**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**

**Assigned On: 6th December 2024**

**Submission Date: 20th December 2024**

package com.example.demo10;  
  
import javafx.application.Application;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
public class HelloApplication extends Application {  
  
 @Override  
 public void start(Stage primaryStage) {  
  
 UserManager userManager = new UserManager();  
  
  
 SceneManager sceneManager = new SceneManager(primaryStage, userManager);  
  
  
 GameManager gameManager = new GameManager(sceneManager);  
 Scene gameScene = gameManager.createGameScene();  
 sceneManager.setGameScene(gameScene);  
  
  
 primaryStage.setTitle("Fruit Catcher Game");  
 primaryStage.setScene(sceneManager.getLoginScene());  
 primaryStage.setResizable(false);  
 primaryStage.show();  
  
  
 primaryStage.sceneProperty().addListener((observable, oldScene, newScene) -> {  
 if (newScene == gameScene) {  
 gameManager.startGame();  
 }  
 });  
 }  
  
 public static void main(String[] args) {  
 *launch*(args);  
 }  
}

package com.example.demo10;  
  
import javafx.fxml.FXML;  
import javafx.scene.control.Label;  
  
public class HelloController {  
 @FXML  
 private Label welcomeText;  
  
 @FXML  
 protected void onHelloButtonClick() {  
 welcomeText.setText("Welcome to JavaFX Application!");  
 }  
}

package com.example.demo10;  
  
import javafx.animation.AnimationTimer;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.canvas.GraphicsContext;  
import javafx.scene.image.Image;  
import javafx.scene.input.KeyCode;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.scene.Scene;  
import javafx.scene.layout.Pane;  
  
import java.util.\*;  
  
public class GameManager {  
  
 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;  
 private double basketY;  
 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 List<GameObject> fruits = new ArrayList<>();  
 private List<GameObject> bombs = new ArrayList<>();  
  
 private Image appleImage;  
 private Image bananaImage;  
 private Image orangeImage;  
 private Image basketImage;  
 private Image tntImage;  
 private Image backgroundImage;  
  
 private long startTime;  
 private Canvas canvas;  
 private GraphicsContext gc;  
  
 private int level = 1; // Default level  
 private SceneManager sceneManager; // Reference to SceneManager  
  
 public GameManager(SceneManager sceneManager) {  
 this.sceneManager = sceneManager;  
  
 basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 basketY = *HEIGHT* - *BASKET\_HEIGHT* - 10;  
  
 appleImage = new Image(getClass().getResourceAsStream("/apple.png"));  
 bananaImage = new Image(getClass().getResourceAsStream("/banana.png"));  
 orangeImage = new Image(getClass().getResourceAsStream("/orange.png"));  
 basketImage = new Image(getClass().getResourceAsStream("/basket.png"));  
 tntImage = new Image(getClass().getResourceAsStream("/tnt.png"));  
 backgroundImage = new Image(getClass().getResourceAsStream("/background.jpg"));  
  
 canvas = new Canvas(*WIDTH*, *HEIGHT*);  
 gc = canvas.getGraphicsContext2D();  
 }  
  
 public void setLevel(int level) {  
 this.level = level;  
 resetGame();  
 }  
  
 public Scene createGameScene() {  
 Pane gameRoot = new Pane();  
 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;  
 }  
 });  
  
 return gameScene;  
 }  
  
 public void startGame() {  
 resetGame();  
 startTime = System.*currentTimeMillis*();  
 AnimationTimer timer = new AnimationTimer() {  
 @Override  
 public void handle(long now) {  
 if (!gameOver && !gameWon) {  
 updateGame();  
 renderGame();  
 }  
 }  
 };  
 timer.start();  
 }  
  
 private void resetGame() {  
 basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 fruits.clear();  
 bombs.clear();  
 score = 0;  
 gameOver = false;  
 gameWon = false;  
  
 }  
  
  
 private void increaseFruitSpeed(int speed) {  
 fruits.forEach(fruit -> fruit.y += speed);  
 }  
  
 private void increaseBombSpawnRate(int rate) {  
 for (int i = 0; i < rate; i++) {  
 bombs.add(new GameObject(random.nextInt(*WIDTH* - *BOMB\_SIZE*), 0, *BOMB\_SIZE*, *BOMB\_SIZE*, "tnt"));  
 }  
 }  
  
 private void updateGame() {  
 long elapsedTime = System.*currentTimeMillis*() - startTime;  
 if (elapsedTime > *GAME\_DURATION*) {  
 gameOver = true;  
 endGame();  
 }  
  
 if (score >= *TARGET\_SCORE*) {  
 gameWon = true;  
 endGame();  
 }  
  
 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 + level)) {  
 String fruitType = getRandomFruitType();  
 fruits.add(new GameObject(random.nextInt(*WIDTH* - *FRUIT\_SIZE*), 0, *FRUIT\_SIZE*, *FRUIT\_SIZE*, fruitType));  
 }  
  
 if (random.nextInt(100) < (2 + level)) {  
 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();  
 endGame();  
 }  
 }  
 }  
  
 private void renderGame() {  
 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 void endGame() {  
 renderGame();  
  
 javafx.animation.Timeline timeline = new javafx.animation.Timeline(  
 new javafx.animation.KeyFrame(  
 javafx.util.Duration.*seconds*(2),  
 event -> sceneManager.redirectToLevelSelection()  
 )  
 );  
 timeline.setCycleCount(1);  
 timeline.play();  
 }  
  
 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;  
 }  
 }  
}

package com.example.demo10;  
  
import javafx.animation.AnimationTimer;  
import javafx.scene.canvas.Canvas;  
import javafx.scene.canvas.GraphicsContext;  
import javafx.scene.image.Image;  
import javafx.scene.input.KeyCode;  
import javafx.scene.paint.Color;  
import javafx.scene.text.Font;  
import javafx.scene.text.FontWeight;  
import javafx.scene.Scene;  
import javafx.scene.layout.Pane;  
  
import java.util.\*;  
  
public class MediumLevelGameManager {  
  
 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 int TARGET\_SCORE ;  
 private int GAME\_DURATION ;  
  
 private double basketX;  
 private double basketY;  
 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 List<GameObject> fruits = new ArrayList<>();  
 private List<GameObject> bombs = new ArrayList<>();  
  
 private Image appleImage;  
 private Image bananaImage;  
 private Image orangeImage;  
 private Image basketImage;  
 private Image tntImage;  
 private Image backgroundImage;  
  
 private long startTime;  
 private Canvas canvas;  
 private GraphicsContext gc;  
  
 private int level;  
 private SceneManager sceneManager;  
  
 public MediumLevelGameManager(SceneManager sceneManager, int level) {  
 this.sceneManager = sceneManager;  
 this.level = level;  
  
 basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 basketY = *HEIGHT* - *BASKET\_HEIGHT* - 10;  
  
 appleImage = new Image(getClass().getResourceAsStream("/apple.png"));  
 bananaImage = new Image(getClass().getResourceAsStream("/banana.png"));  
 orangeImage = new Image(getClass().getResourceAsStream("/orange.png"));  
 basketImage = new Image(getClass().getResourceAsStream("/basket.png"));  
 tntImage = new Image(getClass().getResourceAsStream("/tnt.png"));  
 //backgroundImage = new Image(getClass().getResourceAsStream("/background\_medium.jpg"));  
 // backgroundImage = new Image(getClass().getResourceAsStream("/background\_hard.jpg"));  
 if (level == 2) {  
 backgroundImage = new Image(getClass().getResourceAsStream("/background\_medium.jpg"));  
 } else if (level == 3) {  
 backgroundImage = new Image(getClass().getResourceAsStream("/background\_hard.jpg"));  
 }  
  
  
  
 canvas = new Canvas(*WIDTH*, *HEIGHT*);  
 gc = canvas.getGraphicsContext2D();  
 }  
  
 public Scene createGameScene1() {  
 Pane gameRoot = new Pane();  
 gameRoot.getChildren().add(canvas);  
  
 Scene gameScene1 = new Scene(gameRoot);  
 gameScene1.setOnKeyPressed(event -> {  
 if (event.getCode() == KeyCode.*LEFT*) {  
 leftPressed = true;  
 } else if (event.getCode() == KeyCode.*RIGHT*) {  
 rightPressed = true;  
 }  
 });  
  
 gameScene1.setOnKeyReleased(event -> {  
 if (event.getCode() == KeyCode.*LEFT*) {  
 leftPressed = false;  
 } else if (event.getCode() == KeyCode.*RIGHT*) {  
 rightPressed = false;  
 }  
 });  
  
 return gameScene1;  
 }  
  
 public void startGame() {  
 resetGame();  
 startTime = System.*currentTimeMillis*();  
 AnimationTimer timer = new AnimationTimer() {  
 @Override  
 public void handle(long now) {  
 if (!gameOver && !gameWon) {  
 updateGame();  
 renderGame();  
 }  
 }  
 };  
 timer.start();  
 }  
  
 private void resetGame() {  
 basketX = *WIDTH* / 2 - *BASKET\_WIDTH* / 2;  
 fruits.clear();  
 bombs.clear();  
 score = 0;  
 gameOver = false;  
 gameWon = false;  
 adjustDifficulty();  
 }  
  
 private void adjustDifficulty() {  
 switch (level) {  
 case 2 -> {  
  
 GAME\_DURATION = 20\_000; // 20 seconds  
 TARGET\_SCORE = 15;  
 increaseFruitSpeed(7);  
 increaseBombSpawnRate(7);  
 }  
 case 3 -> {  
  
 GAME\_DURATION = 15\_000; // 15 seconds  
 TARGET\_SCORE = 20;  
 increaseFruitSpeed(8);  
 increaseBombSpawnRate(8);  
 }  
 }  
 }  
  
 private void increaseFruitSpeed(int speed) {  
 fruits.forEach(fruit -> fruit.y += speed);  
 }  
  
 private void increaseBombSpawnRate(int rate) {  
 for (int i = 0; i < rate; i++) {  
 bombs.add(new GameObject(random.nextInt(*WIDTH* - *BOMB\_SIZE*), 0, *BOMB\_SIZE*, *BOMB\_SIZE*, "tnt"));  
 }  
 }  
  
 private void updateGame() {  
 long elapsedTime = System.*currentTimeMillis*() - startTime;  
 if (elapsedTime > GAME\_DURATION) {  
 gameOver = true;  
 endGame();  
 }  
  
 if (score >= TARGET\_SCORE) {  
 gameWon = true;  
 endGame();  
 }  
  
 if (leftPressed) {  
 basketX -= 7; // Faster movement  
 if (basketX < 0) {  
 basketX = 0;  
 }  
 }  
  
 if (rightPressed) {  
 basketX += 7;  
 if (basketX > *WIDTH* - *BASKET\_WIDTH*) {  
 basketX = *WIDTH* - *BASKET\_WIDTH*;  
 }  
 }  
  
 if (random.nextInt(100) < (3 + level)) {  
 String fruitType = getRandomFruitType();  
 fruits.add(new GameObject(random.nextInt(*WIDTH* - *FRUIT\_SIZE*), 0, *FRUIT\_SIZE*, *FRUIT\_SIZE*, fruitType));  
 }  
  
 if (random.nextInt(100) < (2 + level)) {  
 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 + level;  
  
 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 + level;  
  
 if (bomb.y > *HEIGHT*) {  
 bombIterator.remove();  
 } else if (bomb.intersects(basketX, basketY, *BASKET\_WIDTH*, *BASKET\_HEIGHT*)) {  
 gameOver = true;  
 bombIterator.remove();  
 endGame();  
 }  
 }  
 }  
  
 private void renderGame() {  
 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 void endGame() {  
 renderGame();  
  
 javafx.animation.Timeline timeline = new javafx.animation.Timeline(  
 new javafx.animation.KeyFrame(  
 javafx.util.Duration.*seconds*(2),  
 event -> sceneManager.redirectToLevelSelection()  
 )  
 );  
 timeline.setCycleCount(1);  
 timeline.play();  
 }  
  
 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;  
 }  
 }  
}

package com.example.demo10;  
  
import javafx.geometry.Insets;  
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.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.scene.image.Image;  
import javafx.scene.layout.\*;  
import javafx.stage.Stage;  
  
import java.util.HashMap;  
import java.util.Map;  
  
public class SceneManager {  
  
 private final Stage stage;  
 private final UserManager userManager;  
 private Scene loginScene, signUpScene, resetPasswordScene, levelSelectionScene, mediumLevelScene, hardLevelScene, gameScene,gameScene1;  
 private Map<String, String> userDatabase = new HashMap<>();  
  
 public SceneManager(Stage stage, UserManager userManager) {  
 this.stage = stage;  
 this.userManager = userManager;  
 initializeScenes();  
 }  
  
 private void initializeScenes() {  
 loginScene = createLoginScene();  
 signUpScene = createSignUpScene();  
 resetPasswordScene = createResetPasswordScene();  
 levelSelectionScene = createLevelSelectionScene();  
  
 }  
  
 public Scene getLoginScene() {  
 return loginScene;  
 }  
  
 public Scene getSignUpScene() {  
 return signUpScene;  
 }  
  
 public Scene getResetPasswordScene() {  
 return resetPasswordScene;  
 }  
  
 public Scene getLevelSelectionScene() {  
 return levelSelectionScene;  
 }  
  
 public Scene getMediumLevelScene() {  
 return mediumLevelScene;  
 }  
  
 public Scene getHardLevelScene() {  
 return hardLevelScene;  
 }  
  
 public void setGameScene(Scene gameScene) {  
 this.gameScene = gameScene;  
 }  
 public void setGameScene1(Scene gameScene) {  
 this.gameScene = gameScene;  
 }  
  
 private Scene createLoginScene() {  
 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");  
  
 loginButton.setOnAction(e -> {  
 String username = loginUsername.getText();  
 String password = loginPassword.getText();  
  
 if (userDatabase.containsKey(username) && userDatabase.get(username).equals(password)) {  
 stage.setScene(getLevelSelectionScene());  
 } else {  
 showAlert("Error", "Invalid credentials.");  
 }  
 });  
  
 signUpButton.setOnAction(e -> stage.setScene(getSignUpScene()));  
 resetPasswordButton.setOnAction(e -> stage.setScene(getResetPasswordScene()));  
  
 loginBox.getChildren().addAll(  
 new Label("Username:"),  
 loginUsername,  
 new Label("Password:"),  
 loginPassword,  
 loginButton,  
 signUpButton,  
 resetPasswordButton  
 );  
  
 loginPane.add(loginBox, 0, 0);  
 return new Scene(loginPane, 1000, 600);  
 }  
  
 private Scene createSignUpScene() {  
 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");  
  
 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);  
 showAlert("Success", "Registration successful!");  
 stage.setScene(getLoginScene());  
 }  
 });  
  
 backToLoginButton1.setOnAction(e -> stage.setScene(getLoginScene()));  
  
 signUpBox.getChildren().addAll(  
 new Label("Username:"),  
 signUpUsername,  
 new Label("Password:"),  
 signUpPassword,  
 registerButton,  
 backToLoginButton1  
 );  
  
 signUpPane.add(signUpBox, 0, 0);  
 return new Scene(signUpPane, 1000, 600);  
 }  
  
 private Scene createResetPasswordScene() {  
 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");  
  
 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);  
 showAlert("Success", "Password reset successful!");  
 stage.setScene(getLoginScene());  
 }  
 });  
  
 backToLoginButton2.setOnAction(e -> stage.setScene(getLoginScene()));  
  
 resetPasswordBox.getChildren().addAll(  
 new Label("Username:"),  
 resetUsername,  
 new Label("New Password:"),  
 newPassword,  
 resetButton,  
 backToLoginButton2  
 );  
  
 resetPasswordPane.add(resetPasswordBox, 0, 0);  
 return new Scene(resetPasswordPane, 1000, 600);  
 }  
  
  
 private Scene createLevelSelectionScene() {  
 GridPane levelPane = createPaneWithBackground("background1.jpg");  
 VBox levelBox = createStyledBox();  
  
 Label selectLevelLabel = new Label("Select a Level:");  
 selectLevelLabel.setStyle("-fx-font-size: 18px; -fx-font-weight: bold;");  
  
 Button easyButton = createStyledButton("Easy");  
 Button mediumButton = createStyledButton("Medium");  
 Button hardButton = createStyledButton("Hard");  
  
 easyButton.setOnAction(e -> {  
 stage.setScene(gameScene); // Assuming gameScene is set up for Easy  
 });  
  
 mediumButton.setOnAction(e -> {  
 MediumLevelGameManager mediumGame = new MediumLevelGameManager(this, 2);  
 stage.setScene(mediumGame.createGameScene1());  
 mediumGame.startGame();  
 });  
  
 hardButton.setOnAction(e -> {  
 MediumLevelGameManager hardGame = new MediumLevelGameManager(this, 3);  
 stage.setScene(hardGame.createGameScene1());  
 hardGame.startGame();  
 });  
  
 levelBox.getChildren().addAll(  
 selectLevelLabel,  
 easyButton,  
 mediumButton,  
 hardButton  
 );  
  
 levelPane.add(levelBox, 0, 0);  
 return new Scene(levelPane, 1000, 600);  
 }  
  
  
 private GridPane createPaneWithBackground(String backgroundImage) {  
 GridPane pane = new GridPane();  
 pane.setPadding(new Insets(10, 10, 10, 10));  
 pane.setAlignment(Pos.*CENTER*);  
 Image image = new Image(backgroundImage);  
 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 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 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 showAlert(String title, String message) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle(title);  
 alert.setHeaderText(null);  
 alert.setContentText(message);  
 alert.showAndWait();  
 }  
  
 public void redirectToLevelSelection() {  
 stage.setScene(getLevelSelectionScene());  
 }  
}

package com.example.demo10;  
  
import java.io.\*;  
import java.util.HashMap;  
import java.util.Map;  
  
  
public class UserManager {  
  
 private static final String *USER\_DATA\_FILE* = "user\_data.txt";  
 private Map<String, String> users;  
  
 public UserManager() {  
 users = new HashMap<>();  
 loadUserData();  
 }  
  
   
 public void loadUserData() {  
 File file = new File(*USER\_DATA\_FILE*);  
 if (!file.exists()) {  
 return;  
 }  
  
 try (BufferedReader reader = new BufferedReader(new FileReader(file))) {  
 String line;  
 while ((line = reader.readLine()) != null) {  
 String[] user = line.split(":");  
 if (user.length == 2) {  
 users.put(user[0], user[1]);  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 public void saveUserData() {  
 File file = new File(*USER\_DATA\_FILE*);  
 try {  
 if (!file.exists()) {  
 file.createNewFile();  
 }  
  
 try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {  
 for (Map.Entry<String, String> entry : users.entrySet()) {  
 writer.write(entry.getKey() + ":" + entry.getValue());  
 writer.newLine();  
 }  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
  
 public void addUser(String username, String password) {  
 if (!users.containsKey(username)) {  
 users.put(username, password);  
 saveUserData();  
 }  
 }  
  
  
 public boolean validateLogin(String username, String password) {  
 return users.containsKey(username) && users.get(username).equals(password);  
 }  
  
  
  
}

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.geometry.Insets?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.layout.VBox?>  
  
<?import javafx.scene.control.Button?>  
<VBox alignment="CENTER" spacing="20.0" xmlns:fx="http://javafx.com/fxml"  
 fx:controller="com.example.demo10.HelloController">  
 <padding>  
 <Insets bottom="20.0" left="20.0" right="20.0" top="20.0"/>  
 </padding>  
  
 <Label fx:id="welcomeText"/>  
 <Button text="Hello!" onAction="#onHelloButtonClick"/>  
</VBox>

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.example</groupId>  
 <artifactId>demo10</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <name>demo10</name>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <junit.version>5.10.2</junit.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.openjfx</groupId>  
 <artifactId>javafx-controls</artifactId>  
 <version>22.0.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.openjfx</groupId>  
 <artifactId>javafx-fxml</artifactId>  
 <version>22.0.1</version>  
 </dependency>  
 <dependency>  
 <groupId>org.controlsfx</groupId>  
 <artifactId>controlsfx</artifactId>  
 <version>11.2.1</version>  
 </dependency>  
 <dependency>  
 <groupId>com.dlsc.formsfx</groupId>  
 <artifactId>formsfx-core</artifactId>  
 <version>11.6.0</version>  
 <exclusions>  
 <exclusion>  
 <groupId>org.openjfx</groupId>  
 <artifactId>\*</artifactId>  
 </exclusion>  
 </exclusions>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-api</artifactId>  
 <version>${junit.version}</version>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.junit.jupiter</groupId>  
 <artifactId>junit-jupiter-engine</artifactId>  
 <version>${junit.version}</version>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.13.0</version>  
 <configuration>  
 <source>22</source>  
 <target>22</target>  
 </configuration>  
 </plugin>  
 <plugin>  
 <groupId>org.openjfx</groupId>  
 <artifactId>javafx-maven-plugin</artifactId>  
 <version>0.0.8</version>  
 <executions>  
 <execution>  
 <!-- Default configuration for running with: mvn clean javafx:run -->  
 <id>default-cli</id>  
 <configuration>  
 <mainClass>com.example.demo10/com.example.demo10.HelloApplication</mainClass>  
 <launcher>app</launcher>  
 <jlinkZipName>app</jlinkZipName>  
 <jlinkImageName>app</jlinkImageName>  
 <noManPages>true</noManPages>  
 <stripDebug>true</stripDebug>  
 <noHeaderFiles>true</noHeaderFiles>  
 </configuration>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
</project>

module com.example.demo10 {  
 requires javafx.controls;  
 requires javafx.fxml;  
  
 requires org.controlsfx.controls;  
 requires com.dlsc.formsfx;  
  
 opens com.example.demo10 to javafx.fxml;  
 exports com.example.demo10;  
  
  
}