Unit 6: GUI with JavaFX (3 Hrs.)

By:-Kabita Dhital

CSIT 7TH SEM

Content

6.1. Introduction, JavaFX vs Swing, JavaFX Layouts: FlowPane, BorderPane, Hbox, VBox, GridPane

6.2. JavaFX UI Controls: Label, TextField, Button, RadioButton, CheckBox, Hyperlink, Menu, Tooltip, FileChooser.

Java Swing vs. Java FX

No.	JAVA Swing	JAVA FX
1.	Swing has a more sophisticated set of GUI components.	Less component as compared to legacy Swing APIs
2.	With Swing, it is difficult to create beautiful 3-D application.	With JAVAFX one can create beautiful 3-D application.
3.	No new functionality introduction for future.	JavaFX has a rich new toolkit, expected to grow in future
4.	Legacy UI library fully featured	Up and coming to feature-rich UI components
5.	MVC support across components lack consistency.	Friendly with **MVC pattern**.
6.	Swing has a strong community of developers and continues to be actively maintained.	JavaFX has a growing community of developers and is actively maintained by Oracle.
7.	Swing is primarily used for desktop applications but has some support for web deployment through technologies like Java Web Start.	JavaFX is basically designed for creating applications that can run across multiple platforms that includes desktop, web, and mobile.

Introduction-GUI with JavaFX

- **JavaFX** is a software platform for creating and delivering desktop applications, as well as rich web applications that can run across a wide variety of devices.
- **JavaFX** is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.
- Creating a Java GUI application is simple using JavaFX, a strong and contemporary framework. From basic design to complex multimedia integrations, JavaFX simplifies the process of creating user-friendly apps.
- **JavaFX** isn't hard to learn. In fact, any developer with a little bit of object-oriented knowledge and a penchant for desktop development in Java can quickly put together a feature-rich GUI application with the JavaFX framework.
- **JavaFX** enables you to design with Model-View-Controller (MVC), through the use of FXML and Java. The "Model" consists of application-specific domain objects, the "View" consists of FXML, and the "Controller" is Java code that defines the GUI's behavior for interacting with the user.

JavaFX Layouts

Layout panes are container which are used for flexible and dynamic arrangement of UI Controls within a scene graph of a JavaFX application. As a window is resized, the layout pane automatically repositions and resizes the nodes it contains. JavaFX has the following built in panes:- Flowpane, Hbox, Vbox, BorderPane, GridPane, StackPane, TilePane, AnchorPane.

- **FlowPane:-**

A JavaFX FlowPane is a layout component which lays out its child components either vertically or horizontally, and which can wrap the components onto the next row or column if there is not enough space in one row. The JavaFX FlowPane layout component is represented by the class javafx.scene.layout.FlowPane.

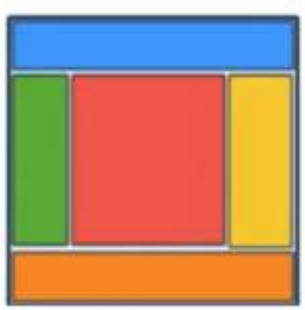
Example of a horizontal flowpane:

```
Image images[] = { ... };
FlowPane flow = new FlowPane();
flow.setVgap(8);
flow.setHgap(4);
flow.setPrefWrapLength(300); // preferred width = 300
for (int i = 0; i < images.length; i++) {
    flow.getChildren().add(new ImageView(image[i]);
}
```

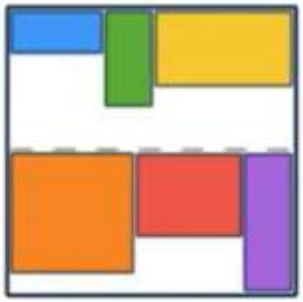
Example of a vertical flowpane:

```
FlowPane flow = new FlowPane(Orientation.VERTICAL);
flow.setColumnHalignment(HPos.LEFT); // align labels on left
flow.setPrefWrapLength(200); // preferred height = 200
for (int i = 0; i < titles.size(); i++) {
    flow.getChildren().add(new Label(titles[i]);
}
```

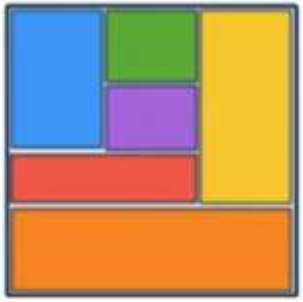
JavaFX Layouts



BorderPane

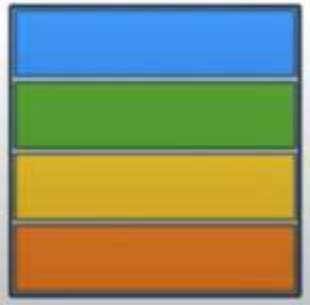


FlowPane



GridPane

StackPane



VBox

HBox

TilePane

AnchorPane

JavaFX Layouts

- **BorderPane:**
- **BorderPane** lays out children in top, left, right, bottom, and center positions. If we use the BorderPane, the nodes are arranged in the Top, Left, Right, Bottom and Center positions.
- The class named **BorderPane** of the package **javafx.scene.layout** represents the BorderPane.
- Example:-

```
ListView list = new ListView();
borderPane.setPrefSize(500,400);
BorderPane.setAlignment(list, Pos.TOP_LEFT);
borderPane.setCenter(list);
BorderPane.setMargin(list, new Insets(12,12,12,12));
```

JavaFX Layouts

- **HBOX:-**

The JavaFX HBox component is a layout component which positions all its child nodes (components) in a horizontal row.

The Java HBox component is represented by the class javafx. scene.

If the hbox has a border and/or padding set, then the contents will be layed out within those insets.

- HBox example:
- HBox hbox = new HBox(8); // spacing = 8
- hbox.getChildren().addAll(new Label("Name:), new TextBox());

JavaFX Layouts

- **Vbox:-**
- VBox lays out its children in a single vertical column. If the vbox has a border and/or padding set, then the contents will be layed out within those insets.
- VBox is a part of JavaFX. VBox lays out its children in form of vertical columns. If the vbox has a border and/or padding set, then the contents will be layed out within those insets. VBox class extends *Pane class*.
- **Constructor of the class:**
- **VBox()**: Creates a VBox layout with spacing = 0 and alignment at TOP_LEFT.
- **VBox(double s)**: Creates a new VBox with specified spacing between children.
- **VBox(double s, Node... c)**: Creates a new VBox with specified nodes and spacing between them.
- **VBox(Node... c)**: Creates an VBox layout with spacing = 0.

JavaFX Layouts

- **GridPane:-**
- The class named **GridPane** of the package **javafx.scene.layout** represents the GridPane.
- If we use Grid Pane in our application, all the nodes that are added to it are arranged in a way that they form a grid of rows and columns. This layout comes handy while creating forms using JavaFX.
- It is particularly useful for **creating forms** (like login forms, registration forms, etc.) where alignment and structure are important.
- **GridPane grid = new GridPane(); // This is the constructor**

{ Notes:- All example of Javafx layout are on the Github}

JAVAFX UI Controls

- In JavaFX, **UI Controls** are components that allow users to interact with the application. These are the building blocks of user interfaces such as **buttons, labels, text fields, combo boxes**, and more.

Control	Purpose	Example
Label	Displays text (non-editable)	new Label("Hello")
Button	Clickable button	new Button("Click Me")
TextField	Single-line text input	new TextField()
PasswordField	Hides input characters	new PasswordField()
TextArea	Multi-line text input	new TextArea()
CheckBox	Binary choice (checked/unchecked)	new CheckBox("Accept")
RadioButton	Used with ToggleGroup for choices	new RadioButton("Male")
ComboBox<T>	Drop-down list (editable or not)	new ComboBox<>()
ListView<T>	Scrollable list of items	new ListView<>()
ChoiceBox<T>	Simple drop-down selection	new ChoiceBox<>()
Slider	Select a value from a range	new Slider(0, 100, 50)
ProgressBar	Displays progress (determinate/indeterminate)	new ProgressBar(0.5)
ProgressIndicator	Circular form of progress bar	new ProgressIndicator()
DatePicker	Select date from a calendar	new DatePicker()
TableView<T>	Display tabular data	new TableView<>()
TreeView<T>	Hierarchical tree structure	new TreeView<>()
MenuBar	Menu with items	new MenuBar()
Tooltip	Hover-help text	new Tooltip("Help!")