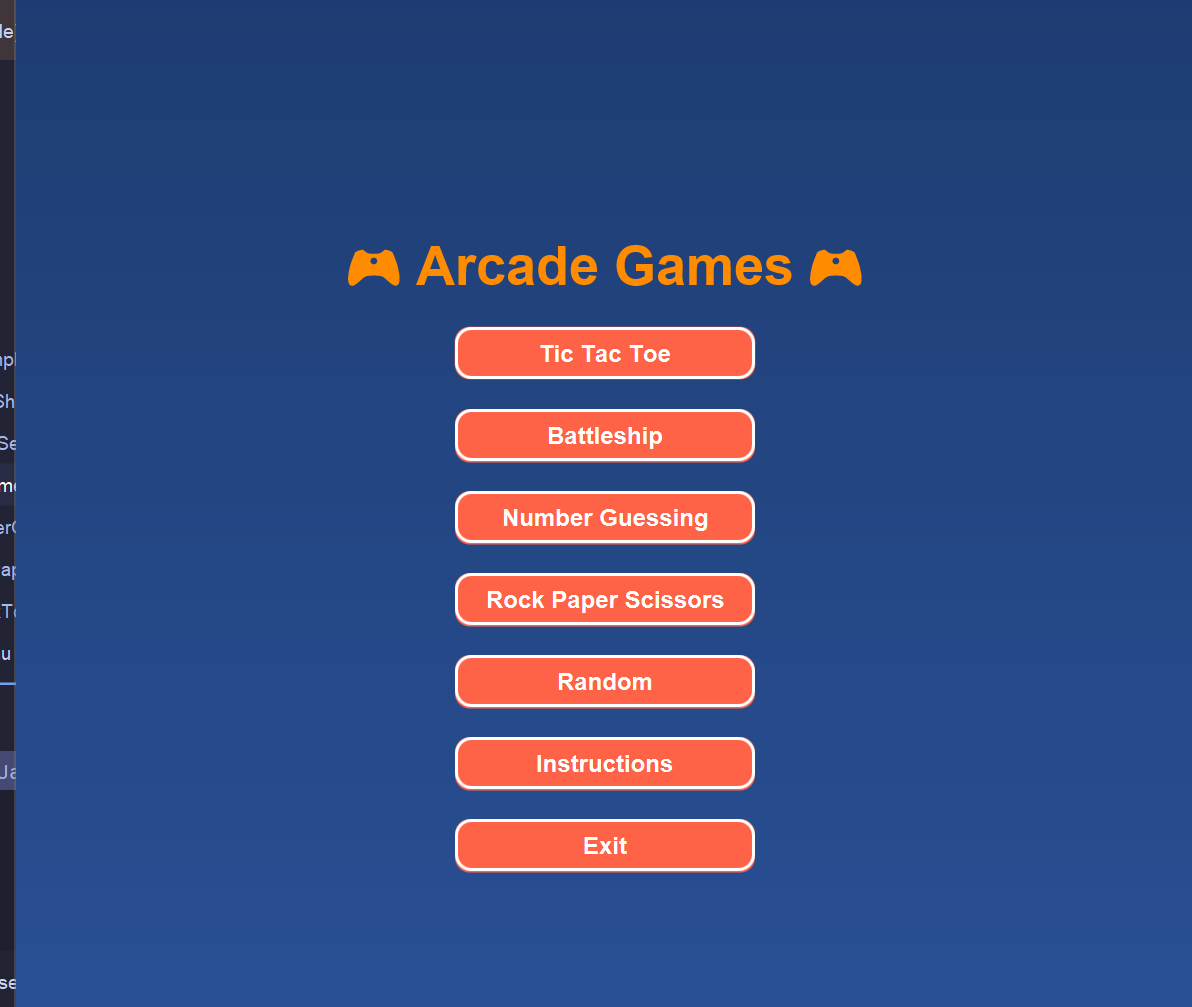
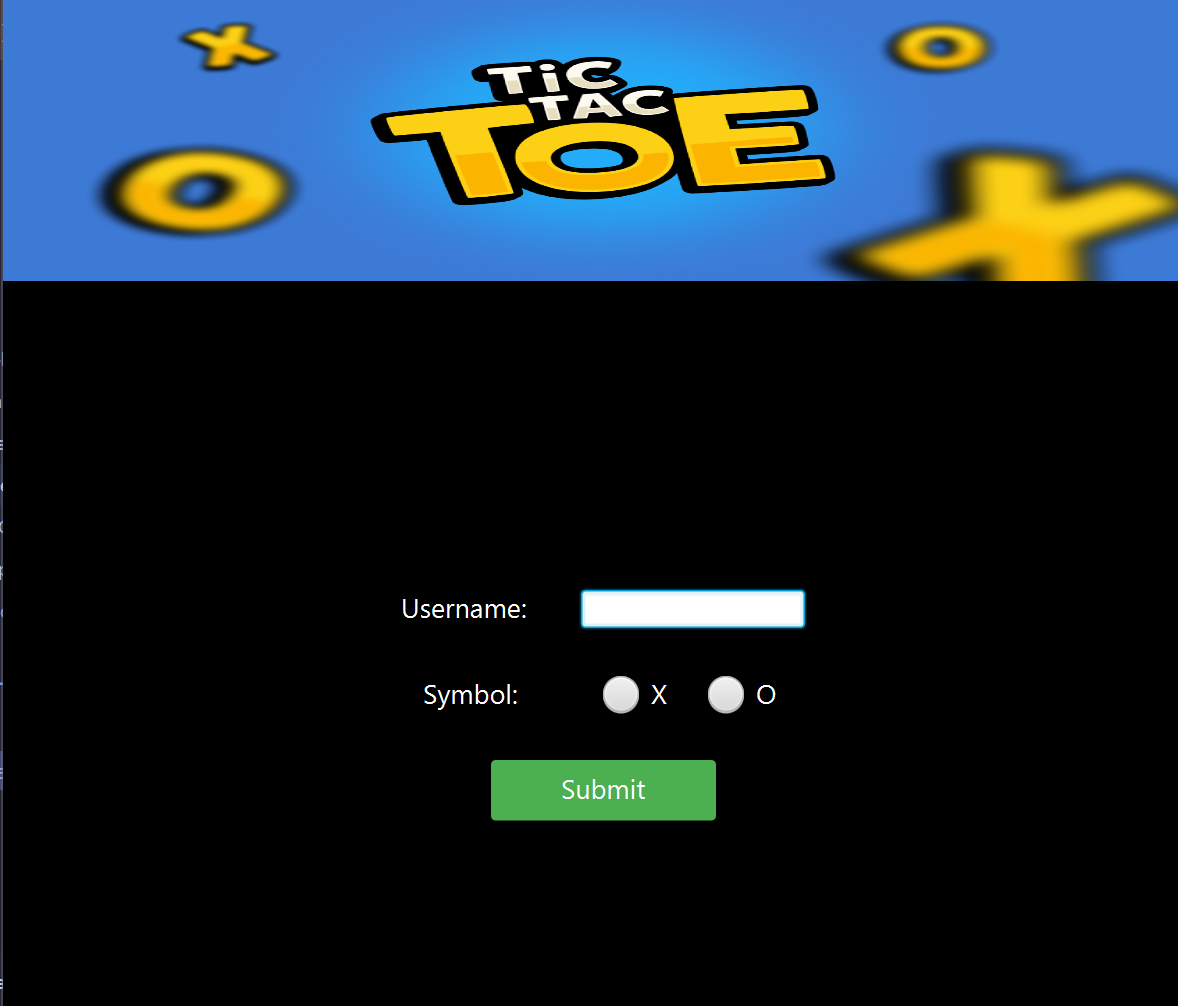
**GAME SELECTION MENU**



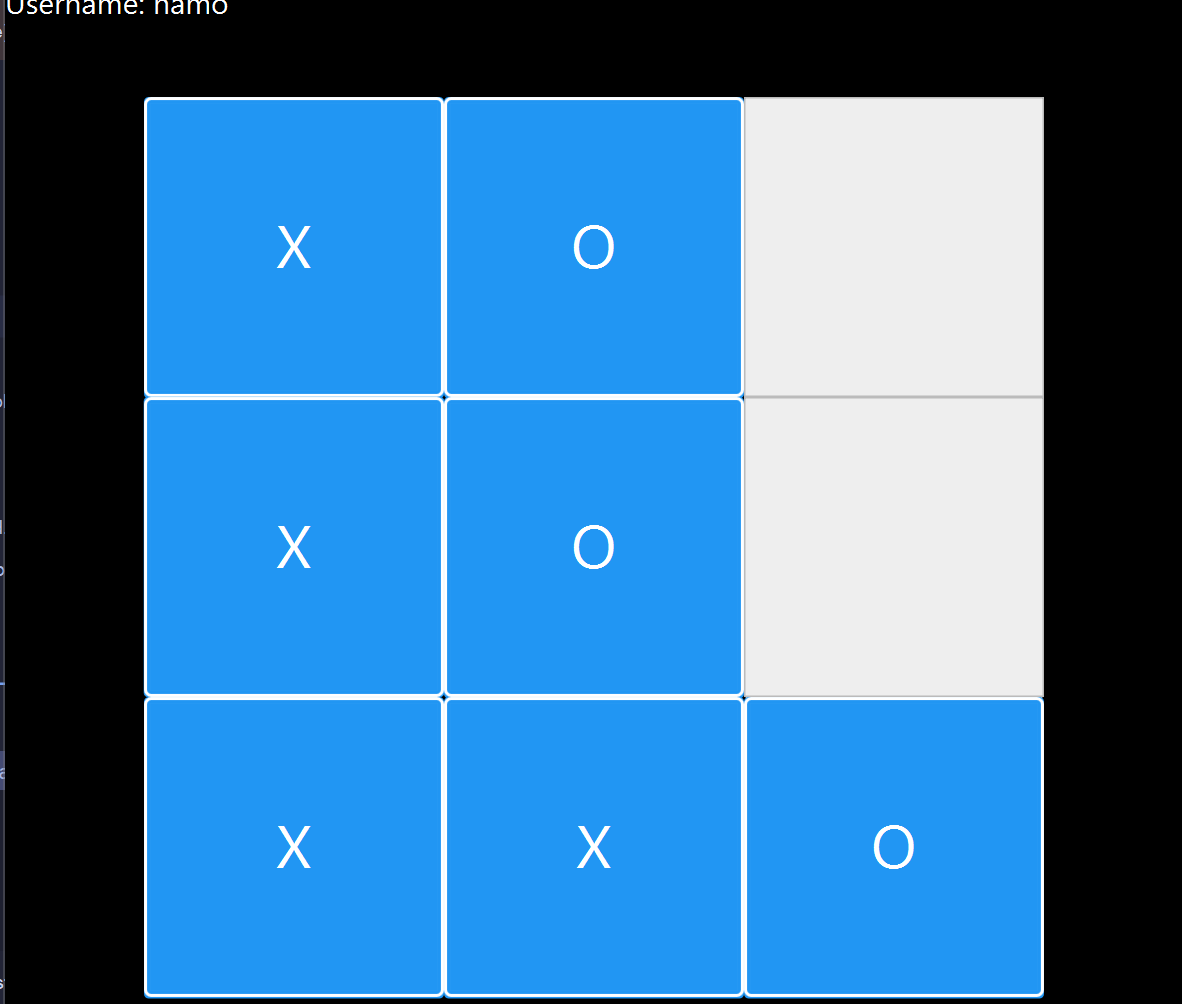
**CODE FOR GAME SELECTION MENU**

*package* com.example.semesterproject.GameSelectionMenu;  
  
*import* javafx.application.Application;  
*import* javafx.geometry.Insets;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.Button;  
*import* javafx.scene.layout.VBox;  
*import* javafx.scene.paint.Color;  
*import* javafx.scene.text.Font;  
*import* javafx.scene.text.FontWeight;  
*import* javafx.scene.text.Text;  
*import* javafx.stage.Stage;  
*import* com.example.semesterproject.TicTakToe.TicTacToeUI;  
*import* com.example.semesterproject.NumberGuessingGame.NumberGuessingGame;  
*import* com.example.semesterproject.RockPaperScissors.RockPaperScissors;  
*import* com.example.semesterproject.BattleShip.BattleShipUI;  
  
*import* java.util.Random;  
  
*public class* GameSelectionMenu *extends* Application {  
  
 @Override  
 *public void* start(Stage primaryStage) {  
  
 Text title = *new* Text("🎮 Arcade Games 🎮");  
 title.setFont(Font.*font*("Arial", FontWeight.BOLD, 36));  
 title.setFill(Color.DARKORANGE);  
  
  
 TicTacToeUI tiktaktoe = *new* TicTacToeUI();  
 NumberGuessingGame numberGuessingGame = *new* NumberGuessingGame();  
 BattleShipUI battleShipUI = *new* BattleShipUI();  
 RockPaperScissors rockPaperScissors = *new* RockPaperScissors();  
  
  
 Button ticTacToeButton = createGameButton("Tic Tac Toe");  
 ticTacToeButton.setOnAction(event -> tiktaktoe.start(primaryStage));  
  
 Button battleshipButton = createGameButton("Battleship");  
 battleshipButton.setOnAction(event -> battleShipUI.start(primaryStage));  
  
 Button numberGuessingButton = createGameButton("Number Guessing");  
 numberGuessingButton.setOnAction(event -> numberGuessingGame.start(primaryStage));  
  
 Button rockPaperScissorsButton = createGameButton("Rock Paper Scissors");  
 rockPaperScissorsButton.setOnAction(event -> rockPaperScissors.start(primaryStage));  
  
 Button randomGameButton = createGameButton("Random");  
 randomGameButton.setOnAction(event -> {  
 Random random = *new* Random();  
 *int* gameNumber = random.nextInt(4);  
 *if* (gameNumber == 0) {  
 tiktaktoe.start(primaryStage);  
 } *else if* (gameNumber == 1) {  
 battleShipUI.start(primaryStage);  
 } *else if* (gameNumber == 2) {  
 numberGuessingGame.start(primaryStage);  
 } *else* {  
 rockPaperScissors.start(primaryStage);  
 }  
 });  
  
 Button exitButton = createGameButton("Exit");  
 exitButton.setOnAction(event -> System.*exit*(0));  
  
  
 Button instructionsButton = createGameButton("Instructions");  
 instructionsButton.setOnAction(event -> showInstructions(primaryStage));  
  
  
 VBox vbox = *new* VBox(20, title, ticTacToeButton, battleshipButton, numberGuessingButton, rockPaperScissorsButton, randomGameButton, instructionsButton, exitButton);  
 vbox.setAlignment(Pos.CENTER);  
 vbox.setPadding(*new* Insets(20));  
 vbox.setStyle("-fx-background-color: linear-gradient(to bottom, #1e3c72, #2a5298);");  
  
  
 Scene scene = *new* Scene(vbox, 800, 800);  
 primaryStage.setWidth(800);  
 primaryStage.setHeight(800);  
 primaryStage.setResizable(*false*);  
  
 *// Stage setup* primaryStage.setTitle("Arcade Game Selection");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 *private* Button createGameButton(String buttonText) {  
 Button button = *new* Button(buttonText);  
 button.setFont(Font.*font*("Arial", FontWeight.BOLD, 16));  
 button.setTextFill(Color.WHITE);  
 button.setStyle("-fx-background-color: #ff6347; -fx-border-color: #ffffff; -fx-border-width: 2px; -fx-border-radius: 10px; -fx-background-radius: 10px;");  
 button.setPrefWidth(200);  
 *return* button;  
 }  
  
  
 *private void* showInstructions(Stage primaryStage) {  
 Text instructionTitle = *new* Text("Game Instructions");  
 instructionTitle.setFont(Font.*font*("Arial", FontWeight.BOLD, 36));  
 instructionTitle.setFill(Color.DARKORANGE);  
  
  
 Text instructionsText = *new* Text("Welcome to the Arcade!\n\n"  
 + "Choose a game and enjoy!\n\n"  
 + "1. Tic Tac Toe: A classic 3x3 grid game.\n"  
 + "2. Battleship: Sink your opponent's ships.\n"  
 + "3. Number Guessing: Guess the number between 1 and 100.\n"  
 + "4. Rock Paper Scissors: A simple hand game.\n\n"  
 + "Click 'Continue' to go back to the game selection menu.");  
  
 instructionsText.setFont(Font.*font*("Arial", 18));  
 instructionsText.setFill(Color.WHITE);  
  
 Button continueButton = createGameButton("Continue");  
 continueButton.setOnAction(event -> {  
 start(primaryStage);  
 });  
  
  
 VBox instructionLayout = *new* VBox(20, instructionTitle, instructionsText, continueButton);  
 instructionLayout.setAlignment(Pos.CENTER);  
 instructionLayout.setPadding(*new* Insets(20));  
 instructionLayout.setStyle("-fx-background-color: linear-gradient(to bottom, #1e3c72, #2a5298);");  
  
  
 Scene instructionsScene = *new* Scene(instructionLayout, 800, 800);  
 primaryStage.setScene(instructionsScene);  
 }  
  
 *public static void* main(String[] args) {  
 *launch*(args);  
 }  
}

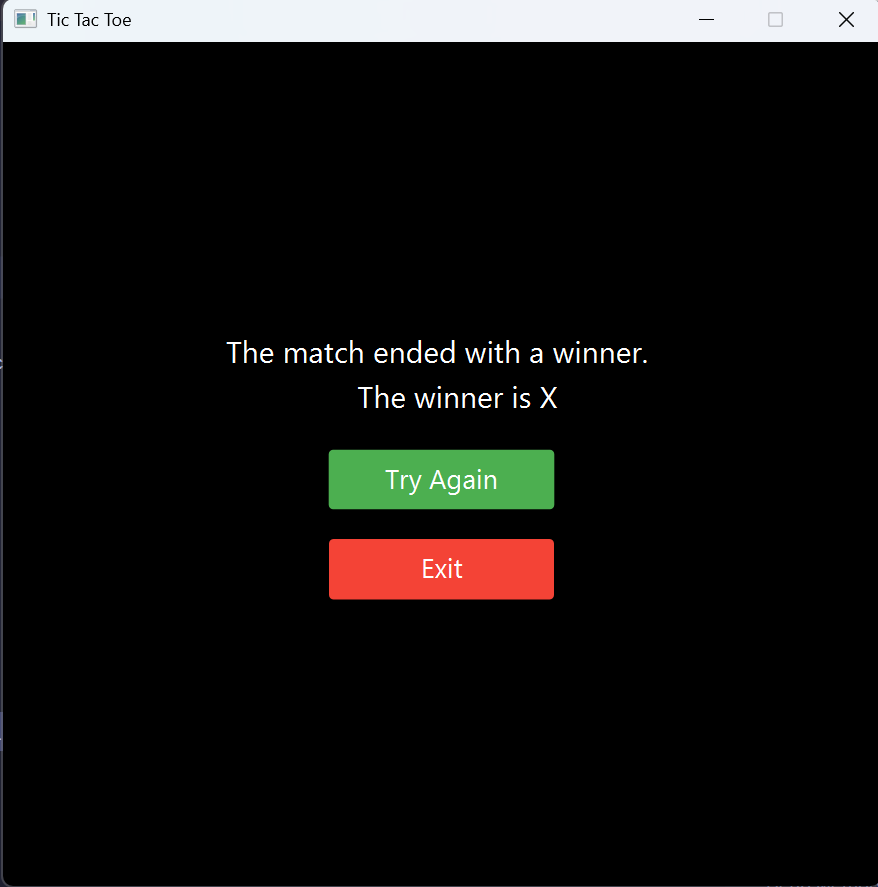
**TIC TAC TOE**



(Username window for tic tac toe game which inputs username from user).



(Main game window for Tic Tac Toe).



(Winner declaration window of game).

**Tic Tac Toe Game**

**Main Menu Window**  
Welcome to Tic Tac Toe! This window introduces the game and lets players enter their names before starting. It also includes a brief overview of how to play. A "Start Game" button launches the match when players are ready.

**Gameplay Window**  
The Tic Tac Toe game is played on a 3x3 grid where two players take turns marking their symbols (X or O). The interface highlights the current player’s turn and displays updates on the game’s progress. The goal is to align three symbols in a row, column, or diagonal. Simple controls ensure an engaging and competitive experience.

**Result Window**  
After the game ends, this window announces the winner or declares a draw if no one wins. It displays the match statistics and provides options to play again or return to the main menu.

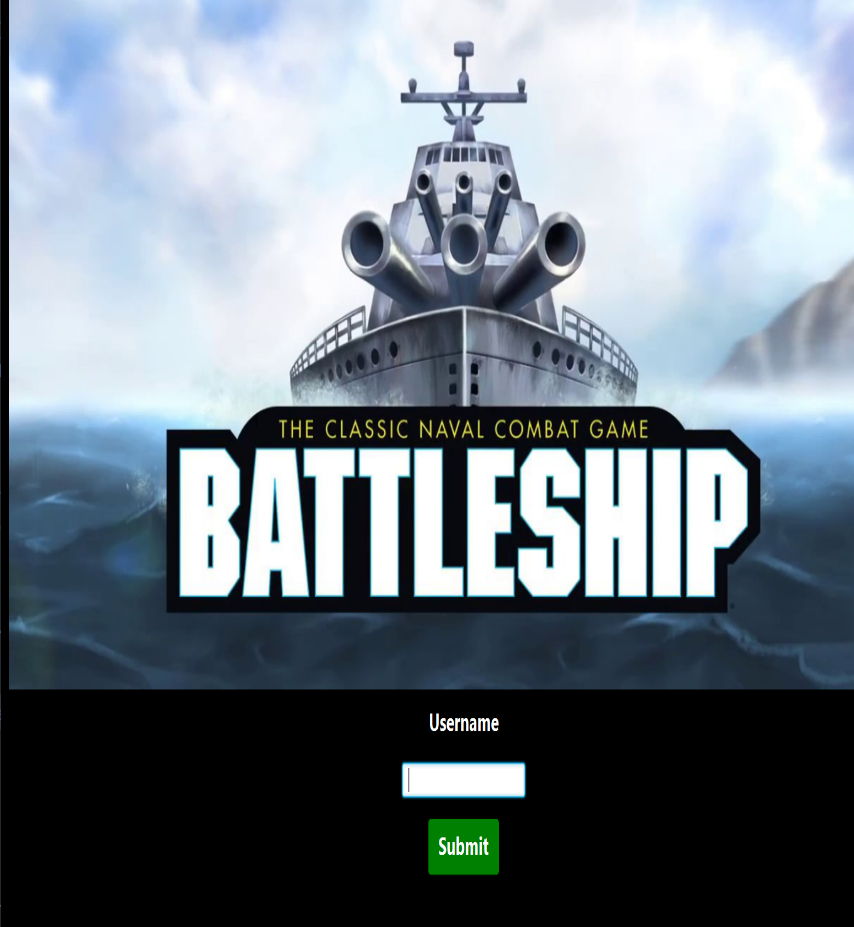
**UI CLASS FOR TIC TAC TOE**

*package* com.example.semesterproject.TicTakToe;  
  
*import* javafx.animation.PauseTransition;  
*import* javafx.application.Application;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.image.ImageView;  
*import* javafx.scene.layout.\*;  
  
*import* javafx.scene.media.Media;  
*import* javafx.scene.media.MediaPlayer;  
*import* javafx.scene.media.MediaView;  
*import* javafx.stage.Stage;  
*import* javafx.util.Duration;  
*import* com.example.semesterproject.GameSelectionMenu.GameSelectionMenu;  
  
  
*import* java.util.ArrayList;  
*import* java.util.Random;  
  
*public class* TicTacToeUI *extends* Application {  
  
 *public void* start(Stage window) {  
 Image image = *new* Image("b.png");  
 ImageView imageView = *new* ImageView(image);  
 imageView.setFitWidth(800);  
 imageView.setFitHeight(200);  
 *// Window setup* window.setHeight(800);  
 window.setWidth(800);  
 window.setResizable(*false*);  
  
 TicTakToe game = *new* TicTakToe();  
  
 *// UI Components* Button submissionButton = *new* Button("Submit");  
 submissionButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 18px;");  
 submissionButton.setPrefWidth(150);  
  
 Label username = *new* Label("Username: ");  
 username.setPrefWidth(100);  
 username.setStyle("-fx-text-fill: white; -fx-font-size: 18px;");  
  
 TextField nameField = *new* TextField();  
 nameField.setPromptText("USERNAME");  
  
 Label symbol = *new* Label("Symbol: ");  
 symbol.setPrefWidth(100);  
 symbol.setStyle("-fx-text-fill: white; -fx-font-size: 18px;");  
 ToggleGroup group = *new* ToggleGroup();  
 RadioButton xButton =*this*.styledRadioButtons("X");  
  
 RadioButton oButton = *this*.styledRadioButtons("O");  
  
 xButton.setToggleGroup(group);  
 oButton.setToggleGroup(group);  
  
  
 *// Layout setup* VBox vbox = *new* VBox();  
 vbox.setStyle("-fx-background-color: black");  
 BorderPane borderPane = *new* BorderPane();  
 borderPane.setTop(imageView);  
 borderPane.setStyle("-fx-background-color: black");  
 HBox nameBox = *new* HBox();  
  
 nameBox.setAlignment(Pos.CENTER);  
 nameBox.setSpacing(20);  
 nameBox.getChildren().addAll(username, nameField);  
  
 HBox symbolBox = *new* HBox();  
 symbolBox.setAlignment(Pos.CENTER);  
 symbolBox.setSpacing(20);  
  
 symbolBox.getChildren().addAll(symbol, xButton,oButton);  
  
 vbox.getChildren().addAll(nameBox, symbolBox, submissionButton);  
 vbox.setSpacing(30);  
 vbox.setAlignment(Pos.CENTER);  
  
 borderPane.setCenter(vbox);  
 borderPane.setStyle("-fx-background-color: #f0f0f0;");  
  
 Scene scene = *new* Scene(borderPane);  
 window.setScene(scene);  
 window.setTitle("Tic Tac Toe");  
 window.show();  
  
 *// Submission button action* submissionButton.setOnAction(e -> {  
 String userName = nameField.getText();  
 RadioButton button = (RadioButton) group.getSelectedToggle();  
 *if* (userName.isEmpty()) {  
 System.out.println("Username cannot be empty.");  
 *return*;  
 }  
 *if*(button==*null*){  
 System.out.println("Game symbol cannot be empty.");  
 *return*;  
 }  
 String gameSymbol = button.getText();  
 String enemySymbol ="";  
  
  
  
 *if*(gameSymbol.equalsIgnoreCase("X")){  
 enemySymbol ="O";  
  
 }  
 *else* enemySymbol ="X";  
 *this*.setupBoard(window, game, userName, gameSymbol,enemySymbol);  
 });  
 }  
  
 *private void* setupBoard(Stage primaryStage, TicTakToe game, String userName, String symbol ,String enemySymbol) {  
 String tapSoundPath ="";  
 *try*{tapSoundPath = getClass().getResource("/tapSound.wav").toExternalForm();}*catch* (Exception e){e.printStackTrace();}  
  
 Media tapSound = *new* Media(tapSoundPath);  
  
  
  
 Label usernameLabel = *new* Label("Username: " + userName);  
 usernameLabel.setAlignment(Pos.TOP\_LEFT);  
 usernameLabel.setPrefWidth(200);  
 usernameLabel.setStyle("-fx-font-size: 20px; -fx-text-fill:white;");  
  
 BorderPane root = *new* BorderPane();  
 root.setPrefHeight(800);  
 root.setPrefWidth(800);  
 root.setStyle("-fx-background-color: black;");  
  
 GridPane grid = *new* GridPane();  
 ArrayList<Button> buttons = *new* ArrayList<>();  
 *this*.initializeBoard(buttons, grid, game, symbol);  
  
 root.setTop(usernameLabel);  
 Label currentPlayerTurn = *new* Label("Turn: " + game.getCurrentPlayer());  
 currentPlayerTurn.setPrefWidth(80);  
 currentPlayerTurn.setStyle("-fx-font-size: 20px; -fx-text-fill: white;");  
  
 *for* (Button button : buttons) {  
 button.setOnAction(event -> {  
  
 *int* row = GridPane.*getRowIndex*(button);  
 *int* col = GridPane.*getColumnIndex*(button);  
  
 *try* {  
 *if* (game.makeMove(row, col)) {  
 MediaPlayer userSound = *new* MediaPlayer(tapSound);  
 button.setText(game.getCurrentPlayer());  
 userSound.play();  
  
  
 button.setStyle("-fx-background-color: #2196F3; -fx-text-fill: white; -fx-font-size: 40px;-fx-border-width: 2px;-fx-border-color: white;-fx-border-radius: 4px;");  
  
 *if* (game.checkTie() || game.checkWinner()) {  
 *if* (game.checkTie()) {  
 *this*.endingOnTieOrWinner(primaryStage, *new* Stage(),"tie");  
 } *else* {  
 *this*.endingOnTieOrWinner(primaryStage,*new* Stage(), "winner is " + game.getCurrentPlayer());  
 }  
 } *else* {  
 game.switchPlayer();  
 currentPlayerTurn.setText("Turn: " + game.getCurrentPlayer());}  
  
 *if*(game.getCurrentPlayer().equalsIgnoreCase(enemySymbol)){  
 PauseTransition pause = *new* PauseTransition(Duration.*seconds*(1));  
  
 pause.play();  
 pause.setOnFinished(e->{  
 MediaPlayer media = *new* MediaPlayer(tapSound);  
 Random random = *new* Random();  
 *int* enemyRow =0;  
 *int* enemyColumn =0;  
 *do* {  
 enemyRow= random.nextInt(3);  
 enemyColumn = random.nextInt(3);  
  
 }*while* (!game.checkForEmptyRowAndColumn(enemyRow,enemyColumn));  
 *try* {  
 game.makeMove(enemyRow,enemyColumn);  
 } *catch* (RowColumnException ex) {  
 *throw new* RuntimeException(ex);  
 }  
 *for*(Button currentButton : buttons){  
 *if*(GridPane.*getRowIndex*(currentButton).equals(enemyRow)&&GridPane.*getColumnIndex*(currentButton).equals(enemyColumn)){  
 currentButton.setText(game.getCurrentPlayer());  
 media.play();  
 currentButton.setStyle("-fx-background-color: #2196F3; -fx-text-fill: white; -fx-font-size: 40px;-fx-border-width: 2px;-fx-border-color: white;-fx-border-radius: 4px;");  
 game.switchPlayer();  
 currentPlayerTurn.setText("Turn: " + game.getCurrentPlayer());  
 *break*;  
 }  
 *if* (game.checkTie() || game.checkWinner()) {  
 *if* (game.checkTie()) {  
 *this*.endingOnTieOrWinner(primaryStage,*new* Stage(), "tie");  
 } *else* {  
 *this*.endingOnTieOrWinner(primaryStage,*new* Stage(), "winner is " + game.getCurrentPlayer());  
 }  
 }  
 }  
  
  
 });  
  
 }  
  
  
  
  
 }  
  
 } *catch* (Exception e) {  
 System.out.println(e.getMessage());  
 }  
 });  
  
 }  
  
 grid.setAlignment(Pos.CENTER);  
  
 Button resetButton = *new* Button("Reset");  
 resetButton.setStyle("-fx-background-color: #f44336; -fx-text-fill: white; -fx-font-size: 18px;");  
 resetButton.setPrefWidth(150);  
 resetButton.setOnAction(event -> *this*.resetBoard(buttons, game, symbol));  
  
 root.setCenter(grid);  
 HBox box = *new* HBox();  
 box.setSpacing(240);  
 box.getChildren().addAll(currentPlayerTurn, resetButton);  
 root.setBottom(box);  
  
 Scene newScene = *new* Scene(root);  
 primaryStage.setScene(newScene);  
 primaryStage.setTitle("Tic Tac Toe");  
 primaryStage.show();  
  
 }  
  
 *public static void* main(String[] args) {  
 *launch*(args);  
 }  
  
 *private* Button styledButtons(String text) {  
 Button button = *new* Button(text);  
 button.setPrefHeight(200);  
 button.setPrefWidth(200);  
 button.setStyle("-fx-font-size: 80px; -fx-text-fill: grey; -fx-background-color: #eeeeee; -fx-border-color: #bbbbbb; -fx-padding: 5px;");  
 *return* button;  
 }  
  
 *private void* initializeBoard(ArrayList<Button> buttons, GridPane grid, TicTakToe game, String symbol) {  
 buttons.clear();  
 grid.getChildren().clear();  
  
 *for* (*int* i = 0; i < 3; i++) {  
 *for* (*int* j = 0; j < 3; j++) {  
 Button button = *this*.styledButtons("");  
 buttons.add(button);  
 grid.add(button, j, i);  
 }  
 }  
 game.initializeBoard(symbol);  
 }  
  
 *private void* resetBoard(ArrayList<Button> buttons, TicTakToe game, String symbol) {  
 *for* (Button button : buttons) {  
 button.setText("");  
 button.setStyle("-fx-font-size: 80px; -fx-text-fill: grey; -fx-background-color: #eeeeee; -fx-border-color: #bbbbbb; -fx-padding: 5px;");  
 }  
 game.resetGame(symbol);  
 }  
  
 *public void* endingOnTieOrWinner(Stage oldStage,Stage stage, String text) {  
 stage.setHeight(600);  
 stage.setWidth(600);  
 stage.setResizable(*false*);  
 String result;  
 *if* (text.equals("tie")) {  
 result = "The match ended in a tie.";  
 } *else* {  
 result = "The match ended with a winner. \n\t\tThe " + text;  
 }  
 Label resultLabel = *new* Label(result);  
 resultLabel.setAlignment(Pos.CENTER);  
 resultLabel.setStyle("-fx-text-fill: white;-fx-font-size: 20px;");  
 VBox vbox = *new* VBox();  
 vbox.setStyle("-fx-background-color: black");  
 Button tryAgain = *new* Button("Try Again");  
 tryAgain.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 18px;");  
 tryAgain.setPrefWidth(150);  
  
 Button exitButton = *new* Button("Exit");  
 exitButton.setStyle("-fx-background-color: #f44336; -fx-text-fill: white; -fx-font-size: 18px;");  
 exitButton.setPrefWidth(150);  
  
 vbox.getChildren().addAll(resultLabel, tryAgain, exitButton);  
 vbox.setSpacing(20);  
 vbox.setAlignment(Pos.CENTER);  
 Scene scene = *new* Scene(vbox);  
  
 tryAgain.setOnAction(event -> {  
 stage.close();  
 *this*.start(oldStage);  
  
  
 });  
 exitButton.setOnAction(event ->  
 {  
 oldStage.close();  
 stage.close();  
 GameSelectionMenu menu = *new* GameSelectionMenu();  
  
 menu.start(*new* Stage());  
  
  
  
 });  
 stage.setTitle(oldStage.getTitle());  
 stage.setScene(scene);  
 stage.show();  
 }  
 *private* RadioButton styledRadioButtons(String text) {  
 RadioButton button = *new* RadioButton(text);  
 button.setStyle("-fx-text-fill: white;-fx-font-size: 18px");  
 button.setPrefWidth(50);  
 *return* button;  
 }  
}

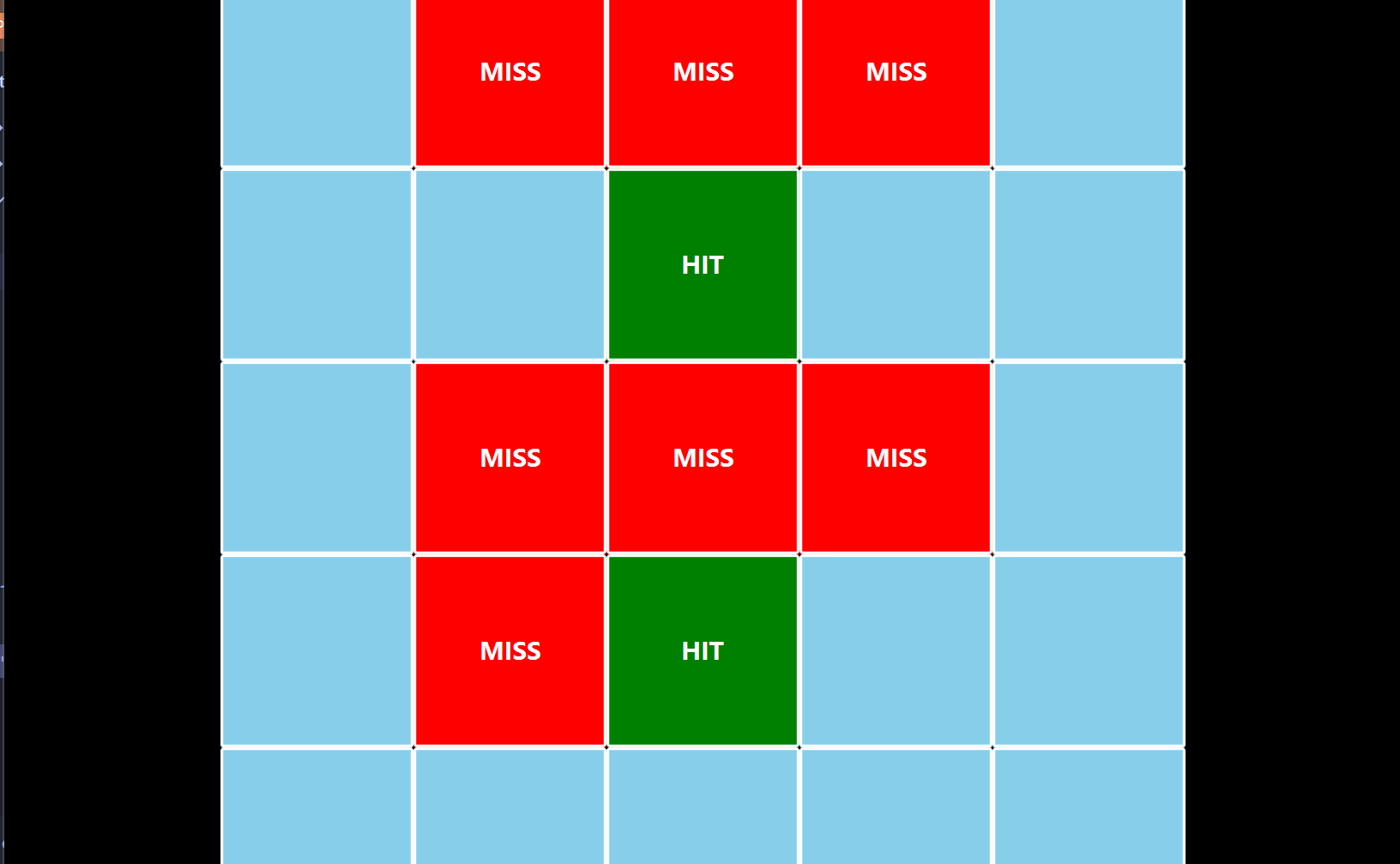
**MAIN LOGIC CLASS FOR GAME**

*package* com.example.semesterproject.TicTakToe;  
  
*public class* TicTakToe {  
  
 *//attributes  
 private* String symbol;  
 *private* String [][] board;  
 *private* String currentPlayer;  
 *private* String winner;  
  
  
 *public* TicTakToe(){  
 *this*.symbol = "";  
 *this*.board = *new* String[3][3];  
 *this*.winner = *null*;  
 *this*.currentPlayer="";  
 }  
 *//Getter and setter  
  
 public void* setSymbol(String symbol) {  
 *this*.symbol = symbol;  
 }  
  
 *public* String getSymbol() {  
 *return* symbol;  
 }  
 *public* String getCurrentPlayer() {  
 *return* currentPlayer;  
 }  
  
 *public void* setCurrentPlayer(String currentPlayer) {  
 *this*.currentPlayer = currentPlayer;  
 }  
  
 *public* String getWinner() {  
 *return* winner;  
 }  
  
 *public void* setWinner(String winner) {  
 *this*.winner = winner;  
 }  
 *public* String[][] getBoard() {  
 *return* board;  
 }  
 *public void* initializeBoard(String currentPlayer){  
 *for* (*int* i = 0; i < 3; i++) {  
 *for* (*int* j = 0; j < 3; j++) {  
 board[i][j] = "";  
 }  
 }  
 *this*.currentPlayer = currentPlayer;  
 *this*.winner =*null*;  
 }  
 *public boolean* makeMove(*int* row, *int* column) *throws* RowColumnException {  
 *if*(row >= 3 || column >= 3 || row < 0 || column < 0){  
 *throw new* RowColumnException("Invalid Rows and columns");  
  
 }  
 *if*(board[row][column].isEmpty()) {  
 board[row][column] = *this*.currentPlayer;  
 *return true*;  
 }  
 *else  
 return false*;  
 }  
 *public void* switchPlayer(){  
 *if*(*this*.currentPlayer.equals("X"))  
 *this*.currentPlayer = "O";  
 *else  
 this*.currentPlayer = "X";  
 }  
 *public boolean* checkWinner(){  
 *//rows check  
 for*(*int* i =0;i<3;i++){  
 *if*(board[i][0].equals(*this*.currentPlayer)&&board[i][1].equals(*this*.currentPlayer)&&board[i][2].equals(*this*.currentPlayer)) {  
 *this*.winner =*this*.currentPlayer;  
 *return true*;  
 }  
  
  
 }  
 *//columns check  
 for*(*int* j=0;j<3;j++){  
 *if*(board[0][j].equals(*this*.currentPlayer)&&board[1][j].equals(*this*.currentPlayer)&&board[2][j].equals(*this*.currentPlayer)) {  
 *this*.winner =*this*.currentPlayer;  
 *return true*;  
 }  
 }  
 *//diagonal check  
 if* ((board[0][0].equals(*this*.currentPlayer) &&  
 board[1][1].equals(*this*.currentPlayer) &&  
 board[2][2].equals(*this*.currentPlayer)) ||  
 (board[0][2].equals(*this*.currentPlayer) &&  
 board[1][1].equals(*this*.currentPlayer) &&  
 board[2][0].equals(*this*.currentPlayer))) {  
 *this*.winner =*this*.currentPlayer;  
 *return true*;  
 }  
  
 *return false*;  
  
  
 }  
 *public boolean* checkTie(){  
 *for*(*int* i=0;i<3;i++){  
 *for*(*int* j=0;j<3;j++){  
 *if*(board[i][j].equals("")){  
 *return false*;  
 }  
 }  
  
 }  
 *return* !(*this*.checkWinner());  
 }  
 *public boolean* checkForEmptyRowAndColumn(*int* row , *int* column){  
 *return* board[row][column].isEmpty();  
 }  
 *public void* resetGame(String userPlayer){  
 *this*.initializeBoard(userPlayer);  
 }  
  
  
}

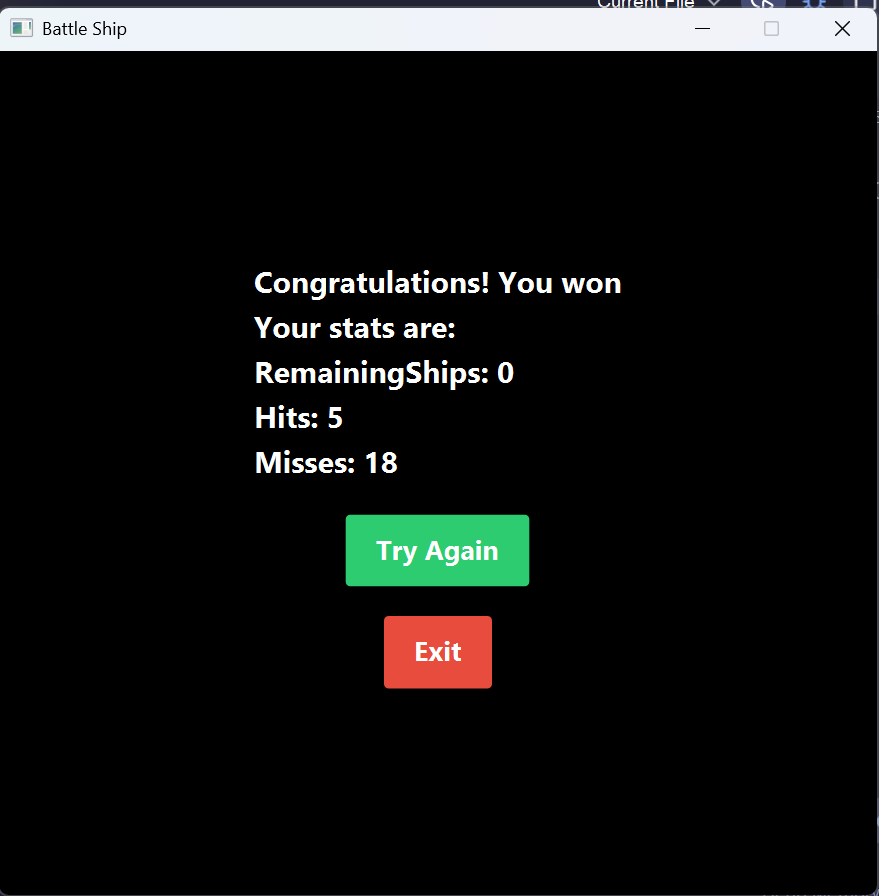
**BATTLESHIP**



(Username window for Battleship which inputs username).



(Main game window for Battleship where you have to hit ships randomly).



(Congratulations window for winner ).

**Battleship Game**

**Username Entry Window**  
This is the entry point for Battleship. Users must enter a valid username to proceed. The window includes a vibrant design with an image and a "Submit" button to start the game. If the username field is left empty, the application provides feedback.

**Gameplay Window**  
The Battleship grid awaits! Players guess ship locations on a 10x10 board by clicking grid cells. Each click reveals a "HIT" (if a ship is present) or a "MISS" (if no ship is there). The interface tracks remaining ships, hits, and misses, offering a "Hint" button to guide the player and a "Reset" option to restart the game at any point.

**Ending Window**  
When all enemy ships are sunk, this window celebrates the player's victory with congratulatory messages and sound effects. It shows final stats, including remaining ships, total hits, and misses. Players can choose to try again or return to the main game selection menu.

**BATTLESHIP CLASS**

*package* com.example.semesterproject.BattleShip;  
*import* com.example.semesterproject.TicTakToe.RowColumnException;  
*import* java.util.ArrayList;  
*import* java.util.Random;  
  
*public class* BattleShip {  
 *//attributes  
 private* String [][] gridfShips;  
 *private* ArrayList<Coordinates> gridCoordinates;  
  
 *int* remainingShips;  
 *int* hits;  
 *int* misses;  
  
 *//constructor  
 public* BattleShip(){  
 gridfShips = *new* String[5][5];  
 {  
 gridCoordinates = *new* ArrayList<>();  
 *for*(*int* i =0;i<gridfShips.length;i++){  
 *for*(*int* j =0;j<gridfShips[i].length;j++){  
 gridCoordinates.add(*new* Coordinates(i,j));  
 }  
 }  
 }  
  
 remainingShips = 0;  
 hits = 0;  
 misses = 0;  
 }  
 *//getter  
 public* String[][] getGridfShips() {  
 *return* gridfShips;  
 }  
  
 *public* ArrayList<Coordinates> getgridCoordinates() {  
 *return* gridCoordinates;  
 }  
  
 *public int* getRemainingShips() {  
 *return* remainingShips;  
 }  
  
 *public int* getHits() {  
 *return* hits;  
 }  
  
 *public int* getMisses() {  
 *return* misses;  
 }  
 *//setter  
  
 public void* setMisses(*int* misses) {  
 *this*.misses = misses;  
 }  
  
 *public void* setHits(*int* hits) {  
 *this*.hits = hits;  
 }  
  
 *public void* setRemainingShips(*int* remainingShips) {  
 *this*.remainingShips = remainingShips;  
 }  
  
 *public void* initialShipGrid(){  
 *for*(*int* i=0;i<5;i++){  
 *for*(*int* j=0;j<5;j++){  
 gridfShips[i][j] = "";  
 }  
 }  
 *this*.setHits(0);  
 *this*.setMisses(0);  
 *this*.remainingShips = 0;  
 *this*.placeShip();  
 }  
 *private void* placeShip(){  
 Random randomShipPosition = *new* Random();  
 *for*(*int* i =0;i<gridfShips.length;i++){  
 *while* (*true*) {  
 *int* shipPosition = randomShipPosition.nextInt(*this*.gridCoordinates.size());  
 *int* row = *this*.gridCoordinates.get(shipPosition).getX();  
 *int* column = *this*.gridCoordinates.get(shipPosition).getY();  
 *if* (gridfShips[row][column].isEmpty()) {  
 gridfShips[row][column] = "Ship";  
 remainingShips++;  
 *break*;  
 }  
 }  
 }  
 }  
 *public boolean* processGuess(*int* row , *int* columns) *throws* RowColumnException{  
 *if*(row>*this*.gridfShips.length-1||columns>gridfShips[row].length-1){  
 *throw new* RowColumnException("\"Invalid input: Row "+row+", Column "+columns+" is out of bounds for a grid of "+*this*.gridfShips.length+"x"+*this*.gridfShips[0].length+".\"\n");  
 }  
 *if*(gridfShips[row][columns].equals("Ship")){  
 gridfShips[row][columns] = "Hit";  
 remainingShips--;  
 hits++;  
 *return true*;  
 }  
 gridfShips[row][columns] = "Miss";  
 misses++;  
  
 *return false*;  
  
 }  
 *public boolean* isGameOver(){  
 *return* remainingShips==0;  
 }  
  
 *public void* restartGame(){  
 *this*.initialShipGrid();  
  
 }  
 *public int* [] hintSystem(){  
 *for*(*int* i =0;i<gridfShips.length;i++){  
 *for*(*int* j =0;j<gridfShips[i].length;j++){  
 *if*(gridfShips[i][j].equals("Ship")){  
 *return new int*[]{i,j};  
 }  
 }  
 }  
 *return null*;  
 }  
  
  
}

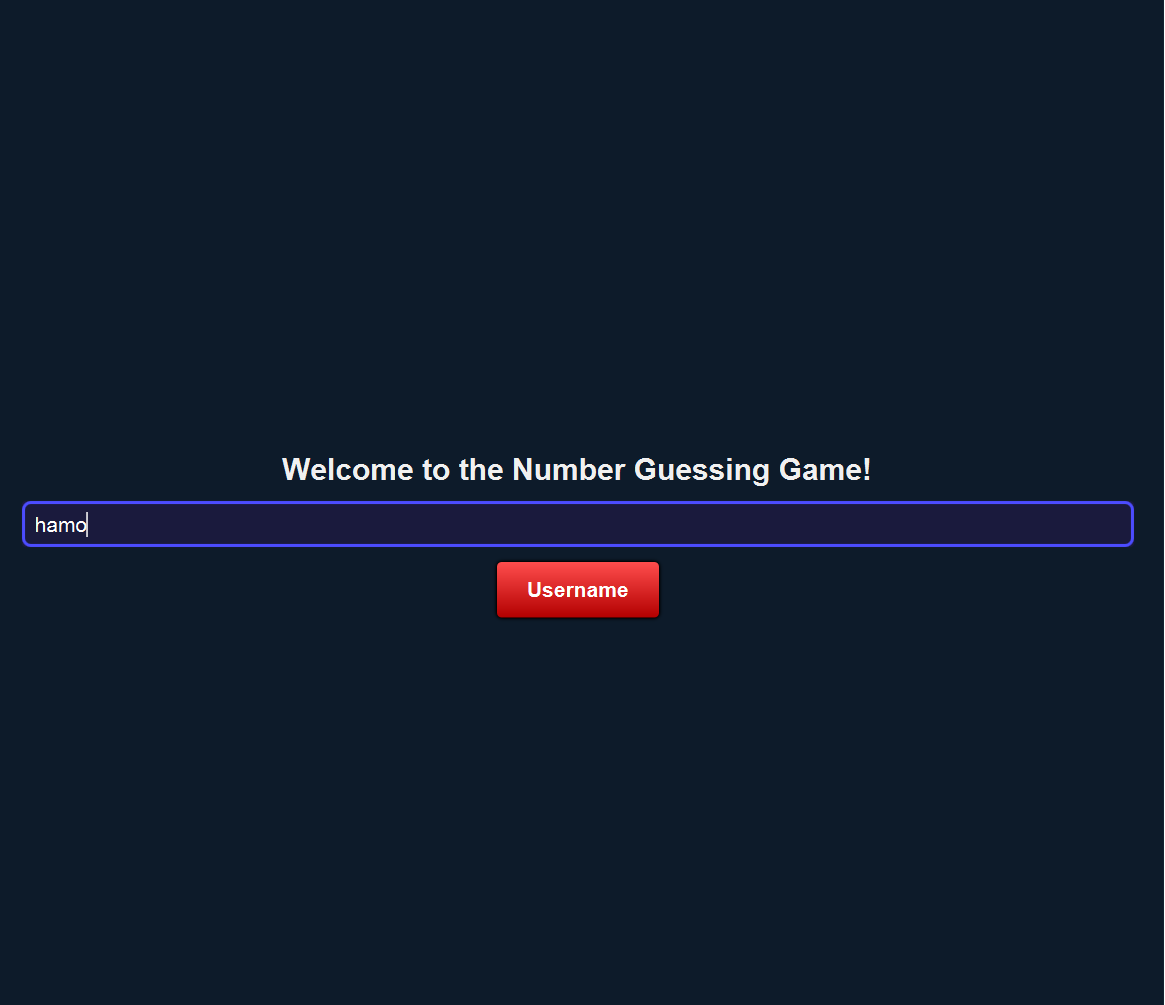
**BATTLESHIP UI CLASS**

*package* com.example.semesterproject.BattleShip;  
  
*import* com.example.semesterproject.GameSelectionMenu.GameSelectionMenu;  
*import* com.example.semesterproject.TicTakToe.RowColumnException;  
*import* javafx.application.Application;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.Alert;  
*import* javafx.scene.control.Button;  
  
*import* javafx.scene.control.Label;  
*import* javafx.scene.control.TextField;  
*import* javafx.scene.image.Image;  
*import* javafx.scene.image.ImageView;  
*import* javafx.scene.layout.\*;  
*import* javafx.scene.media.Media;  
*import* javafx.scene.media.MediaPlayer;  
*import* javafx.stage.Stage;  
  
*import* java.util.ArrayList;  
  
*public class* BattleShipUI *extends* Application {  
 *public void* start(Stage primaryStage) {  
 BorderPane userNameLayout = *new* BorderPane();  
 userNameLayout.setStyle("-fx-background-color: black;-fx-padding: 10px,3px,3px,3px");  
 primaryStage.setWidth(1100);  
 primaryStage.setHeight(1050);  
 Image image = *new* Image("battleship.jpg");  
 ImageView imageView = *new* ImageView(image);  
 imageView.setFitHeight(500);  
 imageView.setFitWidth(1100);  
 userNameLayout.setTop(imageView);  
 Label nameLabel = *new* Label("Username");  
 nameLabel.setStyle("-fx-text-fill: white;-fx-font-weight: bold;-fx-font-size: 18px;");  
 HBox nameHBox = *new* HBox();  
 nameHBox.setPrefWidth(150);  
 nameHBox.setAlignment(Pos.CENTER);  
  
 TextField username = *new* TextField();  
 nameHBox.getChildren().add(username);  
 Label userNameresult = *new* Label();  
 userNameresult.setStyle("-fx-text-fill: white;-fx-font-weight: bold;-fx-font-size: 18px;");  
 username.setPromptText("Username");  
 Button submitButton = *new* Button("Submit");  
 submitButton.setStyle("-fx-background-color: Green ; -fx-text-fill: white;-fx-font-weight: bold; -fx-font-size: 18px;");  
 submitButton.setOnAction(e -> {  
 *if*(username.getText().equals("")) {  
 userNameresult.setText("Username cannot be empty");  
 *return*;  
  
 }  
 primaryStage.close();  
 *this*.gameScene(primaryStage,username.getText());  
 });  
 VBox inputBox = *new* VBox();  
 inputBox.setSpacing(15);  
 inputBox.setStyle("-fx-padding: 10px");  
 inputBox.setAlignment(Pos.TOP\_CENTER);  
 inputBox.getChildren().addAll(nameLabel, nameHBox, submitButton,userNameresult);  
 userNameLayout.setCenter(inputBox);  
  
 primaryStage.setTitle("Battle Ship");  
 primaryStage.setResizable(*false*);  
 primaryStage.setScene(*new* Scene(userNameLayout));  
 primaryStage.show();  
 }  
 *public static void* main(String[] args) {  
 *launch*(args);  
 }  
 *private void* initializeShipGrid(ArrayList<Button> coordinatesOfShips,GridPane gridOfShip,BattleShip backend) {  
 coordinatesOfShips.clear();  
 gridOfShip.getChildren().clear();  
 *for*(*int* i = 0; i <backend.getGridfShips().length ; i++){  
 *for*(*int* j = 0; j < backend.getGridfShips()[0].length; j++){  
 Button shipCoordinates = *new* Button("");  
 *this*.styledShipCoordinatesButton(shipCoordinates);  
 gridOfShip.add(shipCoordinates, i,j);  
 coordinatesOfShips.add(shipCoordinates);  
 }  
 }  
 backend.initialShipGrid();  
  
  
  
  
 }  
 *private void* styledShipCoordinatesButton(Button button){  
  
 button.setPrefHeight(150);  
 button.setPrefWidth(150);  
 button.setStyle("-fx-padding: 2px;-fx-font-size: 30px;-fx-font-weight: bold;-fx-text-fill: white;-fx-background-color:skyblue;-fx-border-width: 2px;-fx-border-style: solid;-fx-border-color: white;-fx-border-radius: 2px");  
  
 }  
 *private* Label getStyledlabel (String text){  
 Label label = *new* Label(text);  
 label.setStyle("-fx-font-weight: bold;-fx-font-size: 30px;-fx-text-fill: white");  
 *return* label;  
 }  
 *private void* updatedStats(Label remainingShips,Label hits,Label Misses,BattleShip backend){  
 remainingShips.setText("RemainingShips: "+backend.remainingShips);  
 hits.setText("Hits: "+backend.hits);  
 Misses.setText("Misses: "+backend.misses);  
  
 }  
 *private void* resetGame(BattleShip backend,ArrayList<Button> coordinatesOfShip,GridPane gridOfShip,Label remainingShips,Label hits,Label Misses){  
 *for*(Button curretnButton : coordinatesOfShip){  
 curretnButton.setText("");  
 *this*.styledShipCoordinatesButton(curretnButton);  
  
 }  
 backend.restartGame();  
 *this*.updatedStats(remainingShips,hits,Misses,backend);  
 }  
 *public void* gameScene(Stage primaryStage ,String userName) {  
  
 BattleShip backend = *new* BattleShip();  
 BorderPane uiLayout = *new* BorderPane();  
 uiLayout.setStyle("-fx-background-color: black;-fx-padding: 10px,3px,3px,3px");  
 Media missSound = *new* Media(*this*.getClass().getResource("/miss.wav").toExternalForm());  
  
 Media hitSound = *new* Media(*this*.getClass().getResource("/Hit.wav").toExternalForm());  
  
  
  
 ArrayList<Button> coordinatesOfShips = *new* ArrayList<>();  
 GridPane gridOfShip = *new* GridPane();  
 gridOfShip.setAlignment(Pos.CENTER);  
 uiLayout.setCenter(gridOfShip);  
 *this*.initializeShipGrid(coordinatesOfShips, gridOfShip,backend);  
 *//setting up the remaining ships,hits and misses updates at bottom left* Label remainingShips = *this*.getStyledlabel("RemainingShips: "+backend.remainingShips);  
 Label hits = *this*.getStyledlabel("Hits: "+backend.hits);  
 Label misses = *this*.getStyledlabel("Misses: "+backend.misses);  
 VBox fieldsBox = *new* VBox();  
  
 fieldsBox.getChildren().addAll(remainingShips, hits, misses);  
 fieldsBox.setAlignment(Pos.BOTTOM\_LEFT);  
 fieldsBox.setSpacing(10);  
  
  
 Button resetButton = *new* Button("Reset");  
 resetButton.setAlignment(Pos.BOTTOM\_CENTER);  
 resetButton.setOnAction(e -> {  
 *this*.resetGame(backend,coordinatesOfShips,gridOfShip,remainingShips,hits,misses);  
 });  
 *//bottom layout* HBox bottomLayout = *new* HBox();  
 bottomLayout.getChildren().addAll(fieldsBox, resetButton);  
 bottomLayout.setPrefWidth(1000);  
 bottomLayout.setSpacing(640);  
  
 resetButton.setStyle("-fx-background-color: #f44336; -fx-text-fill: white; -fx-font-size: 18px;-fx-font-weight: bold");  
 resetButton.setPrefWidth(160);  
 uiLayout.setBottom(bottomLayout);  
 *//top layout* HBox topLayout = *new* HBox();  
 Label username =*this*.getStyledlabel("Username: "+userName);  
 Button hintButton = *new* Button("Hint");  
 hintButton.setStyle("-fx-background-color: Green ;-fx-font-weight: bold; -fx-text-fill: white; -fx-font-size: 18px;");  
 hintButton.setPrefWidth(160);  
 topLayout.setPrefWidth(1100);  
 *if*(userName.length()>=6) {  
 topLayout.setSpacing(610);  
 }  
 *else* topLayout.setSpacing(680);  
 hintButton.setOnAction(actionEvent -> {  
 *int* [] shipPosition = backend.hintSystem();  
 *if*(shipPosition!=*null*) {  
 *for* (Button currentButton : coordinatesOfShips) {  
 *if* (GridPane.*getRowIndex*(currentButton) == shipPosition[0] && GridPane.*getColumnIndex*(currentButton) == shipPosition[1]) {  
 currentButton.setStyle("-fx-padding: 2px;-fx-font-size: 30px;-fx-font-weight: bold;-fx-text-fill: white;-fx-background-color:yellow;-fx-border-width: 2px;-fx-border-style: solid;-fx-border-color: white;-fx-border-radius: 2px");  
 *break*;  
 }  
 }  
 }  
 });  
  
 hintButton.setAlignment(Pos.TOP\_CENTER);  
 topLayout.getChildren().addAll(username,hintButton);  
 uiLayout.setTop(topLayout);  
  
 *for*(Button currentShipCoordinate : coordinatesOfShips){  
 currentShipCoordinate.setOnAction(event -> {  
 *int* row = GridPane.*getRowIndex*(currentShipCoordinate);  
 *int* col = GridPane.*getColumnIndex*(currentShipCoordinate);  
 *try* {  
 *if*(currentShipCoordinate.getText().equalsIgnoreCase("HIT")||currentShipCoordinate.getText().equalsIgnoreCase("MISS")){  
 *return*;  
 }  
 *else if*(backend.processGuess(row,col)){  
 MediaPlayer hitSoundPlayer = *new* MediaPlayer(hitSound);  
 hitSoundPlayer.play();  
 currentShipCoordinate.setStyle("-fx-padding: 2px;-fx-font-size: 20px;-fx-font-weight: bold;-fx-text-fill: white;-fx-background-color:green;-fx-border-width: 2px;-fx-border-style: solid;-fx-border-color: white;-fx-border-radius: 2px");  
 currentShipCoordinate.setText("HIT");  
  
 }  
 *else*{  
 MediaPlayer missSoundPlayer = *new* MediaPlayer(missSound);  
 missSoundPlayer.play();  
 currentShipCoordinate.setStyle("-fx-padding: 2px;-fx-font-size: 20px;-fx-font-weight: bold;-fx-text-fill: white;-fx-background-color:red;-fx-border-width: 2px;-fx-border-style: solid;-fx-border-color: white;-fx-border-radius: 2px");  
 currentShipCoordinate.setText("MISS");  
 }  
  
 *this*.updatedStats(remainingShips,hits,misses,backend);  
 *if*(backend.isGameOver()) {  
 primaryStage.close();  
 *this*.endingScene(primaryStage,userName,backend,remainingShips,hits,misses);  
  
 }  
 } *catch* (RowColumnException e) {  
 *throw new* RuntimeException(e);  
 }  
 });  
 }  
 Scene battleShipScene = *new* Scene(uiLayout);  
 primaryStage.setScene(battleShipScene);  
 primaryStage.show();  
 }  
 *public void* endingScene(Stage primaryStage, String userName, BattleShip backend, Label shipRemaining, Label hits, Label misses) {  
 Media winningsSound = *new* Media(*this*.getClass().getResource("/winning.wav").toExternalForm());  
 MediaPlayer winningSoundPlayer = *new* MediaPlayer(winningsSound);  
 winningSoundPlayer.play();  
 Stage secondaryStage = *new* Stage();  
 secondaryStage.setTitle(primaryStage.getTitle());  
 secondaryStage.setResizable(*false*);  
 secondaryStage.setWidth(600);  
 secondaryStage.setHeight(600);  
  
 Label stats = *this*.getStyledlabel("Congratulations! You won\nYour stats are:\n" + shipRemaining.getText() + "\n" + hits.getText() + "\n" + misses.getText());  
 stats.setStyle("-fx-font-size: 20px; -fx-font-weight: bold; -fx-text-fill: white;");  
  
  
 Button tryAgainButton = *new* Button("Try Again");  
 tryAgainButton.setStyle("-fx-font-size: 18px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: #2ecc71; -fx-padding: 10px 20px;");  
 tryAgainButton.setOnAction(event -> {  
 secondaryStage.close();  
 *this*.gameScene(primaryStage,userName);  
 });  
  
  
 Button exitButton = *new* Button("Exit");  
 exitButton.setOnAction(event -> {  
 secondaryStage.close();  
 GameSelectionMenu menu = *new* GameSelectionMenu();  
 menu.start(*new* Stage());  
 });  
 exitButton.setStyle("-fx-font-size: 18px; -fx-font-weight: bold; -fx-text-fill: white; -fx-background-color: #e74c3c; -fx-padding: 10px 20px;");  
  
 *// Arrange the buttons and label with spacing in a VBox* VBox buttonLayoutAndLabel = *new* VBox();  
 buttonLayoutAndLabel.setStyle("-fx-background-color: black; -fx-alignment: center; -fx-padding: 20px;");  
 buttonLayoutAndLabel.setSpacing(20);  
 buttonLayoutAndLabel.getChildren().addAll(stats,tryAgainButton, exitButton);  
  
 buttonLayoutAndLabel.setAlignment(Pos.CENTER);  
 secondaryStage.setScene(*new* Scene(buttonLayoutAndLabel));  
 secondaryStage.show();  
 }  
}

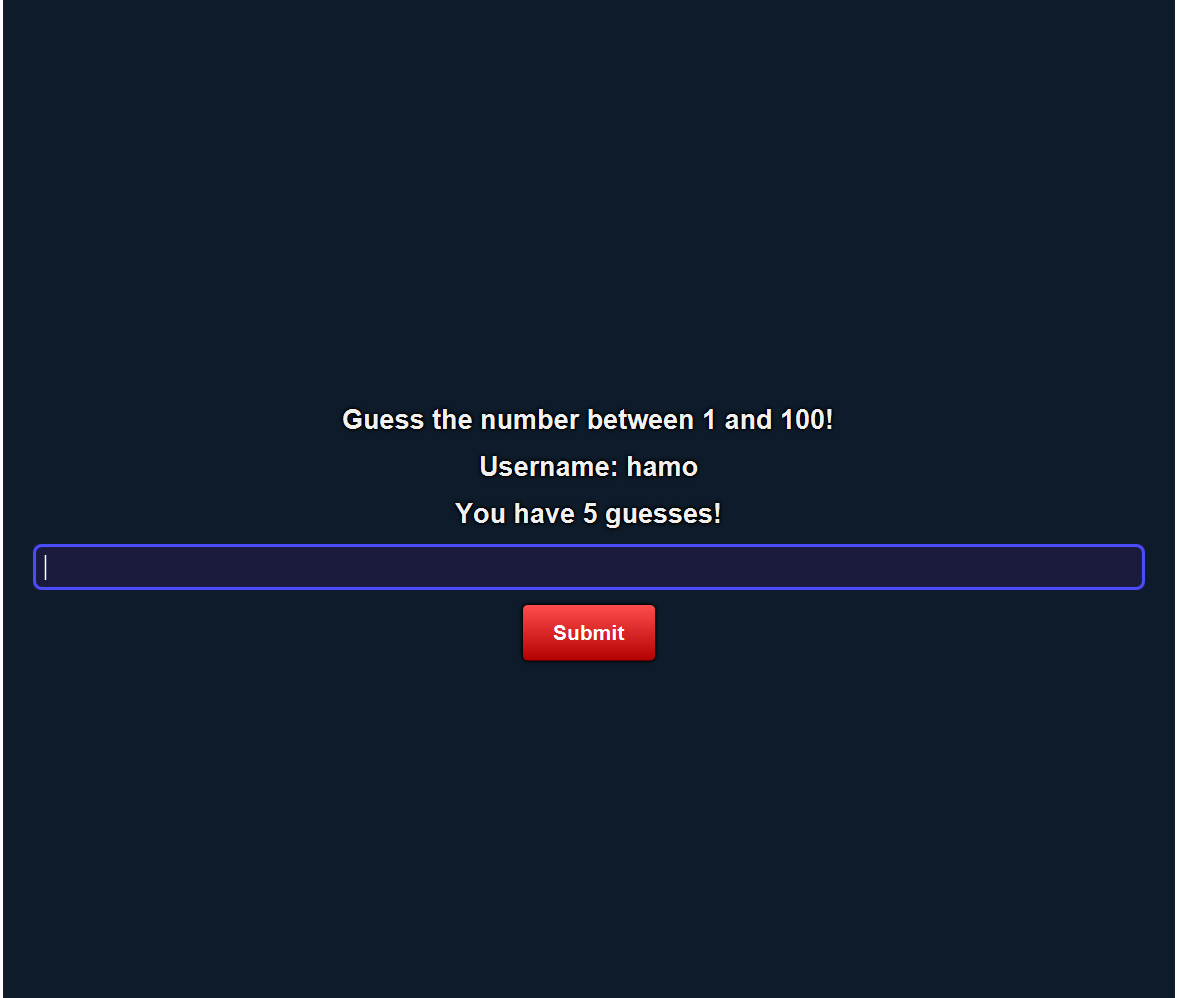
**COORDINATES CLASS**

*package* com.example.semesterproject.BattleShip;  
  
*public class* Coordinates {  
 *private int* x;  
 *private int* y;  
 *public* Coordinates(*int* x, *int* y) {  
 *this*.x = x;  
 *this*.y = y;  
  
 }  
  
 *public int* getX() {  
 *return* x;  
 }  
  
 *public void* setX(*int* x) {  
 *this*.x = x;  
 }  
  
 *public int* getY() {  
 *return* y;  
 }  
  
 *public void* setY(*int* y) {  
 *this*.y = y;  
 }

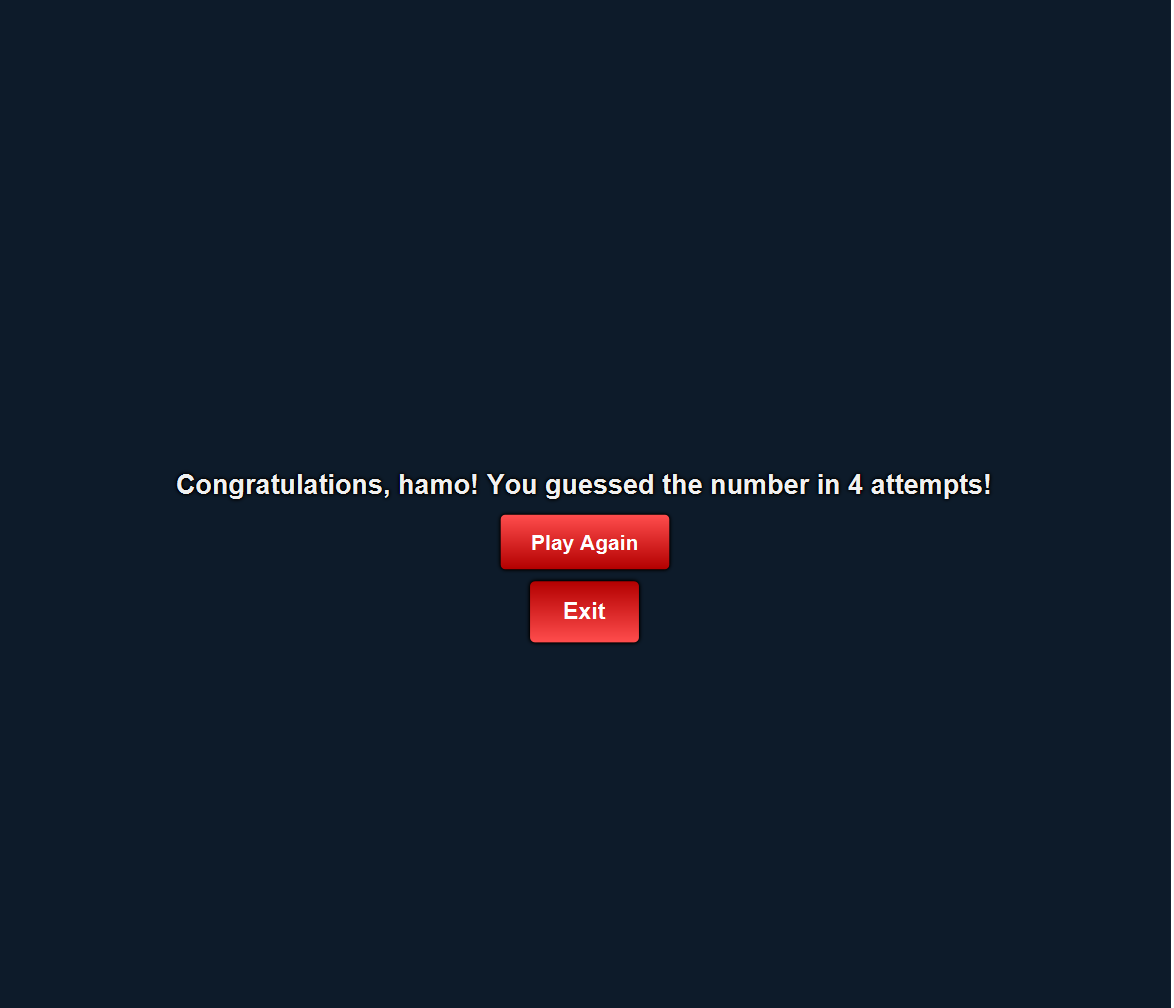
**NUMBER GUESSING**



(Username window which inputs username from user).



(Main game window which provides 5 guesses for guessing the correct number).



(Congratulations window for winner).

**Number Guessing Game**

**Main Menu Window**  
Welcome to the Number Guessing Game! In this window, users will find a simple interface to start the game. You can choose your difficulty level or proceed with the default settings. Click on the "Start Game" button to begin and test your guessing skills!

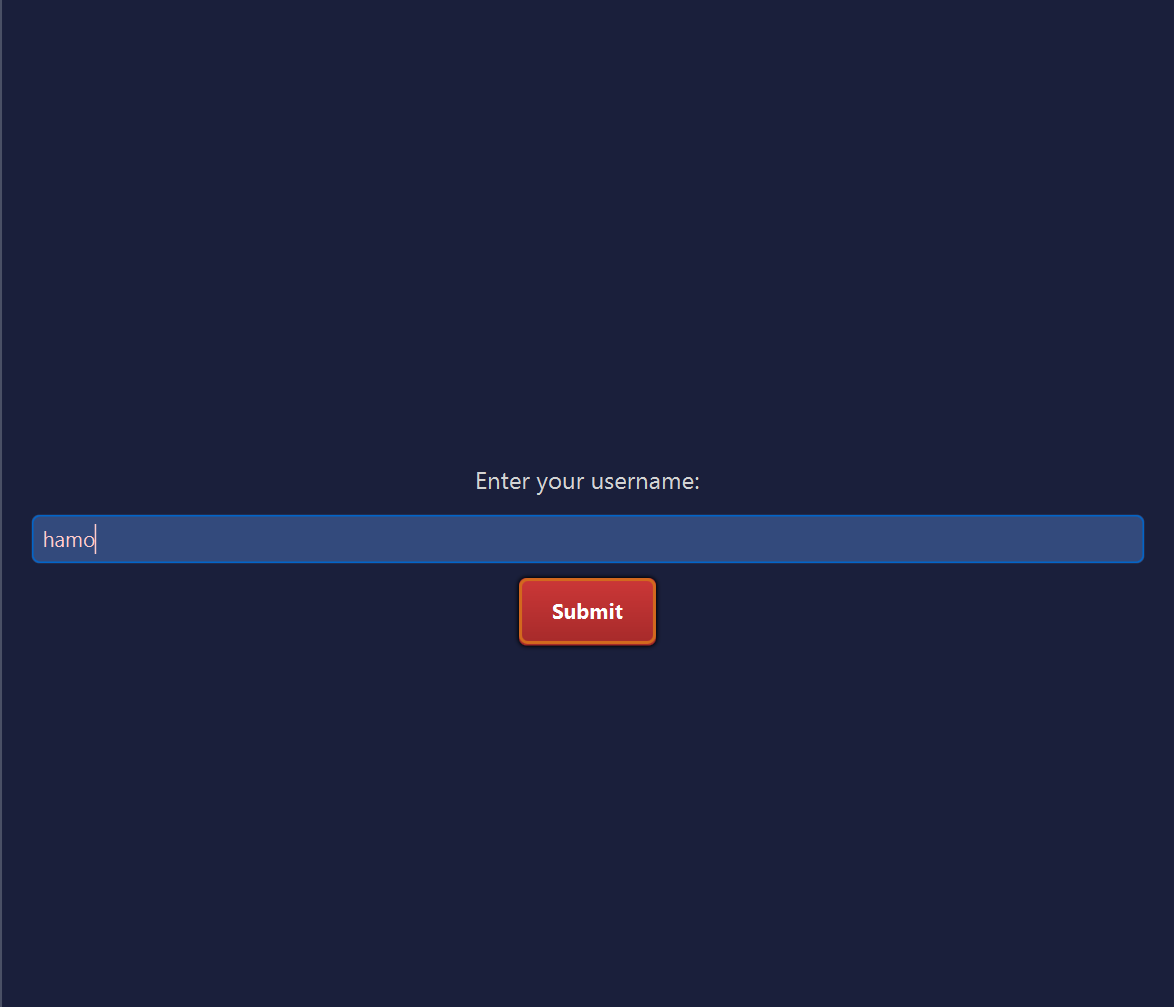
**Gameplay Window**  
The game challenges you to guess a randomly generated number within a specific range. Enter your guess and receive instant feedback: too high, too low, or correct. The interface also displays the remaining attempts and a hint to guide your next move. Aim to guess the number within the allowed attempts to win!

**Result Window**  
Once the game concludes, this window displays your result—either celebrating your win or encouraging you to try again if you didn't guess correctly. It shows the correct number, your total attempts, and an option to restart or return to the main menu.

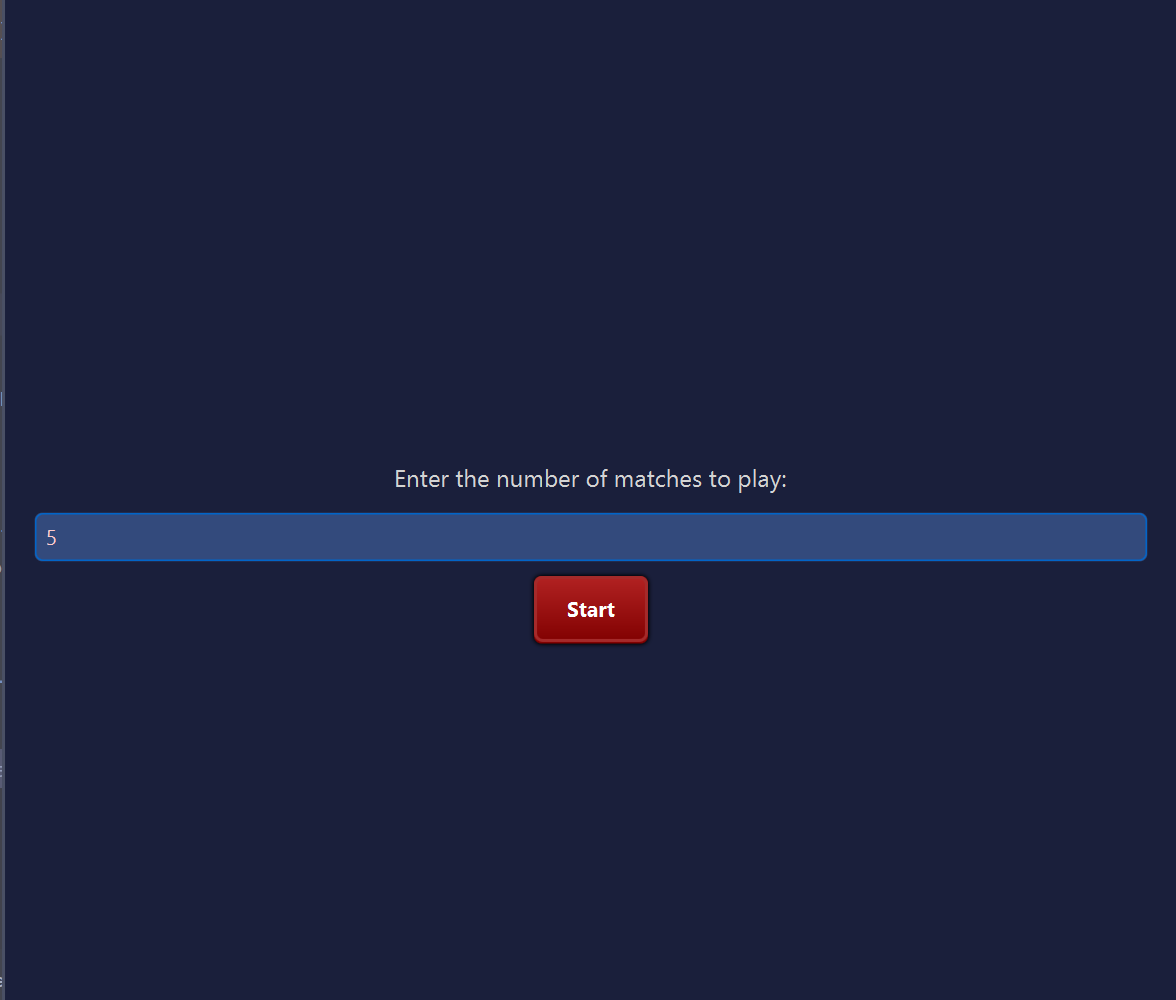
**CODE FOR NUMBER GUESSING GAME**

*package* com.example.semesterproject.NumberGuessingGame;  
  
*import* com.example.semesterproject.GameSelectionMenu.GameSelectionMenu;  
*import* javafx.application.Application;  
*import* javafx.application.Platform;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.layout.\*;  
*import* javafx.scene.media.Media;  
*import* javafx.scene.media.MediaPlayer;  
*import* javafx.stage.Stage;  
*import* com.example.semesterproject.GameSelectionMenu.GameSelectionMenu;  
*import* java.net.URL;  
*import* java.util.Random;  
  
*public class* NumberGuessingGame *extends* Application {  
 *private int* targetNumber;  
 *private int* remainingGuesses;  
 *private int* attempts;  
 *private* String username; *// Store username  
  
 public static void* main(String[] args) {  
 *launch*(args);  
 }  
  
 @Override  
 *public void* start(Stage primaryStage) {  
 showLoginWindow(primaryStage);  
 }  
  
 *private void* showLoginWindow(Stage stage) {  
 VBox loginLayout = *new* VBox(20);  
 loginLayout.setAlignment(Pos.CENTER);  
 loginLayout.getStyleClass().add("login-layout");  
  
 Label welcomeLabel = *new* Label("Welcome to the Number Guessing Game!");  
 welcomeLabel.getStyleClass().add("welcome-label");  
  
 TextField loginField = *new* TextField();  
 loginField.setPromptText("Enter your username");  
 loginField.getStyleClass().add("input-field");  
  
 Button loginButton = *new* Button("Username"); *// Changed button text* loginButton.getStyleClass().add("game-button");  
 Label warningLabel = *new* Label(""); *// Warning label for empty username* warningLabel.getStyleClass().add("warning-label");  
  
 loginButton.setOnAction(e -> {  
 String loginInput = loginField.getText().trim();  
 *if* (loginInput.isEmpty()) {  
 warningLabel.setText("Username cannot be empty. Please enter your username!");  
 } *else* {  
 warningLabel.setText(""); *// Clear warning if input is valid* username = loginInput; *// Save username* targetNumber = *new* Random().nextInt(100) + 1; *// Generate target number* showMainGameWindow(stage, 5); *// Start the game with 5 guesses* }  
 });  
  
 loginLayout.getChildren().addAll(welcomeLabel, loginField, loginButton, warningLabel);  
  
 Scene scene = *new* Scene(loginLayout, 800, 800);  
 applyStylesheet(scene);  
  
 stage.setTitle("Number Guessing Game");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 *private void* showMainGameWindow(Stage stage, *int* guesses) {  
 remainingGuesses = guesses;  
 attempts = 0;  
  
 VBox gameLayout = *new* VBox(15);  
 gameLayout.setAlignment(Pos.CENTER);  
 gameLayout.getStyleClass().add("game-layout");  
  
 Label instructionLabel = *new* Label("Guess the number between 1 and 100!");  
 instructionLabel.getStyleClass().add("instruction-label");  
  
 Label usernameLabel = *new* Label("Username: " + username); *// Display username* usernameLabel.getStyleClass().add("instruction-label");  
  
 Label guessesLabel = *new* Label("You have " + remainingGuesses + " guesses!");  
 guessesLabel.getStyleClass().add("guesses-label");  
  
 TextField guessField = *new* TextField();  
 guessField.setPromptText("Enter your guess (1-100)");  
 guessField.getStyleClass().add("input-field");  
  
 Media media = *new* Media(*this*.getClass().getResource("/tapSound.wav").toExternalForm());  
  
 Button submitButton = *new* Button("Submit");  
 submitButton.getStyleClass().add("game-button");  
  
 Label feedbackLabel = *new* Label("");  
 feedbackLabel.getStyleClass().add("feedback-label");  
  
 submitButton.setOnAction(e -> {  
  
 MediaPlayer mediaPlayer = *new* MediaPlayer(media);  
 mediaPlayer.play();  
 String input = guessField.getText();  
 *try* {  
 *int* guess = Integer.*parseInt*(input);  
 attempts++;  
 remainingGuesses--;  
  
 *if* (guess == targetNumber) {  
 showCongratulationsWindow(stage);  
 } *else if* (remainingGuesses == 0) {  
 *if* (guesses == 5) {  
 showBonusOfferWindow(stage);  
 } *else* {  
 showGameOverWindow(stage);  
 }  
 } *else* {  
 feedbackLabel.setText(guess < targetNumber ? "Too low! Try again." : "Too high! Try again.");  
 guessesLabel.setText("You have " + remainingGuesses + " guesses left!");  
 }  
 } *catch* (NumberFormatException ex) {  
 feedbackLabel.setText("Please enter a valid number!");  
 }  
 guessField.clear();  
 });  
  
 gameLayout.getChildren().addAll(instructionLabel, usernameLabel, guessesLabel, guessField, submitButton, feedbackLabel);  
  
 Scene scene = *new* Scene(gameLayout, 800, 800);  
 applyStylesheet(scene);  
  
 stage.setTitle("Number Guessing Game");  
 stage.setScene(scene);  
 }  
  
 *private void* showBonusOfferWindow(Stage stage) {  
 VBox bonusLayout = *new* VBox(20);  
 bonusLayout.setAlignment(Pos.CENTER);  
 bonusLayout.getStyleClass().add("bonus-layout");  
  
 Label bonusLabel = *new* Label("You are out of guesses! Would you like 3 bonus guesses?");  
 bonusLabel.getStyleClass().add("bonus-label");  
  
 Button yesButton = *new* Button("Yes");  
 yesButton.getStyleClass().add("game-button");  
 yesButton.setOnAction(e -> showMainGameWindow(stage, 3));  
  
 Button noButton = *new* Button("No");  
 noButton.getStyleClass().add("game-button");  
 noButton.setOnAction(e -> showGameOverWindow(stage));  
  
 bonusLayout.getChildren().addAll(bonusLabel, yesButton, noButton);  
  
 Scene scene = *new* Scene(bonusLayout, 800, 800);  
 applyStylesheet(scene);  
  
 stage.setTitle("Bonus Offer");  
 stage.setScene(scene);  
 }  
  
 *private void* showGameOverWindow(Stage stage) {  
 VBox gameOverLayout = *new* VBox(20);  
 gameOverLayout.setAlignment(Pos.CENTER);  
 gameOverLayout.getStyleClass().add("game-over-layout");  
  
 Label gameOverLabel = *new* Label("Game Over! The number was " + targetNumber + ".");  
 gameOverLabel.getStyleClass().add("game-over-label");  
  
 Button playAgainButton = *new* Button("Play Again");  
 playAgainButton.getStyleClass().add("game-button");  
 playAgainButton.setOnAction(e -> showLoginWindow(stage));  
  
 Button exitButton = *new* Button("Exit");  
 exitButton.getStyleClass().add("game-button");  
 exitButton.setOnAction(e -> {stage.close();  
 GameSelectionMenu menu = *new* GameSelectionMenu();  
 menu.start(*new* Stage());}  
 );  
  
 gameOverLayout.getChildren().addAll(gameOverLabel, playAgainButton, exitButton);  
  
 Scene scene = *new* Scene(gameOverLayout, 800, 800);  
 applyStylesheet(scene);  
  
 stage.setTitle("Game Over");  
 stage.setScene(scene);  
 }  
  
 *private void* showCongratulationsWindow(Stage stage) {  
 VBox congratsLayout = *new* VBox(20);  
 congratsLayout.setAlignment(Pos.CENTER);  
 congratsLayout.getStyleClass().add("congrats-layout");  
  
 Label congratsLabel = *new* Label("Congratulations, " + username + "! You guessed the number in " + attempts + " attempts!");  
 congratsLabel.getStyleClass().add("congrats-label");  
  
 Button playAgainButton = *new* Button("Play Again");  
 playAgainButton.getStyleClass().add("game-button");  
 playAgainButton.setOnAction(e -> showLoginWindow(stage));  
  
 Button exitButton = *new* Button("Exit");  
 exitButton.getStyleClass().add("game-button");  
 exitButton.setOnAction(e -> {stage.close();  
 GameSelectionMenu menu = *new* GameSelectionMenu();  
 menu.start(*new* Stage());}  
 );  
  
 congratsLayout.getChildren().addAll(congratsLabel, playAgainButton, exitButton);  
  
 Scene scene = *new* Scene(congratsLayout, 800, 800);  
 applyStylesheet(scene);  
  
 stage.setTitle("Congratulations");  
 stage.setScene(scene);  
 }  
  
 *private void* applyStylesheet(Scene scene) {  
 URL stylesheet = getClass().getResource("/com/example/semesterproject/NumberGuessingGame/Style.css");  
 *if* (stylesheet != *null*) {  
 scene.getStylesheets().add(stylesheet.toExternalForm());  
 } *else* {  
 System.out.println("Error: Style.css not found!");  
 }  
 }

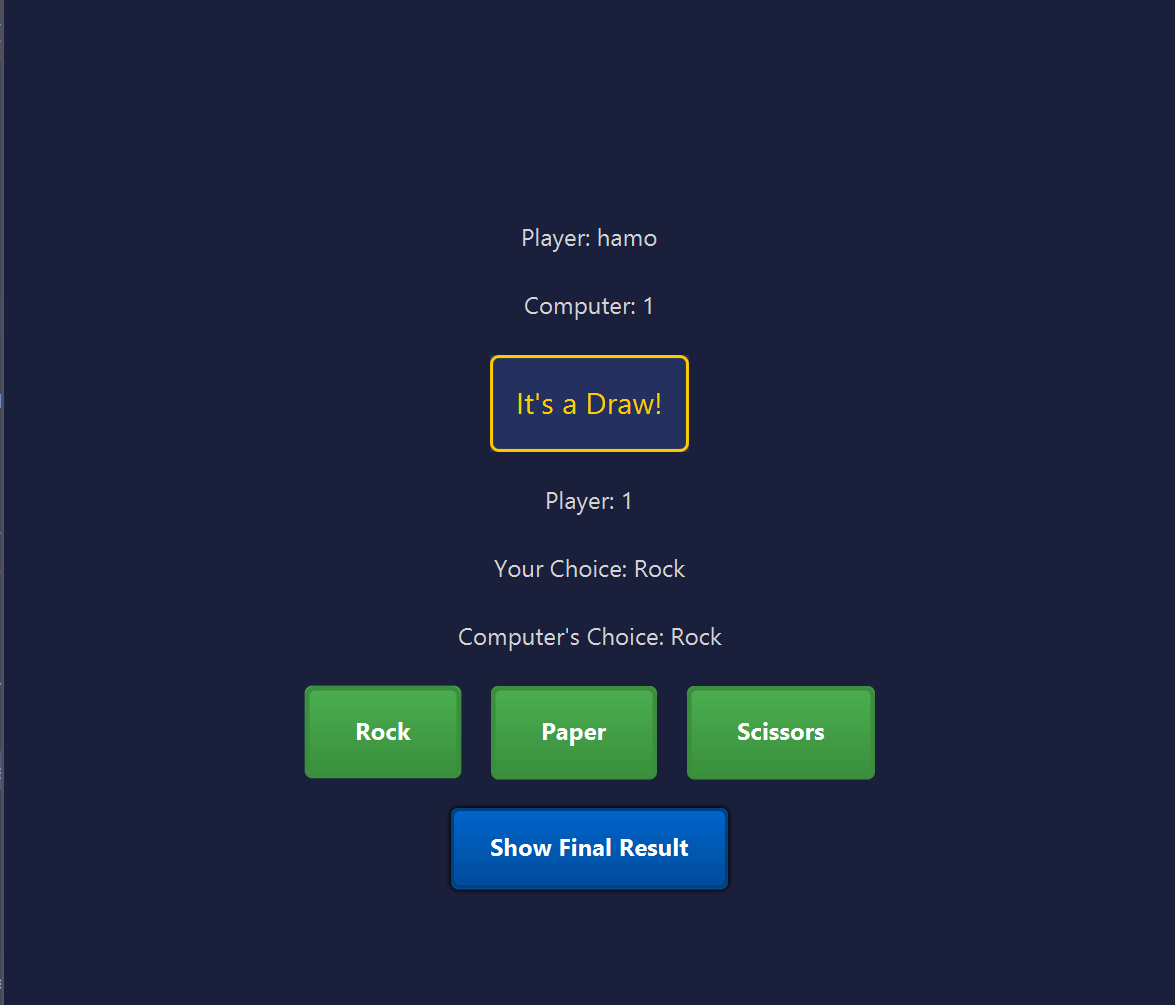
**ROCK PAPER SCISSOR**



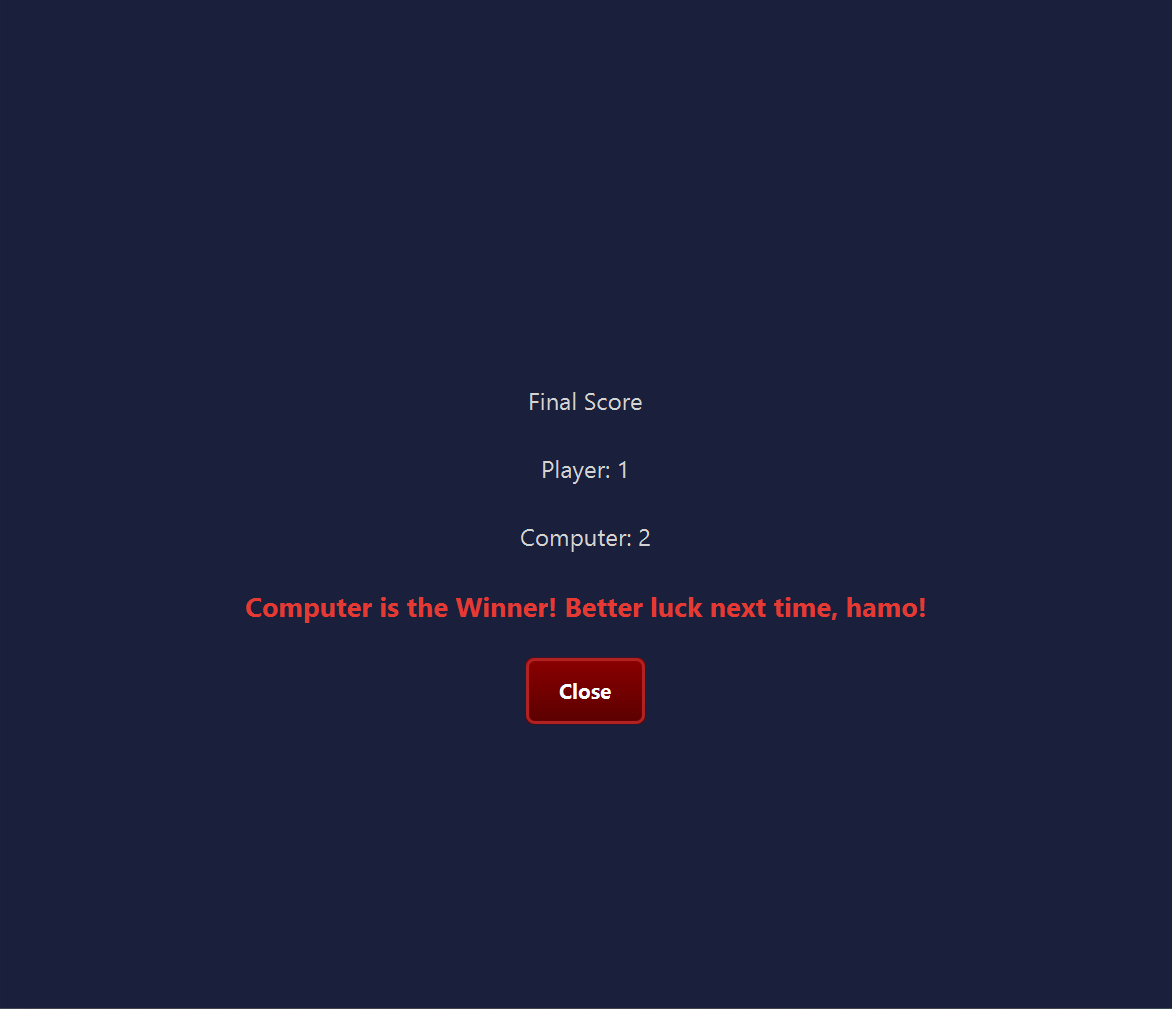
(Username window for taking input from user ).



(Window for selecting the number of matches to play).



(Main game window where user choose among Rock, Paper and Scissors and computer also choose one option and then winner gets the point).



(Congratulations window for congratulating the winner).

**Rock-Paper-Scissors Game**

**Main Menu Window**  
Step into the classic game of Rock-Paper-Scissors! This window provides an introduction and a "Start Game" button to launch the match. Users can also access instructions to learn the game rules.

**Gameplay Window**  
In this window, you play against the computer by selecting Rock, Paper, or Scissors. After making your choice, the computer reveals its move, and the winner is declared instantly based on the classic rules. The interface also keeps track of the score for multiple rounds.

**Result Window**  
This window announces the final winner after the chosen number of rounds is completed. It highlights the player's score, computer's score, and the overall outcome (win, loss, or draw). Users can choose to play again or return to the main menu.

**CODE FOR ROCK PAPER SCISSOR**

*package* com.example.semesterproject.RockPaperScissors;  
  
*import* com.example.semesterproject.GameSelectionMenu.GameSelectionMenu;  
*import* javafx.application.Application;  
*import* javafx.geometry.Pos;  
*import* javafx.scene.Scene;  
*import* javafx.scene.control.\*;  
*import* javafx.scene.layout.\*;  
*import* javafx.stage.Stage;  
*import* java.util.Objects;  
*import* java.util.Random;  
  
*public class* RockPaperScissors *extends* Application {  
  
 *private int* playerScore = 0;  
 *private int* computerScore = 0;  
 *private int* roundsToPlay = 0;  
 *private int* roundsPlayed = 0;  
 *private* String username = "";  
 *private* Stage primaryStage;  
 *private* Scene gameScene, finalResultScene;  
  
 @Override  
 *public void* start(Stage primaryStage) {  
 *this*.primaryStage = primaryStage;  
 showUsernameInputScene();  
 }  
  
 *private void* showUsernameInputScene() {  
 Label usernameLabel = *new* Label("Enter your username:");  
 usernameLabel.getStyleClass().add("label");  
 TextField usernameField = *new* TextField();  
 Button submitButton = *new* Button("Submit");  
 submitButton.getStyleClass().add("game-button");  
  
 VBox inputLayout = *new* VBox(10, usernameLabel, usernameField, submitButton);  
 inputLayout.setAlignment(Pos.CENTER);  
 inputLayout.setPrefSize(800, 800);  
 inputLayout.getStyleClass().add("input-layout");  
  
 submitButton.setOnAction(e -> {  
 username = usernameField.getText().trim();  
 *if* (username.isEmpty()) {  
 Alert alert = *new* Alert(Alert.AlertType.WARNING);  
 alert.setTitle("Warning");  
 alert.setHeaderText("Username Missing");  
 alert.setContentText("Please enter your username to proceed.");  
 alert.showAndWait();  
 } *else* {  
 showMatchInputScene();  
 }  
 });  
  
 Scene usernameScene = *new* Scene(inputLayout);  
 *try* {  
 usernameScene.getStylesheets().add(Objects.*requireNonNull*(getClass().getResource("/com/example/semesterproject/RockPaper/Style.css")).toExternalForm());  
 } *catch* (Exception i) {  
 System.out.println(i.getMessage());  
 }  
  
 primaryStage.setTitle("Rock Paper Scissors");  
 primaryStage.setScene(usernameScene);  
 primaryStage.show();  
 }  
  
 *private void* showMatchInputScene() {  
 Label label = *new* Label("Enter the number of matches to play:");  
 label.getStyleClass().add("label");  
 TextField inputField = *new* TextField("5");  
 Button submitButton = *new* Button("Start");  
 submitButton.getStyleClass().add("game-button");  
  
 VBox inputLayout = *new* VBox(10, label, inputField, submitButton);  
 inputLayout.setAlignment(Pos.CENTER);  
 inputLayout.setPrefSize(800, 800);  
 inputLayout.getStyleClass().add("input-layout");  
  
 submitButton.setOnAction(e -> {  
 *try* {  
 roundsToPlay = Integer.*parseInt*(inputField.getText());  
 *if* (roundsToPlay <= 0) *throw new* NumberFormatException();  
 } *catch* (NumberFormatException ex) {  
 System.out.println("Now the game will be started with default round count which is 5");  
 roundsToPlay = 5; *// Default to 5 if invalid input* }  
 showGameScene();  
 });  
  
 Scene inputScene = *new* Scene(inputLayout);  
 *try* {  
 inputScene.getStylesheets().add(Objects.*requireNonNull*(getClass().getResource("/com/example/semesterproject/RockPaper/Style.css")).toExternalForm());  
 } *catch* (Exception i) {  
 System.out.println(i.getMessage());  
 }  
  
 primaryStage.setTitle("Rock Paper Scissors - Start");  
 primaryStage.setScene(inputScene);  
 primaryStage.show();  
 }  
  
 *private void* showGameScene() {  
 Label usernameLabel = *new* Label("Player: " + username);  
 usernameLabel.getStyleClass().add("label");  
 Label computerLabel = *new* Label("Computer: 0");  
 computerLabel.getStyleClass().add("label");  
 Label playerLabel = *new* Label("Player: 0");  
 playerLabel.getStyleClass().add("label");  
 Label resultLabel = *new* Label("?");  
 resultLabel.getStyleClass().add("result-label");  
 Label playerChoiceLabel = *new* Label("Your Choice: ?");  
 playerChoiceLabel.getStyleClass().add("choice-label");  
 Label computerChoiceLabel = *new* Label("Computer's Choice: ?");  
 computerChoiceLabel.getStyleClass().add("choice-label");  
  
 Button rockButton = *new* Button("Rock");  
 rockButton.getStyleClass().add("rock-button");  
 Button paperButton = *new* Button("Paper");  
 paperButton.getStyleClass().add("paper-button");  
 Button scissorsButton = *new* Button("Scissors");  
 scissorsButton.getStyleClass().add("scissors-button");  
 Button showFinalResultButton = *new* Button("Show Final Result");  
 showFinalResultButton.getStyleClass().add("show-result-button");  
  
 rockButton.setOnAction(e -> playRound("Rock", computerLabel, playerLabel, resultLabel, playerChoiceLabel, computerChoiceLabel));  
 paperButton.setOnAction(e -> playRound("Paper", computerLabel, playerLabel, resultLabel, playerChoiceLabel, computerChoiceLabel));  
 scissorsButton.setOnAction(e -> playRound("Scissors", computerLabel, playerLabel, resultLabel, playerChoiceLabel, computerChoiceLabel));  
 showFinalResultButton.setOnAction(e -> showFinalResultScene());  
  
 HBox buttons = *new* HBox(20, rockButton, paperButton, scissorsButton);  
 buttons.setAlignment(Pos.CENTER);  
  
 VBox layout = *new* VBox(20, usernameLabel, computerLabel, resultLabel, playerLabel, playerChoiceLabel, computerChoiceLabel, buttons, showFinalResultButton);  
 layout.setAlignment(Pos.CENTER);  
 layout.setPrefSize(800, 800);  
 layout.getStyleClass().add("main-layout");  
  
 gameScene = *new* Scene(layout);  
 *try* {  
 gameScene.getStylesheets().add(Objects.*requireNonNull*(getClass().getResource("/com/example/semesterproject/RockPaper/Style.css")).toExternalForm());  
 } *catch* (Exception i) {  
 System.out.println(i.getMessage());  
 }  
  
 primaryStage.setTitle("Rock Paper Scissors - Game");  
 primaryStage.setScene(gameScene);  
 primaryStage.show();  
 }  
  
 *private void* playRound(String playerChoice, Label computerLabel, Label playerLabel, Label resultLabel, Label playerChoiceLabel, Label computerChoiceLabel) {  
 *if* (roundsPlayed >= roundsToPlay) {  
 showFinalResultScene();  
 *return*;  
 }  
  
 String[] options = {"Rock", "Paper", "Scissors"};  
 String computerChoice = options[*new* Random().nextInt(3)];  
  
 playerChoiceLabel.setText("Your Choice: " + playerChoice);  
 computerChoiceLabel.setText("Computer's Choice: " + computerChoice);  
  
 *if* (playerChoice.equals(computerChoice)) {  
 resultLabel.setText("It's a Draw!");  
 } *else if* ((playerChoice.equals("Rock") && computerChoice.equals("Scissors")) ||  
 (playerChoice.equals("Paper") && computerChoice.equals("Rock")) ||  
 (playerChoice.equals("Scissors") && computerChoice.equals("Paper"))) {  
 resultLabel.setText("You Win!");  
 playerScore++;  
 } *else* {  
 resultLabel.setText("Computer Wins!");  
 computerScore++;  
 }  
  
 computerLabel.setText("Computer: " + computerScore);  
 playerLabel.setText("Player: " + playerScore);  
 roundsPlayed++;  
 }  
  
 *private void* showFinalResultScene() {  
 Label finalResultLabel = *new* Label("Final Score");  
 finalResultLabel.getStyleClass().add("label");  
 Label playerScoreLabel = *new* Label("Player: " + playerScore);  
 playerScoreLabel.getStyleClass().add("label");  
 Label computerScoreLabel = *new* Label("Computer: " + computerScore);  
 computerScoreLabel.getStyleClass().add("label");  
  
 Label winnerLabel = *new* Label();  
 winnerLabel.getStyleClass().add("label");  
  
 *if* (playerScore > computerScore) {  
 winnerLabel.setText("Congratulations, " + username + "! You are the Winner!");  
 winnerLabel.setStyle("-fx-font-size: 18px; -fx-text-fill: #4caf50; -fx-font-weight: bold;");  
 } *else if* (computerScore > playerScore) {  
 winnerLabel.setText("Computer is the Winner! Better luck next time, " + username + "!");  
 winnerLabel.setStyle("-fx-font-size: 18px; -fx-text-fill: #e53935; -fx-font-weight: bold;");  
 } *else* {  
 winnerLabel.setText("It's a Draw! Both you and the computer have the same score.");  
 winnerLabel.setStyle("-fx-font-size: 18px; -fx-text-fill: #ffcc00; -fx-font-weight: bold;");  
 }  
  
 Button closeButton = *new* Button("Close");  
 closeButton.getStyleClass().add("close-button");  
 closeButton.setOnAction(e -> {primaryStage.close();  
 GameSelectionMenu menu = *new* GameSelectionMenu();  
 menu.start(*new* Stage());}  
 );  
  
 VBox layout = *new* VBox(20, finalResultLabel, playerScoreLabel, computerScoreLabel, winnerLabel, closeButton);  
 layout.setAlignment(Pos.CENTER);  
 layout.setPrefSize(800, 800);  
 layout.getStyleClass().add("result-layout");  
  
 finalResultScene = *new* Scene(layout);  
 *try* {  
 finalResultScene.getStylesheets().add(Objects.*requireNonNull*(getClass().getResource("/com/example/semesterproject/RockPaper/Style.css")).toExternalForm());  
 } *catch* (Exception i) {  
 System.out.println(i.getMessage());  
 }  
  
 primaryStage.setTitle("Rock Paper Scissors - Final Result");  
 primaryStage.setScene(finalResultScene);  
 primaryStage.show();  
 }  
  
 *public static void* main(String[] args) {  
 *launch*(args);}  
}