VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

JavaFX Controls

Objectives

This session will give the knowledge about

- JavaFX Controls

JavaFX Controls

The JavaFX UI controls are built by using nodes in the scene graph. Therefore, the controls can use the visually rich features of the JavaFX platform.

Because the JavaFX APIs are fully implemented in Java, you can easily integrate the JavaFX UI controls into your existing Java applications.

The package javafx.scene.control provides all the necessary classes for the UI components like Button, Label, etc. Every class represents a specific UI control and defines some methods for their styling.

JavaFX UI Controls

JavaFX Label

- javafx.scene.control.Label class represents label control. As the name suggests, the label is the component that is used to place any text information on the screen.
- It is mainly used to describe the purpose of the other components to the user. You can not set a focus on the label using the Tab key.
- Package: javafx.scene.control

JavaFX Label

- Constructors:
  - Label(): creates an empty Label
  - Label(String text): creates Label with the supplied text
  - Label(String text, Node graphics): creates Label with the supplied text and graphics

JavaFX Button

- JavaFX button control is represented by javafx.scene.control.Button class. A button is a component that can control the behaviour of the Application. An event is generated whenever the button gets clicked.
- Button btn = new Button("My Button");
- Setting the Text of the Button: There are two ways.
  - Passing the text into the class constructor
  - By calling setText("text") method

JavaFX Button

- Wrapping Button Text - We can wrap the text of the button into multiple lines if the text to be displayed is too long.
  - Btn.setWrapText(true);
- Setting the image on the button - Button class contains a constructor which can accept graphics along with the text displayed on the button. The following code implements image on the button.
  - Button btn=new Button("Button 1",img); // img is an Image object

JavaFX Button

- Using setGraphic() method - Button class also provides an instance method named setGraphic(). We have to pass the image view object in this method.
  - btn.setGraphic(img);
- Button Effects - We can apply the effects to a Button. The effects are provided by javafx.scene.effect package. The following code shows how the drop shadow effect can be applied to a button.
  - btn.setEffect(shadow);

JavaFX RadioButton

- The Radio Button is used to provide various options to the user. The user can only choose one option among all. A radio button is either selected or deselected.

```
ToggleGroup group = new ToggleGroup();
RadioButton button1 = new RadioButton("option 1");
RadioButton button2 = new RadioButton("option 2");
button1.setToggleGroup(group);
button2.setToggleGroup(group);
```

JavaFX CheckBox

- The Check Box is used to provide more than one choices to the user. It can be used in a scenario where the user is prompted to select more than one option or the user wants to select multiple options.
- It is different from the radiobutton in the sense that, we can select more than one checkboxes in a scenerio.
- Instantiate javafx.scene.control.CheckBox class to implement CheckBox.
  - CheckBox checkbox = new CheckBox();
  - CheckBox checkbox = new CheckBox("Label Name");

JavaFX TextField

Text Field is basically used to get the input from the user in the form of text. javafx.scene.control.TextField represents TextField.

It provides various methods to deal with textfields in JavaFX. TextField can be created by instantiating TextField class.

TextField

- public TextField(String text) - Creates a TextField with empty text content.

JavaFX TextField

TextField

- public TextField(String text) - Creates a TextField with initial text content.
- Parameters: text - A string for text content.

JavaFX PasswordField

Password field can be created by instantiating javafx.scene.control.PasswordField class.

PasswordField class contains a method named as setPromptText() for showing a prompt text to the user in password field.

The data written in the passwordfield is retrieved by getText() method.

PasswordField

- public PasswordField()
- Creates a default PasswordField instance.

JavaFX HyperLink

In JavaFx, we can use hyper-links to refer the web pages. It is similar to anchor links in HTML. javafx.scene.control.HyperLink class provides all the necessary methods to deal with JavaFX hyper-links.

- Hyperlink hp = new Hyperlink();
- Hyperlink hp = new Hyperlink("http://www.google.com");

JavaFX slider

JavaFX slider is used to provide a pane of option to the user in a graphical form where the user needs to move a slider over the range of values to select one of them. Slider can be created by instantiating javafx.scene.control.Slider class.

- public Slider(double min, double max, double value) - Constructs a Slider control with the specified slider min, max and current value values.
- Parameters: min - Slider minimum value max - Slider maximum value value - Slider current value

JavaFX ProgressBar

Progress Bar is used to show the work progress to the user. It is represented by javafx.scene.control.ProgressBar. The following code implements progrss bar into our application.

- public ProgressBar() - Creates a new indeterminate ProgressBar.
- public ProgressBar(double progress) - Creates a new ProgressBar with the given progress value.

JavaFX Progress Indicator

Progress Indicator is similar to Progress Bar to some extent. Instead of showing the analogue progress to the user, it shows the digital progress so that the user may know the amount of work done in percentage.

It is represented by javafx.scene.control.ProgressIndicator class.

This class needs to be instantiated in order to create Progress Indicator. The following code implements Progress Indicator into our application.

- ProgressIndicator PI=new ProgressIndicator();

JavaFX ScrollBar

JavaFX Scroll Bar is used to provide a scroll bar to the user so that the user can scroll down the application pages.

It can be created by instantiating javafx.scene.control.ScrollBar class.

- ScrollBar s = new ScrollBar();

JavaFX ScrollPane

The JavaFX ScrollPane control is a container that has two scrollbars around the component it contains if the component is larger than the visible area of the ScrollPane.

The scrollbars enable the user to scroll around the component shown inside the ScrollPane, so different parts of the component can be seen.

The JavaFX ScrollPane controls is represented by the JavaFX class javafx.scene.control.ScrollPane.

JavaFX ScrollPane

```
ScrollPane scrollPane = new ScrollPane();
FileInputStream file=null;
try {
        file = new FileInputStream("images/logo.jpg");
} catch (FileNotFoundException e) {
        e.printStackTrace();
}
Image img=new Image(file);
ImageView iv=new ImageView(img);
scrollPane.setContent(iv);
```

JavaFX Menu

JavaFX provides a Menu class to implement menus. Menu is the main component of a any application. In JavaFX, javafx.scene.control.Menu class provides all the methods to deal with menus. This class needs to be instantiated to create a Menu.

- MenuBar menubar = new MenuBar(); //creating MenuBar
- Menu MenuName = new Menu("Menu Name"); //creating Menu
- MenuItem MenuItem1 = new MenuItem("Menu Item 1 Name");
- MenuName.getItems().add(MenuItem1); //add Menu Item to the Menu
- menubar.getMenus().add(MenuName); //adding Menu to the MenuBar

JavaFX TreeView

The JavaFX TreeView enables you to display tree views inside your JavaFX applications. The JavaFX TreeView is represented by the class javafx.scene.control.TreeView.

- TreeItem<String> rootItem = new TreeItem<String>("Java");
- TreeItem<String> webItem = new TreeItem<String>("Web App");
- webItem.getChildren().add(new TreeItem<String>("HTML"));
- rootItem.getChildren().add(webItem);
- TreeView<String> treeView = new TreeView<String>();
- treeView.setRoot(rootItem);

JavaFX ToggleButton

A JavaFX ToggleButton is a button that can be selected or not selected. Like a button that stays in when you press it, and when you press it the next time it comes out again. Toggled - not toggled. The JavaFX ToggleButton is represented by the class javafx.scene.control.ToggleButton.

- ToggleButton toggleButton1 = new ToggleButton("Left");
- ToggleButton toggleButton2 = new ToggleButton("Right");
- ToggleGroup toggleGroup = new ToggleGroup();
- toggleButton1.setToggleGroup(toggleGroup);
- toggleButton2.setToggleGroup(toggleGroup);

JavaFX Controls

```
//import required packages
public class ControlDemo extends Application {
        //declare all controls as instance variables to handle events
        Button btn;
        TextField t1;
        Label l1,image;
        PasswordField pwd;
        Hyperlink hp;
        RadioButton m,f;
        ToggleGroup gender;
        CheckBox opt1,opt2;
```

JavaFX Controls

```
ComboBox<String> skills;
ListView<String> lang;
FileInputStream file;
Slider slider;
ProgressBar progress;
ProgressIndicator PI;

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

JavaFX Controls

```
@Override
public void start(Stage primaryStage) {
        l1=new Label("i am lable");
        l1.setText("enter your name");
        l1.setTextAlignment(TextAlignment.JUSTIFY);
        l1.setTextFill(Color.RED);
        l1.setFont(new Font("Arial",24));

        t1=new TextField();
        t1.setMaxSize(170, 30);
        pwd=new PasswordField();
```

JavaFX Controls

```
hp = new Hyperlink("http://www.google.com");

btn=new Button("click");
btn.setDisable(true);

m=new RadioButton("male");
f=new RadioButton("female");
gender=new ToggleGroup();
m.setToggleGroup(gender);
f.setToggleGroup(gender);
```

JavaFX Controls

```
opt1=new CheckBox("java");
opt2=new CheckBox("python");

skills=new ComboBox<String>();
skills.getItems().add("app develop");
skills.getSelectionModel().selectFirst();
skills.getItems().add("web develop");

lang= new ListView<String>();
lang.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
lang.getItems().add("telugu");
```

JavaFX Controls

```
lang.getItems().add("tamil");
lang.getItems().add("malayalam");

try {
        file = new FileInputStream("images/logo.jpg");
} catch (FileNotFoundException e) {
        e.printStackTrace();
}
Image img=new Image(file);
ImageView iv=new ImageView(img);
image=new Label("",iv);
```

JavaFX Controls

```
                slider = new Slider(1,100,20);
                progress = new ProgressBar();
                PI=new ProgressIndicator();
                //setting layout, scene, stage properties
                VBox root=new VBox();
root.getChildren().addAll(l1,t1,pwd,hp,btn,image,m,f,opt1,opt2,skills,lang,slider,p
rogress,PI);
                Scene scene = new Scene(root,300,300);
                primaryStage.setScene(scene);
                primaryStage.setTitle("Control demo");
                primaryStage.show();   }   }
```

Summary

We have discussed about

- JavaFX UI Controls