**Application Program Development: Grouping Components Together**

**Learning Outcomes**

1. **Understanding of Grouping Components**: Learn how to effectively group UI components.
2. **Designing Simple Applications**: Create simple applications using grouped components to reduce code redundancy and improve maintainability.

**Introduction to Organizing Components**

When developing applications with graphical user interfaces (GUI), it's important to keep components organized. Often, components are reused across different parts of the application. For example, you might need to enter a user's name, address, and phone number in multiple windows. By grouping these components together, you can simplify your code and enhance reusability.

**Benefits of Grouping Components**

1. **Code Reusability**: By creating reusable component groups, you can avoid writing the same code multiple times.
2. **Maintainability**: Updates to the grouped components need to be made only once, rather than in every instance.
3. **Consistency**: Using the same component group across different windows ensures a consistent look and feel.

**Using Panes to Group Components**

In JavaFX, **Pane** objects are used to group components together. This allows you to:

* Move grouped components around as a unit within a window.
* Share grouped components between different windows with minimal code changes.

**Example: Creating an Address Pane**

**Explanation**

The **AddressPane** is a reusable component that groups labels and text fields for entering contact information. This pane can be used in multiple windows without duplicating the code for each label and text field.

**AddressPane Code**

java

Copy code

import javafx.geometry.Pos;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.Pane;

public class AddressPane extends Pane {

public AddressPane(String title) {

Pane innerPane = new Pane();

innerPane.setStyle("-fx-background-color: white; " +

"-fx-border-color: gray; " +

"-fx-padding: 4 4;");

// Create labels and text fields

Label label1 = new Label("Name:");

label1.relocate(10, 20);

label1.setPrefSize(80, 30);

Label label2 = new Label("Street:");

label2.relocate(10, 55);

label2.setPrefSize(80, 30);

Label label3 = new Label("City:");

label3.relocate(10, 90);

label3.setPrefSize(80, 30);

Label label4 = new Label("Province:");

label4.relocate(10, 125);

label4.setPrefSize(80, 30);

Label label5 = new Label("Postal Code:");

label5.relocate(10, 160);

label5.setPrefSize(80, 30);

TextField nameField = new TextField();

nameField.relocate(100, 20);

nameField.setPrefSize(300, 30);

TextField streetField = new TextField();

streetField.relocate(100, 55);

streetField.setPrefSize(300, 30);

TextField cityField = new TextField();

cityField.relocate(100, 90);

cityField.setPrefSize(300, 30);

TextField provinceField = new TextField();

provinceField.relocate(100, 125);

provinceField.setPrefSize(300, 30);

TextField postalField = new TextField();

postalField.relocate(100, 160);

postalField.setPrefSize(300, 30);

// Add components to inner pane

innerPane.getChildren().addAll(label1, label2, label3, label4, label5,

nameField, streetField, cityField, provinceField, postalField);

// Create and style the title label

Label titleLabel = new Label();

titleLabel.setText(title);

titleLabel.setStyle("-fx-background-color: white; \n" +

"-fx-translate-y: -8; \n" +

"-fx-translate-x: 10;");

getChildren().addAll(innerPane, titleLabel);

}

}

} }

**Using AddressPane in Applications**

**Explanation**

We'll use the **AddressPane** in two different applications to demonstrate its reusability.

1. **App1**: Uses **AddressPane** with a **ComboBox** to select different address types.
2. **App2**: Uses **AddressPane** twice within the same window, each with a different title.

**Application 1: OneApp**

**Explanation**

* **ComboBox**: A drop-down list for selecting address types.
* **AddressPane**: A reusable pane for entering contact information.

**Code for OneApp**

java

Copy code

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.ComboBox;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

public class OneApp extends Application {

public void start(Stage primaryStage) {

Pane aPane = new Pane();

// Create drop-down list (ComboBox)

ObservableList<String> options = FXCollections.observableArrayList(

"Home Address", "Work Address", "Alternate Address"

);

ComboBox<String> addressBox = new ComboBox<>(options);

addressBox.setValue("Work Address");

addressBox.relocate(10, 10);

addressBox.setPrefSize(410, 30);

aPane.getChildren().add(addressBox);

// Add AddressPane

AddressPane myPanel = new AddressPane("CONTACT ADDRESS");

myPanel.relocate(10, 50);

aPane.getChildren().add(myPanel);

// Setup and show the stage

primaryStage.setTitle("App 1");

primaryStage.setResizable(false);

primaryStage.setScene(new Scene(aPane, 420, 250));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**Application 2: TwoApp**

**Explanation**

* **ListView**: Displays a list of names.
* **AddressPane**: Two instances of **AddressPane** for different address types.

**Code for TwoApp**

java

Copy code

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.ComboBox;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

public class OneApp extends Application {

public void start(Stage primaryStage) {

Pane aPane = new Pane();

// Create drop-down list (ComboBox)

ObservableList<String> options = FXCollections.observableArrayList(

"Home Address", "Work Address", "Alternate Address"

);

ComboBox<String> addressBox = new ComboBox<>(options);

addressBox.setValue("Work Address");

addressBox.relocate(10, 10);

addressBox.setPrefSize(410, 30);

aPane.getChildren().add(addressBox);

// Add AddressPane

AddressPane myPanel = new AddressPane("CONTACT ADDRESS");

myPanel.relocate(10, 50);

aPane.getChildren().add(myPanel);

// Setup and show the stage

primaryStage.setTitle("App 1");

primaryStage.setResizable(false);

primaryStage.setScene(new Scene(aPane, 420, 250));

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

**Summary**

* **Grouping Components**: Use **Pane** objects to group related components for easy reuse.
* **Reusable Layouts**: Create reusable component groups like **AddressPane** for consistent UI design.
* **Applications**: Create applications (**OneApp** and **TwoApp**) that reuse **AddressPane** in different ways, showcasing its versatility.

By organizing and grouping components, you can streamline your code, reduce redundancy, and maintain consistency across your applications.