

**Project Report**

**Tic Tac Toe Game**

**Java Programmin Language**

**Faculty**

Teacher: **Ms. Aman Farooq**

**Group Members**

Student Name: **Muhammad Ubaid**

Student ID: **FA18-BSSE-0050**

Student Name: **ABDIRAHMAN AHMED KHALIF**

Student ID: **FA18-BSSE-0047**

**Table of Contents**

[Introduction 3](#_Toc11964445)

[**Game Discretion** 3](#_Toc11964446)

[Hardware/ Software Requirements 4](#_Toc11964447)

[**Hardware** 4](#_Toc11964448)

[**Software** 4](#_Toc11964449)

[UML DIAGRAM 5](#_Toc11964450)

[Game Snap Shots 6](#_Toc11964451)

[DEVELOPER GUIDE 8](#_Toc11964452)

[**Game Code** 8](#_Toc11964453)

# **Introduction**

We are group of two person Mohammad Ubaid, ID: FA18-BSSE-0050 and Abdirahman Ahmed Khalif, ID: FA18-BSSE-0047. And we made Tic Tac Toe game in Java Programming Language and we used IntelliJ IDEA as editor.

## **Game Discretion**

**Tic**-**tac**-**toe** (American English), noughts and crosses (British English), or Xs and Os is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

The following example game is won by the first player, X:

Game of Tic-tac-toe, won by X

We made the game as an application that when we run it starts the game to play the two players. When every player clicks on any button the button will be disable to clicked again till the game end. When the game end and one of the players has got horizontal, vertical, or diagonal row the program pops up a message box that says the X player or O player wins the game, and also asks a question that if the players need to play again or not. And it gives three options to select, and the option they are YES, NOT and CENCAL. If the players select YES, the game will start again, but if they choose NO or CENCAL the application or game will end and it will be closed.

And another thing that we add the program is a filling system that when the game ends and the application closed, it will report the activities of the players like how many games they play and how many points that every player have got in every game.

# **Hardware/ Software Requirements**

## **Hardware**

* A minimum computer system that will help you access all the tools in the courses is a Pentium 166 or better
* 64 Megabytes of RAM or better
* Windows 2000 (or higher if possible)
* Java Virtual Machine

## **Software**

* Notepad/Java editor/IntelliJ IDEA
* Jdk-1.8.1\_181

# **UML DIAGRAM**

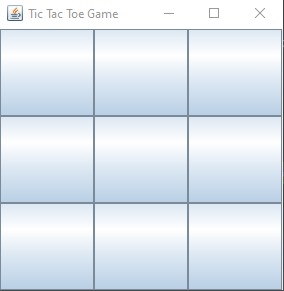
|  |
| --- |
| **TicTacToe** |
| button : JBotton  alternate : int  winPoint : int  games : int  numbeOfGames : int  xWin : int  yWin : int |
| + TicTacToe ()  + initializeButton () : void  + resetButton () : void |

|  |
| --- |
| **JPanel** |
|  |

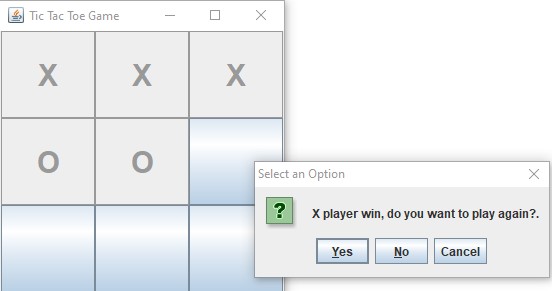
|  |
| --- |
| **<<interface>>**  **ActionListener** |
|  |

|  |
| --- |
| **buttonListener** |
|  |
| + actionPerformed ( e : ActionEvent ) : void  + checkForWin () : void  + checkAdjacent ( a: int , b : int ) : void |

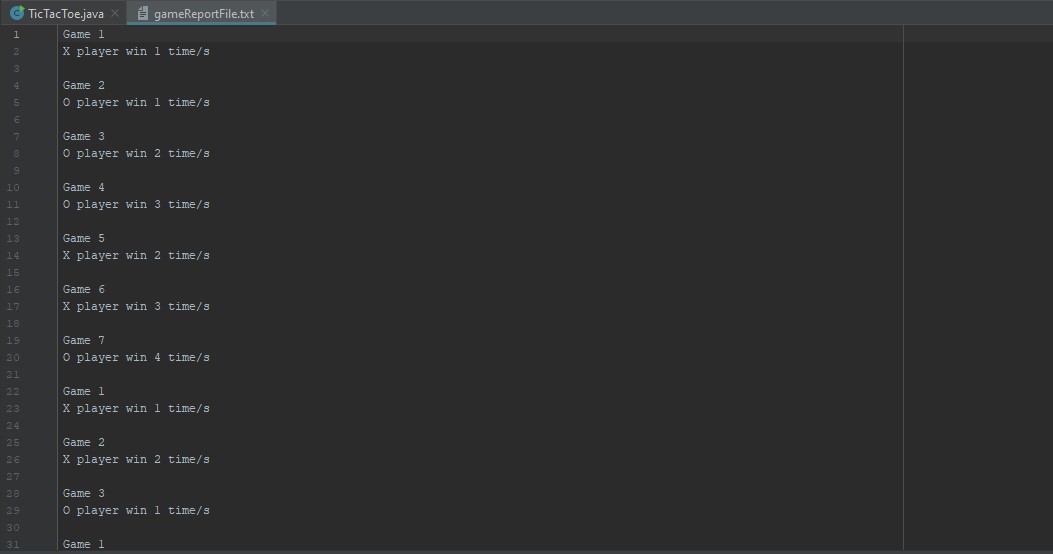
# **Game Snap Shots**



The start window of the application game



When one of the players win the game and it pops up the message box that tells the the winner and asks the if they need to play again or not, and give the option like YES, NOT and CELCAL.



The report file of the activities of the players that tells number games they play and how many points that every player have got in every game.

# **DEVELOPER GUIDE**

## **Game Code**

import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.io.BufferedWriter;  
import java.io.FileWriter;  
import java.io.IOException;  
import javax.swing.\*;  
import static com.sun.deploy.uitoolkit.ToolkitStore.*dispose*;  
  
public class TicTacToe extends JPanel {  
 JButton buttons[] = new JButton[9];  
 int alternate = 0;   
  
 int winPointsc,games,numberOfGames,xwin,ywin;  
  
 public TicTacToe()  
 {  
 setLayout(new GridLayout(3,3));  
 initializeButtons();  
 numberOfGames = 0;  
 games = 0;  
 xwin=0;  
 ywin=0;  
 }  
  
 public void initializeButtons()  
 {  
 for(int i = 0; i <= 8; i++)  
 {  
 buttons[i] = new JButton();  
 buttons[i].setText("");  
 buttons[i].addActionListener(new ButtonListener());  
  
 add(buttons[i]);  
 }  
 }  
 public void resetButtons()  
 {  
 for(int i = 0; i <= 8; i++)  
 {  
 buttons[i].setText("");  
 buttons[i].setEnabled(true);  
 }  
 }  
  
   
 private class ButtonListener implements ActionListener {  
  
 public void actionPerformed(ActionEvent e)  
 {  
  
  
 JButton buttonClicked = (JButton)e.getSource();  
 buttonClicked.setFont(new Font("Century Schoolbook L", 1, 30));  
 if(buttonClicked == e.getSource())  
 {  
 buttonClicked.setEnabled(false);  
  
 }  
  
 if(alternate%2 == 0) {  
 buttonClicked.setText("X");  
 if(checkForWin() == true)  
 {  
  
 FileWriter fileWriter = null;  
 BufferedWriter bufferedWriter = null;  
  
 try {  
 fileWriter = new FileWriter("gameReportFile.txt", true);  
 bufferedWriter = new BufferedWriter(fileWriter);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 try {  
 while (checkForWin() == true) {  
  
 if (numberOfGames >= 0) {  
 games++;  
 bufferedWriter.write("Game " + games);  
 bufferedWriter.newLine();  
 }  
 if (winPoints >= 0) {  
 xwin++;  
 bufferedWriter.write("X player win " + xwin + " time/s");  
 }  
 break;  
 }  
  
 bufferedWriter.newLine();  
 bufferedWriter.newLine();  
 bufferedWriter.close();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 int dialogButton = JOptionPane.*showConfirmDialog*(null,"X player win, do you want to play again?.");  
 if (dialogButton == JOptionPane.*YES\_OPTION*) {  
 resetButtons();  
  
 }  
 else if (dialogButton == JOptionPane.*NO\_OPTION* || dialogButton == JOptionPane.*CANCEL\_OPTION*) {  
 setVisible(false);  
 try {  
 *dispose*();  
 System.*exit*(0);  
 } catch (Exception e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
 }  
 else if(alternate %2 == 1) {  
 buttonClicked.setText("O");  
 if(checkForWin() == true)  
 {  
  
 FileWriter fileWriter = null;  
 BufferedWriter bufferedWriter = null;  
  
 try {  
 fileWriter = new FileWriter("gameReportFile.txt", true);  
 bufferedWriter = new BufferedWriter(fileWriter);  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 try {  
  
  
 while (checkForWin() == true) {  
 if (numberOfGames >= 0) {  
 games++;  
 bufferedWriter.write("Game " + games);  
 bufferedWriter.newLine();  
 }  
 if (winPoints >= 0) {  
 ywin++;  
 bufferedWriter.write("O player win " + ywin + " time/s");  
 }  
 break;  
 }  
  
 bufferedWriter.newLine();  
 bufferedWriter.newLine();  
 bufferedWriter.close();  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
  
 int dialogButton = JOptionPane.*showConfirmDialog*(null,"O player win, do you want to play again?.");  
 if (dialogButton == JOptionPane.*YES\_OPTION*) {  
 resetButtons();  
 }  
 else if (dialogButton == JOptionPane.*NO\_OPTION* || dialogButton == JOptionPane.*CANCEL\_OPTION*) {  
 setVisible(false);  
 try {  
 *dispose*();  
 System.*exit*(0);  
 } catch (Exception e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
 }  
  
 else if (alternate <= 9 && checkForWin() != true) {  
 int dialogButton = JOptionPane.*showConfirmDialog*(null, "No one win, do you want to play again?.");  
 if (dialogButton == JOptionPane.*YES\_OPTION*) {  
 resetButtons();  
 } else if (dialogButton == JOptionPane.*NO\_OPTION* || dialogButton == JOptionPane.*CANCEL\_OPTION*) {  
 setVisible(false);  
 try {  
 *dispose*();  
 System.*exit*(0);  
 } catch (Exception e1) {  
 e1.printStackTrace();  
 }  
 }  
 }  
  
 alternate++;  
  
 }  
  
 public boolean checkForWin() {  
 //horizontal win check  
 if( checkAdjacent(0,1) && checkAdjacent(1,2) ) { //no need to put " == true" because the default check is for true  
 return true;  
 }  
 else if( checkAdjacent(3,4) && checkAdjacent(4,5) ) {  
 return true;  
 }  
 else if ( checkAdjacent(6,7) && checkAdjacent(7,8)) {  
 return true;  
 }  
  
 //vertical win check  
 else if ( checkAdjacent(0,3) && checkAdjacent(3,6)) {  
 return true;  
 }  
 else if ( checkAdjacent(1,4) && checkAdjacent(4,7)) {  
 return true;  
 }  
 else if ( checkAdjacent(2,5) && checkAdjacent(5,8)) {  
 return true;  
 }  
  
 //diagonal win check  
 else if ( checkAdjacent(0,4) && checkAdjacent(4,8)) {  
 return true;  
 }  
 else if ( checkAdjacent(2,4) && checkAdjacent(4,6)) {  
 return true;  
 }  
 else {  
 return false;  
 }  
  
  
 }  
  
 public boolean checkAdjacent(int a, int b)  
 {  
 if ( buttons[a].getText().equals(buttons[b].getText()) && !buttons[a].getText().equals("") )  
 return true;  
 else  
 return false;  
 }  
  
 }  
  
 public static void main(String[] args)  
 {  
 JFrame window = new JFrame("Tic Tac Toe Game");  
 window.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 window.getContentPane().add(new TicTacToe());  
 window.setBounds(550,300,300,300);  
 window.setVisible(true);  
 }  
}