Table of Contents

Introduction

In this project, we set out to develop a complete Pong game using JavaFX, featuring multiple interactive screens for a smooth user experience. Our original plan included creating several screens, such as the Main Screen, Setting Screen, Login Screen, Forgot Password Screen, Sign Up Screen, Main Menu Screen, Main Menu Screen (Guest), Profile Screen, Change Password Screen, How to Play Screen, Game Screen, Results Screen, and Pause Screen.

As we progressed, we successfully implemented the Main Screen, Setting Screen, Login Screen, Main Menu Screen, Main Menu Screen (Guest), Profile Screen, How to Play Screen, Game Screen, Results Screen, and Pause Screen. However, we decided to simplify the project by removing the Forgot Password and Change Password screens due to the lack of a database for storing user data. Additionally, we opted to use a preset array for user information instead of allowing new user registrations, which led to the removal of the Sign-Up screen.

Despite these changes, we have created a functional local Pong game that offers essential features and a seamless user experience. In future versions, we plan to integrate a database to manage user information and bring back the removed features to further enhance the game's capabilities and user experience.

Main Screen

Source Code

MainScreen.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.effect.Glow?>

<?import javafx.scene.effect.Reflection?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.paint.LinearGradient?>

<?import javafx.scene.paint.Stop?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<AnchorPane id="AnchorPane" prefHeight="378.0" prefWidth="536.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.FXMLDocumentController">

    <children>

      <Pane prefHeight="378.0" prefWidth="589.0" style="-fx-background-color: #000000;" />

      <ImageView fx:id="BGmain_ID" fitHeight="378.0" fitWidth="614.0" layoutX="-31.0" layoutY="18.0"
pickOnBounds="true" preserveRatio="true">

         <image>

            <Image url="@Images/background.png" />

         </image>

      </ImageView>

        <Label fx:id="label" layoutX="87.0" layoutY="143.0" minHeight="16" minWidth="69" />

      <Button fx:id="login_ID" alignment="CENTER" cacheShape="false" focusTraversable="false"
layoutX="133.0" layoutY="180.0" mnemonicParsing="false" onAction="#loginbtn" prefHeight="72.0"
prefWidth="115.0" rotate="-10.0" scaleShape="false" text="LOGIN" textAlignment="CENTER">

         <font>

            <Font name="Britannic Bold" size="24.0" />

         </font>

         <effect>

            <Reflection fraction="0.69" topOpacity="0.3" />
```

```
         </effect>
         <textFill>
            <LinearGradient endX="1.0" endY="1.0">
               <stops>
                  <Stop>
                     <color>
                        <Color red="0.18896198272705078" green="0.43421053886413574"
blue="0.24209916591644287" />
                     </color>
                  </Stop>
                  <Stop offset="1.0">
                     <color>
                        <Color red="0.7105262875556946" green="0.27368420362472534"
blue="0.16447368264198303" />
                     </color>
                  </Stop>
               </stops>
            </LinearGradient>
         </textFill></Button>
      <Button fx:id="Guest_ID" alignment="CENTER" cacheShape="false" focusTraversable="false"
layoutX="342.0" layoutY="180.0" mnemonicParsing="false" onAction="#Guestbtn" prefHeight="72.0"
prefWidth="115.0" rotate="10.0" scaleShape="false" text="GUEST" textAlignment="CENTER">
         <font>
            <Font name="Britannic Bold" size="24.0" />
         </font>
         <effect>
            <Reflection fraction="0.69" topOpacity="0.3" />
         </effect>
         <textFill>
            <LinearGradient endX="1.0" endY="1.0">
               <stops>
                  <Stop>
                     <color>
                        <Color red="0.18896198272705078" green="0.43421053886413574"
blue="0.24209916591644287" />
                     </color>
                  </Stop>
                  <Stop offset="1.0">
                     <color>
```

```
                        <Color red="0.127964586019516" green="0.040631093084812164"
blue="0.30263158679008484" />
                     </color>
                  </Stop>
               </stops>
            </LinearGradient>
         </textFill></Button>
      <Text fill="WHITE" layoutX="190.0" layoutY="86.0" stroke="#7c4694" strokeType="OUTSIDE"
strokeWidth="2.0" text="PONG" textAlignment="CENTER" wrappingWidth="209.041015625">
         <font>
            <Font name="Calisto MT Bold Italic" size="61.0" />
         </font>
      </Text>
      <ImageView fitHeight="159.0" fitWidth="160.0" layoutX="7.0" layoutY="1.0" pickOnBounds="true"
preserveRatio="true">
         <image>
            <Image url="@Images/moon.png" />
         </image>
      </ImageView>
      <ImageView fitHeight="72.0" fitWidth="119.0" layoutX="426.0" layoutY="14.0" pickOnBounds="true"
preserveRatio="true">
         <image>
            <Image url="@Images/star.png" />
         </image>
         <effect>
            <Glow level="1.0" />
         </effect>
      </ImageView>
      <ImageView fitHeight="50.0" fitWidth="55.0" layoutX="227.0" layoutY="310.0" pickOnBounds="true"
preserveRatio="true" rotate="139.4">
         <image>
            <Image url="@Images/star_1.png" />
         </image>
      </ImageView>
      <ImageView fitHeight="91.0" fitWidth="69.0" layoutX="485.0" layoutY="148.0" pickOnBounds="true"
preserveRatio="true" rotate="-135.0">
         <image>
            <Image url="@Images/star_1.png" />
         </image>
      </ImageView>
```

```
      <ImageView fitHeight="55.0" fitWidth="45.0" layoutX="197.0" layoutY="112.0" pickOnBounds="true"
preserveRatio="true" rotate="-90.0">
         <image>
            <Image url="@Images/star_1.png" />
         </image>
      </ImageView>
      <ImageView fitHeight="72.0" fitWidth="119.0" layoutX="27.0" layoutY="274.0" pickOnBounds="true"
preserveRatio="true" rotate="-110.0">
         <image>
            <Image url="@Images/star.png" />
         </image>
         <effect>
            <Glow level="1.0" />
         </effect>
      </ImageView>
    </children>
</AnchorPane>
```

FXMLDocumentController.java

```
package javafxaassignment;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.image.ImageView;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class FXMLDocumentController implements Initializable {
```

```
@FXML
private Label label;
@FXML
private ImageView BGmain_ID;
@FXML
private Button login_ID;
@FXML
private Button Sign_ID;
@FXML
private Button Guest_ID;

private void handleButtonAction(ActionEvent event) {
    System.out.println("You clicked me!");
    label.setText("Hello World!");
}

@Override
public void initialize(URL url, ResourceBundle rb) {

}

@FXML
private void loginbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}


@FXML
private void Guestbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainMenu.fxml"));
```

```
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void signupbtn(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("signup.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Screenshot

Explanation

To give it a simple explanation here is the Main Screen, essentially this page hold access to allow user to either proceed as guest to directly move to the main menu for guest. While the login button redirects us to the login page.

The page is done with scene builder using FXML and FXML controller.

Settings

Source Code

SettingsController.java

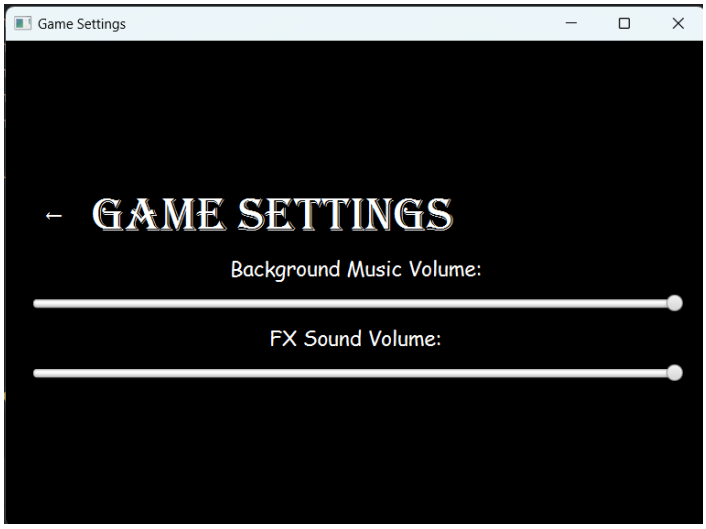
```
package javafxaassignment;

import javafx.fxml.FXML;
import javafx.scene.control.Slider;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.stage.Stage;
import java.util.prefs.Preferences;
import javafx.event.ActionEvent;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;

public class SettingsController {

    @FXML
    private Slider bgMusicSlider;

    @FXML
    private Slider fxSoundSlider;
    private Preferences prefs;
    private MediaPlayer bgMusicPlayer;
    private MediaPlayer fxSoundPlayer;
    private Stage stage;
    @FXML
    private Button back;

    public void setStage(Stage stage) {
        this.stage = stage;
    }

    private void initialize() {
        prefs = Preferences.userNodeForPackage(getClass());
        String bgMusicFile = "/demo/main/background_music.mp3";
        String fxSoundFile = "/demo/main/fx_sound.mp3";

        Media bgMusicMedia = new Media(getClass().getResource(bgMusicFile).toString());
        Media fxSoundMedia = new Media(getClass().getResource(fxSoundFile).toString());

        bgMusicPlayer = new MediaPlayer(bgMusicMedia);
        fxSoundPlayer = new MediaPlayer(fxSoundMedia);

        bgMusicSlider.valueProperty().addListener((observable, oldValue, newValue) -> {
```

```
            double volume = newValue.doubleValue() / 100.0;
            bgMusicPlayer.setVolume(volume);
            System.out.println("Background Music Volume: " + volume);
        });

        fxSoundSlider.valueProperty().addListener((observable, oldValue, newValue) -> {
            double volume = newValue.doubleValue() / 100.0;
            fxSoundPlayer.setVolume(volume);
            System.out.println("FX Sound Volume: " + volume);
        });

        loadPreferences();
        bgMusicPlayer.play();
        fxSoundPlayer.play();
    }

    private void loadPreferences() {
        double bgVolume = prefs.getDouble("bgMusicVolume", 50.0);
        double fxVolume = prefs.getDouble("fxSoundVolume", 50.0);

        bgMusicSlider.setValue(bgVolume);
        fxSoundSlider.setValue(fxVolume);

        bgMusicPlayer.setVolume(bgVolume / 100.0);
        fxSoundPlayer.setVolume(fxVolume / 100.0);
    }

    private void stop() {
        prefs.putDouble("bgMusicVolume", bgMusicSlider.getValue());
        prefs.putDouble("fxSoundVolume", fxSoundSlider.getValue());

        bgMusicPlayer.stop();
        fxSoundPlayer.stop();
    }

    private void handleBack() {
        stage.close();
    }

    @FXML
    private void Back(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Settings.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.geometry.Insets?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.Slider?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.layout.VBox?>


<VBox alignment="CENTER" spacing="10" style="-fx-background-color: black;"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/22"
fx:controller="javafxaassignment.SettingsController">

    <padding>

        <Insets bottom="20" left="20" right="20" top="20" />

    </padding>

    <HBox alignment="CENTER_LEFT" spacing="10">

        <Button fx:id="back" onAction="#Back" style="-fx-font-size: 20px; -fx-text-fill: white; -fx-
background-color: transparent;" text="←" />

        <Label style="-fx-font-size: 42px; -fx-text-fill: white; -fx-font-family: 'Algerian';"
text="Game Settings" />

    </HBox>

    <Label style="-fx-font-size: 18px; -fx-text-fill: white; -fx-font-family: 'Comic Sans MS';"
text="Background Music Volume:" />

    <Slider fx:id="bgMusicSlider" max="100" min="0" value="50" />

    <Label style="-fx-font-size: 18px; -fx-text-fill: white; -fx-font-family: 'Comic Sans MS';"
text="FX Sound Volume:" />

    <Slider fx:id="fxSoundSlider" max="100" min="0" value="50" />

</VBox>
```

Screenshot



Explanation

I have imported a few special libraries like slider to perform the slide bar on the volume state; media to import the mp3 from the files; media player to play the audio; and preferences to store user audio volumes preference. The user preference is to ensure once the user change the volume state, they are unneeded to change at the next time when they reopen the game.

So, at first, I initiate a preference object to store user preference, and use string variables to store the files location. Then I declare media objects to load the audio files and add listeners to update the volume metrics of the slide bar whenever the values change. After that, I use the loadPreference method to load the change in to the preference. I used the _Player.play() to start the audio.

In the loadPreference method, I use a double to retrieve the changed volume values of initially volume values at 50. By using the _Slider.setValue(_Volume) to set the slide bar stop point and _Player.setVolume(_Volume/100.0) to set the actual playing/hearing volume.

I implementeda stop method to make sure the audio stop when the app is close or exit. The prefs.putDouble("_Volume", _Slider.getValue()); will save the last or current volume to the preference so that it will use this values when the app restart. And then I use _Player.stop() to stop the audio.

Login Screen

The Login program allows users to log in as two types of accounts user and administrator. Additionally, users can be authenticated by verifying saved username and password to change password.

Source Code

login.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.PasswordField?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.effect.Bloom?>

<?import javafx.scene.effect.InnerShadow?>

<?import javafx.scene.effect.Light.Distant?>

<?import javafx.scene.effect.Lighting?>

<?import javafx.scene.effect.SepiaTone?>

<?import javafx.scene.effect.Shadow?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.LoginController">

   <left>

      <AnchorPane prefHeight="400.0" prefWidth="91.0" BorderPane.alignment="CENTER" />

   </left>

   <center>

      <AnchorPane prefHeight="400.0" prefWidth="387.0" BorderPane.alignment="CENTER">

         <children>
```

```
            <Pane  layoutX="-94.0"  prefHeight="400.0"  prefWidth="607.0"  style="-fx-background-color:
#000000;">

               <children>

                  <ImageView  fitHeight="340.0"  fitWidth="591.0"  layoutY="35.0"  pickOnBounds="true"
preserveRatio="true">

                     <image>

                        <Image url="@Images/background.png" />

                     </image>

                  </ImageView>

                  <ImageView    fitHeight="290.0"    fitWidth="347.0"    layoutX="5.0"    layoutY="99.0"
pickOnBounds="true" preserveRatio="true">

                     <image>

                        <Image url="@Images/shootingstar.png" />

                     </image>

                  </ImageView>

                  <Pane     layoutX="101.0"     layoutY="94.0"     opacity="0.93"     prefHeight="255.0"
prefWidth="406.0" style="-fx-background-color: #808080;">

                     <children>

                        <Label   alignment="CENTER"   layoutX="14.0"   layoutY="64.0"   prefHeight="50.0"
prefWidth="181.0" text="Username:" textFill="WHITE">

                           <font>

                              <Font name="Agency FB Bold" size="28.0" />

                           </font>

                        </Label>

                        <TextField fx:id="username" alignment="CENTER" layoutX="180.0" layoutY="69.0"
prefHeight="33.0" prefWidth="156.0" promptText="Username">

                           <font>

                              <Font name="System Bold" size="18.0" />

                           </font>

                        </TextField>

                        <PasswordField     fx:id="password"     alignment="CENTER"     layoutX="180.0"
layoutY="133.0" prefHeight="39.0" prefWidth="157.0" promptText="Password">

                           <font>

                              <Font name="System Bold" size="18.0" />

                           </font>

                        </PasswordField>

                        <Label  alignment="CENTER"  layoutX="12.0"  layoutY="124.0"  prefHeight="58.0"
prefWidth="185.0" text="Password :" textFill="WHITE">

                           <font>

                              <Font name="Agency FB Bold" size="28.0" />

                           </font>
```

```
                        </Label>
                        <Button fx:id="login" alignment="CENTER" layoutX="170.0" layoutY="205.0"
mnemonicParsing="false" onAction="#userlogin" prefHeight="36.0" prefWidth="66.0" text="Login"
textAlignment="JUSTIFY">
                           <font>
                              <Font name="Bell MT Bold" size="14.0" />
                           </font>
                           <effect>
                              <Lighting>
                                 <bumpInput>
                                    <Shadow />
                                 </bumpInput>
                                 <light>
                                    <Light.Distant>
                                       <color>
                                          <Color red="1.0" green="0.8888888955116272"
blue="0.8888888955116272" />
                                       </color>
                                    </Light.Distant>
                                 </light>
                              </Lighting>
                           </effect>
                        </Button>
                     </children>
                     <effect>
                        <SepiaTone level="0.05" />
                     </effect>
                  </Pane>
                  <Button fx:id="backlogin" layoutX="13.0" layoutY="18.0" mnemonicParsing="false"
onAction="#backlogin" text="&lt;">
                     <font>
                        <Font name="Britannic Bold" size="21.0" />
                     </font>
                     <effect>
                        <Bloom threshold="0.95" />
                     </effect>
                  </Button>
                  <Text fill="WHITE" fontSmoothingType="LCD" layoutX="216.0" layoutY="64.0"
opacity="0.84" strokeType="OUTSIDE" strokeWidth="0.0" text="LOGIN" textAlignment="CENTER"
wrappingWidth="177.09765625">
```

```
                        <font>
                           <Font name="Haettenschweiler" size="59.0" />
                        </font>
                        <effect>
                           <Lighting             diffuseConstant="2.0"             specularConstant="1.82"
specularExponent="14.71" surfaceScale="10.0">
                              <light>
                                 <Light.Distant elevation="57.41" />
                              </light>
                              <bumpInput>
                                 <InnerShadow />
                              </bumpInput>
                           </Lighting>
                        </effect>
                     </Text>
                     <ImageView    fitHeight="79.0"    fitWidth="75.0"    layoutX="178.0"    layoutY="4.0"
pickOnBounds="true" preserveRatio="true">
                        <image>
                           <Image url="@Images/starry.png" />
                        </image>
                     </ImageView>
                     <ImageView    fitHeight="79.0"    fitWidth="75.0"    layoutX="356.0"    layoutY="16.0"
pickOnBounds="true" preserveRatio="true">
                        <image>
                           <Image url="@Images/starry.png" />
                        </image>
                     </ImageView>
                     <ImageView   fitHeight="65.0"   fitWidth="49.0"   layoutX="526.0"   layoutY="100.0"
pickOnBounds="true" preserveRatio="true" rotate="108.4">
                        <image>
                           <Image url="@Images/star_1.png" />
                        </image>
                     </ImageView>
                     <ImageView   fitHeight="82.0"   fitWidth="63.0"   layoutX="532.0"   layoutY="283.0"
pickOnBounds="true" preserveRatio="true">
                        <image>
                           <Image url="@Images/star_1.png" />
                        </image>
                     </ImageView>
```

```
                  <ImageView    fitHeight="82.0"    fitWidth="63.0"    layoutX="34.0"    layoutY="110.0"
pickOnBounds="true" preserveRatio="true">
                     <image>
                        <Image url="@Images/star_1.png" />
                     </image>
                  </ImageView>
               </children>
            </Pane>
            <Label fx:id="wronglogin" layoutX="171.0" layoutY="137.0" textFill="RED" />
         </children>
      </AnchorPane>
   </center>
</BorderPane>
```

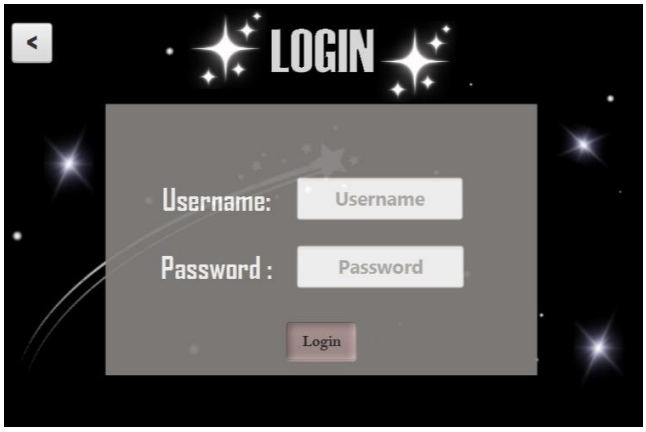
LoginController.java

```
package javafxaassignment;

import java.net.URL;
import javafx.scene.Node;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;


public class LoginController implements Initializable {

    @FXML
    private TextField username;
```

```
@FXML
private PasswordField password;
@FXML
private Button login;
@FXML
private Label wronglogin;
@FXML
private Button backlogin;

@Override
public void initialize(URL url, ResourceBundle rb) {

}



@FXML
private void userlogin(ActionEvent event) throws Exception{

    if(username.getText().equals("zora") && password.getText().equals("123")) {

        wronglogin.setText("Success");

    Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();

    }

if(username.getText().equals("Jen") && password.getText().equals("456")) {

        wronglogin.setText("Success");

    Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
```

```
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();

        }

    else if(username.getText().isEmpty() && password.getText().isEmpty()) {
        wronglogin.setText("Please Enter your credentials.");
    }

    else{
    wronglogin.setText("Wrong Credentials");

}
}

    @FXML
    private void backlogin(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainScreen.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Screenshot



Explanation

Key Components

1. Imports and Class Declaration

   The necessary JavaFX packages are imported. The 'LoginController' class implements 'Initializable', which requires implementing the initialize method.

```
package javafxaassignment;

import java.net.URL;
import javafx.scene.Node;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import javafx.stage.Stage;
```

```
public class LoginController implements Initializable {
```

2. FXML Annotations

   '@FXML' annotations link JavaFX UI components (defined in the FXML file) with the controller's fields and methods.

```
    @FXML
```

3. Fields

```
@FXML
private TextField username;
@FXML
private PasswordField password;
@FXML
private Button login;
@FXML
private Label wronglogin;
@FXML
private Button backlogin;
```

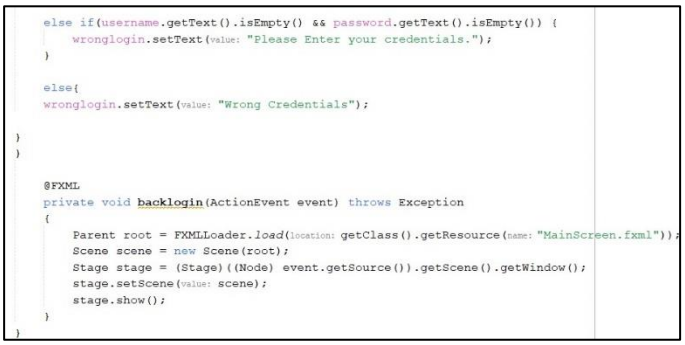
   - 'TextField' username: Text field for the user to enter their username.
   - 'PasswordField' password: Password field for the user to enter their password.
   - 'Button login': Button to trigger the login process.
   - 'Label wronglogin': Label to display login success or error messages.
   - 'Button backlogin': Button to navigate back to the main screen.

4. Methods

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}
```

- ‘initialize(URL url, ResourceBundle rb)’: This method is called to initialize the controller after the FXML file has been loaded.

```
@FXML
private void userlogin(ActionEvent event) throws Exception{

    if(username.getText().equals(anObject: "zora") && password.getText().equals(anObject: "123")) {

        wronglogin.setText(value: "Success");

    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
    }

if(username.getText().equals(anObject: "Jen") && password.getText().equals(anObject: "456")) {

        wronglogin.setText(value: "Success");

    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();

    }
```



```
    else if(username.getText().isEmpty() && password.getText().isEmpty()) {
        wronglogin.setText(value: "Please Enter your credentials.");
    }

    else{
    wronglogin.setText(value: "Wrong Credentials");

}
}


    @FXML
    private void backlogin(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainScreen.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(value: scene);
        stage.show();
    }
}
```

- ‘userlogin(ActionEvent event)’: This method handles the login process:
  - It checks the entered username and password.
  - If the credentials are correct, it sets the label text to "Success" and navigates to the MainMenuLogin.fxml scene.
  - If the fields are empty, it prompts the user to enter credentials.
  - If the credentials are incorrect, it displays "Wrong Credentials".
- ‘backlogin(ActionEvent event)’: This method navigates back to the MainScreen.fxml scene when the back button is pressed.

OOP Concepts

1. Encapsulation

   Fields like ‘username’, ‘password’, ‘login’, ‘wronglogin’, and ‘backlogin’ are encapsulated within the ‘LoginController’ class. They are marked private, meaning they can only be accessed within the class itself. Methods like ‘userlogin’ and ‘backlogin’ provide controlled access to the functionality, ensuring the internal state of the controller is managed properly.

```
@FXML
private void userlogin(ActionEvent event) throws Exception{

    if(username.getText().equals(anObject: "zora") && password.getText().equals(anObject: "123")) {

        wronglogin.setText(value: "Success");

    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
    }

if(username.getText().equals(anObject: "Jen") && password.getText().equals(anObject: "456")) {

        wronglogin.setText(value: "Success");

    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
    }
```

```
    else if(username.getText().isEmpty() && password.getText().isEmpty()) {
        wronglogin.setText(value: "Please Enter your credentials.");
    }

    else{
    wronglogin.setText(value: "Wrong Credentials");

}
}


    @FXML
    private void backlogin(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainScreen.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(value: scene);
        stage.show();
    }
}
```

2. Abstraction

   The controller abstracts the login functionality from the UI. The UI interacts with the controller through defined methods without needing to know the implementation details of the login logic. The 'FXML' annotations hide the details of how UI components are linked to the controller, providing a clean separation of concerns.

```
@FXML
private TextField username;
@FXML
private PasswordField password;
@FXML
private Button login;
@FXML
private Label wronglogin;
@FXML
private Button backlogin;
```

3. Inheritance

   The 'LoginController' class extends 'Initializable'. Although 'Initializable' is an interface rather than a class, this shows a form of inheritance where 'LoginController' inherits the contract to implement the 'initialize' method.

```
public class LoginController implements Initializable {
```

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}
```

4. Polymorphism

   While this specific example doesn't heavily rely on polymorphism, it can be extended to do so. For instance, if there were multiple types of login (e.g., user login, admin login), you could create an interface or an abstract class for login handlers and implement different login behaviors using polymorphism. The 'FXMLLoader.load' method demonstrates polymorphism by loading different FXML files based on the provided path, allowing for different UI screens to be shown.

Main Menu Screen (Guest)

Source Code

MainMenu.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.effect.InnerShadow?>

<?import javafx.scene.effect.Light.Distant?>

<?import javafx.scene.effect.Lighting?>

<?import javafx.scene.effect.Reflection?>

<?import javafx.scene.effect.Shadow?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<AnchorPane id="AnchorPane" fx:id="Mainmenu_ID" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxaassignment.MainMenuController">

   <children>

      <Pane prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #000000;">

         <children>

            <ImageView fitHeight="344.0" fitWidth="622.0" layoutX="-21.0" layoutY="14.0"
pickOnBounds="true" preserveRatio="true">

               <image>

                  <Image url="@Images/background.png" />

               </image>

            </ImageView>

            <Button fx:id="plays2" alignment="CENTER" layoutX="153.0" layoutY="200.0"
mnemonicParsing="false" onAction="#Playbtn" prefHeight="66.0" prefWidth="115.0" text="PLAY">

               <font>

                  <Font name="Rockwell Extra Bold" size="24.0" />

               </font>

               <effect>

                  <InnerShadow choke="0.23" height="201.0" radius="113.5" width="255.0">

                     <color>
```

```
                        <Color red="0.6196078658103943" green="0.18431372940540314"
blue="0.5529412031173706" />
                     </color>
                     <input>
                        <Reflection />
                     </input>
                  </InnerShadow>
               </effect>
            </Button>
            <Button fx:id="Signout_ID" layoutX="327.0" layoutY="200.0" mnemonicParsing="false"
onAction="#Signoutbtn" prefHeight="66.0" prefWidth="115.0" text="LOGIN">
               <font>
                  <Font name="Rockwell Extra Bold" size="21.0" />
               </font>
               <effect>
                  <InnerShadow height="200.5" radius="113.375" width="255.0">
                     <color>
                        <Color red="0.1725490242242813" green="0.35686275362968445"
blue="0.7882353067398071" />
                     </color>
                     <input>
                        <Reflection />
                     </input>
                  </InnerShadow>
               </effect>
            </Button>
            <ImageView fitHeight="125.0" fitWidth="124.0" layoutX="364.0" layoutY="15.0"
pickOnBounds="true" preserveRatio="true">
               <image>
                  <Image url="@Images/,moon.png" />
               </image>
            </ImageView>
            <Text fill="WHITE" layoutX="175.0" layoutY="77.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="GUEST MENU">
               <font>
                  <Font name="Segoe Print Bold" size="35.0" />
               </font>
            </Text>
            <ImageView fitHeight="97.0" fitWidth="114.0" layoutX="49.0" layoutY="29.0"
pickOnBounds="true" preserveRatio="true">
               <image>
```

```
                  <Image url="@Images/fireball.png" />
               </image>
            </ImageView>
            <Button fx:id="settings2" layoutX="546.0" layoutY="354.0" mnemonicParsing="false"
onAction="#settings2" prefHeight="31.0" prefWidth="40.0" text="⚙">
               <font>
                  <Font size="15.0" />
               </font>
               <effect>
                  <Lighting specularConstant="0.62" specularExponent="35.63">
                     <bumpInput>
                        <Shadow />
                     </bumpInput>
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="1.0" green="0.8156862854957581" blue="0.6666666865348816" />
                           </color>
                        </Light.Distant>
                     </light>
                  </Lighting>
               </effect>
            </Button>
            <Button fx:id="htp2" alignment="CENTER" layoutX="497.0" layoutY="354.0"
mnemonicParsing="false" onAction="#htp2" prefHeight="31.0" prefWidth="40.0" text="???"
textAlignment="CENTER">
               <font>
                  <Font name="System Bold" size="13.0" />
               </font>
               <effect>
                  <Lighting>
                     <bumpInput>
                        <Shadow />
                     </bumpInput>
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="0.9725490212440491" green="0.8627451062202454"
blue="0.8627451062202454" />
                           </color>
```

```
                        </Light.Distant>
                     </light>
                  </Lighting>
               </effect>
            </Button>
            <ImageView fitHeight="42.0" fitWidth="61.0" layoutX="184.0" layoutY="156.0"
pickOnBounds="true" preserveRatio="true">
               <image>
                  <Image url="@Images/astronaut3.png" />
               </image>
            </ImageView>
            <ImageView fitHeight="60.0" fitWidth="69.0" layoutX="354.0" layoutY="140.0"
pickOnBounds="true" preserveRatio="true">
               <image>
                  <Image url="@Images/astronaut2.png" />
               </image>
            </ImageView>
            <Button fx:id="X_ID" layoutX="542.0" layoutY="14.0" mnemonicParsing="false"
onAction="#xbtn" prefHeight="25.0" prefWidth="25.0" text="X">
               <font>
                  <Font name="System Bold" size="24.0" />
               </font>
               <effect>
                  <InnerShadow choke="0.7" height="36.0" radius="17.5" width="36.0">
                     <color>
                        <Color red="0.24313725531101227" green="0.14509804546833038"
blue="0.6313725709915161" />
                     </color>
                  </InnerShadow>
               </effect>
            </Button>
         </children>
      </Pane>
   </children>
</AnchorPane>
```

MainMenuController.java

```
package javafxaassignment;


import java.net.URL;
```

```
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.stage.Stage;
import javafx.scene.layout.AnchorPane;

public class MainMenuController implements Initializable {

    @FXML
    private Button Profile_ID;
    @FXML
    private Button X_ID;



    Stage stage;
    @FXML
    private AnchorPane Mainmenu_ID;
    @FXML
    private Button Signout_ID;
    @FXML
    private Button plays2;
    @FXML
    private Button settings2;
    @FXML
    private Button htp2;

    @Override
    public void initialize(URL url, ResourceBundle rb) {


    }
```

```
@FXML
private void Signoutbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainScreen.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
}

@FXML
private void Playbtn(ActionEvent event) throws Exception{
    Parent root = FXMLLoader.load(getClass().getResource("Game_1.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}

@FXML
private void xbtn(ActionEvent event)
{
    Alert alert = new Alert(AlertType.CONFIRMATION);
    alert.setTitle("DIEEE");
    alert.setHeaderText("You're about to die!");
    alert.setContentText("Do you want to die before saving?:");

    if(alert.showAndWait().get()== ButtonType.OK)
    {
    stage = (Stage) Mainmenu_ID.getScene().getWindow();
    stage.close();
}
}

@FXML
private void Profilebtn(ActionEvent event) {
}

@FXML
```

```
    private void settings2(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("settings_1.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void htp2(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("HowToPlay_1.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Screenshot

Explanation

It shares most of its counterpart (Main Menu Screen). It also acts as a hub for user navigation of the application. Albeit missing certain features such access to the profile stage. Another thing, it is not as secure as its counterpart as it can be directly accessed via Main Screen. This method access contains no security requirement.

Main Menu Screen (Login)

Source Code

MainMenuLogin.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.effect.InnerShadow?>

<?import javafx.scene.effect.Light.Distant?>

<?import javafx.scene.effect.Lighting?>

<?import javafx.scene.effect.Shadow?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<AnchorPane id="AnchorPane" fx:id="Mainmenu_ID" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxaassignment.MainMenuLoginController">

   <children>

      <Pane prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #000000;">

         <children>

            <ImageView fitHeight="377.0" fitWidth="646.0" layoutX="-46.0" layoutY="20.0"
pickOnBounds="true" preserveRatio="true">

               <image>

                  <Image url="@Images/background.png" />

               </image>

            </ImageView>

            <Pane layoutX="245.0" layoutY="128.0" />

            <Button fx:id="Play_ID" alignment="CENTER" cacheShape="false" centerShape="false"
contentDisplay="CENTER" focusTraversable="false" layoutX="246.0" layoutY="140.0"
mnemonicParsing="false" onAction="#Playbtn" prefHeight="60.0" prefWidth="108.0" scaleShape="false"
text="PLAY">

               <effect>

                  <InnerShadow choke="0.23" height="147.0" radius="100.0" width="255.0">

                     <color>
```

```
                        <Color red="0.6196078658103943" green="0.18431372940540314"
blue="0.5529412031173706" />

                     </color>

                  </InnerShadow>

               </effect>

               <font>

                  <Font name="Rockwell Extra Bold" size="24.0" />

               </font>

            </Button>

            <Button fx:id="Profile_ID" layoutX="345.0" layoutY="272.0" mnemonicParsing="false"
onAction="#Profilebtn" prefHeight="60.0" prefWidth="108.0" text="PROFILE">

               <font>

                  <Font name="Rockwell Extra Bold" size="16.0" />

               </font>

               <effect>

                  <InnerShadow height="146.5" radius="99.875" width="255.0">

                     <color>

                        <Color red="0.1725490242242813" green="0.35686275362968445"
blue="0.7882353067398071" />

                     </color>

                  </InnerShadow>

               </effect>

            </Button>

            <Button fx:id="Back_ID" layoutX="146.0" layoutY="272.0" mnemonicParsing="false"
onAction="#Backbtn" prefHeight="60.0" prefWidth="108.0" text="BACK">

               <font>

                  <Font name="Rockwell Extra Bold" size="21.0" />

               </font>

               <effect>

                  <InnerShadow height="39.0" radius="73.0" width="255.0">

                     <color>

                        <Color red="0.7631579041481018" green="0.18372319638729095"
blue="0.18372319638729095" />

                     </color>

                  </InnerShadow>

               </effect>

            </Button>

            <Button fx:id="X_ID" layoutX="537.0" layoutY="14.0" mnemonicParsing="false"
onAction="#xbtn" prefHeight="51.0" prefWidth="49.0" text="X">

               <font>
```

```
                <Font name="System Bold" size="24.0" />

            </font>

            <effect>

                <InnerShadow choke="0.7" height="36.0" radius="17.5" width="36.0">

                    <color>

                        <Color red="0.24313725531101227" green="0.14509804546833038"
blue="0.6313725709915161" />

                    </color>

                </InnerShadow>

            </effect>

        </Button>

        <ImageView fitHeight="60.0" fitWidth="61.0" layoutX="369.0" layoutY="216.0"
pickOnBounds="true" preserveRatio="true">

            <image>

                <Image url="@Images/astronaut1.png" />

            </image>

        </ImageView>

        <ImageView fitHeight="60.0" fitWidth="69.0" layoutX="170.0" layoutY="212.0"
pickOnBounds="true" preserveRatio="true">

            <image>

                <Image url="@Images/astronaut2.png" />

            </image>

        </ImageView>

        <ImageView fitHeight="42.0" fitWidth="61.0" layoutX="274.0" layoutY="98.0"
pickOnBounds="true" preserveRatio="true">

            <image>

                <Image url="@Images/astronaut3.png" />

            </image>

        </ImageView>

        <ImageView fitHeight="123.0" fitWidth="114.0" layoutX="342.0" layoutY="20.0"
pickOnBounds="true" preserveRatio="true">

            <image>

                <Image url="@Images/,moon.png" />

            </image>

        </ImageView>

        <Text fill="WHITE" layoutX="188.0" layoutY="78.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="MAIN MENU">

            <font>

                <Font name="Segoe Print Bold" size="35.0" />

            </font>
```

```
            </Text>
            <Button fx:id="howtoplay" layoutX="500.0" layoutY="354.0" mnemonicParsing="false"
onAction="#howtoplayBtn" prefHeight="32.0" prefWidth="37.0" text="???" textAlignment="CENTER">
               <font>
                  <Font name="System Bold" size="13.0" />
               </font>
               <effect>
                  <Lighting>
                     <bumpInput>
                        <Shadow />
                     </bumpInput>
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="0.9725490212440491" green="0.8627451062202454"
blue="0.8627451062202454" />
                           </color>
                        </Light.Distant>
                     </light>
                  </Lighting>
               </effect>
            </Button>
            <Button fx:id="Setting_ID" layoutX="543.0" layoutY="354.0" mnemonicParsing="false"
onAction="#Settingbtn" prefHeight="32.0" prefWidth="37.0" text="⚙">
               <font>
                  <Font name="System Bold" size="15.0" />
               </font>
               <effect>
                  <Lighting specularConstant="0.62" specularExponent="35.63">
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="1.0" green="0.8156862854957581" blue="0.6666666865348816" />
                           </color>
                        </Light.Distant>
                     </light>
                     <bumpInput>
                        <Shadow />
```

```
                    </bumpInput>
                </Lighting>
            </effect>
        </Button>
        <ImageView fitHeight="97.0" fitWidth="114.0" layoutX="59.0" layoutY="30.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@Images/fireball.png" />
            </image>
        </ImageView>
        <ImageView fitHeight="97.0" fitWidth="114.0" layoutX="69.0" layoutY="40.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@Images/fireball.png" />
            </image>
        </ImageView>
        <ImageView fitHeight="55.0" fitWidth="49.0" layoutX="479.0" layoutY="119.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@Images/star_1.png" />
            </image>
        </ImageView>
        <ImageView fitHeight="42.0" fitWidth="37.0" layoutX="261.0" layoutY="272.0"
pickOnBounds="true" preserveRatio="true">
            <image>
                <Image url="@Images/star_1.png" />
            </image>
        </ImageView>
        <ImageView fitHeight="51.0" fitWidth="49.0" layoutX="57.0" layoutY="148.0"
pickOnBounds="true" preserveRatio="true" rotate="107.4">
            <image>
                <Image url="@Images/star_1.png" />
            </image>
        </ImageView>
      </children>
    </Pane>
  </children>
</AnchorPane>
```

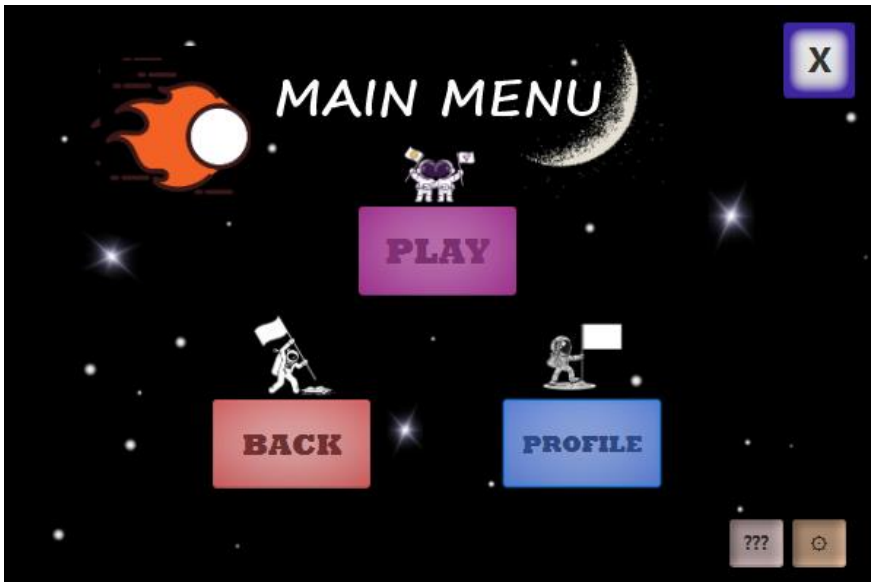
MainMenuLoginController.java

```
package javafxaassignment;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class MainMenuLoginController implements Initializable {

    @FXML
    private AnchorPane Mainmenu_ID;
    @FXML
    private Button Back_ID;
    @FXML
    private Button Profile_ID;
    @FXML
    private Button Play_ID;
    @FXML
    private Button Setting_ID;
    @FXML
    private Button X_ID;

    Stage stage;
    @FXML
    private Button howtoplay;
    @FXML
```

```
private Button pausee;

@Override
public void initialize(URL url, ResourceBundle rb) {

}

@FXML
private void Backbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainScreen.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
}

@FXML
private void Playbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}

@FXML
private void Settingbtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("settings.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}

@FXML
```

```
    private void xbtn(ActionEvent event) {
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);
        alert.setTitle("DIEEE");
        alert.setHeaderText("You're about to die!");
        alert.setContentText("Do you want to die before saving?:");

        if(alert.showAndWait().get()== ButtonType.OK)
        {
        stage = (Stage) Mainmenu_ID.getScene().getWindow();
        stage.close();
    }

    @FXML
    private void Profilebtn(ActionEvent event) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("profile.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
    }

    @FXML
    private void howtoplayBtn(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("HowToPlay.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
    }
}
```

Screenshot



Explanation

Like its predecessor this stage allow user to navigate the application. It doesn't contain any special function or methods. It borrows most of the function from the main screen stage. As such it acts as hub for the application. Compare to its counterpart it requires passing the login page to access this stage. This also give this page access to the profile stage only accessible via this stage.

Profile Screen

Source Code

backButtonController.java

```
package javafxaassignment;


import javafx.fxml.FXML;

import javafx.fxml.Initializable;

import javafx.scene.control.Button;

import javafx.event.ActionEvent;

import java.net.URL;

import java.util.ResourceBundle;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.control.TextField;

import javafx.stage.Stage;


public class ProfileController implements Initializable {


    @FXML

    private Button backProfile;


    @FXML

    private TextField OnePlayerRecord;


    @FXML

    private TextField TwoPlayerRecord;



    public void handleButtonAction(ActionEvent actionEvent) {
```

```
        //add action for button
        System.out.println("Back Button Clicked");
        backProfile.setText("Returns to Main Menu");
    }



    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {


    }


    @FXML
    private void backProfile(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }


}
```

Profile.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.effect.Bloom?>
<?import javafx.scene.effect.InnerShadow?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
```

```
<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.paint.LinearGradient?>

<?import javafx.scene.paint.Stop?>

<?import javafx.scene.shape.Ellipse?>

<?import javafx.scene.shape.Line?>

<?import javafx.scene.shape.Polygon?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<Pane style="-fx-background-color: #000000;" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.ProfileController">

   <children>

      <BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-
Infinity" prefHeight="400.0" prefWidth="600.0">

         <left>

            <Pane prefHeight="400.0" prefWidth="604.0" BorderPane.alignment="CENTER">

               <children>

                  <ImageView fitHeight="413.0" fitWidth="675.0" layoutX="-75.0" pickOnBounds="true"
preserveRatio="true">

                     <image>

                        <Image url="@Images/background.png" />

                     </image>

                  </ImageView>

                  <Button fx:id="backProfile" layoutX="14.0" layoutY="14.0" mnemonicParsing="false"
onAction="#backProfile" prefHeight="51.0" prefWidth="49.0" text="&lt;">

                     <font>

                        <Font name="System Bold" size="24.0" />

                     </font>

                     <effect>

                        <InnerShadow choke="0.7" height="36.0" radius="17.5" width="36.0">

                           <color>
```

```
                              <Color red="0.24313725531101227" green="0.14509804546833038"
blue="0.6313725709915161" />

                           </color>

                        </InnerShadow>

                     </effect>

                  </Button>

                  <Text fill="WHITE" layoutX="177.0" layoutY="54.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="PROFILE" textAlignment="CENTER" wrappingWidth="243.70556640625">

                     <font>

                        <Font name="Segoe Print Bold" size="40.0" />

                     </font>

                  </Text>

                  <Line endX="332.0" layoutX="268.0" layoutY="93.0" startX="-268.0" stroke="WHITE"
strokeWidth="2.0" />

                  <Line endX="-191.0" endY="250.99996948242188" layoutX="491.0" layoutY="149.0"
startX="-191.0" startY="-55.999969482421875" stroke="WHITE" strokeWidth="2.0" />

                  <Text fill="WHITE" layoutX="63.0" layoutY="139.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="User Details" underline="true">

                     <font>

                        <Font name="BB Text Bold Italic" size="32.0" />

                     </font>

                  </Text>

                  <Text fill="WHITE" layoutX="394.0" layoutY="139.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Records" underline="true">

                     <font>

                        <Font name="BB Text Bold Italic" size="32.0" />

                     </font>

                  </Text>

                  <Text fill="WHITE" layoutX="94.0" layoutY="196.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Username">

                     <font>

                        <Font name="BB Text Regular Italic" size="23.0" />

                     </font>

                  </Text>
```

```
                  <Text fill="WHITE" layoutX="119.0" layoutY="295.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Email">
                     <font>
                        <Font name="BB Text Regular Italic" size="23.0" />
                     </font>
                  </Text>
                  <TextField layoutX="68.0" layoutY="302.0" opacity="0.45" prefHeight="37.0"
prefWidth="160.0" />
                  <Text fill="WHITE" layoutX="314.0" layoutY="223.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="ONE PLAYER" wrappingWidth="160.443359375">
                     <font>
                        <Font name="Forte" size="25.0" />
                     </font>
                  </Text>
                  <Text fill="WHITE" layoutX="314.0" layoutY="327.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="TWO PLAYERS" wrappingWidth="160.443359375">
                     <font>
                        <Font name="Forte" size="25.0" />
                     </font>
                  </Text>
                  <Polygon layoutX="522.0" layoutY="220.0" points="-48.0, 40.000030517578125, 50.0,
40.0, 0.0, -60.0" stroke="BLACK" strokeType="INSIDE">
                     <fill>
                        <LinearGradient endX="1.0" endY="1.0">
                           <stops>
                              <Stop>
                                 <color>
                                    <Color red="0.14059758186340332" green="0.04111842066049576"
blue="0.3552631437778473" opacity="0.8999999761581421" />
                                 </color>
                              </Stop>
                              <Stop offset="1.0">
                                 <color>
                                    <Color red="1.0" green="1.0" blue="1.0" />
```

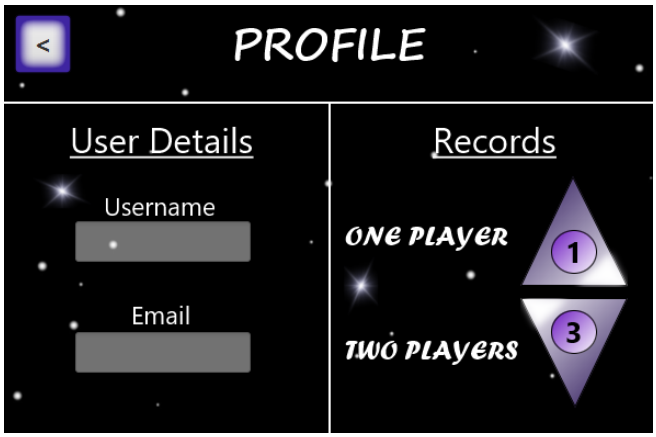
```
                                    </color>
                                 </Stop>
                              </stops>
                           </LinearGradient>
                        </fill>
                        <effect>
                           <Bloom threshold="0.8" />
                        </effect>
                     </Polygon>
                     <Polygon layoutX="522.0" layoutY="331.0" points="-48.0, 40.000030517578125, 50.0,
40.0, 0.0, -60.0" rotate="180.0" stroke="BLACK" strokeType="INSIDE">
                        <fill>
                           <LinearGradient endX="1.0" endY="1.0">
                              <stops>
                                 <Stop>
                                    <color>
                                       <Color red="0.14059758186340332" green="0.04111842066049576"
blue="0.3552631437778473" opacity="0.8999999761581421" />
                                    </color>
                                 </Stop>
                                 <Stop offset="1.0">
                                    <color>
                                       <Color red="1.0" green="1.0" blue="1.0" />
                                    </color>
                                 </Stop>
                              </stops>
                           </LinearGradient>
                        </fill>
                        <effect>
                           <Bloom threshold="0.82" />
                        </effect>
                     </Polygon>
```

```
                  <Ellipse layoutX="523.0" layoutY="229.0" radiusX="21.0" radiusY="20.0"
stroke="BLACK" strokeType="INSIDE">
                     <fill>
                        <LinearGradient endX="1.0" endY="1.0">
                           <stops>
                              <Stop>
                                 <color>
                                    <Color red="0.4156862795352936" green="0.08235294371843338"
blue="0.7490196228027344" />
                                 </color>
                              </Stop>
                              <Stop offset="1.0">
                                 <color>
                                    <Color red="1.0" green="1.0" blue="1.0" />
                                 </color>
                              </Stop>
                           </stops>
                        </LinearGradient>
                     </fill>
                  </Ellipse>
                  <Ellipse layoutX="523.0" layoutY="302.0" radiusX="21.0" radiusY="20.0"
stroke="BLACK" strokeType="INSIDE">
                     <fill>
                        <LinearGradient endX="1.0" endY="1.0">
                           <stops>
                              <Stop>
                                 <color>
                                    <Color red="0.4156862795352936" green="0.08235294371843338"
blue="0.7490196228027344" />
                                 </color>
                              </Stop>
                              <Stop offset="1.0">
                                 <color>
```

```
                                    <Color red="1.0" green="1.0" blue="1.0" />
                                </color>
                            </Stop>
                        </stops>
                    </LinearGradient>
                </fill>
            </Ellipse>
            <Text layoutX="515.0" layoutY="239.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="1" textAlignment="CENTER" wrappingWidth="16.46875">
                <font>
                    <Font name="System Bold" size="27.0" />
                </font>
            </Text>
            <Text layoutX="515.0" layoutY="312.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="3" textAlignment="CENTER" wrappingWidth="16.46875">
                <font>
                    <Font name="System Bold" size="27.0" />
                </font>
            </Text>
            <ImageView fitHeight="75.0" fitWidth="70.0" layoutX="486.0" layoutY="2.0"
pickOnBounds="true" preserveRatio="true" rotate="108.4">
                <image>
                    <Image url="@Images/star_1.png" />
                </image>
            </ImageView>
            <ImageView fitHeight="54.0" fitWidth="45.0" layoutX="307.0" layoutY="233.0"
pickOnBounds="true" preserveRatio="true">
                <image>
                    <Image url="@Images/star_1.png" />
                </image>
            </ImageView>
            <ImageView fitHeight="54.0" fitWidth="45.0" layoutX="35.0" layoutY="146.0"
pickOnBounds="true" preserveRatio="true" rotate="90.0">
                <image>
```

```
                        <Image url="@Images/star_1.png" />
                    </image>
                </ImageView>
                <TextField layoutX="68.0" layoutY="201.0" opacity="0.45" prefHeight="37.0"
prefWidth="160.0" />
            </children>
        </Pane>
    </left>
</BorderPane>
</children>
</Pane>
```

Screenshot



Explanation

The profile.fxml file defines the layout and structure of the User Profile page in the JavaFX application. I use ‘AnchorPane’ as the root container which allows me to have precise positioning of its children elements. Several labels' components have been used here to display text such as “USER PROFILE”, “USER DETAILS”, “RECORDS”, etc. “USER DETAILS” AND “RECORDS” both have their own respective colors. The button at the top left corner of the page is linked to an action method in the controller file that handles the interaction. In the “RECORDS” side of the page, there is “ONE PLAYER” and “TWO PLAYER” text with their respective boxes. These boxes show the past battle wins depending on the game mode and will constantly update depending on the player wins.

ProfileController Explanation

```
@FXML
private Button backButton;

@FXML
private TextField OnePlayerRecord;

@FXML
private TextField TwoPlayerRecord;
```

1. Fields:

- backButton: A ‘button’ object that triggers an action to return to the Main Menu.
- OnePlayerRecord: A ‘text’ object that displays the past battle wins in Player vs Bots mode.
- TwoPlayerRecord: A ‘text’ object that displays the past battle wins in Player vs Player mode.

2. Event Handlers

```
public void handleButtonAction(ActionEvent actionEvent) {
    //add action for button
    System.out.println("Back Button Clicked");
    backButton.setText("Returns to Main Menu");
}
```

- HandleButtonAction: When the button is clicked, it directs back to the Main Menu.

OOP Concepts

1. Encapsulation – is demonstrated by keeping the UI elements (‘Button’, ‘TextField’) as private fields and providing public method to manipulate these fields. This ensures that the internal state of the controller is not directly accessible from the outside, protecting the integrity of the data.

```
@FXML
private Button backButton;

@FXML
private TextField OnePlayerRecord;

@FXML
private TextField TwoPlayerRecord;
```

How To Play Screen

Source Code

HowToPlayController.java

```
package javafxaassignment;


import javafx.fxml.FXML;

import javafx.scene.control.Button;

import javafx.scene.control.TextArea;

import javafx.stage.Stage;


import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import javafx.event.ActionEvent;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;


public class HowToPlayController {

    @FXML

    private TextArea instructionsTextArea;




    private Stage stage;

    @FXML

    private Button backhtp;


    public void setStage(Stage stage) {

        this.stage = stage;
```

```
}

private void initialize() {
    loadInstructions();
}

private void loadInstructions() {
    String filePath = "/demo/main/instructions.txt";
    StringBuilder instructions = new StringBuilder();

    try (InputStream inputStream = getClass().getResourceAsStream(filePath);
         BufferedReader br = new BufferedReader(new InputStreamReader(inputStream))) {
        String line;
        while ((line = br.readLine()) != null) {
            instructions.append(line).append("\n");
        }
    } catch (IOException e) {
        instructions.append("Error loading instructions.");
    }

    instructionsTextArea.setText(instructions.toString());
}

private void handleBack() {
    stage.close();
}

@FXML
private void backhtps(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
```

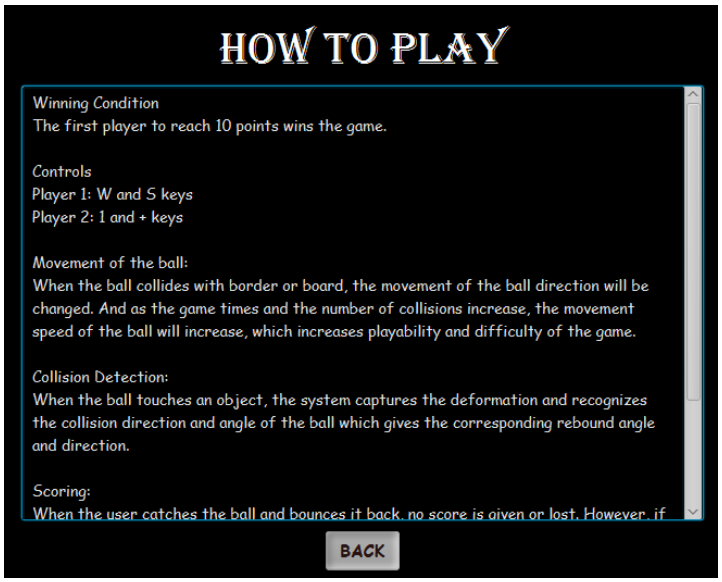
```
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
    }
}
```

HowToPlay.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TextArea?>
<?import javafx.scene.effect.InnerShadow?>
<?import javafx.scene.effect.Light.Distant?>
<?import javafx.scene.effect.Lighting?>
<?import javafx.scene.effect.Shadow?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.paint.Color?>
<?import javafx.scene.text.Font?>

<VBox alignment="CENTER" spacing="10" style="-fx-background-color: black;"
xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxaassignment.HowToPlayController">
    <padding>
        <Insets bottom="20" left="20" right="20" top="20" />
    </padding>
    <Label style="-fx-font-size: 42px; -fx-text-fill: white; -fx-font-family: 'Algerian';" text="How
to Play" />
    <TextArea fx:id="instructionsTextArea" editable="false" opacity="0.89" prefHeight="400"
prefWidth="300" style="-fx-font-size: 14px; -fx-text-fill: white; -fx-control-inner-background: black;
-fx-font-family: 'Comic Sans MS';" text="Winning Condition&#10;The first player to reach 10 points
wins the game.&#10;&#10;Controls&#10;Player 1: W and S keys&#10;Player 2: 1 and +
keys&#10;&#10;Movement of the ball:&#10;When the ball collides with border or board, the movement of
the ball direction will be changed. And as the game times and the number of collisions increase, the
movement speed of the ball will increase, which increases playability and difficulty of the
```

```
game.&#10;&#10;Collision Detection:&#10;When the ball touches an object, the system captures the deformation and recognizes the collision direction and angle of the ball which gives the corresponding rebound angle and direction.&#10;&#10;Scoring:&#10;When the user catches the ball and bounces it back, no score is given or lost. However, if the ball collides with the edge of the screen, the attacker gains a point.&#10;&#10;Played Continuously:&#10;The game does not end until the limit of the number of rounds is covered. When the user does not receive the ball, the turn is over, but the game continues. And the ball is returned to the center line." wrapText="true">
      <font>
         <Font name="AdobeKaitiStd-Regular" size="13.0" />
      </font></TextArea>
    <Button fx:id="backhtp" onAction="#backhtps" prefHeight="35.0" prefWidth="68.0" text="BACK" textFill="#250303">
      <effect>
         <InnerShadow>
            <input>
               <Lighting>
                  <bumpInput>
                     <Shadow />
                  </bumpInput>
                  <light>
                     <Light.Distant />
                  </light>
               </Lighting>
            </input>
            <color>
               <Color red="0.7764706015586853" green="0.7764706015586853" blue="0.7764706015586853" />
            </color>
         </InnerShadow>
      </effect>
      <font>
         <Font name="Comic Sans MS Bold" size="16.0" />
      </font></Button>
</VBox>
```

Screenshot



Explanation

I used an InputStream to open the instructions file, then wrapped it in an InputStreamReader to read the file's characters, and finally used a BufferedReader to read the text line by line. I created a String variable named line to hold each line of text as it is read. Each line is then appended to a StringBuilder object called instructions. After reading the entire file, I set the content of the instructions object into the instructionsTextArea to display it on the screen using a text area. There is a back button to redirect you back to the previous page.

Game Screen

Source Code

Temporary Game Screen:

Game.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.effect.Light.Distant?>

<?import javafx.scene.effect.Lighting?>

<?import javafx.scene.effect.Shadow?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.BorderPane?>

<?import javafx.scene.layout.Pane?>

<?import javafx.scene.paint.Color?>

<?import javafx.scene.text.Font?>


<BorderPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.GameController">

   <top>

        <Pane fx:id="gamePane" prefHeight="402.0" prefWidth="600.0" style="-fx-background-color:
#000000;" BorderPane.alignment="CENTER">

         <children>

            <ImageView fitHeight="423.0" fitWidth="600.0" pickOnBounds="true" preserveRatio="true">

               <image>

                  <Image url="@Images/gamepong.png" />

               </image>

            </ImageView>

              <Button fx:id="settingsButton" layoutX="8.0" layoutY="6.0" onAction="#handleSettings"
prefHeight="42.0" prefWidth="44.0" text="⚙">

               <font>

                  <Font name="System Bold" size="20.0" />

               </font>

               <effect>

                  <Lighting>

                     <bumpInput>

                        <Shadow />
```

```
                        </bumpInput>
                        <light>
                           <Light.Distant>
                              <color>
                                 <Color red="1.0" green="0.8156862854957581" blue="0.6666666865348816" />
                              </color>
                           </Light.Distant>
                        </light>
                     </Lighting>
                  </effect></Button>
               <Button   fx:id="exitButton"   layoutX="519.0"   layoutY="6.0"   onAction="#handleExit"
prefHeight="49.0" prefWidth="67.0" text="EXIT">
                  <effect>
                     <Lighting>
                        <bumpInput>
                           <Shadow />
                        </bumpInput>
                        <light>
                           <Light.Distant>
                              <color>
                                 <Color       red="0.9098039269447327"       green="0.32549020648002625"
blue="0.32549020648002625" />
                              </color>
                           </Light.Distant>
                        </light>
                     </Lighting>
                  </effect>
                  <font>
                     <Font name="System Bold" size="18.0" />
                  </font></Button>
               <Button   fx:id="pauseButton"   layoutX="441.0"   layoutY="7.0"   onAction="#handlePause"
prefHeight="49.0" prefWidth="67.0" text="PAUSE">
                  <font>
                     <Font name="System Bold" size="14.0" />
                  </font>
                  <effect>
                     <Lighting>
                        <bumpInput>
```

```
                        <Shadow />
                     </bumpInput>
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="0.47058823704719543" green="0.49803921580314636" blue="1.0" />
                           </color>
                        </Light.Distant>
                     </light>
                  </Lighting>
               </effect></Button>
            <Button   fx:id="gameresult"   layoutX="256.0"   layoutY="176.0"   mnemonicParsing="false"
onAction="#handleResult" prefHeight="50.0" prefWidth="89.0" text="RESULT">
               <effect>
                  <Lighting>
                     <bumpInput>
                        <Shadow />
                     </bumpInput>
                     <light>
                        <Light.Distant>
                           <color>
                              <Color red="0.16078431904315948" green="1.0" blue="0.21960784494876862" />
                           </color>
                        </Light.Distant>
                     </light>
                  </Lighting>
               </effect>
               <font>
                  <Font name="System Bold" size="18.0" />
               </font>
            </Button>
         </children></Pane>
   </top>
</BorderPane>
```

GameController.java

```
package javafxaassignment;
```

```
import java.net.URL;
import java.util.ResourceBundle;
import javafx.animation.AnimationTimer;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.input.KeyCode;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.stage.Stage;


public class GameController implements Initializable {

    @FXML
    private Pane gamePane;
    @FXML
    private Button pauseButton;
    @FXML
    private Button exitButton;
    @FXML
    private Button settingsButton;

    private GamePane customGamePane;
    private Paddle paddle1;
    private Paddle paddle2;
    private Ball ball;
    private Score score;
    private Label gameOverLabel;
```

```
private boolean isGameOver = false;
private boolean isPaused = false;

private AnimationTimer gameLoop;
@FXML
private Button gameresult;

@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}

public void initialize() {
    gamePane.setPrefSize(GamePane.GAME_WIDTH, GamePane.GAME_HEIGHT);
    gamePane.setStyle("-fx-background-color: black;");

    newPaddles();
    newBall();
    score = new Score(GamePane.GAME_WIDTH, GamePane.GAME_HEIGHT);
    gamePane.getChildren().addAll(paddle1, paddle2, ball, score);
    score.draw();

    gameOverLabel = new Label();
    gameOverLabel.setTextFill(Color.WHITE);
    gameOverLabel.setFont(new Font("Consolas", 30));
    gameOverLabel.setVisible(false);
    gamePane.getChildren().add(gameOverLabel);

    pauseButton.setOnAction(e -> togglePause());
    exitButton.setOnAction(e -> exitGame());

    gameLoop = new AnimationTimer() {
        private long lastUpdate = 0;

        @Override
        public void handle(long now) {
```

```
            if (!isGameOver && !isPaused && now - lastUpdate >= 16_000_000) {
                move();
                checkCollision();
                score.draw();
                lastUpdate = now;
            }
        }
    };

    gameLoop.start();

    gamePane.setFocusTraversable(true);
    gamePane.setOnKeyPressed(e -> handleKeyPressed(e.getCode()));
    gamePane.setOnKeyReleased(e -> handleKeyReleased(e.getCode()));
    gamePane.requestFocus();
}

private void handleKeyPressed(KeyCode code) {
    if (!isPaused) {
        switch (code) {
            case W:
                paddle1.setYDirection(-20);
                break;
            case S:
                paddle1.setYDirection(20);
                break;
            case UP:
                paddle2.setYDirection(-20);
                break;
            case DOWN:
                paddle2.setYDirection(20);
                break;
            default:
                break;
        }
    }
}
```

```
private void handleKeyReleased(KeyCode code) {
    if (!isPaused) {
        switch (code) {
            case W:
            case S:
                paddle1.setYDirection(0);
                break;
            case UP:
            case DOWN:
                paddle2.setYDirection(0);
                break;
            default:
                break;
        }
    }
}

private void newBall() {
    ball = new Ball((GamePane.GAME_WIDTH / 2) - (GamePane.BALL_DIAMETER / 2),
                    (GamePane.GAME_HEIGHT / 2) - (GamePane.BALL_DIAMETER / 2),
                    GamePane.BALL_DIAMETER, GamePane.BALL_DIAMETER);
}

private void newPaddles() {
    paddle1 = new Paddle(0, (GamePane.GAME_HEIGHT / 2) - (GamePane.PADDLE_HEIGHT / 2),
                         GamePane.PADDLE_WIDTH, GamePane.PADDLE_HEIGHT, 1);
    paddle2 = new Paddle(GamePane.GAME_WIDTH - GamePane.PADDLE_WIDTH,
                         (GamePane.GAME_HEIGHT / 2) - (GamePane.PADDLE_HEIGHT / 2),
                         GamePane.PADDLE_WIDTH, GamePane.PADDLE_HEIGHT, 2);
}

private void move() {
    paddle1.move();
    paddle2.move();
    ball.move();
}
```

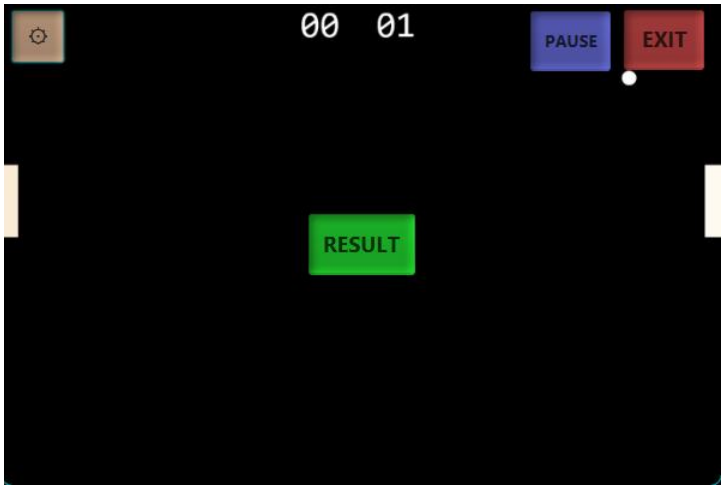
```
    private void checkCollision() {
        double ballCenterY = ball.getCenterY();
        double gameHeightMinusBallDiameter = GamePane.GAME_HEIGHT - GamePane.BALL_DIAMETER;

        if (ballCenterY <= 0 || ballCenterY >= gameHeightMinusBallDiameter) {
            ball.setYDirection(-ball.yVelocity);
        }

        if (ball.getBoundsInParent().intersects(paddle1.getBoundsInParent()) ||
                ball.getBoundsInParent().intersects(paddle2.getBoundsInParent())) {

            boolean isPaddle1 = ball.getBoundsInParent().intersects(paddle1.getBoundsInParent());
            ball.setXDirection(isPaddle1 ? Math.abs(ball.xVelocity) : -Math.abs(ball.xVelocity));

            ball.setXDirection(Math.min(Math.abs(ball.xVelocity) + 1, 10) * (isPaddle1 ? 1 : -1));
            ball.setYDirection(ball.yVelocity > 0 ?
                    Math.min(ball.yVelocity + 1, 10) :
                    Math.max(ball.yVelocity - 1, -10));
        }

        double ballCenterX = ball.getCenterX();
        if (ballCenterX <= 0) {
            score.player2++;
            resetGame();
        } else if (ballCenterX >= GamePane.GAME_WIDTH - GamePane.BALL_DIAMETER) {
            score.player1++;
            resetGame();
        }

        if (score.player1 >= 10 || score.player2 >= 10) {
            endGame();
        }

        paddle1.setY(Math.max(0, Math.min(paddle1.getY(), GamePane.GAME_HEIGHT -
GamePane.PADDLE_HEIGHT)));
        paddle2.setY(Math.max(0, Math.min(paddle2.getY(), GamePane.GAME_HEIGHT -
GamePane.PADDLE_HEIGHT)));
```

```
}

private void endGame() {
    isGameOver = true;
    gameOverLabel.setText("Game Over! " + (score.player1 >= 10 ? "Player 1" : "Player 2") + " Wins!");
    gameOverLabel.setVisible(true);
    gameOverLabel.setLayoutX((GamePane.GAME_WIDTH - gameOverLabel.getWidth()) / 2);
    gameOverLabel.setLayoutY((GamePane.GAME_HEIGHT - gameOverLabel.getHeight()) / 2);
}

private void resetGame() {
    if (!isGameOver) {
        gamePane.getChildren().removeAll(ball, paddle1, paddle2);
        newBall();
        newPaddles();
        gamePane.getChildren().addAll(paddle1, paddle2, ball);
    }
}

private void togglePause() {
    isPaused = !isPaused;
    pauseButton.setText(isPaused ? "Resume" : "Pause");
    gamePane.requestFocus();
}

private void exitGame() {
    Stage stage = (Stage) gamePane.getScene().getWindow();
    stage.close();
}

@FXML
private void handlePause(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("pausemenu.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
```

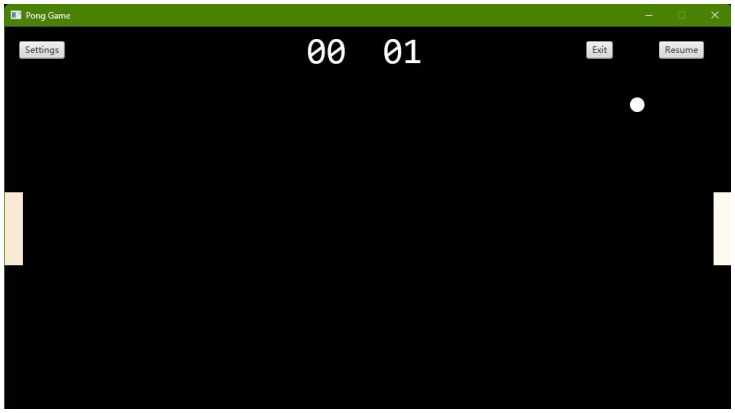
```
        stage.show();
    }

    @FXML
    private void handleExit(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void handleSettings(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("settings_2.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void handleResult(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Results.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Screenshot

Temporary Game Screen:



Game Screen:



Game Class

GamePane: Displays paddles, balls, and scores. Handles game logic, collision detection, ball rebound, score settlement, pause and end game. And uses the AnimationTimer to create game loops, ensuring that the game runs at about 60 frames per second. Prevents unstable behavior of game elements.

```
gameLoop = new AnimationTimer() {                    // Initialize and start
    2 usages
    private long lastUpdate = 0;

    @Override
    public void handle(long now) {
        // Update game state if the game is not over or paused, roughly 60 FPS
        if (!isGameOver && !isPaused && now - lastUpdate >= 16_000_000) {
            move();
            checkCollision();
            score.draw();
            lastUpdate = now;
        }
    }
};
```

Use **AnimationTimer** for game loops. Dynamically update the game state (e.g., ball movement, collapse checking) and update game elements (corresponding to turn changes). Ensure that the game runs smoothly and interacts with user input.

isPaused: Boolean function confirming whether the game is paused or not. When the user presses the pause button, the game loops to respond to it.

lastUpdate: Ensure that the game is updated at the same frequency for all updates (e.g., every 16 milliseconds for 60 FPS).

```
    gameLoop.start();

    setFocusTraversable(true);
    setOnKeyPressed(e -> handleKeyPressed(e.getCode()));
    setOnKeyReleased(e -> handleKeyReleased(e.getCode()));
    requestFocus();
}
```

setFocusTraversable(true), Allows the game window to receive keyboard events to bridge keyboard correspondence and paddle movement.

handleKeyPressed , handles the key pressed event, setting the direction of paddle movement.

handleKeyReleased, handles the key released event and stops the paddle movement.

```
private void handleKeyPressed(KeyCode code) {
    if (!isPaused) {
        switch (code) {
            case W -> paddle1.setYDirection(-20); // Move paddle1 up
            case S -> paddle1.setYDirection(20);  // Move paddle1 down
            case UP -> paddle2.setYDirection(-20); // Move paddle2 up
            case DOWN -> paddle2.setYDirection(20); // Move paddle2 down
        }
    }
}
```

```
public void newBall() {
    ball = new Ball( x: (GAME_WIDTH / 2) - (BALL_DIAMETER / 2), y: (GAME_HEIGHT / 2) - (BALL_DIAMETER / 2), BALL_DIAMETER, BALL_DIAMETER);
}

                                        // Initialize paddles at starting positions
2 usages
public void newPaddles() {
    paddle1 = new Paddle( x: 0,  y: (GAME_HEIGHT / 2) - (PADDLE_HEIGHT / 2), PADDLE_WIDTH, PADDLE_HEIGHT,  id: 1);
    paddle2 = new Paddle( x: GAME_WIDTH - PADDLE_WIDTH,  y: (GAME_HEIGHT / 2) - (PADDLE_HEIGHT / 2), PADDLE_WIDTH, PADDLE_HEIGHT,  id: 2);
}
```

Set the x, y coordinates of the ball to **(GAME_WIDTH / 2) - (BALL_DIAMETER / 2)** to center it horizontally and center it vertically to keep the game equitable.

The y-coordinate of both paddles is set to **(GAME_HEIGHT / 2) - (PADDLE_HEIGHT / 2)** to center them vertically.

```
public void checkCollision() {
    double ballCenterY = ball.getCenterY();
    double gameHeightMinusBallDiameter = GAME_HEIGHT - BALL_DIAMETER;
                                                        // collision with top and bottom
    if (ballCenterY <= 0 || ballCenterY >= gameHeightMinusBallDiameter) {
        ball.setYDirection(-ball.yVelocity);
    }


                                                        // collision with paddles
    if (ball.getBoundsInParent().intersects(paddle1.getBoundsInParent()) ||
            ball.getBoundsInParent().intersects(paddle2.getBoundsInParent())) {

        boolean isPaddle1 = ball.getBoundsInParent().intersects(paddle1.getBoundsInParent());
        ball.setXDirection(isPaddle1 ? Math.abs(ball.xVelocity) : -Math.abs(ball.xVelocity));
```

checkCollision method: check whether the ball intersects with the paddle. If an interaction is detected it is a collision. If collision then the direction and speed of the ball is adjusted.

Reverses the direction of the ball when it hits the top or bottom boundaries. Set the ball to change its vertical direction while maintaining horizontal motion.

```
                                                    // speed increase
    ball.setXDirection(Math.min(Math.abs(ball.xVelocity) + 1, 10) * (isPaddle1 ? 1 : -1));
    ball.setYDirection(ball.yVelocity > 0 ?
            Math.min(ball.yVelocity + 1, 10) :
            Math.max(ball.yVelocity - 1, -10));
```

Limit the speed increase to prevent the ball from moving too rapidly. If the ball is moving too fast, the difficulty of the game will increase. The player's experience may decrease.

When the ball intersects the paddle, reverse their horizontal direction (**xVelocity**) and increase their speed slightly. Along with that the vertical direction (**yVelocity**) is also changed to add randomness and playability.

```
    double ballCenterX = ball.getCenterX();
    if (ballCenterX <= 0) {
        score.player2++;
        resetGame();                                    // Reset game after player 2 scores
    } else if (ballCenterX >= GAME_WIDTH - BALL_DIAMETER) {
        score.player1++;
        resetGame();                                    // Reset game after player 1 scores
    }
                                                        // Check for game over
    if (score.player1 >= 10 || score.player2 >= 10) {
        endGame();
    }
                                                        // Ensure paddles stay within game boundaries
    paddle1.setY(Math.max(0, Math.min(paddle1.getY(), GAME_HEIGHT - PADDLE_HEIGHT)));
    paddle2.setY(Math.max(0, Math.min(paddle2.getY(), GAME_HEIGHT - PADDLE_HEIGHT)));
}
```

Math.max and **Math.min** : Limit the position of the paddles to make sure they don't go beyond the top and bottom boundaries. Paddle1's Y-coordinate is constrained to 0 (the top boundary), and GAME_HEIGHT - PADDLE_HEIGHT (bottom boundary).

```
private void endGame() {
    isGameOver = true;
    gameOverLabel.setText("Game Over! " + (score.player1 >= 10 ? "Player 1" : "Player 2") + " Wins!");
    gameOverLabel.setVisible(true);
    gameOverLabel.setLayoutX((GAME_WIDTH - gameOverLabel.getWidth()) / 2);
    gameOverLabel.setLayoutY((GAME_HEIGHT - gameOverLabel.getHeight()) / 2);
}
```

While game is in progress, the system dynamically checks weather the ball crosses left or right border. If it does, the corresponding player's score is increased and reset game. If the player's score reaches 10-point limit, the game ends.

```
private void resetGame() {
    if (!isGameOver) {
        getChildren().removeAll(ball, paddle1, paddle2);
        newBall();
        newPaddles();
        getChildren().addAll(paddle1, paddle2, ball);
    }
}
```

At the start of a new turn, the game components are all restored to their default positions to keep the game equitable.

```
private void togglePause() {
    isPaused = !isPaused;
    pauseButton.setText(isPaused ? "Resume" : "Pause");
    requestFocus(); // Ensure the GamePane has focus to capture key events
    if (!isPaused) {
        setOnKeyPressed(e -> handleKeyPressed(e.getCode()));
        setOnKeyReleased(e -> handleKeyReleased(e.getCode()));
    }
}
```

```
private void exitGame() {
    Stage stage = (Stage) getScene().getWindow();
    stage.close();
}
```

Ball Class

Allows the ball to interact with game boundaries and paddles, handling collisions and changes in direction.

```
int initialSpeed = 2;                    // Initial speed of the ball
```

initialSpeed Sets the initial speed of the ball. Ensure the ball moves at the same speed at the beginning of each game or round to ensure the fairness of the game.

```
public Ball(double x, double y, double width, double height) {
    super(x, y, v2: width / 2, Color.WHITE);
    int randomXDirection = Math.random() < 0.5 ? -1 : 1;
    setXDirection(randomXDirection * initialSpeed);
    int randomYDirection = Math.random() < 0.5 ? -1 : 1;
    setYDirection(randomYDirection * initialSpeed);
}
```

super(x, y, width / 2, Color.WHITE), Because the Circle class in JavaFX uses radius instead of diameter. To set the radius of the ball, divide the width by two.

Math.random() < 0.5 ? -1 : 1, Randomize the initial direction of the ball. If the random number is less than 0.5, set the direction to -1 (left or up). Otherwise, set it to 1 (right or down). Randomizing the initial direction of the ball creates uncertainty when playing, which can increase the playability of the game and thus enhance the player's playing experience.

```
public void setXDirection(int randomXDirection) { xVelocity = randomXDirection; }

// Set the y direction velocity
3 usages
public void setYDirection(int randomYDirection) { yVelocity = randomYDirection; }

// Move the ball based on its velocity
1 usage
public void move() {
    setCenterX(getCenterX() + xVelocity); // Update the x position of the ball
    setCenterY(getCenterY() + yVelocity); // Update the y position of the ball
}
```

The **move** method: Updates the position of the ball: matches the current speed (xVelocity and yVelocity) to the current position of the ball (getCenterX and getCenterY).

Score Class

Displays and updates the score.

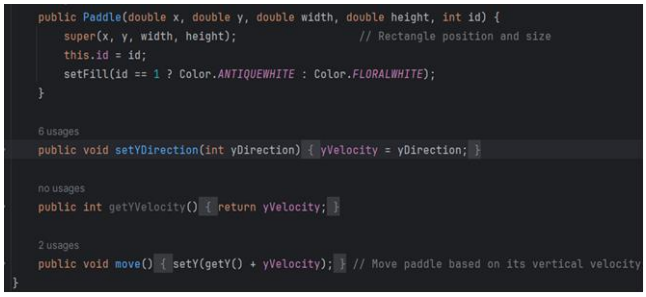
```
    public void draw() {
        GraphicsContext gc = getGraphicsContext2D();
        gc.clearRect( v: 0,  v1: 0, getWidth(), getHeight());   // Clear previous drawing
        gc.setFill(Color.WHITE);
        gc.setFont(new Font( s: "Consolas",  v: 50));          // Set scores font

        gc.fillText(String.format("%02d", player1),  v: (GAME_WIDTH / 2) - 85,  v1: 50);
        gc.fillText(String.format("%02d", player2),  v: (GAME_WIDTH / 2) + 20,  v1: 50);

        // center line
        gc.strokeLine( v: GAME_WIDTH / 2,  v1: 0,  v2: GAME_WIDTH / 2, GAME_HEIGHT);
    }
}
```

draw: This method removes the previous drawing. It then uses GraphicsContext to draw the current score.

GraphicsContext: Method for drawing shapes, text and images on the canvas.

Paddle Class

Responds to key presses with corresponding movements.



```
public Paddle(double x, double y, double width, double height, int id) {
    super(x, y, width, height);                  // Rectangle position and size
    this.id = id;
    setFill(id == 1 ? Color.ANTIQUEWHITE : Color.FLORALWHITE);
}

6 usages
public void setYDirection(int yDirection) { yVelocity = yDirection; }

no usages
public int getYVelocity() { return yVelocity; }

2 usages
public void move() { setY(getY() + yVelocity); } // Move paddle based on its vertical velocity
}
```

id: used to distinguish between player 1 (id = 1) and player 2 (id = 2).

setFill(id == 1 ? Color.ANTIQUEWHITE : Color.FLORALWHITE);: associate paddle's id with the player's id. Player 1's racket is **Color.ANTIQUEWHITE**, player 2's racket is **Color.FLORALWHITE**.

PongGame Class

Initializes the game window and layout. Specifies that the game window frame has focus to capture paddle movement and game events.

```
public class PongGame extends Application {
    @Override
    public void start(Stage primaryStage) {
        GamePane gamePane = new GamePane();
        Scene scene = new Scene(gamePane, GamePane.GAME_WIDTH, GamePane.GAME_HEIGHT);
        primaryStage.setTitle("Pong Game");
        primaryStage.setScene(scene);
        primaryStage.setResizable(false);          // Prevent game window from being resizable
        primaryStage.show();

        gamePane.requestFocus();                   // Only focus game pane to ensure it receives key events
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

gamePane.requestFocus();: When the gamePane is running, it immediately tracks the focus to make sure it receives key events, and continuously captures player input to better correspond to racket movement. Users don't need to click on the game area to start game interactions, optimizing the user gaming experience.

Results

Results screens have 2 results, each for a different main menu. Results is for those who sign up and login to play the game, while another one is Results1 for those who didn't do any sign up and login to the game.

Source Code

FXML

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.text.Font?>

<?import javafx.scene.text.Text?>


<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: white;"
xmlns="http://javafx.com/javafx/22" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="javafxaassignment.ResultsController">

   <children>

      <ImageView fx:id="backgroundImage" fitHeight="380.0" fitWidth="581.0" layoutX="10.0"
layoutY="10.0">

         <image>

            <Image url="@Images/360_F_289726988_cF9wzIATeSAOBxCg23NqnLeaKVfXWvdy.jpg" />

         </image>

      </ImageView>

      <Text fill="WHITE" layoutX="203.0" layoutY="68.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="RESULTS" textAlignment="CENTER">

         <font>

            <Font name="Gill Sans Ultra Bold" size="36.0" />

         </font>

      </Text>

      <Text fx:id="winnerText" fill="WHITE" layoutX="169.0" layoutY="116.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="PLAYER X WINS!">

         <font>

            <Font name="Gill Sans Ultra Bold Condensed" size="36.0" />

         </font>

      </Text>
```

```
      <Text fill="WHITE" layoutX="232.0" layoutY="280.0" strokeType="OUTSIDE" strokeWidth="0.0"
text="Congratulations!" textAlignment="CENTER">

         <font>

            <Font name="Forte" size="18.0" />

         </font>

      </Text>

      <Button fx:id="playAgainButton" layoutX="169.0" layoutY="318.0" mnemonicParsing="false"
onAction="#handlePlayAgainAction" text="Play Again" textAlignment="CENTER">

         <font>

            <Font name="Berlin Sans FB" size="18.0" />

         </font>

      </Button>

      <Button fx:id="mainMenuButton" layoutX="322.0" layoutY="318.0" mnemonicParsing="false"
onAction="#Menubtn" text="Main Menu" textAlignment="CENTER">

         <font>

            <Font name="Berlin Sans FB" size="18.0" />

         </font>

      </Button>

      <ImageView fx:id="trophyImage" fitHeight="115.0" fitWidth="104.0" layoutX="250.0"
layoutY="143.0">

         <image>

            <Image url="@Images/winnertrophy.png" />

         </image>

      </ImageView>

   </children>

</AnchorPane>
```

Results Controller (MainMenuLogin)

```
package javafxaassignment;


import java.net.URL;

import java.util.ResourceBundle;

import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.fxml.Initializable;

import javafx.scene.Node;

import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ResultsController implements Initializable {

    @FXML
    private Text winnerText;

    @FXML
    private ImageView backgroundImage;

    @FXML
    private ImageView trophyImage;

    @FXML
    private Button playAgainButton;

    @FXML
    private Button mainMenuButton;

    @Override
    public void initialize(URL url, ResourceBundle rb) {

    }

    @FXML
    private void handlePlayAgainAction(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
```

```
    }

    @FXML
    private void Menubtn(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    public void setWinner(String winner) {
        winnerText.setText(winner + " WINS!");
    }

    private void displayImage(){
        Image trophy = new Image(getClass().getResourceAsStream("winnertrophy.png"));
        trophyImage.setImage(trophy);
    }

    private void displayImage2(){
        Image background = new Image(getClass().getResourceAsStream("spacebackground.jpg"));
        backgroundImage.setImage(background);
    }
}
```

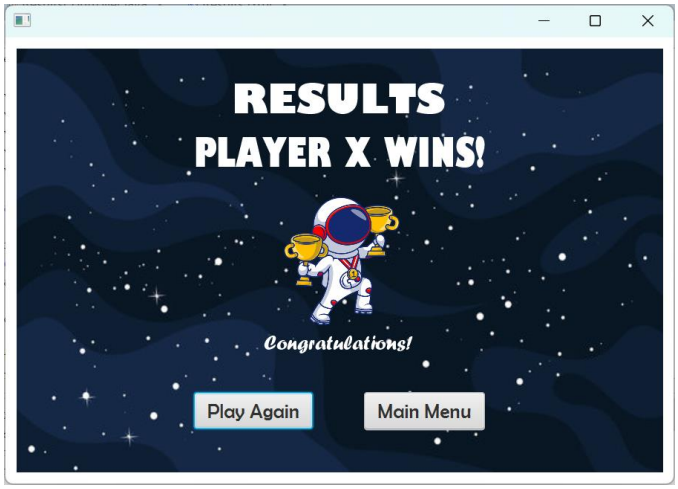
Results1Controller (GuestMainMenu)

Same as results controller, the only difference is in the ‘Menubtn’.

```
    @FXML
    private void Menubtn(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("MainMenu.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
```

```
        stage.show();
    }
```

Screenshot



Explanation of ResultsController (MainMenuLogin)

Key Components

1. Package Declaration and Imports

   The 'package javafxaassignment' declares the class is part of the 'javafxaassignment' package. The import statements bring in necessary classes from the JavaFX library and the standard library for handling events, loading FXML files, managing scenes, controlling UI elements, and working with images.

```
package javafxaassignment;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.text.Text;
import javafx.stage.Stage;
```

2. Class Declarations

   Declares the class named 'ResultsController' and this class implements the 'Initializable' interface, requiring the implementation of the initialize method.

```
public class ResultsController implements Initializable {
```

3. Private Fields

   The private fields correspond to the UI components defined in the FXML file. The '@FXML' annotation tells the JavaFX runtime to inject the corresponding components from the FXML file into these fields.

```
@FXML
private Text winnerText;

@FXML
private ImageView backgroundImage;

@FXML
private ImageView trophyImage;

@FXML
private Button playAgainButton;

@FXML
private Button mainMenuButton;
```

4. Initialize Method

   The 'initialize' method is called automatically after the FXML file has been loaded. Any initialization code for the controller can be placed here. Currently, it does not contain any setup logic.

```
@Override
public void initialize(URL url, ResourceBundle rb) {

}
```

5. Event Handlers

   'handlePlayAgainAction' Method

```
@FXML
private void handlePlayAgainAction(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

   This method is called when the "Play Again' button is clicked.

   - Parent root = FXMLLoader.load(getClass().getResource("Game.fxml")); loads the Game.fxml file.
   - Scene scene = new Scene(root); creates a new scene with the loaded FXML root.
   - Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow(); retrieves the current stage (window) from the event source.

- stage.setScene(scene); sets the new scene on the stage.
- stage.show(); displays the stage with the new scene.

'Menubtn' Method

```
@FXML
private void Menubtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

This method handles transitioning to the main menu when the "Main Menu" button is clicked. It follows the same logic as handlePlayAgainAction, but it loads the MainMenuLogin.fxml file to switch to the main menu screen for user that login to the game.

6. Utility Methods

```
public void setWinner(String winner) {
    winnerText.setText(winner + " WINS!");
}

private void displayImage(){
    Image trophy = new Image(getClass().getResourceAsStream("winnertrophy.png"));
    trophyImage.setImage(trophy);
}

private void displayImage2(){
    Image background = new Image(getClass().getResourceAsStream("spacebackground.jpg"));
    backgroundImage.setImage(background);
}
```

- 'setWinner' Method

  This method sets the text of the winnerText field to display the winning player's name. It can be called externally to update the winner text dynamically.
- 'displayImage' and 'displayImage2' Method

  The 'displayImage' loads the image file winnertrophy.png and sets it to the trophyImage field, while the 'displayImage2' loads the image file spacebackground.jpg and sets it to the backgroundImage field.

OOP Concepts

1. Encapsulation

   Encapsulation is the mechanism of restricting access to some of an object's components and protecting the object's internal state from unauthorized access or modification. This is done by using access modifiers such as private, protected, and public.

   - Private Fields: The fields 'winnerText', 'backgroundImage', 'trophyImage', 'playAgainButton', and 'mainMenuButton' are declared private to prevent direct access

from outside the class. Instead, these fields are manipulated through public or private methods.

```
@FXML
private Text winnerText;

@FXML
private ImageView backgroundImage;

@FXML
private ImageView trophyImage;

@FXML
private Button playAgainButton;

@FXML
private Button mainMenuButton;
```

- Public Methods: The 'setWinner' method is public to allow other classes to set the winner's name. This method provides controlled access to modify the 'winnerText' field.

```
public void setWinner(String winner) {
    winnerText.setText(winner + " WINS!");
}
```

- Private Methods: The 'displayImage' and 'displayImage2' methods are private because they are utility methods used only within the 'ResultsController' class.

```
private void displayImage(){
    Image trophy = new Image(getClass().getResourceAsStream("winnertrophy.png"));
    trophyImage.setImage(trophy);
}

private void displayImage2(){
    Image background = new Image(getClass().getResourceAsStream("spacebackground.jpg"));
    backgroundImage.setImage(background);
}
```

2. Inheritance

Inheritance is the mechanism by which one class inherits the properties and behavior of another class. In this case, 'ResultsController' inherits from the 'Initializable' interface. By implementing the Initializable interface, ResultsController is required to define the initialize method. This method is called to initialize a controller after its root element has been completely processed.

```
public class ResultsController implements Initializable {
```

```
@Override
public void initialize(URL url, ResourceBundle rb) {

}
```

3. Abstraction

Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. The methods 'handlePlayAgainAction' and 'Menubtn'

abstract the details of how the scene transition happens. They provide a simple interface for handling button clicks without exposing the underlying complexity.

```
@FXML
private void handlePlayAgainAction(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

```
@FXML
private void Menubtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

4. Composition

   Composition is a design principle where a class is composed of one or more objects from other classes. It represents a "has-a" relationship. The ‘ResultsController’ class is composed of several JavaFX components, such as ‘Text’, ‘ImageView’, and ‘Button’. These components are part of the class and are used to build the UI and handle interactions.

```
@FXML
private Text winnerText;

@FXML
private ImageView backgroundImage;

@FXML
private ImageView trophyImage;

@FXML
private Button playAgainButton;

@FXML
private Button mainMenuButton;
```

Explanation of Results1Controller (GuestMainMenu)

Key Components

Results1Controller shares the same components with the results controller, the only difference is that the event handler of the main menu button. Results1Controller is to handle the set screen to the guest main menu.

1. Event Handling Methods

```
@FXML
private void handlePlayAgainAction(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

'handlePlayAgainAction': This method handles the action when the "Play Again" button is clicked. It loads the "Game1.fxml" scene and sets it on the current stage.

```
@FXML
private void Menubtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenu.fxml"));
    Scene scene = new Scene(parent: root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

'Menubtn': This method handles the action when the "Main Menu" button is clicked. It loads the "MainMenu.fxml" scene and sets it on to the guest main menu if the user didn't sign up and login at all.

OOP Concepts

1. Encapsulation

   The controller's internal state (fields like 'winnerText', 'backgroundImage', 'trophyImage', 'playAgainButton', and 'mainMenuButton') is kept private, and interaction with these fields is controlled through public methods.

```
@FXML
private Text winnerText;

@FXML
private ImageView backgroundImage;

@FXML
private ImageView trophyImage;

@FXML
private Button playAgainButton;

@FXML
private Button mainMenuButton;
```

2. Inheritance

   The ResultsController1 class implements the Initializable interface, inheriting the contract to implement the initialize method.

```
public class ResultsController implements Initializable {
```

```
@Override
public void initialize(URL url, ResourceBundle rb) {

}
```

3. Abstraction

Abstraction is the concept of hiding the complex implementation details and showing only the necessary features of an object. The methods 'handlePlayAgainAction' and 'Menubtn' abstract the details of how the scene transition happens. They provide a simple interface for handling button clicks without exposing the underlying complexity.

```
@FXML
private void handlePlayAgainAction(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(scene);
    stage.show();
}
```

```
@FXML
private void Menubtn(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenu.fxml"));
    Scene scene = new Scene(parent: root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

4. Composition

The controller uses composition to include various JavaFX UI components (Text, ImageView, Button) as part of its structure.

```
@FXML
private Text winnerText;

@FXML
private ImageView backgroundImage;

@FXML
private ImageView trophyImage;

@FXML
private Button playAgainButton;

@FXML
private Button mainMenuButton;
```

Pause

For Pause, there are 2 pauses each for different main menu, there's one for those with login credentials and those that continue as guest without login. So, pausemenu is for those with login credentials while pausemenu_1 is for those that do not login.

Source Code

pausemenu.fxml

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.control.Button?>

<?import javafx.scene.image.Image?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.Pane?>


<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.PauseMenuController">

   <Pane layoutY="1.0" prefHeight="400.0" prefWidth="601.0" style="-fx-background-color: #000000;" />

   <ImageView fitHeight="312.0" fitWidth="879.0" pickOnBounds="true" preserveRatio="true">

      <image>

         <Image url="@Images/gamepong.png" />

      </image>

   </ImageView>

   <ImageView fitHeight="326.0" fitWidth="327.0" layoutX="137.0" layoutY="38.0" pickOnBounds="true"
preserveRatio="true">

      <image>

         <Image url="@Images/paused_screen.png" />

      </image>

   </ImageView>

    <Button fx:id="resume" layoutX="184.0" layoutY="252.0" onAction="#handleResume" opacity="0.0"
prefHeight="56.0" prefWidth="222.0" />

    <Button fx:id="restart" layoutX="186.0" layoutY="144.0" onAction="#handleRestart" opacity="0.0"
prefHeight="82.0" prefWidth="85.0" />

    <Button fx:id="mainmenu" layoutX="318.0" layoutY="144.0" onAction="#handleMainMenu" opacity="0.0"
prefHeight="82.0" prefWidth="86.0" />

    <Button fx:id="settings" layoutX="150.0" layoutY="48.0" onAction="#handleSettings" opacity="0.0"
prefHeight="28.0" prefWidth="29.0" />

</AnchorPane>
```

PauseMenuController.java

```
package javafxaassignment;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class PauseMenuController {

    private Stage stage;
    private Scene mainMenuScene;
    private Scene settingsScene;

    @FXML
    private Button resume;
    @FXML
    private Button restart;
    @FXML
    private Button mainmenu;
    @FXML
    private Button settings;

    @FXML
    private void handleResume(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
```

```
    @FXML
    private void handleRestart(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Game.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }


    @FXML
    private void handleSettings(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("settings_4.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void handleMainMenu(ActionEvent event) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

pausemenu_1.fxml

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.image.Image?>
```

```
<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.Pane?>


<AnchorPane id="AnchorPane" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/22"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="javafxaassignment.PauseMenuController1">

   <Pane prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #000000;">

      <children>

         <ImageView fitHeight="352.0" fitWidth="600.0" pickOnBounds="true" preserveRatio="true">

            <image>

               <Image url="@Images/gamepong.png" />

            </image>

         </ImageView>

      </children>

   </Pane>

   <ImageView fitHeight="326.0" fitWidth="327.0" layoutX="137.0" layoutY="38.0" pickOnBounds="true"
preserveRatio="true">

      <image>

         <Image url="@Images/paused_screen.png" />

      </image>

   </ImageView>

    <Button fx:id="resume" layoutX="184.0" layoutY="252.0" onAction="#handleResume" opacity="0.0"
prefHeight="56.0" prefWidth="222.0" />

    <Button fx:id="restart" layoutX="186.0" layoutY="144.0" onAction="#handleRestart" opacity="0.0"
prefHeight="82.0" prefWidth="85.0" />

    <Button fx:id="mainmenu" layoutX="318.0" layoutY="144.0" onAction="#handleMainMenu" opacity="0.0"
prefHeight="82.0" prefWidth="86.0" />

    <Button fx:id="settings" layoutX="150.0" layoutY="48.0" onAction="#handleSettings" opacity="0.0"
prefHeight="28.0" prefWidth="29.0" />

</AnchorPane>
```

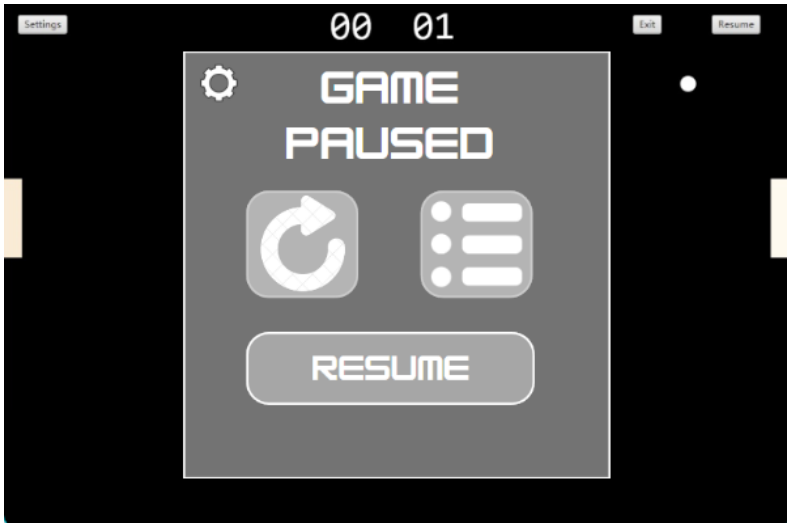
PauseMenuController1.java

```
package javafxaassignment;


import javafx.event.ActionEvent;

import javafx.fxml.FXML;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;
```

```
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class PauseMenuController1 {

    private Stage stage;
    private Scene mainMenuScene;
    private Scene settingsScene;

    @FXML
    private Button resume;
    @FXML
    private Button restart;
    @FXML
    private Button mainmenu;
    @FXML
    private Button settings;

    @FXML
    private void handleResume(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Game_1.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }

    @FXML
    private void handleRestart(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("Game_1.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
```

```
    }


    @FXML
    private void handleSettings(ActionEvent event) throws Exception
    {
        Parent root = FXMLLoader.load(getClass().getResource("settings_5.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }


    @FXML
    private void handleMainMenu(ActionEvent event) throws Exception{
        Parent root = FXMLLoader.load(getClass().getResource("MainMenu.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(scene);
        stage.show();
    }
}
```

Screenshot



Explanation

pausemenu.fxml

It is the FXML file that represents the UI for the pause menu in the game designed for users who have logged in.

Components:

ImageView: Displays an image (paused_screen.png) as the background.

Buttons:

- Resume with OnAction of #handleResume
- Restart with OnAction of #handleRestart
- Mainmenu with OnAction of #handleMainMenu
- Settings with OnAction of #handleSettings

The controller is PauseMenuController.java

PauseMenuController.java

This Java class serves as the controller for pausemenu.fxml. It manages the stage and scenes for navigation and also defines the method.

```
package javafxaassignment;

import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
```

The package is javafxaassignment. The rest are imports necessary for the codes to function.

```
public class PauseMenuController {

    @FXML
    private Button resume;
    @FXML
    private Button restart;
    @FXML
    private Button mainmenu;
    @FXML
    private Button settings;
```

The public class PauseMenuController is a class and under that the @FXML annotation is used for the resume, restart, mainmenu and settings buttons that show that these buttons are injected from the FXML file into the controller. These buttons allow users to interact with the pause

menu and trigger methods that handle resuming the game, restarting the game, opening the settings menu or returning to the main menu.

```
@FXML
private void handleResume(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

This is the event handler for handleResume(ActionEvent event) that resumes the game by loading Game.fxml then set the scene and show it.

```
@FXML
private void handleRestart(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

This is the event handler for handleRestart(ActionEvent event) that restarts the game by loading Game.fxml then set the scene and show it.

```
@FXML
private void handleSettings(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "settings_4.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

This is the event handler for handleSettings(ActionEvent event) that opens the settings menu by loading settings_4.fxml then set the scene and show it.

```
    @FXML
    private void handleMainMenu(ActionEvent event) throws Exception{
        Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
        Scene scene = new Scene(root);
        Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
        stage.setScene(value: scene);
        stage.show();
    }
}
```

This is the event handler for handleMainMenu(ActionEvent event) that takes the user to the main menu by loading MainMenuLogin.fxml then set the scene and show it. Then at last the } is the closing for the class.

pausemenu_1.fxml

This FXML file represents the UI for the pause menu designed for users who continue as guests (without login). All the components and design are the same as pausemenu.fxml, the only difference is that the controller for this fxml is PauseMenuController1.java

PauseMenuController1.java

This Java class serves as the controller for pausemenu_1.fxml. The imports and buttons are the same as PauseMenuController.java. The only difference here is the different scene for each event handler. The class is also different.

```
public class PauseMenuController1
```

```
@FXML
private void handleResume(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}

@FXML
private void handleRestart(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "Game.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}


@FXML
private void handleSettings(ActionEvent event) throws Exception
{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "settings_4.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}

@FXML
private void handleMainMenu(ActionEvent event) throws Exception{
    Parent root = FXMLLoader.load(location: getClass().getResource(name: "MainMenuLogin.fxml"));
    Scene scene = new Scene(root);
    Stage stage = (Stage)((Node) event.getSource()).getScene().getWindow();
    stage.setScene(value: scene);
    stage.show();
}
```

The event handler button actions - handleResume, handleRestart and handleSettings each use a different scene respectively which is Game_1.fxml,Game_1.fxml, settings_5.fxml and MainMenu.fxml which is all based on the user's login status.

Pause Menu Summary

These files and classes together provide two versions of a pause menu UI (pausemenu.fxml and pausemenu_1.fxml) and their respective controllers (PauseMenuController.java and PauseMenuController1.java). They handle navigation within a JavaFX application for logged-in and guest users and allow them to resume gameplay, restart the game, access settings or return to the main menu based on their actions.