|  |  |
| --- | --- |
| **Name** | **Hammas Ullah** |
| **Reg No** | **Sp24-BSE-033** |
| **Section** | **BSE-B** |
| **Submitted to** | **Sir Shahid Bhatti** |

**Snake Game Code**

**Login Menu**

package com.example.snakegameproject;  
  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.layout.\*;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.stage.Stage;  
  
import java.io.\*;  
import java.util.ArrayList;  
import java.util.List;  
  
public class LoginMenu extends Application {  
  
  
 private List<User> users = new ArrayList<>();  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
  
 @Override  
 public void start(Stage primaryStage) {  
  
 loadUsers();  
// Image image = new Image("C:\\Users\\Home\\Pictures\\presentation\\downlaod.png");  
// StackPane stackPane = new StackPane();  
// // stackPane.getChildren().add(image);  
// primaryStage.getIcons().add(image);  
 primaryStage.setResizable(false);  
 showLoginScreen(primaryStage);  
 }  
  
 private void loadUsers() {  
 try (BufferedReader reader = new BufferedReader(new FileReader("user.txt"))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(" ");  
 if (parts.length == 2) {  
 users.add(new User(parts[0], parts[1]));  
 }  
 }  
 } catch (IOException e) {  
 System.err.println("Error loading user data: " + e.getMessage());  
 }  
 }  
  
 private void saveUser(String username, String password) {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter("user.txt", true))) {  
 writer.write(username + ":" + password);  
 writer.newLine();  
 } catch (IOException e) {  
 System.err.println("Error saving user data: " + e.getMessage());  
 }  
 }  
  
 private boolean userExists(String username) {  
 for (User user : users) {  
 if (user.getUsername().equals(username)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private boolean validateUser(String username, String password) {  
 for (User user : users) {  
 if (user.getUsername().equals(username) && user.getPassword().equals(password)) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private void showLoginScreen(Stage stage) {  
 GridPane grid = createStyledGrid();  
  
 Label title = createStyledLabel("Login to Snake Game", 20, Color.WHITE);  
 Label usernameLabel = createStyledLabel("Username:", 16, Color.WHITE);  
 Label passwordLabel = createStyledLabel("Password:", 16, Color.WHITE);  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter username");  
 usernameField.setStyle("-fx-background-color: #b49159; -fx-text-fill: white;");  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter password");  
 passwordField.setStyle("-fx-background-color: #b49159; -fx-text-fill: white;");  
  
 Button loginButton = createStyledButton("Login", Color.GREEN);  
 Button signUpButton = createStyledButton("Sign Up", Color.DODGERBLUE);  
 Label messageLabel = createStyledLabel("", 14, Color.RED);  
  
 loginButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
  
 if (validateUser(username, password)) {  
 messageLabel.setText("Login successful!");  
 SnakeGame game = new SnakeGame();  
 game.start(stage);  
 System.out.println("Logged in successfully!");  
 } else {  
 messageLabel.setText("Invalid username. Please sign up.");  
 }  
 });  
  
 signUpButton.setOnAction(e -> showSignUpScreen(stage));  
  
 VBox vbox = new VBox(10, title, grid, loginButton, signUpButton, messageLabel);  
 vbox.setAlignment(Pos.CENTER);  
 vbox.setPadding(new Insets(20));  
  
 grid.add(usernameLabel, 0, 0);  
 grid.add(usernameField, 1, 0);  
 grid.add(passwordLabel, 0, 1);  
 grid.add(passwordField, 1, 1);  
  
 Scene scene = createStyledScene(vbox, 400, 400);  
 stage.setScene(scene);  
 stage.setTitle("Snake Game Login");  
 stage.show();  
 }  
  
 private void showSignUpScreen(Stage stage) {  
 GridPane grid = createStyledGrid();  
  
 Label title = createStyledLabel("Sign Up", 20, Color.WHITE);  
 Label usernameLabel = createStyledLabel("Username:", 16, Color.WHITE);  
 Label passwordLabel = createStyledLabel("Password:", 16, Color.WHITE);  
  
 TextField usernameField = new TextField();  
 usernameField.setPromptText("Enter username");  
 usernameField.setStyle("-fx-background-color: #333; -fx-text-fill: white;");  
 PasswordField passwordField = new PasswordField();  
 passwordField.setPromptText("Enter password");  
 passwordField.setStyle("-fx-background-color: #333; -fx-text-fill: white;");  
  
 Button submitButton = createStyledButton("Submit", Color.GREEN);  
 Button backButton = createStyledButton("Back", Color.RED);  
 Label messageLabel = createStyledLabel("", 14, Color.RED);  
  
 submitButton.setOnAction(e -> {  
 String username = usernameField.getText();  
 String password = passwordField.getText();  
  
 if (userExists(username)) {  
 messageLabel.setText("User already exists. Please log in.");  
 } else {  
 users.add(new User(username, password));  
 saveUser(username, password);  
 messageLabel.setText("User saved. You can now log in.");  
 }  
 });  
  
 backButton.setOnAction(e -> showLoginScreen(stage));  
  
 VBox vbox = new VBox(10, title, grid, submitButton, backButton, messageLabel);  
 vbox.setAlignment(Pos.CENTER);  
 vbox.setPadding(new Insets(20));  
  
 grid.add(usernameLabel, 0, 0);  
 grid.add(usernameField, 1, 0);  
 grid.add(passwordLabel, 0, 1);  
 grid.add(passwordField, 1, 1);  
  
 Scene scene = createStyledScene(vbox, 400, 300);  
 stage.setScene(scene);  
 stage.setTitle("Sign Up");  
 }  
  
 private GridPane createStyledGrid() {  
 GridPane grid = new GridPane();  
 grid.setPadding(new Insets(20));  
 grid.setHgap(10);  
 grid.setVgap(10);  
 grid.setStyle("-fx-background-color: #222;");  
 return grid;  
 }  
  
 private Label createStyledLabel(String text, int fontSize, Color color) {  
 Label label = new Label(text);  
 label.setFont(Font.font("Segoe UI", FontWeight.BOLD, fontSize));  
 label.setTextFill(color);  
 return label;  
 }  
  
 private Button createStyledButton(String text, Color color) {  
 Button button = new Button(text);  
 button.setFont(Font.font("Segoe UI", FontWeight.SEMI\_BOLD, 14));  
 button.setStyle("-fx-background-color: " + toString(color) + "; -fx-text-fill: white; -fx-background-radius: 10;");  
 button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: white; -fx-text-fill: " + toString(color) + "; -fx-background-radius: 10;"));  
 button.setOnMouseExited(e -> button.setStyle("-fx-background-color: " + toString(color) + "; -fx-text-fill: white; -fx-background-radius: 10;"));  
 return button;  
 }  
  
 private Scene createStyledScene(Region content, int width, int height) {  
 StackPane root = new StackPane(content);  
 root.setStyle("-fx-background-color: #b49159;");  
 return new Scene(root, width, height);  
 }  
  
 private String toString(Color color) {  
 return "#" + color.toString().substring(2, 8).toUpperCase();  
 }  
  
 private static class User {  
 private final String username;  
 private final String password;  
  
 public User(String username, String password) {  
 this.username = username;  
 this.password = password;  
 }  
  
 public String getUsername() {  
 return username;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
 }  
}

**SnakeGame**

package com.example.snakegameproject;  
  
import javafx.animation.AnimationTimer;  
import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.scene.layout.Pane;  
import javafx.scene.paint.Color;  
import javafx.scene.shape.Circle;  
import javafx.scene.text.Text;  
import javafx.stage.Stage;  
import javafx.scene.input.KeyCode;  
import java.util.ArrayList;  
import java.util.Random;  
  
  
public class SnakeGame extends Application {  
  
 private static final int *WIDTH* = 600;  
 private static final int *HEIGHT* = 400;  
 private static final int *TILE\_SIZE* = 22;  
 private static final int *SPEED* = 6;  
  
 private ArrayList<Circle> snake = new ArrayList<>();  
 private Circle food = new Circle(*TILE\_SIZE* / 2, Color.*RED*);  
 private String direction = "RIGHT";  
 private boolean running = true;  
 private int score = 0;  
  
 private Text scoreText = new Text("Score: 0");  
 private Text gameOverText = new Text("Game Over!!! "+"Your Score is \n: "+score +" Press R to Restart");  
  
 private Pane pane = new Pane();  
  
 @Override  
 public void start(Stage stage) {  
 pane.setPrefSize(*WIDTH*, *HEIGHT*);  
  
 pane.setStyle("-fx-background-color: #2e2b28;");  
  
  
 Circle head = new Circle(*WIDTH* / 2, *HEIGHT* / 2, *TILE\_SIZE* / 2, Color.*GREEN*);  
 snake.add(head);  
 pane.getChildren().add(head);  
 generateFood();  
  
 scoreText.setFill(Color.*WHITE*);  
 scoreText.setX(20);  
 scoreText.setY(30);  
 pane.getChildren().add(scoreText);  
  
 gameOverText.setFill(Color.*WHITE*);  
 gameOverText.setX(*WIDTH* / 2 -100);  
 gameOverText.setY(*HEIGHT* /2 );  
 gameOverText.setVisible(false);  
 pane.getChildren().add(gameOverText);  
  
 AnimationTimer timer = new AnimationTimer() {  
 private long lastUpdateTime = 0;  
  
 @Override  
 public void handle(long now) {  
 if (running && now - lastUpdateTime >= 1000000000 / *SPEED*) {  
 lastUpdateTime = now;  
 update();  
 }  
 }  
 };  
  
 timer.start();  
  
 Scene scene = new Scene(pane);  
 scene.setOnKeyPressed(e -> {  
 KeyCode code = e.getCode();  
 if (code == KeyCode.*UP* && !direction.equals("DOWN")) direction = "UP";  
 else if (code == KeyCode.*DOWN* && !direction.equals("UP")) direction = "DOWN";  
 else if (code == KeyCode.*LEFT* && !direction.equals("RIGHT")) direction = "LEFT";  
 else if (code == KeyCode.*RIGHT* && !direction.equals("LEFT")) direction = "RIGHT";  
 else if (code == KeyCode.*R* && !running) restartGame();  
 });  
  
 stage.setScene(scene);  
 stage.setTitle("Snake Game");  
 stage.show();  
 }  
  
 private void update() {  
 Circle head = snake.get(0);  
 head.setFill(Color.*WHITE*);  
  
 Circle newHead = new Circle(*TILE\_SIZE* / 2, Color.*GREEN*);  
 switch (direction) {  
 case "UP": newHead.setCenterX(head.getCenterX()); newHead.setCenterY(head.getCenterY() - *TILE\_SIZE*); break;  
 case "DOWN": newHead.setCenterX(head.getCenterX()); newHead.setCenterY(head.getCenterY() + *TILE\_SIZE*); break;  
 case "LEFT": newHead.setCenterX(head.getCenterX() - *TILE\_SIZE*); newHead.setCenterY(head.getCenterY()); break;  
 case "RIGHT": newHead.setCenterX(head.getCenterX() + *TILE\_SIZE*); newHead.setCenterY(head.getCenterY()); break;  
 }  
  
 if (newHead.getCenterX() < 0) newHead.setCenterX(*WIDTH* - *TILE\_SIZE* / 2);  
 if (newHead.getCenterX() >= *WIDTH*) newHead.setCenterX(*TILE\_SIZE* / 2);  
 if (newHead.getCenterY() < 0) newHead.setCenterY(*HEIGHT* - *TILE\_SIZE* / 2);  
 if (newHead.getCenterY() >= *HEIGHT*) newHead.setCenterY(*TILE\_SIZE* / 2);  
  
 snake.add(0, newHead);  
 pane.getChildren().add(newHead);  
  
 if (Math.*abs*(newHead.getCenterX() - food.getCenterX()) < *TILE\_SIZE* &&  
 Math.*abs*(newHead.getCenterY() - food.getCenterY()) < *TILE\_SIZE*) {  
 score++;  
 updateScore();  
 generateFood();  
 } else {  
 Circle tail = snake.remove(snake.size() - 1);  
 pane.getChildren().remove(tail);  
 }  
  
 if (isCollidingWithSelf()) {  
 running = false;  
 displayGameOver();  
 }  
 }  
 private void displayGameOver() {  
 gameOverText.setText("Game Over!!! "+"Your Score is: "+score +"\n\n Press R to Restart");  
 gameOverText.setVisible(true);  
 }  
  
  
 private boolean isCollidingWithSelf() {  
 Circle head = snake.get(0);  
  
 for (int i = 1; i < snake.size(); i++) {  
 if (head.getCenterX() == snake.get(i).getCenterX() && head.getCenterY() == snake.get(i).getCenterY()) {  
 return true;  
 }  
 }  
 return false;  
 }  
  
 private void generateFood() {  
 Random rand = new Random();  
 food.setCenterX(rand.nextInt(*WIDTH* / *TILE\_SIZE*) \* *TILE\_SIZE* + *TILE\_SIZE* / 2);  
 food.setCenterY(rand.nextInt(*HEIGHT* / *TILE\_SIZE*) \* *TILE\_SIZE* + *TILE\_SIZE* / 2);  
 pane.getChildren().remove(food);  
 pane.getChildren().add(food);  
 }  
  
 private void updateScore() {  
 scoreText.setText("Score: " + score);  
 }  
  
 private void restartGame() {  
 snake.clear();  
 pane.getChildren().clear();  
 score = 0;  
 direction = "RIGHT";  
 running = true;  
 scoreText.setText("Score: " + score);  
 gameOverText.setVisible(false);  
  
 Circle head = new Circle(*WIDTH* / 2, *HEIGHT* / 2, *TILE\_SIZE* / 2, Color.*GREEN*);  
 snake.add(head);  
 pane.getChildren().add(head);  
  
 generateFood();  
 }  
  
  
  
 public static void main(String[] args) {  
 *launch*();  
 }  
}