through 8 Possible winning positions
booleanhasWinner = false ;
            for (i = 0; i< 8; i++) {
                Icon icon1 = b[(a[i][1] - 1)].getIcon();
```

```
            Icon icon2 = b[(a[i][2] - 1)].getIcon();
            Icon icon3 = b[(a[i][3] - 1)].getIcon();
            if ((icon1 == icon2) && (icon2 == icon3) && (icon1 != null)) {
               if (icon1 == ic1) {
                  b[(a[i][1] - 1)].setIcon(ic11);
                  b[(a[i][2] - 1)].setIcon(ic11);
                  b[(a[i][3] - 1)].setIcon(ic11);
JOptionPane.showMessageDialog(TTT1.this, "!!!PLAYER 1 WON!!! CLICK
RESET");
hasWinner = true ;
                  break;
               } else if (icon1 == ic2) {
                  b[(a[i][1] - 1)].setIcon(ic22);
                  b[(a[i][2] - 1)].setIcon(ic22);
                  b[(a[i][3] - 1)].setIcon(ic22);
JOptionPane.showMessageDialog(TTT1.this, "!!!PLAYER 2 WON!!! CLICK
RESET");
hasWinner = true ;
                  break;
               }
            }
         }

         if(!hasWinner) {
booleanhasEmptyButtons = false ;
for(JButton button : b) {
               if(button.getIcon() == null) {
hasEmptyButtons = true;
```

```
                    break;
                }
            }

            if(!hasEmptyButtons) {
JOptionPane.showMessageDialog(this, "Its a draw");
            }
        }



    }//eoflogicfriend
    else if (type == false) {                          //  complogic
        if (e.getSource() == reset) {
            for (i = 0; i<= 8; i++) {
                b[i].setIcon(null);
            }//eof for
            for (i = 0; i<= 7; i++)
                for (j = 0; j <= 4; j++)
                    a[i][j] = a1[i][j];   //again initialsing array
        } else {  //complogic
            for (i = 0; i<= 8; i++) {
                if (e.getSource() == b[i]) {
                    if (b[i].getIcon() == null) {
                        b[i].setIcon(ic1);
                        if (b[4].getIcon() == null) {
                            b[4].setIcon(ic2);
this.check(5);
```

```
                    } else {
this.complogic(i);
                    }
                  }
               }
            }//eof for
         }

booleanhasWinner = false ;
         for (i = 0; i<= 7; i++) {

            Icon icon1 = b[(a[i][1] - 1)].getIcon();
            Icon icon2 = b[(a[i][2] - 1)].getIcon();
            Icon icon3 = b[(a[i][3] - 1)].getIcon();
            if ((icon1 == icon2) && (icon2 == icon3) && (icon1 != null)) {
               if (icon1 == ic1) {
                  b[(a[i][1] - 1)].setIcon(ic11);
                  b[(a[i][2] - 1)].setIcon(ic11);
                  b[(a[i][3] - 1)].setIcon(ic11);
JOptionPane.showMessageDialog(TTT1.this, "!!!YOU WON!!! CLICK
RESET");
hasWinner = true ;
                  break;
               } else if (icon1 == ic2) {
                  b[(a[i][1] - 1)].setIcon(ic22);
                  b[(a[i][2] - 1)].setIcon(ic22);
                  b[(a[i][3] - 1)].setIcon(ic22);
```

```
JOptionPane.showMessageDialog(TTT1.this, "!!!AWK (COMPUTER)
WON!!! CLICK RESET");
hasWinner = true ;
                    break;
                }
            }
        }

        if(!hasWinner) {
booleanhasEmptyButtons = false ;
for(JButton button : b) {
                if(button.getIcon() == null) {
hasEmptyButtons = true;
                    break;
                }
            }

            if(!hasEmptyButtons) {
JOptionPane.showMessageDialog(this, "Its a draw");
            }
        }
    }//eofcomplogic
  }//eofactionperformed

  public static void main(String[] args) {
    new TTT1();
  }//eof main
}//eof class
```

CHAPTER 8

RESULTS AND DISCUSSION

OUTPUT SCREEN 1:

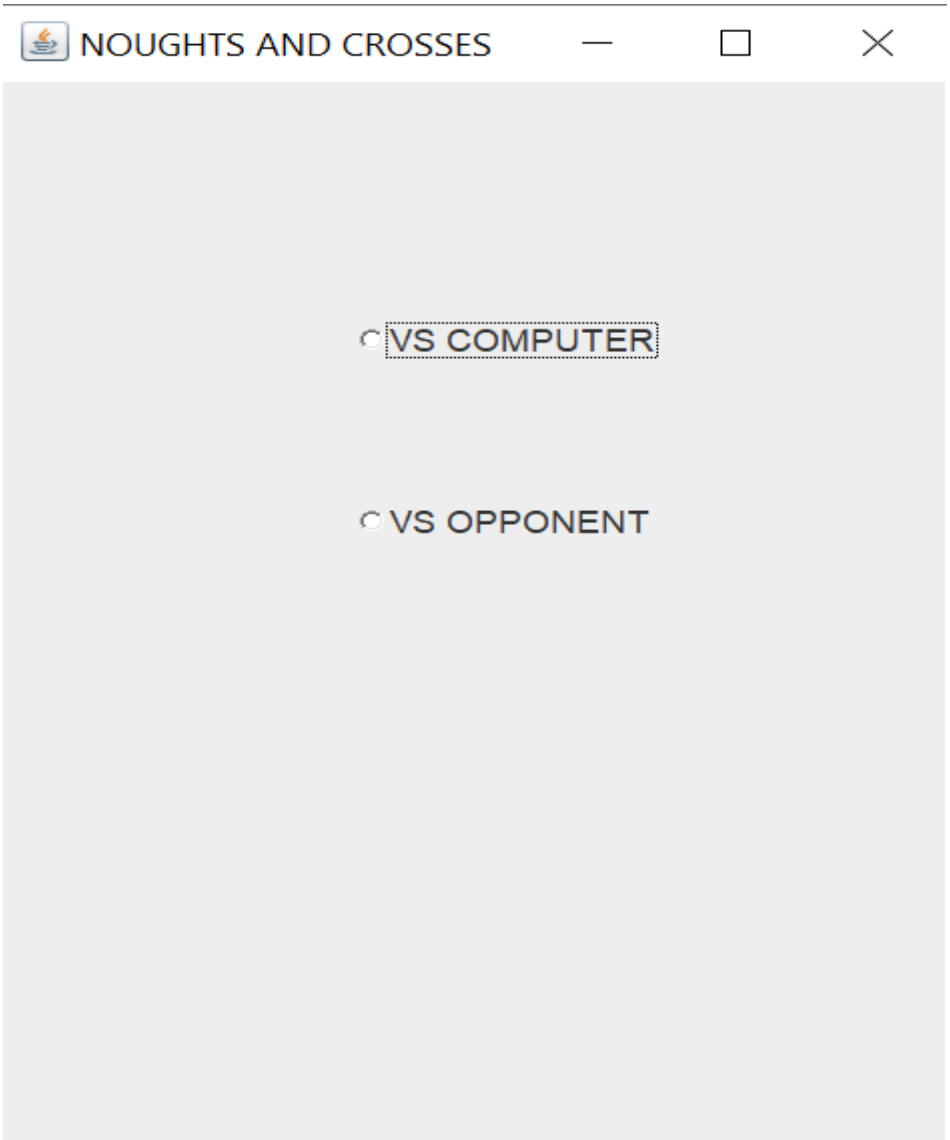

Figure 1 Screen display for choosing the opponent either

It is the first interface screen given to the user to choose whether he/she wants to play with computer or with opponent. Here we have used a radio button to choose a single option.

OUTPUT SCREEN 2:

Vs Computer

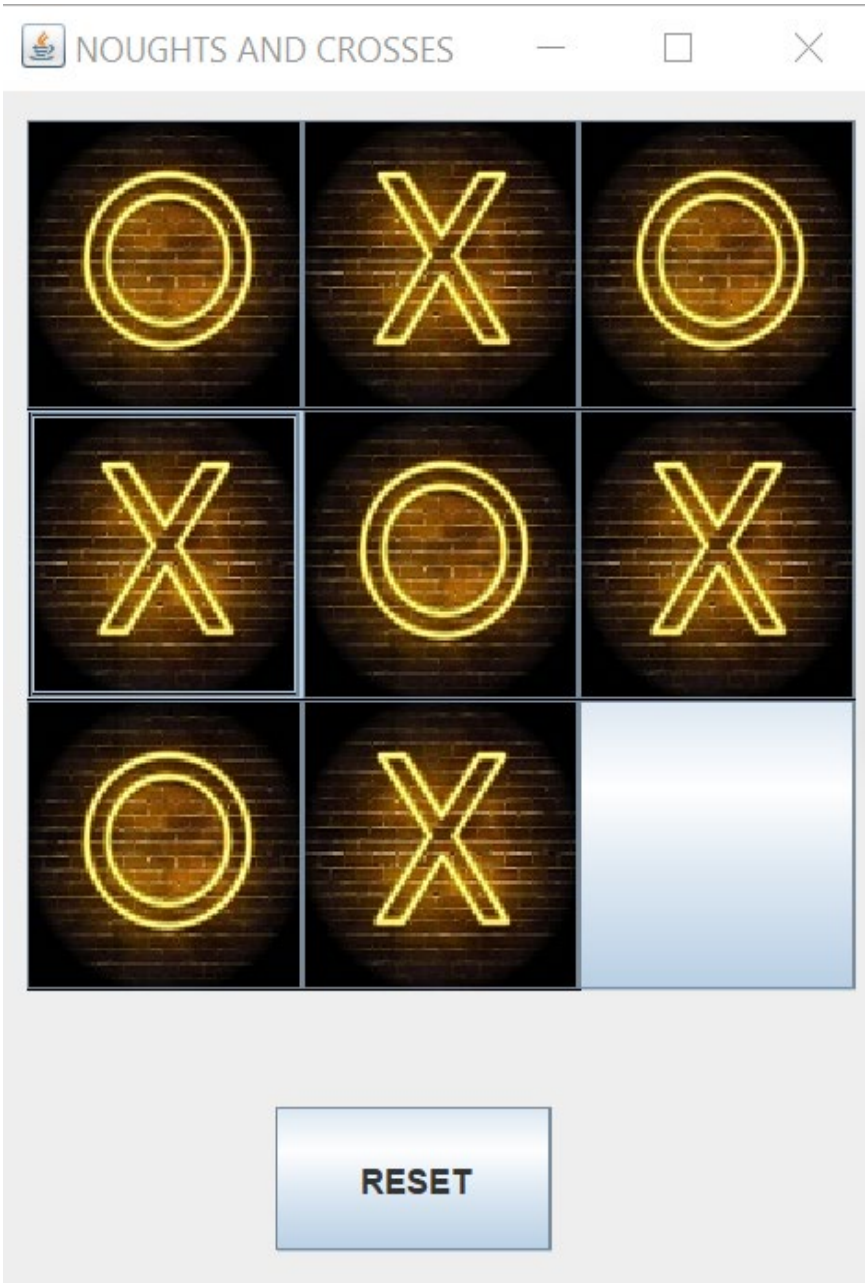

Figure 2 Screen display of game board

It is the second interface screen where each player is given their chance to mark their move

OUTPUT SCREEN 2.1:

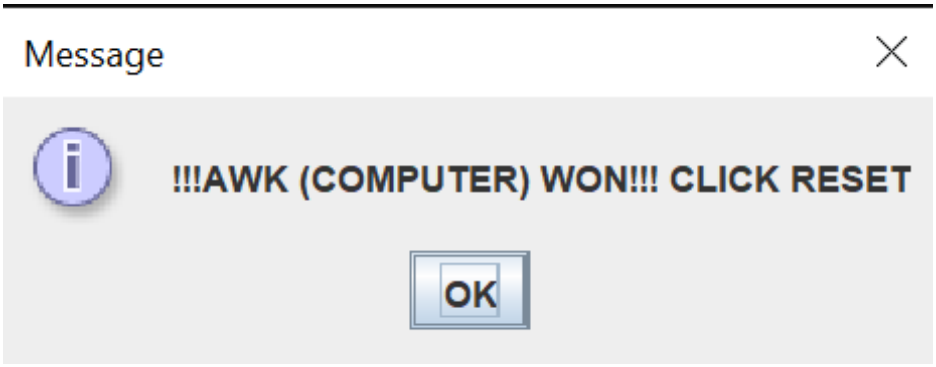

Figure2.1 Screen display when computer wins

Using JOptionPanel we have created an interface to print the message as COMPUTER WON when the game was won by the computer. The OK button is used to exit the message panel.

OUTPUT SCREEN 2.2:

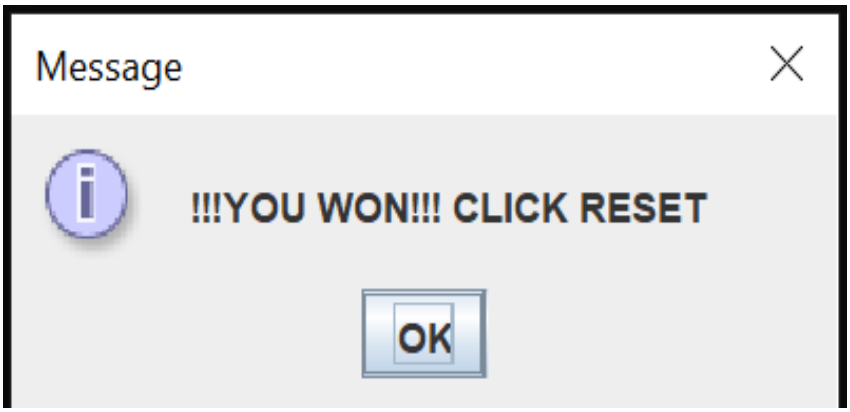

Figure2.2 Screen display when user wins

Using JOptionPanel we have created an interface to print the message as YOU WON when the game was won by the user. The OK button is used to exit the message panel.

OUTPUT SCREEN 3:

Vs Opponent

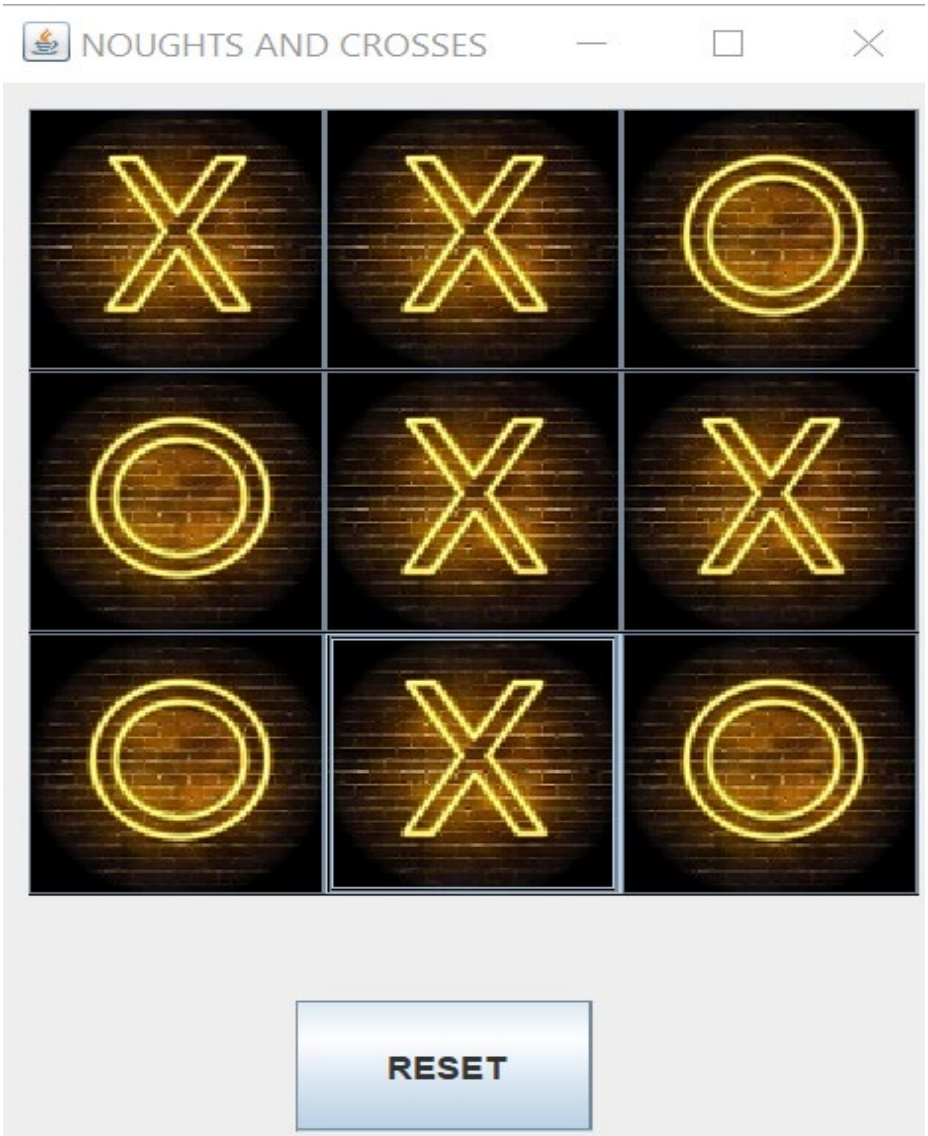

Figure3 Screen display of game board after all boxes are filled

It is the interface screen where each player is given a chance to mark their move when the user chooses to play against another opponent.

OUTPUT SCREEN 3.1:

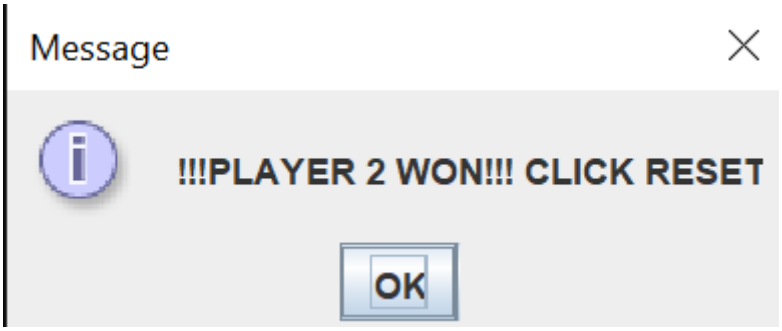

Figure 3.1 Screen display when player 2 wins

Using JOptionPanel we have created an interface to print the message as PLAYER 1 WON when the game was won by player 1. The OK button is used to exit the message panel.

OUTPUT SCREEN 3.2:

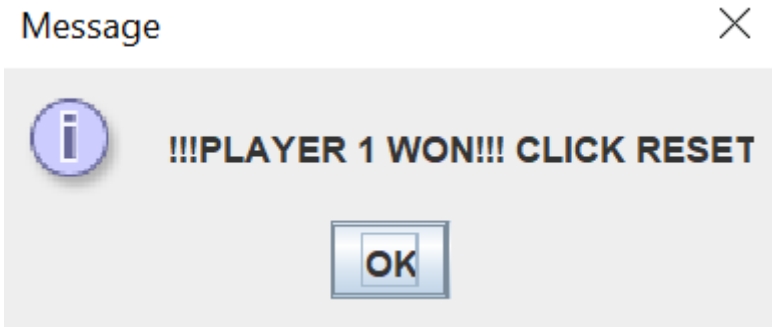

Figure 3.2 Screen display when player 1 wins

Using JOptionPanel we have created an interface to print the message as PLAYER 2 WON when the game was won by player 1. The OK button is used to exit the message panel.

OUTPUT SCREEN 4:

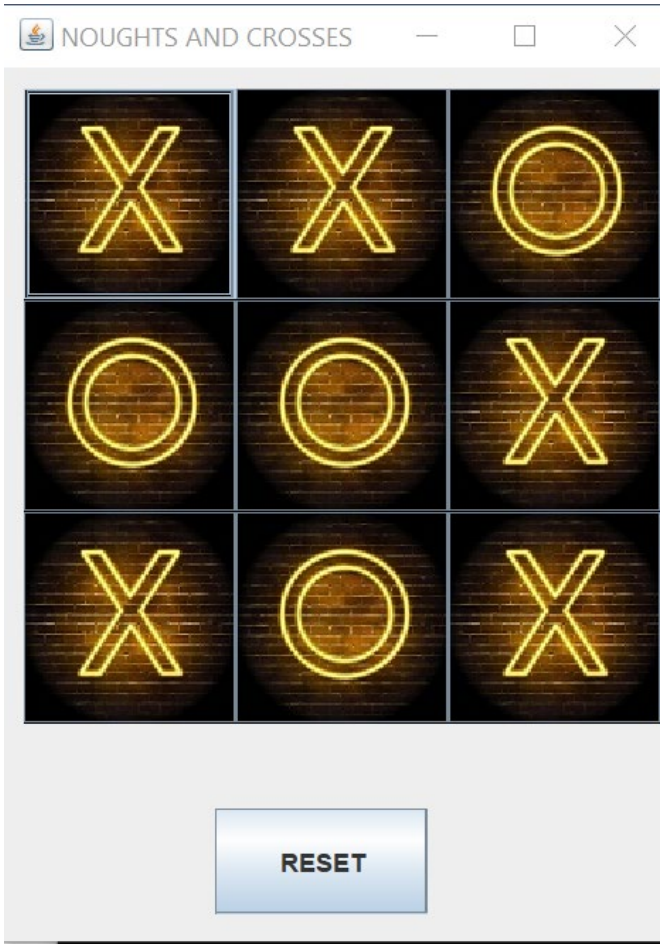

Figure 4 Screen display when the match is draw

OUTPUT SCREEN 4.1:

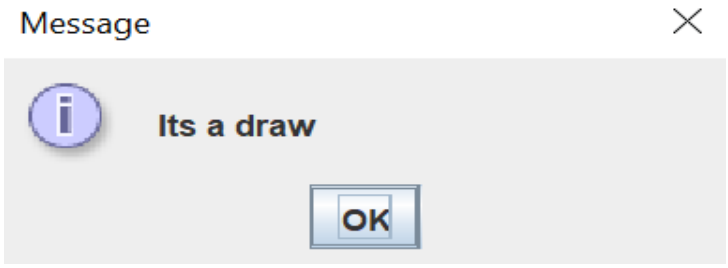

Figure 4.1 Screen display when player 1 wins

Using JOptionPanel we have created an interface to print the message as Its a draw when the all the boxes are filled and there is no winner. The OK button is used to exit the message panel and to reset the game.

CHAPTER 9
CONCLUSION

It helps children develop strategy at an early age. though not a hard strategy it requires some thought as a youngster, such as blocking the other player and keeping them from winning while trying to win yourself. These tic tac toe panels improve hand eye coordination and encourage better social interaction by better collaborative play.The tic tac toe game is most familiar among all age groups. intelligence can be property of any purpose driven maker. this basic idea has been suggested many items. as algorithm of playing tic tac toe has been presented and tested that works in efficient way. overall, the system works without any bugs.

CHAPTER 10

REFERENCE

1. Herbert Schildt(1997) –"Java the complete reference" (comprehensive coverage of the java language), eleventh edition .

2.Ken Arnold James Gosling ,David Holmes (2005) – " The Java Programming Language" ,Fourth Edition

3.Joshua Bloch (2001)-"Effective java" , Third Edition .

4. James J.Gosling,Bill joy,Jr.Steele ,Guy L.,Gilad Bracha,Alex Buckley,Guy L.Steele Jr.,(1996) -" The Java Language Specification" ,Java SE 8 Edition .

5. John Zukowski –"java AWT reference ,"O'Reilly & Associates, Inc. 103A Morris St. Sebastopol, CA,United States.

6. Randy Chapman - "Java Applet, Awt and Util Class Reference "Paperback – Import, 1 December 1996.

7. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/