KP14603 OBJECT ORIENTED PROGRAMMING CONCEPT

SEMESTER II

SESSION 2019/2020

INDIVIDUAL PROJECT

(TIC-TAC-TOE GAME)

LECTURER NAME: MADAM SITI HASNAH TANALOL

NAME	**MATRIC NUMBER**
IZYAN UZMA BINTI SHAMSU	BI19110036

CONTENTS

1.0 INTRODUCTION

Most of the research nowadays is focused towards problems that deal with complexity or are influenced by some kind of random events. Interesting about these problems are that if they are deterministic, then a solution is expected to exist, at least a theoretical one. These problems are often inspired by games, such as mathematical and theory games. On the other hand, the point of randomness involved in these problems increases the difficulty of prediction on the possible solution or in some situations outcome.

During this Covid-19, the educational games like Tic Tac Toe is one of the first ways of learning, which allows new insights through child's own experience. In addition, this kind of games teaches good sportsmanship and it helps children apply their logic and develop strategy at an early age. Also, it prepares children for more complex games because they have to think of multiple things at one time and improve their concentration. Hence, Tic Tac Toe game helps develop coordination, fine motor and visual skills. Therefore, in this individual project, I have created and designed a Tic Tac Toe game implemented by a GUI concept that can be used by everyone to play and enhance their creativity. GUI (Graphical User Interface) is an easy to use visual experience to build Java applications. It is mainly made of graphical components like buttons, labels, etc. through which the user can interact with the applications.

Tic Tac Toe game is often played by at least 2 players who take turns marking the spaces in a 3 x 3 grid which is nine squares. It consists of player 'X' and player 'O', out of which the other one is called opponent. The game usually begins with the player 'X' and the player who manage to place three respective marks in any direction such as horizontal, vertical or diagonal row will wins the game. If no one wins, then the game is said to be draw.

2.0 OBJECTIVES

1. To develop the well-known board game Tic Tac Toe for two players by implementing Java GUI.
2. To create and design a game that can naturally develop people's logical thinking especially children and young generation which can help them in subjects such as math and engineering in the future.
3. To create a game that develop an advanced interpersonal understanding which result to better negotiation of conflicts with other people.

3.0 JAVA CODE

1. Login Game

```
package logingame;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.FileWriter;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JTextField;

import javax.swing.SwingUtilities;


public class LoginGame {

    private static JLabel userLabel;

    private static JTextField userText;

    private static JLabel passwordLabel;

    private static JPasswordField passwordText;

    private static JButton button1, button2, button3;

    private static JLabel accountLabel;

    private static FileWriter fileWriter;


    public static void main(String[] args) {

        JPanel panel = new JPanel();
```

```
JFrame frame = new JFrame("Tic-Tac-Toe Login");
frame.setSize(380, 330);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.add(panel);
panel.setLayout(null);
userLabel = new JLabel("Username");
userLabel.setBounds(30, 20, 80, 25);
panel.add(userLabel);
userText = new JTextField(20);
userText.setBounds(110, 20, 165, 25);
panel.add(userText);
passwordLabel = new JLabel("Password");
passwordLabel.setBounds(30, 50, 80, 25);
panel.add(passwordLabel);
passwordText = new JPasswordField();
passwordText.setBounds(110, 50, 165, 25);
panel.add(passwordText);

button1 = new JButton("Login");
button1.setBounds(30, 100, 80, 25);
button1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        if (ae.getActionCommand() == button1.getActionCommand()) {
            try {
                fileWriter = new FileWriter("AdminUser.txt");
                fileWriter.write(userLabel.getText() + " : " + userText.getText());
                fileWriter.write(passwordLabel.getText() + " : " + passwordText.getText());
                fileWriter.close();
            } catch (Exception e) {
            }
        }
        TicTacToeGame second = new TicTacToeGame();
        second.setVisible(true);
        frame.dispose();
    }
});
panel.add(button1);

accountLabel = new JLabel("Don't have a Tic-Tac-Toe account?");
accountLabel.setBounds(30, 140, 200, 25);
panel.add(accountLabel);

button2 = new JButton("Create New Account");
button2.setBounds(30, 160, 180, 25);
button2.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        NewGameAccount third = new NewGameAccount();
        third.setVisible(true);
        frame.dispose();
    }
```

```
        });
        panel.add(button2);


        button3 = new JButton("ADMIN ONLY");
        button3.setBounds(30, 240, 120, 25);
        button3.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent ae) {
            Admin fourth = new Admin();
            fourth.setVisible(true);
            frame.dispose();
            }
            });


        panel.add(button3);
        frame.setVisible(true);
        SwingUtilities.invokeLater(
                new Runnable() {
            @Override
            public void run() {
                new LoginGame();
            }
        }
        );
    }
}
```

2. Tic Tac Toe Game

```
package logingame;

import java.awt.Color;

import java.awt.Container;

import java.awt.Font;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import static javax.swing.JFrame.EXIT_ON_CLOSE;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.SwingUtilities;


public class TicTacToeGame extends JFrame {

    private Container frame;

    private String Player;

    private JButton[][] board;

    private boolean Winner;

    private JMenuBar menuBar;

    private JMenu menu;

    private JMenuItem quit;

    private JMenuItem newGame;
```

```
public TicTacToeGame() {
    super();
    frame = getContentPane();
    frame.setLayout(new GridLayout(3, 3));
    setTitle("Tic-Tac-Toe");
    setSize(450, 450);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setVisible(true);
    Player = "X";
    board = new JButton[3][3];
    Winner = false;
    Board();
    MenuBar();
}
private void MenuBar() { //Initialize MenuBar
    menuBar = new JMenuBar();
    menu = new JMenu("File");

    newGame = new JMenuItem("New Game");
    newGame.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent ae) { //Handle button events
            resetBoard();
        }
    });
    quit = new JMenuItem("Quit"); //To create the file menu item which is "Quit".
    quit.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            System.exit(0);
        }
    });
    menu.add(newGame);
    menu.add(quit);
    menuBar.add(menu);
    setJMenuBar(menuBar);
}
private void resetBoard() {
    Player = "X";
    Winner = false;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            board[i][j].setText("");
        }
    }
}
private void Board() { //Initialize Board
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            JButton btn = new JButton("");
            btn.setFont(new Font(Font.SANS_SERIF, Font.BOLD, 45));
            btn.setBackground(Color.PINK);

            btn.setForeground(Color.BLACK);

            board[i][j] = btn;

            btn.addActionListener(new ActionListener() {

                @Override

                public void actionPerformed(ActionEvent e) {

                    if (((JButton) e.getSource()).getText().equals("") && Winner == false) {

                        btn.setText(Player);

                        Winner();

                        nextPlayer();

                    }

                }

            });

            frame.add(btn);

        }

    }

}

public boolean isFull() { //To determine if game board is full and no winner then it will be draw game.

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

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

            if (board[i][j].getText() == "") {

                return false;

            }

        }

    }
```

```
 return true; }

   private void nextPlayer() { //Change move method, when player X move, then player O will be the next turn.

       if (Player.equals("X")) {

           Player = "O";

       } else {

           Player = "X";

       }

   }

   private void Winner() { // To determine who is the winner in that game or game draw.

       if (board[0][0].getText().equals(Player) && board[1][0].getText().equals(Player) && board[2][0].getText().equals(Player)) {

           JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

           Winner = true;

       } else if (board[0][1].getText().equals(Player) && board[1][1].getText().equals(Player) && board[2][1].getText().equals(Player)) {

           JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

           Winner = true;

       } else if (board[0][2].getText().equals(Player) && board[1][2].getText().equals(Player) && board[2][2].getText().equals(Player)) {

           JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

           Winner = true;

       } else if (board[0][0].getText().equals(Player) && board[0][1].getText().equals(Player) && board[0][2].getText().equals(Player)) {

           JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

           Winner = true;

       } else if (board[1][0].getText().equals(Player) && board[1][1].getText().equals(Player) && board[1][2].getText().equals(Player)) {
            JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

            Winner = true;

        } else if (board[2][0].getText().equals(Player) && board[2][1].getText().equals(Player)
&& board[2][2].getText().equals(Player)) {

            JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

            Winner = true;

        } else if (board[0][0].getText().equals(Player) && board[1][1].getText().equals(Player)
&& board[2][2].getText().equals(Player)) {

            JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

            Winner = true;

        } else if (board[0][2].getText().equals(Player) && board[1][1].getText().equals(Player)
&& board[2][0].getText().equals(Player)) {

            JOptionPane.showMessageDialog(null, "Player " + Player + " Won!");

            Winner = true;

        } else if (isFull()) {

            JOptionPane.showMessageDialog(null, "Draw Game!");

        }

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

                new TicTacToeGame();

            }

        });

    }

}
```

3. Creating New Game Account

```
package logingame;

import java.io.FileWriter;

public class NewGameAccount extends javax.swing.JFrame {

    public NewGameAccount() {

        initComponents();

    }

    @SuppressWarnings("unchecked")

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

if (evt.getActionCommand() == jButton1.getActionCommand()) {

            try {

                FileWriter fileWriter = new FileWriter("AdminUser.txt");

                fileWriter.write(jLabel2.getText() + " : " + jTextField1.getText());

                fileWriter.write(jLabel3.getText() + " : " + jPasswordField1.getText());

                fileWriter.close();

            } catch (Exception e) {

            }

        }

        String user = jTextField1.getText();

        String password = jPasswordField1.getText();

        if (user.equals(user) && password.equals(password)) {

            TicTacToeGame second = new TicTacToeGame();
        second.setVisible(true);

        this.dispose();

}

}

public static void main(String args[]) {

    java.awt.EventQueue.invokeLater(new Runnable() {

        public void run() {

            new NewGameAccount().setVisible(true);

        }

    });

}

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPanel jPanel1;

private javax.swing.JPasswordField jPasswordField1;

private javax.swing.JTextField jTextField1;
```

4. Admin User

```
package logingame;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

public class Admin extends javax.swing.JFrame {

    public Admin() {

        initComponents();

    }

    @SuppressWarnings("unchecked")

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

File file = new File("AdminUser.txt");

                StringBuffer contents = new StringBuffer();

                BufferedReader reader = null;

                try {

                    reader = new BufferedReader(new FileReader(file));

                    String text = null;

                    while ((text = reader.readLine()) != null) {

                        contents.append(text)

                                .append(System.getProperty(

                                        "line.separator"));

                    }

                } catch (FileNotFoundException e) {
```

```
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    if (reader != null) {
                        reader.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                jTextArea1.append(contents.toString());
            }
}
private javax.swing.JButton jButton1;
private javax.swing.JLabel jLabel1;
private javax.swing.JPanel jPanel1;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextArea jTextArea1;
```

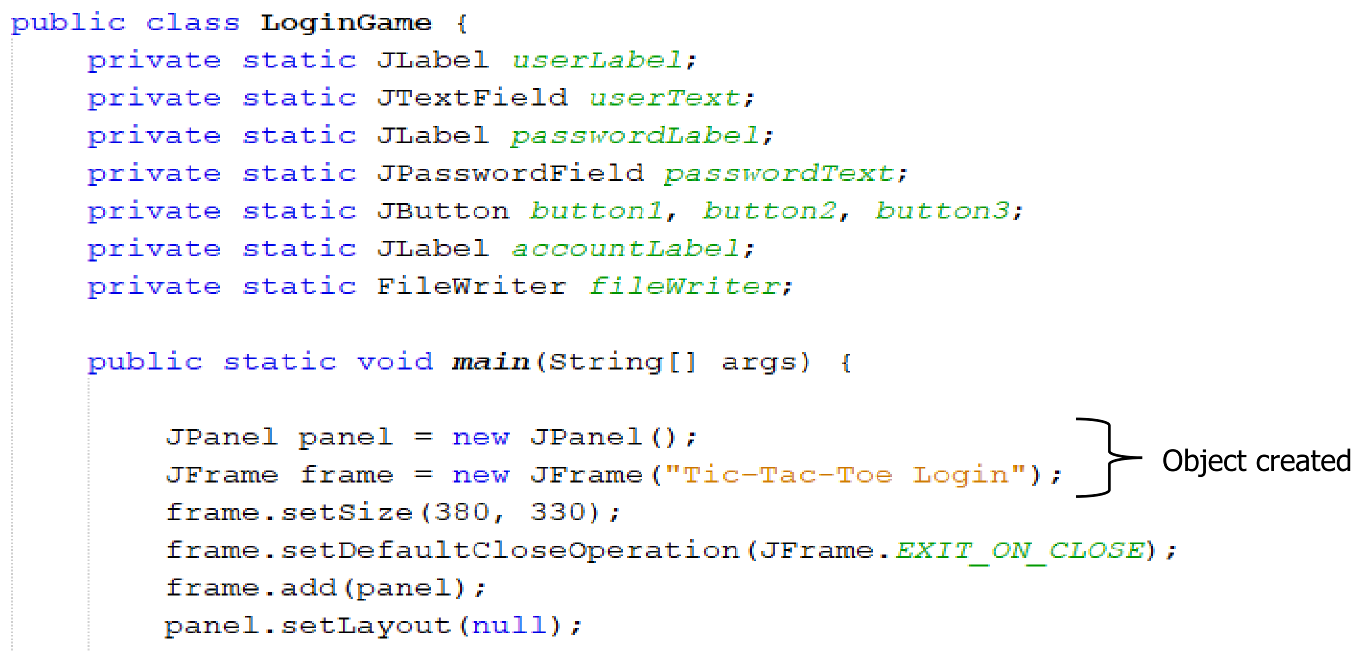
4.0 OBJECT ORIENTED CONCEPTS IMPLEMENTATION

In this project, I have created 4 different jFrame for this Tic Tac Toe game which are Login Game, Tic Tac Toe Game, New Game Account and Admin. Therefore, I have used four different classes which are class LoginGame, class TicTacToeGame, class NewGameAccount and class Admin. Also, there are several concepts of object oriented programming that I have implemented in this project including Object and Classes, Inheritance, Polymorphism, Encapsulation and Inner Class.

1. Object and Classes

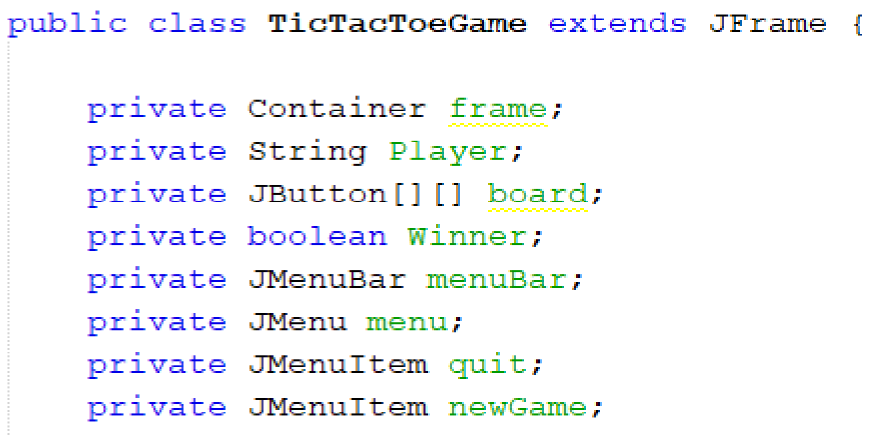
A class is a blueprint from which individual objects are created and object then use to access data during runtime. Object is an instance of class. A class is created by using the keyword class.

i) class LoginGame



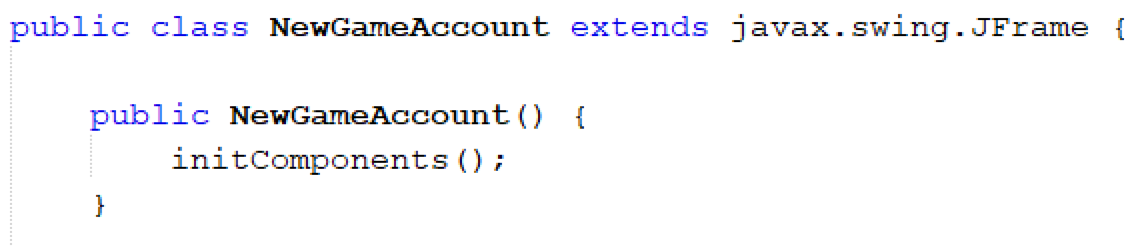
```java
public class LoginGame {
    private static JLabel userLabel;
    private static JTextField userText;
    private static JLabel passwordLabel;
    private static JPasswordField passwordText;
    private static JButton button1, button2, button3;
    private static JLabel accountLabel;
    private static FileWriter fileWriter;

    public static void main(String[] args) {

        JPanel panel = new JPanel();
        JFrame frame = new JFrame("Tic-Tac-Toe Login");
        frame.setSize(380, 330);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(panel);
        panel.setLayout(null);
```

ii) class TicTacToeGame



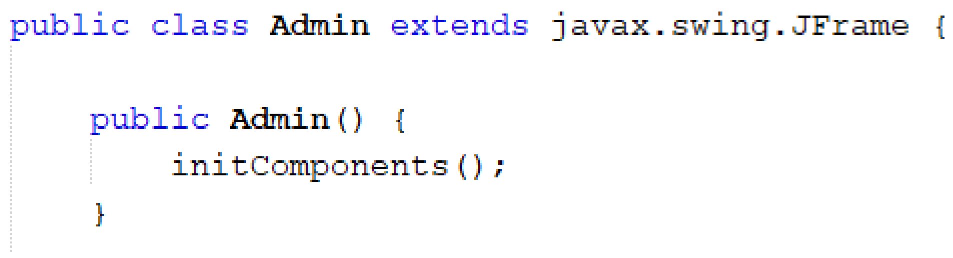
```
public class TicTacToeGame extends JFrame {

    private Container frame;
    private String Player;
    private JButton[][] board;
    private boolean Winner;
    private JMenuBar menuBar;
    private JMenu menu;
    private JMenuItem quit;
    private JMenuItem newGame;
```

iii) class NewGameAccount



```
public class NewGameAccount extends javax.swing.JFrame {

    public NewGameAccount() {
        initComponents();
    }
```

iv) class Admin

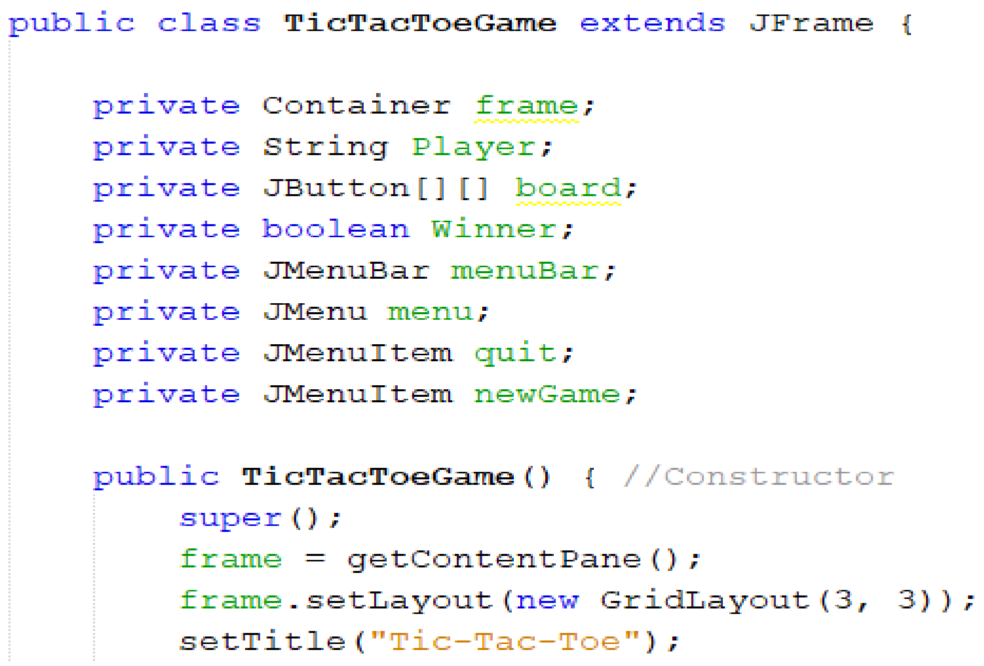


```
public class Admin extends javax.swing.JFrame {

    public Admin() {
        initComponents();
    }
```

2. Inheritance

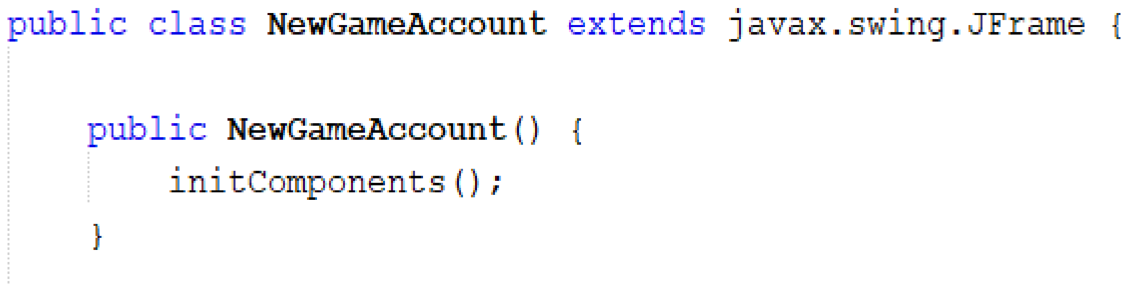
Inheritance is a mechanism in which one class is allow to inherit the features (fields and methods) of another class. The class which inherits the properties of other is known as subclass (child class) and the class whose properties are inherited is known as superclass (parent class). **'extends'** is the keyword used to inherit the properties of a class. The meaning of **'extends'** is to increase the functionality which allowing one class to incorporate another class into its declaration.

i) TicTacToeGame (subclass) inherits JFrame (superclass)



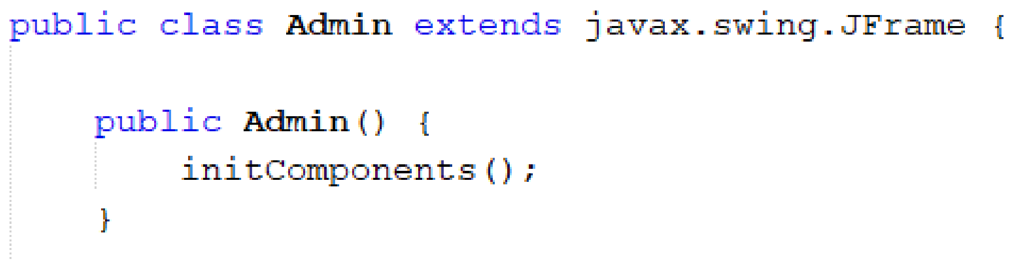
```
public class TicTacToeGame extends JFrame {

    private Container frame;
    private String Player;
    private JButton[][] board;
    private boolean Winner;
    private JMenuBar menuBar;
    private JMenu menu;
    private JMenuItem quit;
    private JMenuItem newGame;

    public TicTacToeGame() { //Constructor
        super();
        frame = getContentPane();
        frame.setLayout(new GridLayout(3, 3));
        setTitle("Tic-Tac-Toe");
```

ii) NewGameAccount (subclass) inherits javax.swing.JFrame (superclass)



```
public class NewGameAccount extends javax.swing.JFrame {

    public NewGameAccount() {
        initComponents();
    }
```

iii) Admin (subclass) inherits javax.swing.JFrame (superclass)

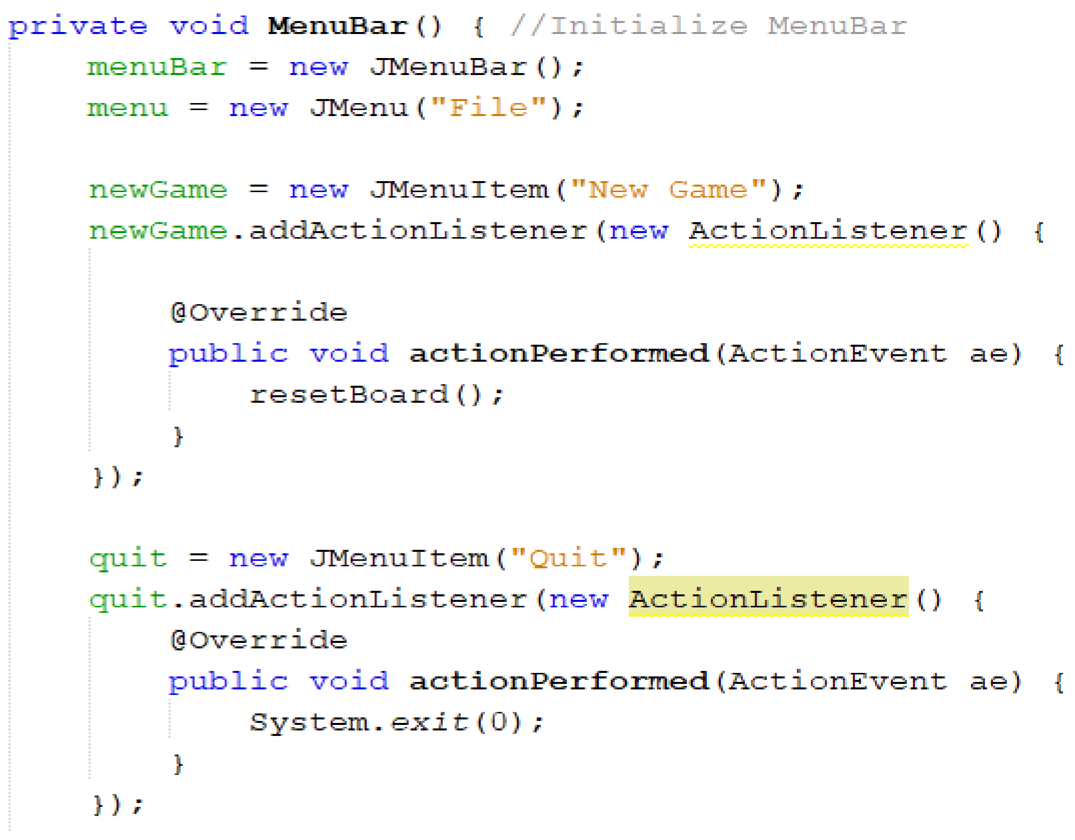


```
public class Admin extends javax.swing.JFrame {

    public Admin() {
        initComponents();
    }
```

3. Polymorphism

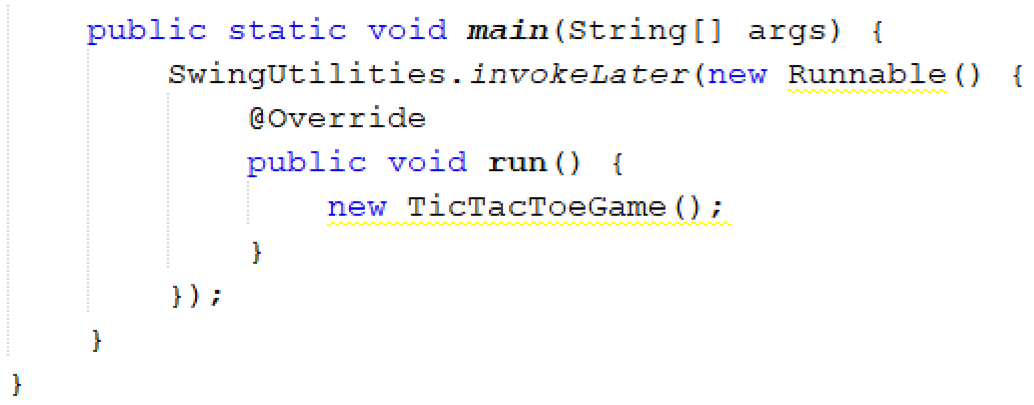
Runtime polymorphism also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by method overriding. Overridden methods are another way that Java implements the "one interface, multiple methods" aspects of polymorphism.

i)



```
private void MenuBar() { //Initialize MenuBar
    menuBar = new JMenuBar();
    menu = new JMenu("File");

    newGame = new JMenuItem("New Game");
    newGame.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent ae) {
            resetBoard();
        }
    });

    quit = new JMenuItem("Quit");
    quit.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            System.exit(0);
        }
    });
```

ii)



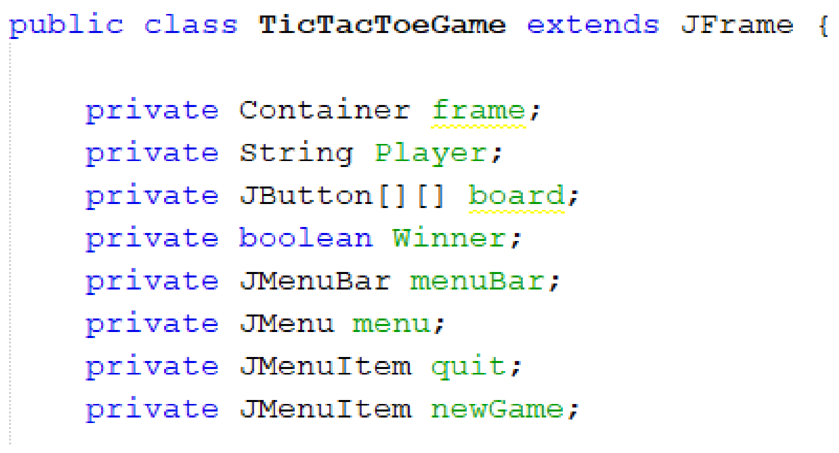
```
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new TicTacToeGame();
            }
        });
    }
}
```

4. Encapsulation

Encapsulation is a programming mechanism that binds together code and the data it manipulates, and that keeps both safe from outside interface and misuse. It is a protective shield that prevents the data from being accessed by the code outside this shield. Encapsulation is achieved by declaring all the variables in the class as private and writing public methods in the class to set and get the values of the variables.

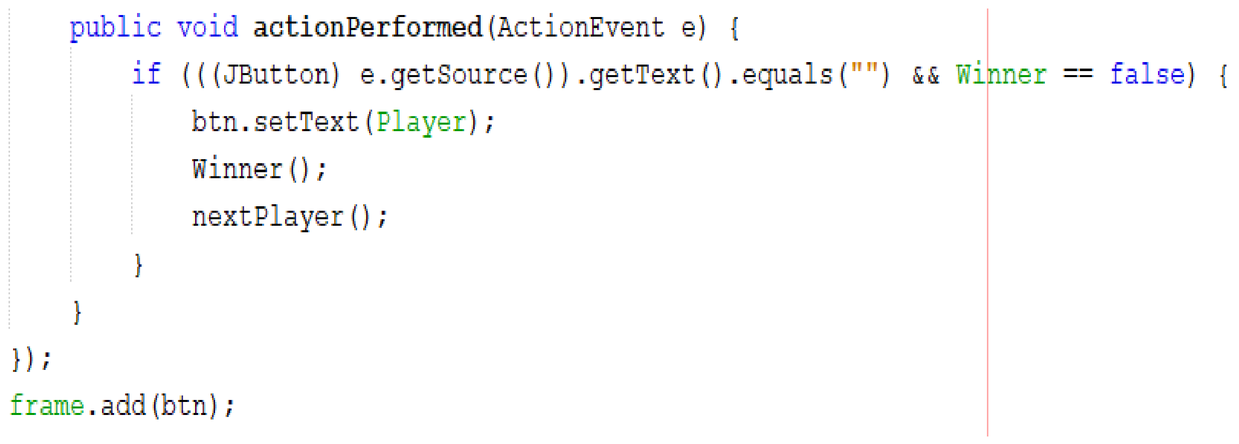
Example:

i)

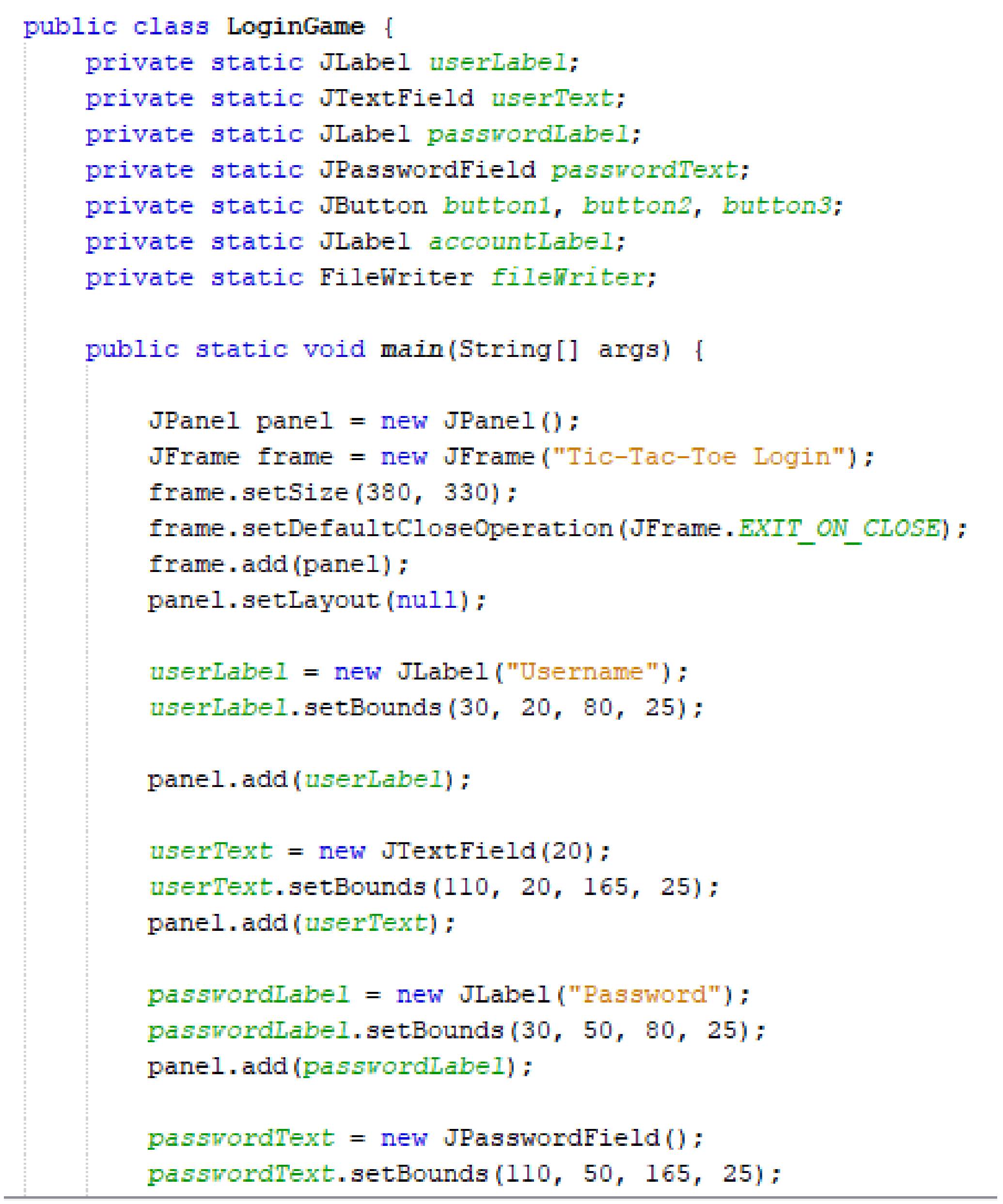


```
public class TicTacToeGame extends JFrame {

    private Container frame;
    private String Player;
    private JButton[][] board;
    private boolean Winner;
    private JMenuBar menuBar;
    private JMenu menu;
    private JMenuItem quit;
    private JMenuItem newGame;
```

public void actionPerformed :



```
    public void actionPerformed(ActionEvent e) {
        if (((JButton) e.getSource()).getText().equals("") && Winner == false) {
            btn.setText(Player);
            Winner();
            nextPlayer();
        }
    }
});
frame.add(btn);
```

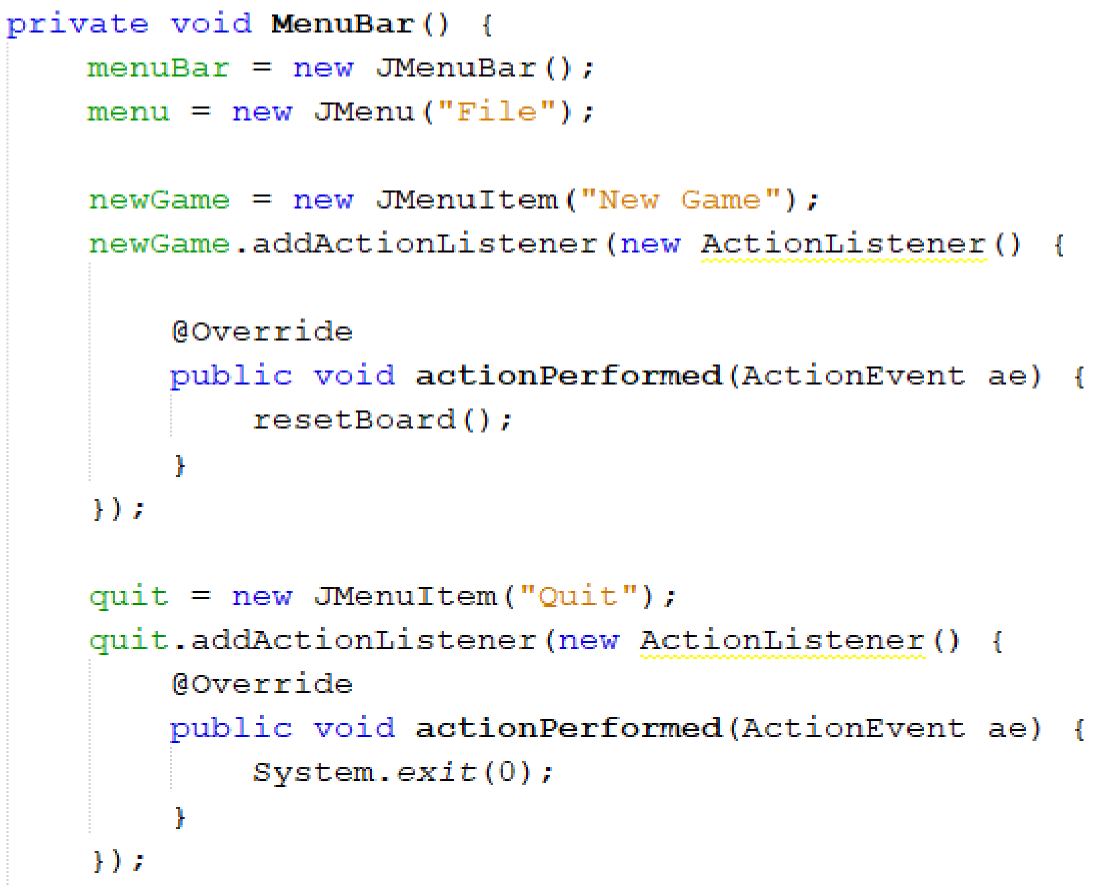
ii) 

```
public class LoginGame {
    private static JLabel userLabel;
    private static JTextField userText;
    private static JLabel passwordLabel;
    private static JPasswordField passwordText;
    private static JButton button1, button2, button3;
    private static JLabel accountLabel;
    private static FileWriter fileWriter;

    public static void main(String[] args) {

        JPanel panel = new JPanel();
        JFrame frame = new JFrame("Tic-Tac-Toe Login");
        frame.setSize(380, 330);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(panel);
        panel.setLayout(null);

        userLabel = new JLabel("Username");
        userLabel.setBounds(30, 20, 80, 25);

        panel.add(userLabel);

        userText = new JTextField(20);
        userText.setBounds(110, 20, 165, 25);
        panel.add(userText);

        passwordLabel = new JLabel("Password");
        passwordLabel.setBounds(30, 50, 80, 25);
        panel.add(passwordLabel);

        passwordText = new JPasswordField();
        passwordText.setBounds(110, 50, 165, 25);
```

5. Inner Class

Inner class is one class which is a member of another class. Inner Class used to logically group classes and interfaces in one place so that it can be more readable and maintainable. There are basically four types of inner classes in Java and the one that I used in this project in anonymous inner class. Anonymous inner classes are declared without any name at all. An anonymous inner class can implement only one interface at one time.

i) An anonymous inner class is created that implements the ActionListener interface. Action Listener defines a method that receives a notification when an action (clicking a button) take place.



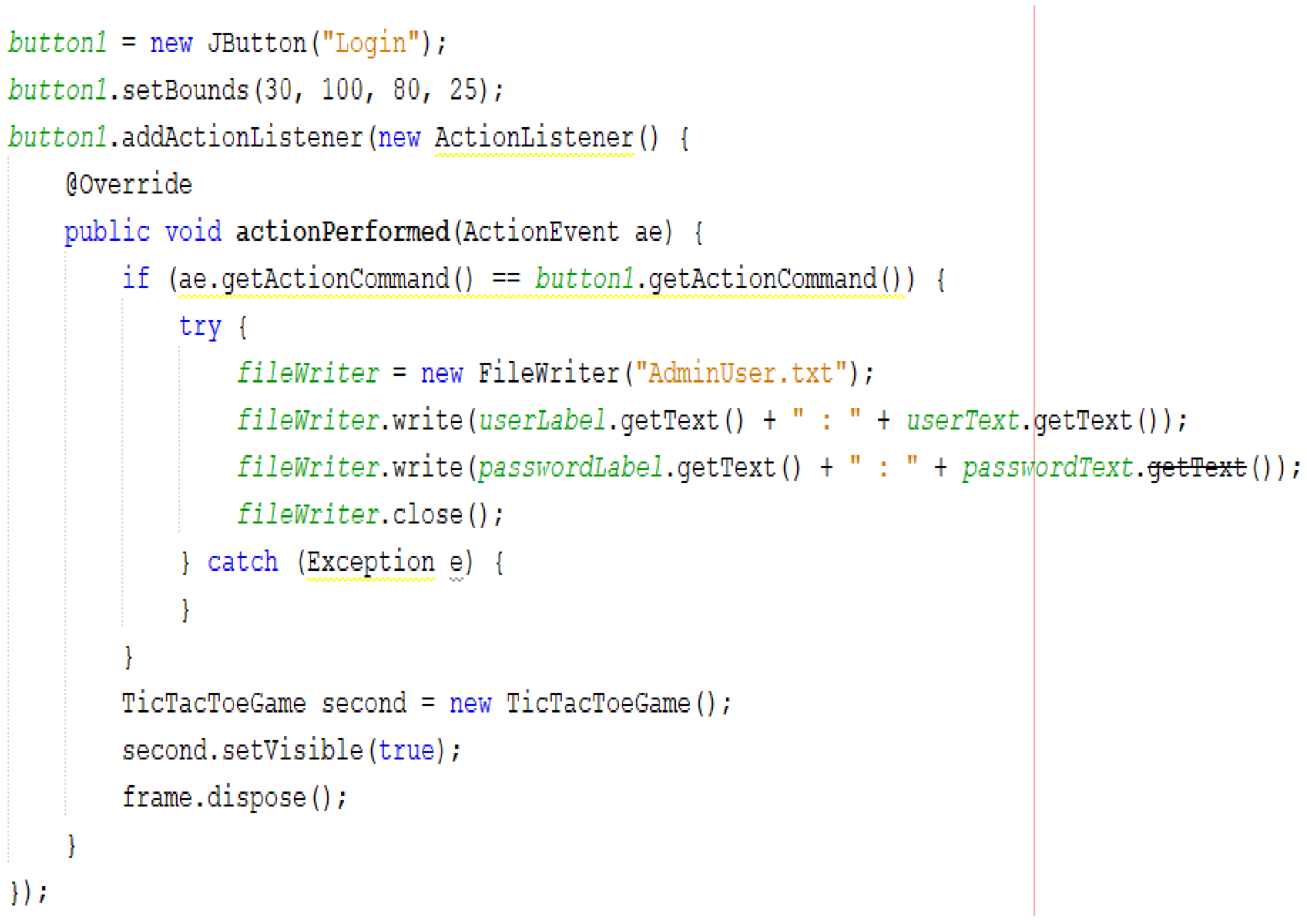
```
private void MenuBar() {
    menuBar = new JMenuBar();
    menu = new JMenu("File");

    newGame = new JMenuItem("New Game");
    newGame.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent ae) {
            resetBoard();
        }
    });

    quit = new JMenuItem("Quit");
    quit.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent ae) {
            System.exit(0);
        }
    });
```

5.0 READ AND WRITE IMPLEMENTATION

In programming, we often have the need to read or write data to file. In this project, I have implemented the common ways to read and write data to a file using Java I/O package. The java.io package contains nearly every class that need to perform input and output (I/O) in Java. In this project, Java character streams are used to perform input and output for 16-bit Unicode. The most frequently used classes are FileReader and FileWriter. Also, the new BufferedReader( ) method is used to opens a file for reading and returning a BufferedReader that can be used to read text from a file in an efficient manner.

i) Write Implementation

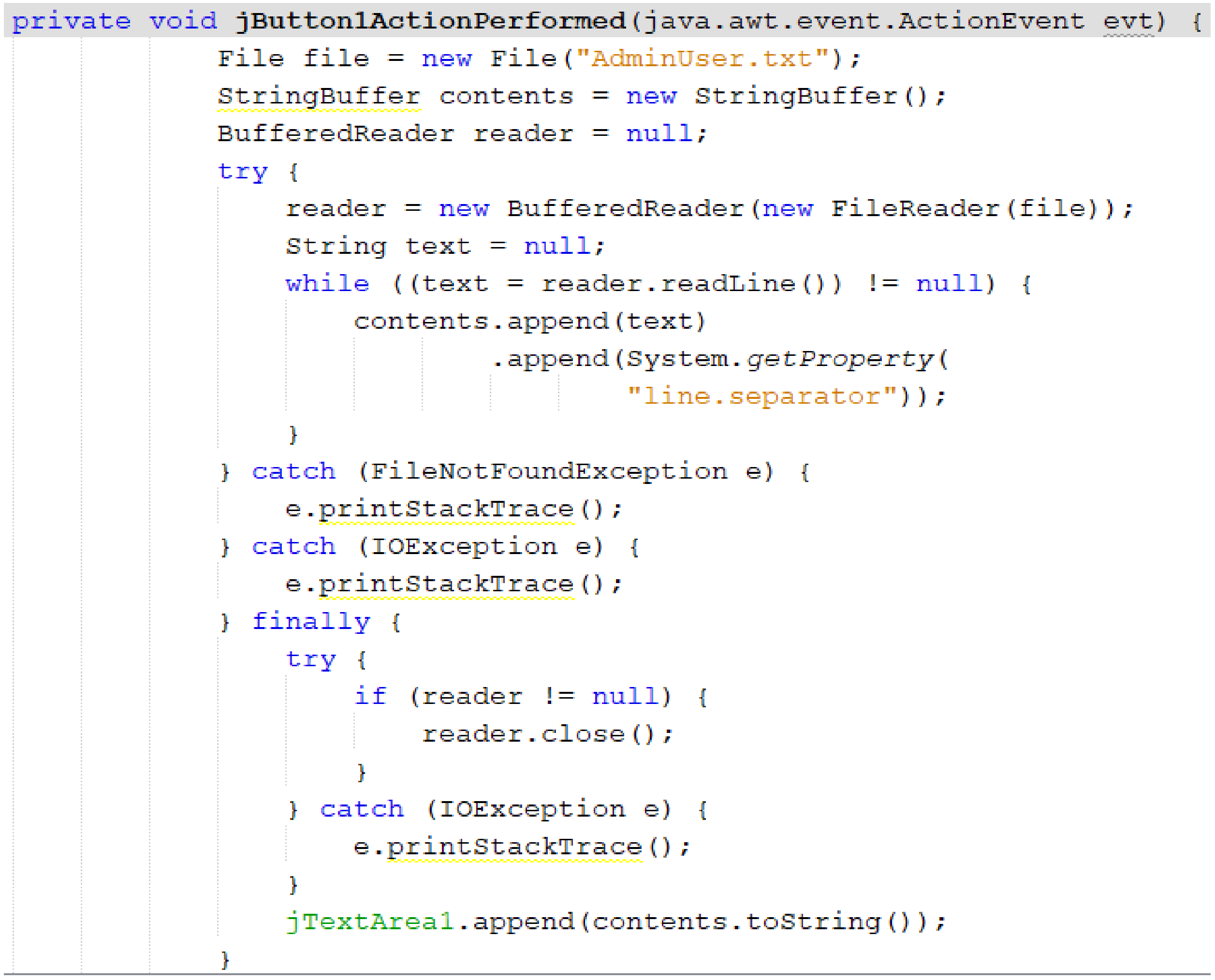
When the button "Login" is clicked by the user, the input in Username and Password will automatically write to AdminUser,txt file.



```
button1 = new JButton("Login");
button1.setBounds(30, 100, 80, 25);
button1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent ae) {
        if (ae.getActionCommand() == button1.getActionCommand()) {
            try {
                fileWriter = new FileWriter("AdminUser.txt");
                fileWriter.write(userLabel.getText() + " : " + userText.getText());
                fileWriter.write(passwordLabel.getText() + " : " + passwordText.~~getText~~());
                fileWriter.close();
            } catch (Exception e) {
            }
        }
        TicTacToeGame second = new TicTacToeGame();
        second.setVisible(true);
        frame.dispose();
    }
});
```

ii) Read Implementation

In Admin jFrame, when the admin clicked “READ DATA FROM FILE” button, the text in AdminUser.txt file will be read and pop up at the jTextArea1.



```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
            File file = new File("AdminUser.txt");
            StringBuffer contents = new StringBuffer();
            BufferedReader reader = null;
            try {
                reader = new BufferedReader(new FileReader(file));
                String text = null;
                while ((text = reader.readLine()) != null) {
                    contents.append(text)
                            .append(System.getProperty(
                                    "line.separator"));
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                try {
                    if (reader != null) {
                        reader.close();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                jTextArea1.append(contents.toString());
            }
```

6.0 USER MANUAL

TIC TAC TOE INSTRUCTIONS

You are X's and your opponent is O's. On your turn, click anywhere on the grid to place an X in that square. Your goal is to get three in a row before your opponent does. This game is not against a computer, so you need to play it with your friend or partner on the same computer.

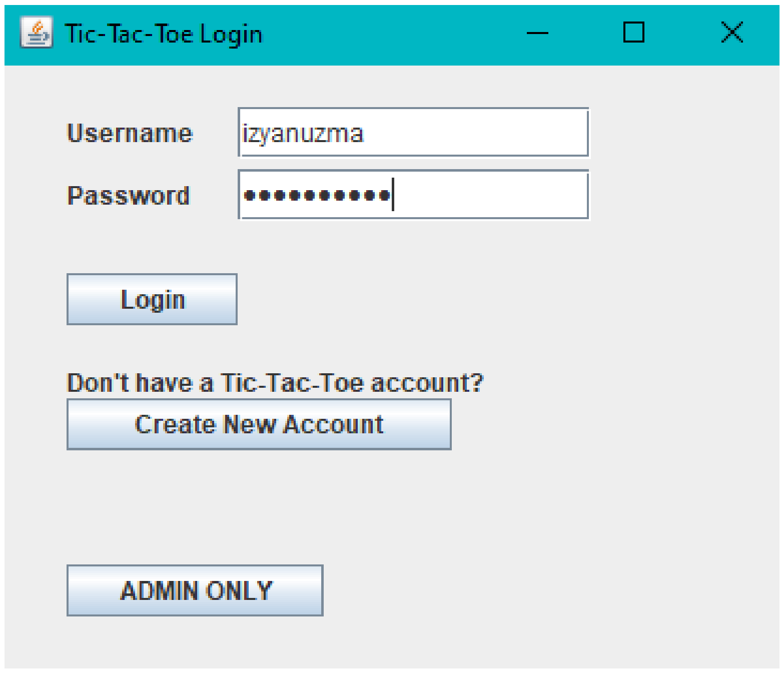
TIC TAC TOE TIPS AND TRICKS

Watch your opponent. Take note of where they place their O's. Keep your eyes open for those winning spots so you can block them before they get three in row. Also, don't ignore the corners because most players go for the middle space whenever they can.

HOW TO USE THE SYSTEM

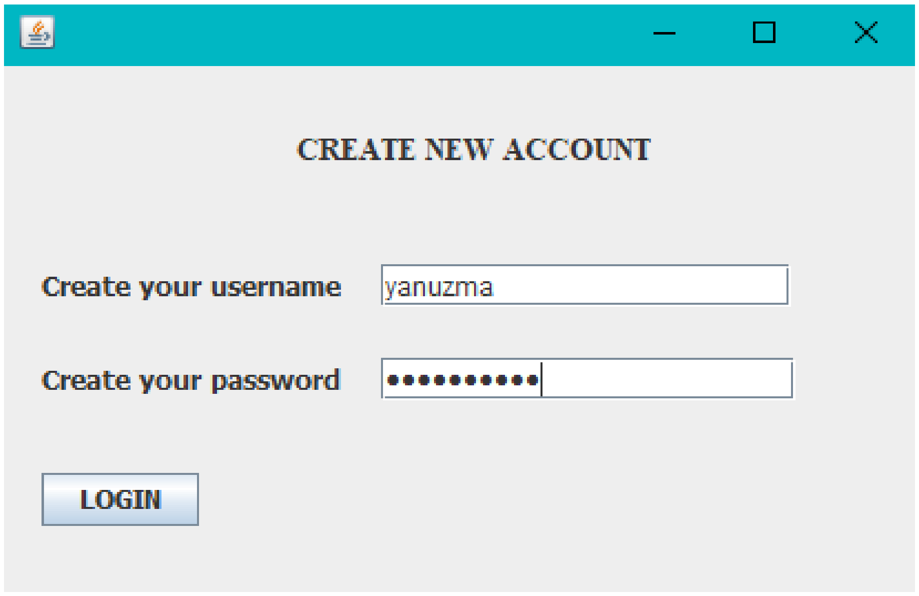
FIRST STEP

First step you need to Login into your Tic Tac Toe account before you can start playing. Fill in your Username and Password as shown in figure below. If you don't have a Tic Tac Toe account, you can create the new account by clicking the "Create New Account" below. The "ADMIN ONLY" button is only for admin to read the user or player data from text file.



SECOND STEP

For the new user or player, when you click to the "Create New Account" button, the window as shown in figure below will open and you just need just fill in and create your new username and new password. Then, just click button "LOGIN" and you are successfully done creating the new account, and it will directly open the game as shown in step three and then you can start playing.

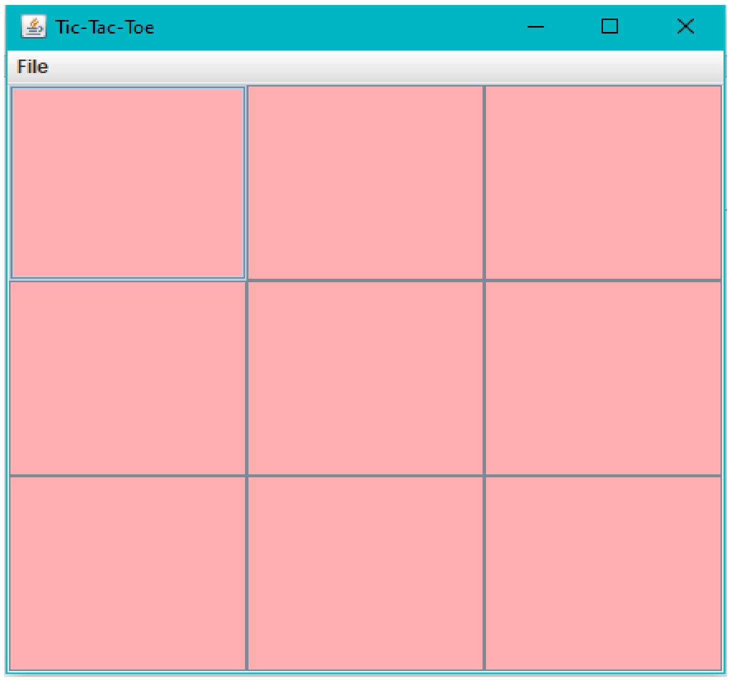

CREATE NEW ACCOUNT

Create your username yanuzma

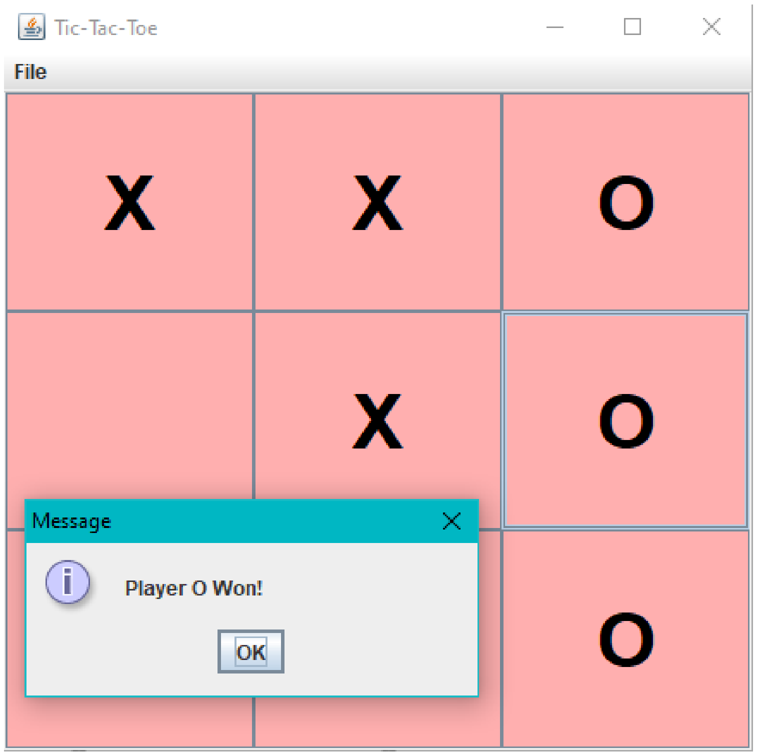
Create your password ●●●●●●●●●●●

THIRD STEP

Next, when you click button "Login" as shown in the first step's figure or after you are creating new account, the Tic Tac Toe game will open and you can start playing with your friend or partner.

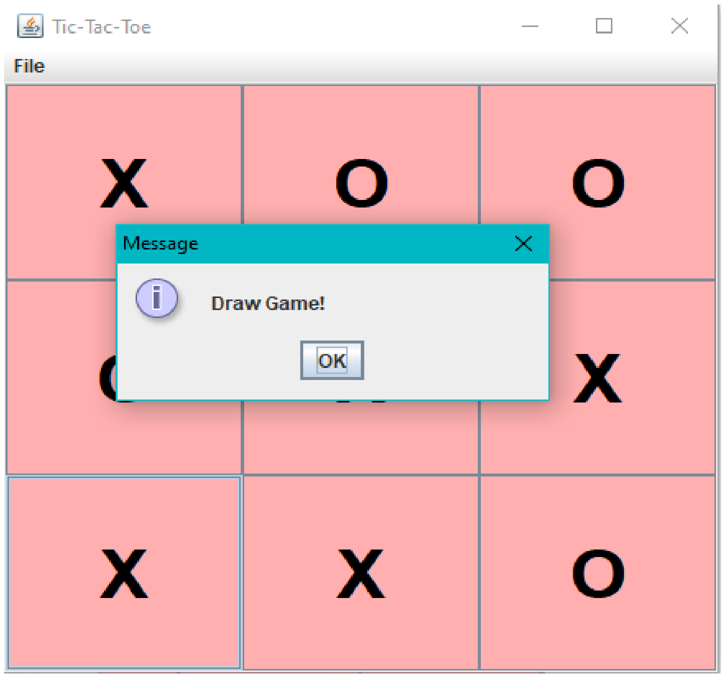


If one of you win the game, the Winner message will pop out as shown in figure below.

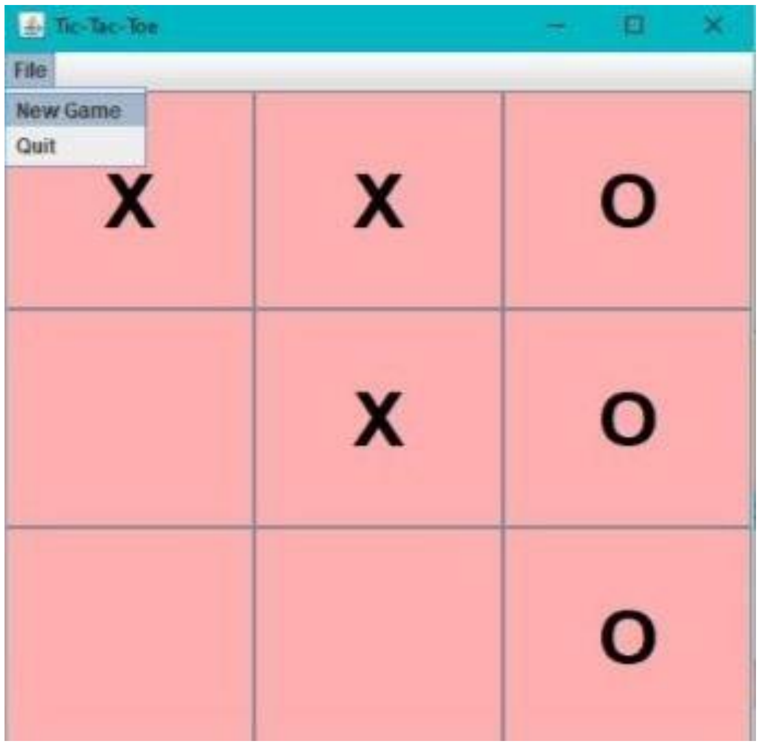


Player O Won!

But if there is no winner, then the message "Draw Game" will pop out.

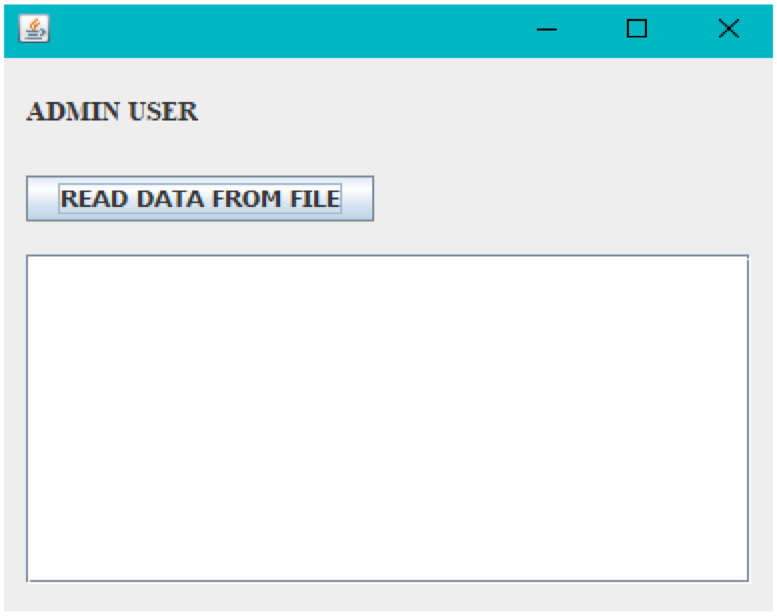


Also, there is a "File" at the Menu bar on the left top of the window that contents "New Game" and "Quit". If you want to start a new game again, then just click the "New Game" button at file otherwise click "Quit" to exit the game.

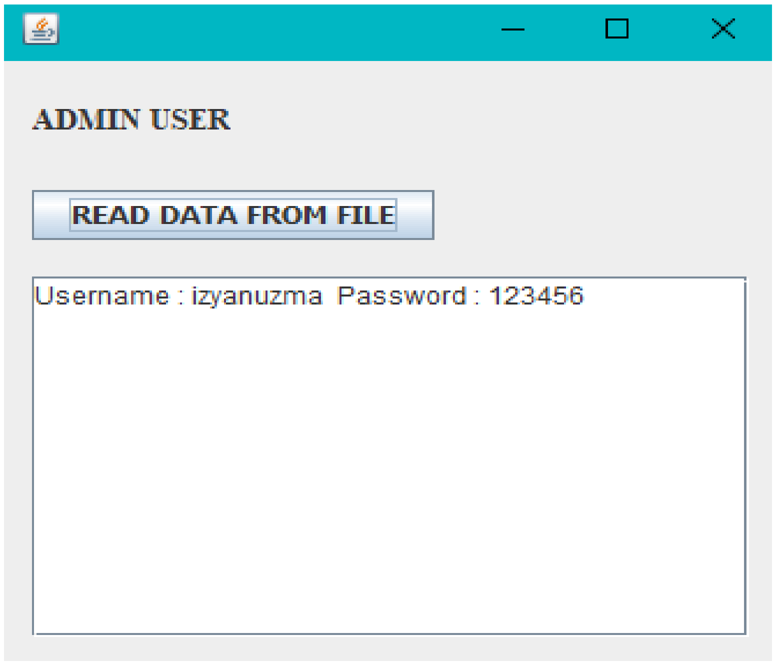


FOURTH STEP

Step four in only for Admin User. This jFrame was created for admin to read the user data from AdminUser.txt file. When the user or player fill in their username and password or creating new username and new password, the admin automatically can read their data from the AdminUser.txt file.



This is an example of input that has been read from the text file:



```
Username : izyanuzma  Password : 123456
```

7.0 CONCLUSION

In a nutshell, this project was successfully created by implementing Java GUI and several object oriented programming concepts. The use of Java GUI makes it easier for players or users to play and it is even more attractive. Tic Tac Toe is a game of predictability. This predictability is what helps foster strategic thinking of people especially young generation. They can learn through observation what their opponents' next move is and think ways on how to block them. In order to figure out what else they can do in the game to win, they are encourage to think more logically. Although this game is quite old for this era, but this game still has good intellectual quality. Tic Tac Toe is a game with rules and this can help people in their intellectual, socio moral and personality development. In fact, some parents recommend that these educational games with rules be a vital part of children's early education. Also, activities such as these can develop the bond between our family.