Course code : **CSE2005**

Course title : **Object Oriented Programming**

JavaFX Layouts

Objectives

This session will give the knowledge about

- JavaFX Layouts

JavaFX Layouts

- This arrangement of the components within the scene is called the Layout of the scene.
- JavaFX provides several predefined layouts such as HBox, VBox, Border Pane, Stack Pane, Text Flow, Anchor Pane, Title Pane, Grid Pane, Flow Panel, etc.
- Each of the above mentioned layout is represented by a class and all these classes belongs to the package javafx.layout. The class named Pane is the base class of all the layouts in JavaFX.

Steps to apply Layouts

To create a layout, you need to −

- Create node.
- Instantiate the respective class of the required layout.
- Set the properties of the layout.
- Add all the created nodes to the layout.

JavaFX HBox

HBox layout pane arranges the nodes in a single row. It is represented by javafx.scene.layout.HBox class. We just need to instantiate HBox class in order to create HBox layout.

• setAlignment(Double)	This represents the alignment of the nodes.
• setFillHeight(boolean)	This is a boolean property. If you set this property to true the height of the nodes will become equal to the height of the HBox.
• setSpacing(Double)	This represents the space between the nodes in the HBox. It is of double type.

JavaFX HBox

• public final void setPadding(Insets value)	The top, right, bottom, and left padding around the region's content.This space will be included in the calculation of the region'sminimum and preferred sizes. By default padding is Insets.EMPTY. Setting thevalue to null should be avoided.

Constructors - The HBox class contains two constructors that are given below.

- new HBox() : create HBox layout with 0 spacing
- new Hbox(Double spacing) : create HBox layout with a spacing value

JavaFX HBox

```
Button b[]=new Button[10];
for(int i=0;i<10;i++){
        b[i]=new Button("b"+i);
        b[i].setMaxWidth(Double.MAX_VALUE);
        b[i].setMaxHeight(Double.MIN_NORMAL);
}
HBox root = new HBox();
root.setSpacing(10);
root.setPadding(new Insets(20, 20, 10, 20));

root.getChildren().addAll(b);
```

JavaFX VBox

Instead of arranging the nodes in horizontal row, Vbox Layout Pane arranges the nodes in a single vertical column. It is represented by javafx.scene.layout.VBox class. This class needs to be instantiated.

setAlignment(Double)	This represents the alignment of the nodes.
setFillWidth(boolean)	This property is of the boolean type. The Widtht of resizeable nodes can be made equal to the Width of the VBox by setting this property to true.
setSpacing(Double)	This represents the space between the nodes in the HBox. It is of double type.

JavaFX VBox

Constructors

- VBox() : creates layout with 0 spacing
- Vbox(Double spacing) : creates layout with a spacing value of double type
- Vbox(Double spacing, Node? children) : creates a layout with the specified spacing among the specified child nodes
- Vbox(Node? children) : creates a layout with the specified nodes having 0 spacing among them

JavaFX VBox

```
Button b[]=new Button[10];
for(int i=0;i<10;i++){
        b[i]=new Button("b"+i);
        b[i].setMaxWidth(Double.MAX_VALUE);
        b[i].setMaxHeight(Double.MIN_NORMAL);
}
VBox root = new VBox();
root.setSpacing(10);
root.setPadding(new Insets(20, 20, 10, 20));
root.setFillWidth(true);
root.getChildren().addAll(b);
```

JavaFX BorderPane

BorderPane arranges the nodes at the left, right, centre, top and bottom of the screen. It is represented by javafx.scene.layout.BorderPane class. This class provides various methods like setRight(), setLeft(), setCenter(), setBottom() and setTop() which are used to set the position for the specified nodes. We need to instantiate BorderPane class to create the BorderPane.

- BorderPane() : create the empty layout
- BorderPane(Node Center) : create the layout with the center node
- BorderPane(Node Center, Node top, Node right, Node bottom, Node left) : create the layout with all the nodes

JavaFX BorderPane

setBottom()	Add the node to the bottom of the screen
setCentre()	Add the node to the centre of the screen
setLeft()	Add the node to the left of the screen
setRight()	Add the node to the right of the screen
setTop()	Add the node to the top of the screen

JavaFX BorderPane

```
Button b[]=new Button[10];
for(int i=0;i<6;i++){
        b[i]=new Button("b"+i);
}
BorderPane root = new BorderPane();
root.setPadding(new Insets(20, 20, 10, 20));
root.setTop(b[0]);
root.setLeft(b[1]);
root.setRight(b[2]);
root.setCenter(b[3]);
root.setBottom(b[4]);
```

JavaFX FlowPane

- FlowPane layout pane organizes the nodes in a flow that are wrapped at the flowpane's boundary.
- The horizontal flowpane arranges the nodes in a row and wrap them according to the flowpane's width.
- The vertical flowpane arranges the nodes in a column and wrap them according to the flowpane's height.
- FlowPane layout is represented by javafx.scene.layout.FlowPane class. We just need to instantiate this class to create the flowpane layout.

JavaFX FlowPane

• setAlignment(Pos value)	The overall alignment of the flowpane's content.
• setColumnHalignment(HPos Value)	The horizontal alignment of nodes within the columns.
• setHgap(Double value)	Horizontal gap between the columns.
• setOrientation(Orientation value)	Orientation of the flowpane
• setPrefWrapLength(double value)	The preferred height or width where content will wrap in horizontal or vertical.
• setRowValignment(VPos value)	The vertical alignment of the nodes within the rows.
• setVgap(Double value)	The vertical gap among the rows

JavaFX FlowPane - Constructors

- FlowPane()
- FlowPane(Double Hgap, Double Vgap)
- FlowPane(Double Hgap, Double Vgap, Node? children)
- FlowPane(Node... Children)
- FlowPane(Orientation orientation)
- FlowPane(Orientation orientation, double Hgap, Double Vgap)
- FlowPane(Orientation orientation, double Hgap, Double Vgap, Node? children)
- FlowPane(Orientation orientation, Node... Children)

JavaFX FlowPane

```
Button b[]=new Button[10];
for(int i=0;i<10;i++){
        b[i]=new Button("b"+i);
}

FlowPane root = new FlowPane();
root.setVgap(10);
root.setHgap(20);
root.setPrefWrapLength(250);
root.getChildren().addAll(b);
```

JavaFX GridPane

GridPane Layout pane allows us to add the multiple nodes in multiple rows and columns. It is seen as a flexible grid of rows and columns where nodes can be placed in any cell of the grid.

It is represented by javafx.scence.layout.GridPane class. We just need to instantiate this class to implement GridPane.

Constructors

- Public GridPane(): creates a gridpane with 0 hgap/vgap.

JavaFX GridPane

• setAlignment(Pos value)	Represents the alignment of the grid within the GridPane.
• setGridLinesVisible(Boolean value)	This property is intended for debugging. Lines can be displayed to show the gidpane's rows and columns by setting this property to true.
• setHgap(Double value)	Horizontal gaps among the columns
• setVgap(Double value)	Vertical gaps among the rows

JavaFX GridPane

```
Button button1 = new Button("Button 1");
button1.setMaxSize(135, 70);
Button button2 = new Button("Button 2");
Button button3 = new Button("Button 3");
Button button4 = new Button("Button 4");
Button button5 = new Button("Button 5");
Button button6 = new Button("Button 6");
```

JavaFX GridPane

```
GridPane gridPane = new GridPane();
gridPane.add(button1, 0, 0, 2, 1);
gridPane.add(button2, 2, 0, 1, 1);
gridPane.add(button3, 2, 1, 1, 1);
gridPane.add(button4, 0, 2, 1, 1);
gridPane.add(button5, 1, 2, 1, 1);
gridPane.add(button6, 2, 2, 1, 1);
gridPane.setHgap(10);
gridPane.setVgap(10);
Scene scene = new Scene(gridPane, 400, 350);
```

JavaFX StackPane

The StackPane layout pane places all the nodes into a single stack where every new node gets placed on the top of the previous node.

It is represented by javafx.scene.layout.StackPane class. We just need to instantiate this class to implement StackPane layout into our application.

Constructors

- StackPane()
- StackPane(Node? Children)

JavaFX StackPane – Demo-1

```
Button b[]=new Button[10];
for(int i=0;i<10;i++){
        b[i]=new Button("b"+i);
        b[i].setMaxWidth(Double.MAX_VALUE);
        b[i].setMaxHeight(Double.MIN_NORMAL);
}

StackPane root = new StackPane();
root.setPadding(new Insets(20, 20, 10, 20));
root.getChildren().addAll(b);
```

JavaFX StackPane – Demo-2

```
Circle circle = new Circle(100, 100, 70);
circle.setFill(Color.GREEN);

Rectangle rectangle = new Rectangle(100, 100, 180, 160);
rectangle.setFill(Color.BLUE);

StackPane root = new StackPane();
root.getChildren().addAll(circle,rectangle);
```

JavaFX TilePane

A JavaFX TilePane is a layout component which lays out its child components in a grid of equally sized cells.

The JavaFX TilePane layout component is represented by the class javafx.scene.layout.TilePane

Creating a TilePane

- TilePane tilePane = new TilePane();

JavaFX TilePane

```
Button b[]=new Button[10];
for(int i=0;i<10;i++){
        b[i]=new Button("b"+i);
}

TilePane root = new TilePane();
root.setVgap(10);
root.setHgap(20);
root.getChildren().addAll(b);
```

Summary

We have discussed about

- JavaFX Layouts