**Comsats University Islamabad, Lahore Campus**

**Course Title: Object Oriented Programming**

**Assignment #4**

**Submitted by: Attika Hussain (SP24-BSE-017)**

**Fatima Chaudhry (SP24-BSE-024)**

**Uswa Maryam (SP24-BSE-121)**

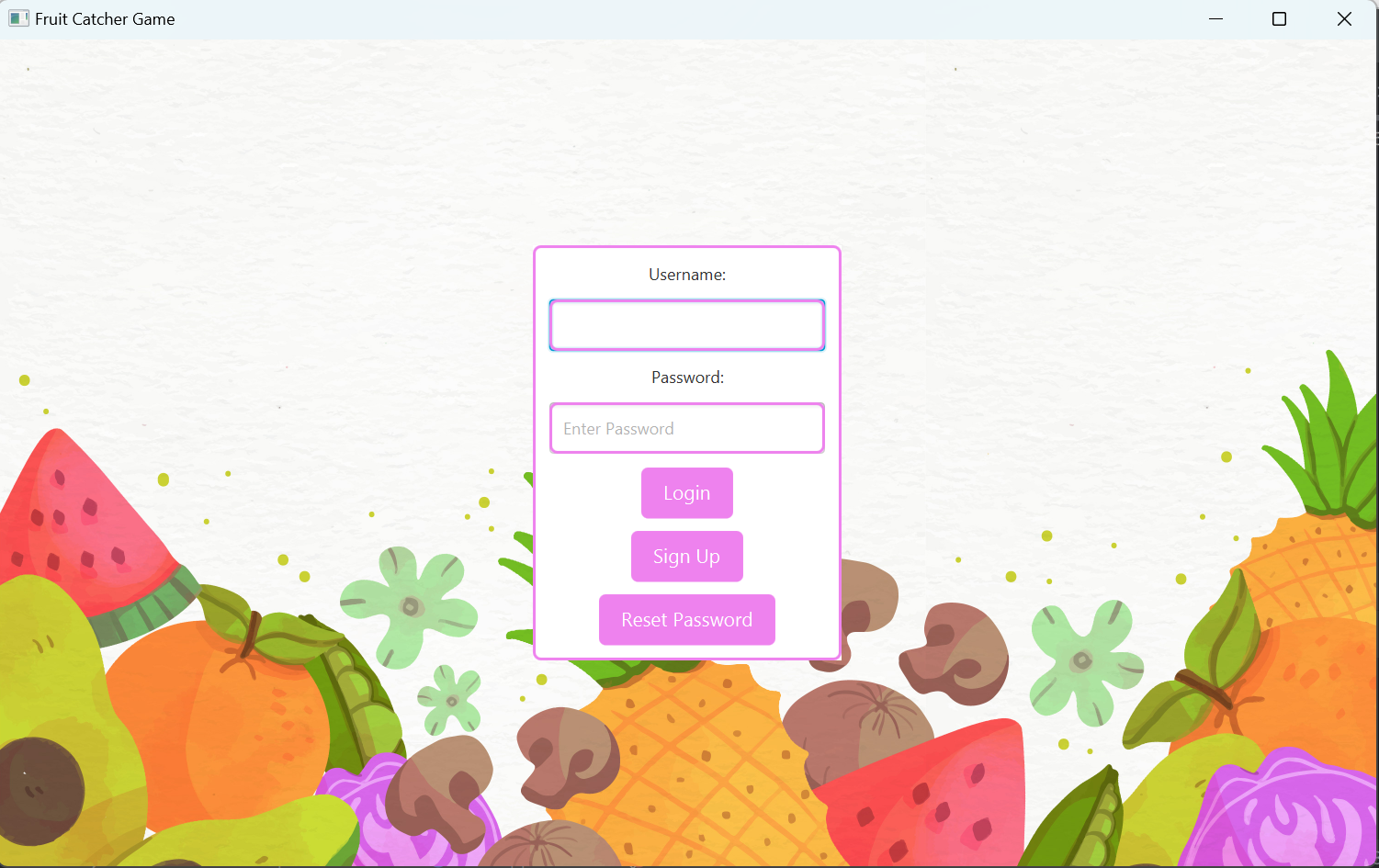
**Section: A**

**Submitted to: Sir Shahid Bhatti**

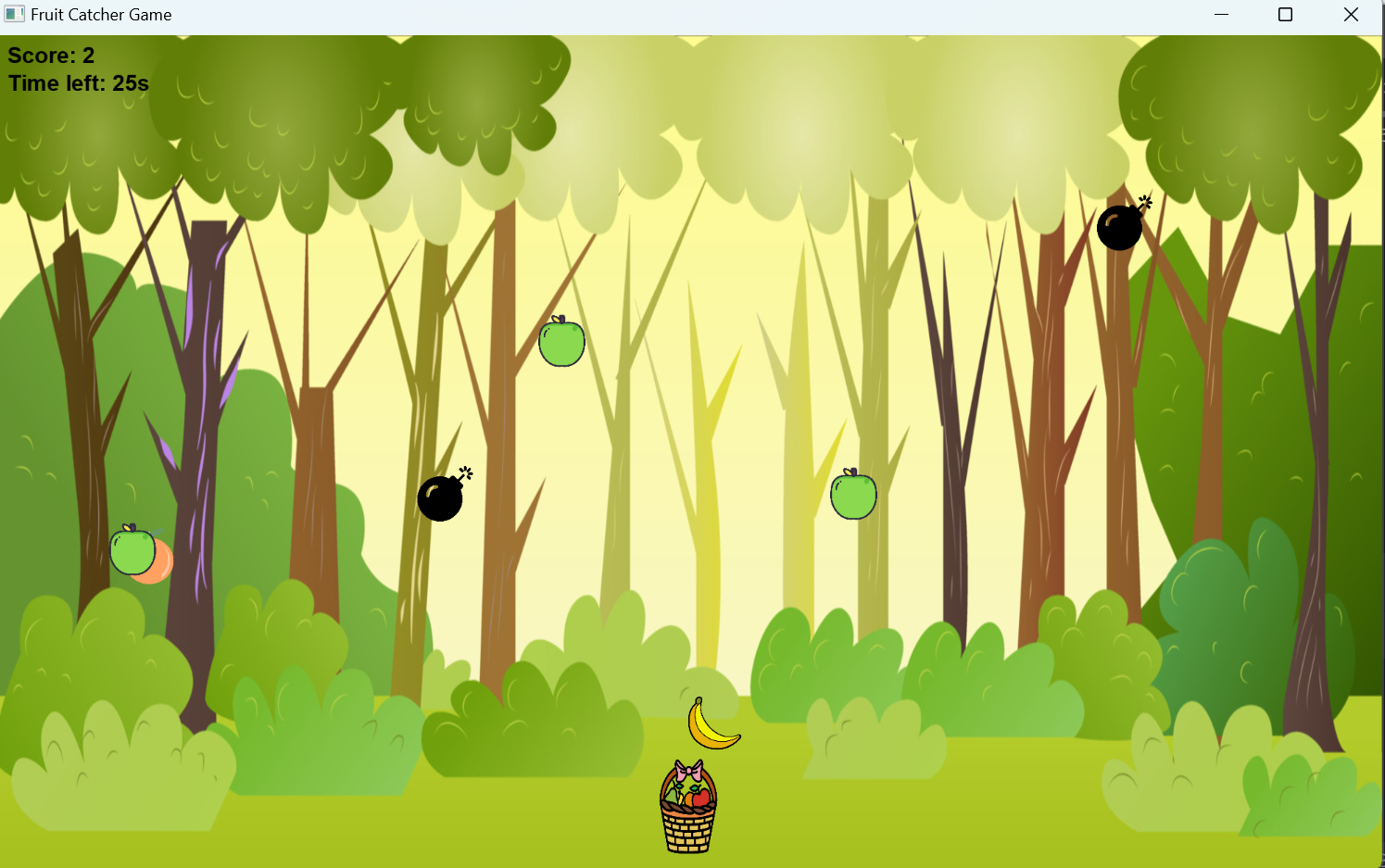
**Submission Date: 13th December 2024**

**Fruit Catcher Game**

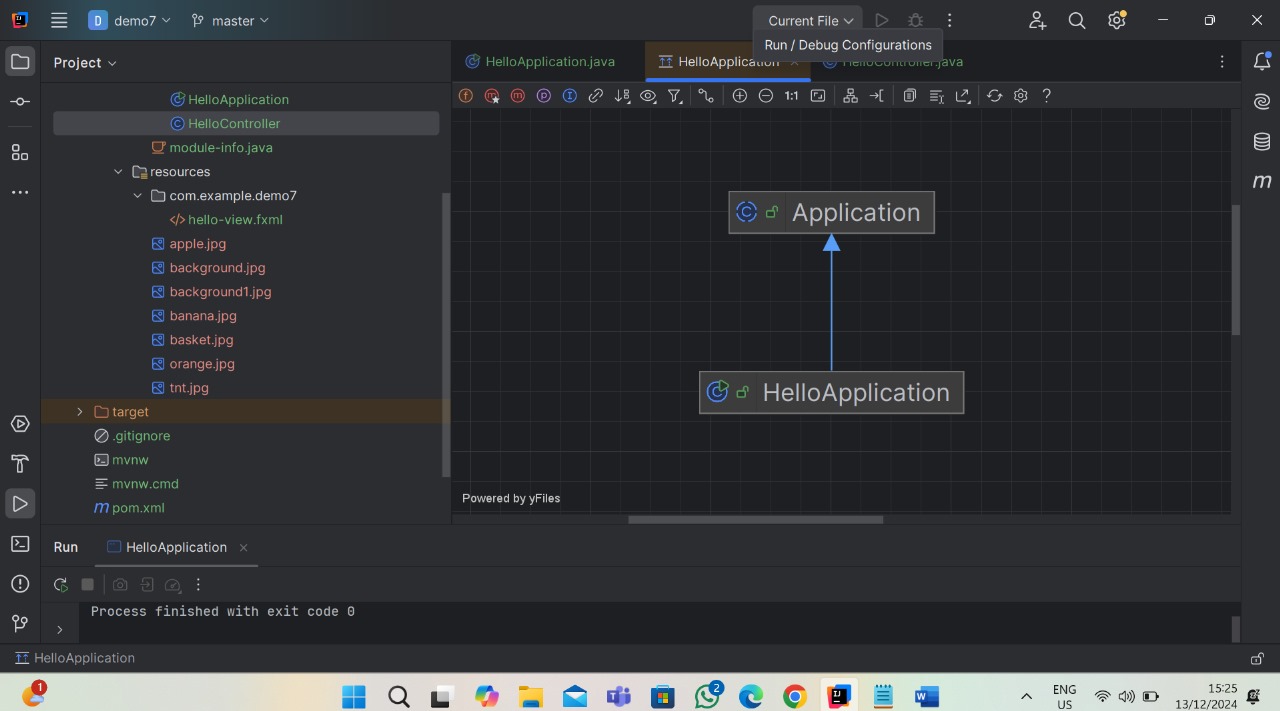
* Login Page containing text fields username and passwords and buttons for signing up, log in and resetting password.

****

* This is the game layout. The fruits and bombs are falling using animation timer. The basket at the bottom is supposed to catch 10 fruits within 30 seconds while avoiding the bombs. If the bomb touches the basket or 10 fruits are not caught within 30 seconds the Game ends.



**UML Diagram**

****

**Code**

**package com.example.demo7;  
  
import javafx.animation.AnimationTimer;  
import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.geometry.Pos;  
import javafx.scene.Scene;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.canvas.GraphicsContext;  
import javafx.scene.control.\*;  
import javafx.scene.image.Image;  
import javafx.scene.input.KeyCode;  
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.\*;  
  
public class HelloApplication extends Application {  
  
 private Map<String, String> userDatabase = new HashMap<>();  
 private final String FILE\_NAME = "user\_data.txt";  
  
 private static final int *WIDTH* = 1000;  
 private static final int *HEIGHT* = 600;  
 private static final int *BASKET\_WIDTH* = 50;  
 private static final int *BASKET\_HEIGHT* = 70;  
 private static final int *FRUIT\_SIZE* = 40;  
 private static final int *BOMB\_SIZE* = 40;  
 private static final int *TARGET\_SCORE* = 10;  
 private static final int *GAME\_DURATION* = 30\_000;  
  
 private double basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 private double basketY = *HEIGHT* - *BASKET\_HEIGHT* - 10;  
  
 private List<GameObject> fruits = new ArrayList<>();  
 private List<GameObject> bombs = new ArrayList<>();  
  
 private boolean leftPressed = false;  
 private boolean rightPressed = false;  
  
 private int score = 0;  
 private boolean gameOver = false;  
 private boolean gameWon = false;  
  
 private Random random = new Random();  
 private Image appleImage = new Image(getClass().getResourceAsStream("/apple.png"));  
 private Image bananaImage = new Image(getClass().getResourceAsStream("/banana.png"));  
 private Image orangeImage = new Image(getClass().getResourceAsStream("/orange.png"));  
 private Image basketImage = new Image(getClass().getResourceAsStream("/basket.png"));  
 private Image tntImage = new Image(getClass().getResourceAsStream("/tnt.png"));  
 private Image backgroundImage = new Image(getClass().getResourceAsStream("/background.jpg"));  
  
 private long startTime;  
  
 @Override  
 public void start(Stage primaryStage) {  
 loadUserData();  
  
 primaryStage.setTitle("Fruit Catcher Game");  
  
 // Login Page  
 GridPane loginPane = createPaneWithBackground("background1.jpg");  
 VBox loginBox = createStyledBox();  
  
 TextField loginUsername = new TextField();  
 loginUsername.setPromptText("Enter Username");  
 styleTextField(loginUsername);  
  
 PasswordField loginPassword = new PasswordField();  
 loginPassword.setPromptText("Enter Password");  
 styleTextField(loginPassword);  
  
 Button loginButton = createStyledButton("Login");  
 Button signUpButton = createStyledButton("Sign Up");  
 Button resetPasswordButton = createStyledButton("Reset Password");  
  
 loginBox.getChildren().addAll(  
 new Label("Username:"),  
 loginUsername,  
 new Label("Password:"),  
 loginPassword,  
 loginButton,  
 signUpButton,  
 resetPasswordButton  
 );  
  
 loginPane.add(loginBox, 0, 0);  
 Scene loginScene = new Scene(loginPane, 1000, 600);  
  
 // Sign-Up Page  
 GridPane signUpPane = createPaneWithBackground("background1.jpg");  
 VBox signUpBox = createStyledBox();  
  
 TextField signUpUsername = new TextField();  
 signUpUsername.setPromptText("Choose Username");  
 styleTextField(signUpUsername);  
  
 PasswordField signUpPassword = new PasswordField();  
 signUpPassword.setPromptText("Choose Password");  
 styleTextField(signUpPassword);  
  
 Button registerButton = createStyledButton("Register");  
 Button backToLoginButton1 = createStyledButton("Back to Login");  
  
 signUpBox.getChildren().addAll(  
 new Label("Username:"),  
 signUpUsername,  
 new Label("Password:"),  
 signUpPassword,  
 registerButton,  
 backToLoginButton1  
 );  
  
 signUpPane.add(signUpBox, 0, 0);  
 Scene signUpScene = new Scene(signUpPane, 1000, 600);  
  
 // Reset Password Page  
 GridPane resetPasswordPane = createPaneWithBackground("background1.jpg");  
 VBox resetPasswordBox = createStyledBox();  
  
 TextField resetUsername = new TextField();  
 resetUsername.setPromptText("Enter Username");  
 styleTextField(resetUsername);  
  
 PasswordField newPassword = new PasswordField();  
 newPassword.setPromptText("Enter New Password");  
 styleTextField(newPassword);  
  
 Button resetButton = createStyledButton("Reset Password");  
 Button backToLoginButton2 = createStyledButton("Back to Login");  
  
 resetPasswordBox.getChildren().addAll(  
 new Label("Username:"),  
 resetUsername,  
 new Label("New Password:"),  
 newPassword,  
 resetButton,  
 backToLoginButton2  
 );  
  
 resetPasswordPane.add(resetPasswordBox, 0, 0);  
 Scene resetPasswordScene = new Scene(resetPasswordPane, 1000, 600);  
  
 // Fruit Game Page  
 Pane gameRoot = new Pane();  
 Canvas canvas = new Canvas(*WIDTH*, *HEIGHT*);  
 GraphicsContext gc = canvas.getGraphicsContext2D();  
 gameRoot.getChildren().add(canvas);  
 Scene gameScene = new Scene(gameRoot);  
  
 gameScene.setOnKeyPressed(event -> {  
 if (event.getCode() == KeyCode.*LEFT*) {  
 leftPressed = true;  
 } else if (event.getCode() == KeyCode.*RIGHT*) {  
 rightPressed = true;  
 }  
 });  
  
 gameScene.setOnKeyReleased(event -> {  
 if (event.getCode() == KeyCode.*LEFT*) {  
 leftPressed = false;  
 } else if (event.getCode() == KeyCode.*RIGHT*) {  
 rightPressed = false;  
 }  
 });  
  
 loginButton.setOnAction(e -> {  
 String username = loginUsername.getText();  
 String password = loginPassword.getText();  
  
 if (userDatabase.containsKey(username) && userDatabase.get(username).equals(password)) {  
 showAlert("Success", "Login successful!");  
 resetGame();  
 primaryStage.setScene(gameScene);  
 startGame(gc);  
 } else {  
 showAlert("Error", "Invalid credentials.");  
 }  
 });  
  
 signUpButton.setOnAction(e -> primaryStage.setScene(signUpScene));  
  
 resetPasswordButton.setOnAction(e -> primaryStage.setScene(resetPasswordScene));  
  
 registerButton.setOnAction(e -> {  
 String username = signUpUsername.getText();  
 String password = signUpPassword.getText();  
  
 if (username.isEmpty() || password.isEmpty()) {  
 showAlert("Error", "Fields cannot be empty.");  
 } else if (userDatabase.containsKey(username)) {  
 showAlert("Error", "Username already exists.");  
 } else {  
 userDatabase.put(username, password);  
 saveUserData();  
 showAlert("Success", "Registration successful!");  
 primaryStage.setScene(loginScene);  
 }  
 });  
  
 backToLoginButton1.setOnAction(e -> primaryStage.setScene(loginScene));  
  
 resetButton.setOnAction(e -> {  
 String username = resetUsername.getText();  
 String password = newPassword.getText();  
  
 if (username.isEmpty() || password.isEmpty()) {  
 showAlert("Error", "Fields cannot be empty.");  
 } else if (!userDatabase.containsKey(username)) {  
 showAlert("Error", "Username does not exist.");  
 } else {  
 userDatabase.put(username, password);  
 saveUserData();  
 showAlert("Success", "Password reset successful!");  
 primaryStage.setScene(loginScene);  
 }  
 });  
  
 backToLoginButton2.setOnAction(e -> primaryStage.setScene(loginScene));  
  
 primaryStage.setScene(loginScene);  
 primaryStage.show();  
 }  
  
 private void startGame(GraphicsContext gc) {  
 startTime = System.*currentTimeMillis*();  
 AnimationTimer timer = new AnimationTimer() {  
 @Override  
 public void handle(long now) {  
 if (!gameOver && !gameWon) {  
 updateGame();  
 renderGame(gc);  
 }  
 }  
 };  
 timer.start();  
 }  
  
 private void resetGame() {  
 basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 fruits.clear();  
 bombs.clear();  
 score = 0;  
 gameOver = false;  
 gameWon = false;  
 }  
  
 private void updateGame() {  
 long elapsedTime = System.*currentTimeMillis*() - startTime;  
 if (elapsedTime > *GAME\_DURATION*) {  
 gameOver = true;  
 }  
  
 if (score >= *TARGET\_SCORE*) {  
 gameWon = true;  
 }  
  
 if (leftPressed) {  
 basketX -= 5;  
 if (basketX < 0) {  
 basketX = 0;  
 }  
 }  
  
 if (rightPressed) {  
 basketX += 5;  
 if (basketX > *WIDTH* - *BASKET\_WIDTH*) {  
 basketX = *WIDTH* - *BASKET\_WIDTH*;  
 }  
 }  
  
 if (random.nextInt(100) < 3) {  
 String fruitType = getRandomFruitType();  
 fruits.add(new GameObject(random.nextInt(*WIDTH* - *FRUIT\_SIZE*), 0, *FRUIT\_SIZE*, *FRUIT\_SIZE*, fruitType));  
 }  
  
 if (random.nextInt(100) < 2) {  
 bombs.add(new GameObject(random.nextInt(*WIDTH* - *BOMB\_SIZE*), 0, *BOMB\_SIZE*, *BOMB\_SIZE*, "tnt"));  
 }  
  
 Iterator<GameObject> fruitIterator = fruits.iterator();  
 while (fruitIterator.hasNext()) {  
 GameObject fruit = fruitIterator.next();  
 fruit.y += 5;  
  
 if (fruit.y > *HEIGHT*) {  
 fruitIterator.remove();  
 } else if (fruit.intersects(basketX, basketY, *BASKET\_WIDTH*, *BASKET\_HEIGHT*)) {  
 score++;  
 fruitIterator.remove();  
 }  
 }  
  
 Iterator<GameObject> bombIterator = bombs.iterator();  
 while (bombIterator.hasNext()) {  
 GameObject bomb = bombIterator.next();  
 bomb.y += 5;  
  
 if (bomb.y > *HEIGHT*) {  
 bombIterator.remove();  
 } else if (bomb.intersects(basketX, basketY, *BASKET\_WIDTH*, *BASKET\_HEIGHT*)) {  
 gameOver = true;  
 bombIterator.remove();  
 }  
 }  
 }  
  
 private void renderGame(GraphicsContext gc) {  
 gc.drawImage(backgroundImage, 0, 0, *WIDTH*, *HEIGHT*);  
 gc.drawImage(basketImage, basketX, basketY, *BASKET\_WIDTH*, *BASKET\_HEIGHT*);  
  
 for (GameObject fruit : fruits) {  
 switch (fruit.type) {  
 case "apple" -> gc.drawImage(appleImage, fruit.x, fruit.y, fruit.width, fruit.height);  
 case "banana" -> gc.drawImage(bananaImage, fruit.x, fruit.y, fruit.width, fruit.height);  
 case "orange" -> gc.drawImage(orangeImage, fruit.x, fruit.y, fruit.width, fruit.height);  
 }  
 }  
  
 for (GameObject bomb : bombs) {  
 gc.drawImage(tntImage, bomb.x, bomb.y, bomb.width, bomb.height);  
 }  
  
 gc.setFill(Color.*BLACK*);  
 gc.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 16));  
 gc.fillText("Score: " + score, 10, 20);  
 gc.fillText("Time left: " + Math.*max*(0, (*GAME\_DURATION* - (System.*currentTimeMillis*() - startTime)) / 1000) + "s", 10, 40);  
  
 if (gameOver) {  
 gc.setFill(Color.*RED*);  
 gc.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 gc.fillText("Game Over!", *WIDTH* / 2 - 60, *HEIGHT* / 2);  
 }  
  
 if (gameWon) {  
 gc.setFill(Color.*GREEN*);  
 gc.setFont(Font.*font*("Arial", FontWeight.*BOLD*, 24));  
 gc.fillText("You Win!", *WIDTH* / 2 - 60, *HEIGHT* / 2);  
 }  
 }  
  
 private String getRandomFruitType() {  
 int randomIndex = random.nextInt(3);  
 return switch (randomIndex) {  
 case 0 -> "apple";  
 case 1 -> "banana";  
 case 2 -> "orange";  
 default -> "apple";  
 };  
 }  
  
 private static class GameObject {  
 double x, y, width, height;  
 String type;  
  
 GameObject(double x, double y, double width, double height, String type) {  
 this.x = x;  
 this.y = y;  
 this.width = width;  
 this.height = height;  
 this.type = type;  
 }  
  
 boolean intersects(double otherX, double otherY, double otherWidth, double otherHeight) {  
 return x < otherX + otherWidth && x + width > otherX && y < otherY + otherHeight && y + height > otherY;  
 }  
 }  
  
 private void loadUserData() {  
 try (BufferedReader reader = new BufferedReader(new FileReader(FILE\_NAME))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] parts = line.split(",");  
 if (parts.length == 2) {  
 userDatabase.put(parts[0], parts[1]);  
 }  
 }  
 } catch (IOException e) {  
 System.*out*.println("No user data found, starting fresh.");  
 }  
 }  
  
 private void saveUserData() {  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(FILE\_NAME))) {  
 for (Map.Entry<String, String> entry : userDatabase.entrySet()) {  
 writer.write(entry.getKey() + "," + entry.getValue());  
 writer.newLine();  
 }  
 } catch (IOException e) {  
 System.*out*.println("Error saving user data: " + e.getMessage());  
 }  
 }  
  
 private GridPane createPaneWithBackground(String imageName) {  
 GridPane pane = new GridPane();  
 pane.setPadding(new Insets(10, 10, 10, 10));  
 pane.setAlignment(Pos.*CENTER*);  
 Image image = new Image(imageName);  
 BackgroundImage bgImage = new BackgroundImage(  
 image,  
 BackgroundRepeat.*NO\_REPEAT*,  
 BackgroundRepeat.*NO\_REPEAT*,  
 BackgroundPosition.*CENTER*,  
 new BackgroundSize(BackgroundSize.*AUTO*, BackgroundSize.*AUTO*, false, false, true, true)  
 );  
 pane.setBackground(new Background(bgImage));  
 return pane;  
 }  
  
 private VBox createStyledBox() {  
 VBox box = new VBox(10);  
 box.setAlignment(Pos.*CENTER*);  
 box.setPadding(new Insets(20));  
 box.setStyle("-fx-background-color: white; -fx-border-color: violet; -fx-border-width: 2px; -fx-border-radius: 5px; -fx-padding: 10px;");  
 return box;  
 }  
  
 private Button createStyledButton(String text) {  
 Button button = new Button(text);  
 button.setStyle("-fx-background-color: violet; -fx-text-fill: white; -fx-font-size: 14px; -fx-padding: 8px 16px; -fx-border-radius: 5px; -fx-background-radius: 5px;");  
 return button;  
 }  
  
 private void styleTextField(TextField textField) {  
 textField.setStyle("-fx-padding: 8px; -fx-border-color: violet; -fx-border-width: 2px; -fx-border-radius: 5px;");  
 textField.setPrefWidth(200);  
 }  
  
 private void showAlert(String title, String message) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}**