Application Program Development

Segment : Working with Multiple Views

Mahboob Ali

Outcomes

- Creating multiple views in JavaFX
- How to work with Multiple views
- Communicating with multiple views

Introduction

- Thus far our graphical interfaces have always included only one view.
- Now, we'll become familiar with user interfaces containing multiple views.
- Generally, the views are created as Scene-objects and the transitioning between them happens with events bound to the application.
- The example below has two Scene objects which both have their own content and an event related to the content.
- Instead of having an object for laying out components (such as BorderPane) in the example Scene objects, both objects have just one user interface component.

Example – two button on same scene

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
public class MultiScene extends Application {
  @Override
  public void start(Stage ps) {
      Button sc1 = new Button("Scene 1 ..");
      Button sc2 = new Button("Scene 2.");
      Scene first = new Scene(sc1);
      Scene second = new Scene(sc2);

      sc1.setOnAction((event) -> {
          ps.setScene(second);
      });
      sc2.setOnAction((event) -> {
          ps.setScene(first);
      });
      ps.setScene(first);
      ps.show();
  }
  public static void main(String[] args) {      launch(args);  } }
```

One layout for each view

- Let's get familiar with an example containing two different views.
- In the first view user is asked to input a password.
- If the user types a wrong password, the application informs the user about the mistake.
- If the user types the correct password, the application switches to the next view.
- Switching between views happens like in the previous example.
- The concrete switching event has been bound to the login button.
- When pressing the button, the application checks the password typed to the password field – here we're hoping that the user writes "password".
- If the password is correct, the view of the window is changed. In our example the view only includes the text "Welcome, this is the beginning!".

```java
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.PasswordField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class TwoLayoutsOneStage extends Application {
        @Override
        public void start(Stage ps) throws Exception {
        // 1. Creating the view for asking a password
                Label instructionText = new Label("Write the password and press Log in");
                PasswordField passwordField = new PasswordField();
                Button startButton = new Button("Log in");
                Label errorMessage = new Label("");
                // Layout 1
                GridPane layout = new GridPane();
                layout.add(instructionText, 0, 0);
                layout.add(passwordField, 0, 1);
                layout.add(startButton, 0, 2);
                layout.add(errorMessage, 0, 3);
```

```
        layout.setPrefSize(300, 180);
        layout.setAlignment(Pos.CENTER);
        layout.setVgap(10);
        layout.setHgap(10);
        layout.setPadding(new Insets(20, 20, 20, 20));

        Scene passwordView = new Scene(layout);
        Label welcomeText = new Label("Welcome, this is the beginning!");
        // Layout 2
        StackPane welcomeLayout = new StackPane();
        welcomeLayout.setPrefSize(300, 180);
        welcomeLayout.getChildren().add(welcomeText);
        welcomeLayout.setAlignment(Pos.CENTER);

        Scene welcomeView = new Scene(welcomeLayout);
        startButton.setOnAction((event) -> {
                if (!passwordField.getText().trim().equals("password")) {
                        errorMessage.setText("Unknown password!");
                        return;
                }
        window.setScene(welcomeView);
        });
        window.setScene(passwordView);
        window.show();
        }
public static void main(String[] args) { launch(args); } }
```

View with same layout

- Sometimes one wants an application to have a permanent view whose parts are swapped when needed.
- Typically applications that have some kind of menus function in this manner.
- In the example below, there is a application which contains a main menu and an area with variable content.
- When pressing the buttons on the main menu the the content of the content area changes.

```
class CirclePane extends StackPane {
        private Circle circle = new Circle(50);

        public CirclePane() {
                getChildren().add(circle);
                circle.setStroke(Color.BLACK);
                circle.setFill(Color.WHITE);
        }
        public void enlarge() {
                circle.setRadius(circle.getRadius() + 2);
        }
        public void shrink() {
                circle.setRadius(circle.getRadius() > 2 ? circle.getRadius() - 2 :
                                                        circle.getRadius());
        }
}
```

View with same layout

```
public class ControlCircleWithMouseAndKey extends Application {
        private CirclePane circlePane = new CirclePane();


        @Override // Override the start method in the Application class
        public void start(Stage primaryStage) {
                // Hold two buttons in an HBox
                HBox hBox = new HBox();
                hBox.setSpacing(10);
                hBox.setAlignment(Pos.CENTER);
                Button btEnlarge = new Button("Enlarge");
                Button btShrink = new Button("Shrink");
                hBox.getChildren().add(btEnlarge);
                hBox.getChildren().add(btShrink);
                // Create and register the handler
                btEnlarge.setOnAction(e -> circlePane.enlarge());
                btShrink.setOnAction(e -> circlePane.shrink());

                BorderPane borderPane = new BorderPane();
                borderPane.setCenter(circlePane);
                borderPane.setBottom(hBox);
                BorderPane.setAlignment(hBox, Pos.CENTER);
```

View with same layout

```
        // Create a scene and place it in the stage
        Scene scene = new Scene(borderPane, 200, 150);
        primaryStage.setTitle("ControlCircle"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage

        circlePane.setOnMouseClicked(e -> {
            if (e.getButton() == MouseButton.PRIMARY) {
                circlePane.enlarge();
            } else if (e.getButton() == MouseButton.SECONDARY) {
                circlePane.shrink();
            }
        });
        scene.setOnKeyPressed(e -> {
            if (e.getCode() == KeyCode.UP) { circlePane.enlarge(); }
            else if (e.getCode() == KeyCode.DOWN) { circlePane.shrink(); }
        });
    }

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