java --module-path "C:\Program Files\Java\javafx-sdk-23\lib" --add-modules javafx.controls,javafx.media AVMain1

F:\Java-pgm>javac --module-path "C:\Program Files\Java\javafx-sdk-23\lib" --add-modules javafx.controls,javafx.media AVMain1.java

JavaFX UI Controls :

The graphical user interface of every desktop application mainly considers UI elements, layouts and behaviour.

The UI elements are the one which are actually shown to the user for interaction or information exchange. Layout defines the organization of the UI elements on the screen. Behaviour is the reaction of the UI element when some event is occurred on it.

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.

SN	Control	Description
1	Label	Label is a component that is used to define a simple text on the screen. Typically, a label is placed with the node, it describes.
2	Button	Button is a component that controls the function of the application. Button class is used to create a labelled button.
3	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.
4	CheckBox	Check Box is used to get the kind of information from the user which contains various choices. User marked the checkbox either on (true) or off(false).
5	TextField	Text Field is basically used to get the input from the user in the form of

		text. javafx.scene.control.TextField represents TextField
6	Slider	Slider is used to provide a pane of options 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.
7	ScrollBar	JavaFX Scroll Bar is used to provide a scroll bar to the user so that the user can scroll down the application pages.
8	ProgressBar	Progress Bar is used to show the work progress to the user. It is represented by the class javafx.scene.control.ProgressBar.

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

Constructors:

1. Label( ): creates an empty Label
2. Label(String text): creates Label with the supplied text
3. Label(String text, Node graphics): creates Label with the supplied text and graphics

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.

Lets see an example where the user is shown the two text boxes and prompted to fill its user-id and password.

Program on Label & Text Box

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
public class LabTxt extends Application {

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

@Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    Label user_id=new Label("User Name");
    Label password = new Label("Password");
    TextField tf1=new TextField();
    TextField tf2=new TextField();
    Button b = new Button("Submit");
    GridPane root = new GridPane();
    root.addRow(0, user_id, tf1);
    root.addRow(1, password, tf2);
    root.addRow(2, b);
    Scene scene=new Scene(root,800,200);
```

```
        primaryStage.setScene(scene);
        primaryStage.setTitle("Text Field Example");
        primaryStage.show();
    }
    }
```

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.

How to create a Button?
Button can be created by instantiating Button class. Use the following line to create button object.

1. Button btn = new Button("My Button");

Adding a Button to the scene graph
To visualize the button on the screen, we must attach it to the scene object. The following code creates a button and adds it to the scene object.

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.

1. CheckBox chkbox = new CheckBox(“ ”));

Use the following line to attach a label with the checkbox.

1. CheckBox checkbox = new CheckBox("Label Name");

We can change the CheckBox Label at any time by calling an instance method setText("text"). We can make it selected by calling setSelected("true")

Program on Check Box:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.CheckBox;
import javafx.scene.control.Label;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
public class Chk extends Application {



public static void main(String[] args) {
launch(args);
}
  @Override
public void start(Stage primaryStage) throws Exception {
    // TODO Auto-generated method stub
    Label l = new Label("Languages Known:     ");
    CheckBox c1 = new CheckBox("Hindi");
    CheckBox c2 = new CheckBox("English");
    CheckBox c3 = new CheckBox("Gujrati");
       HBox root = new HBox();
    root.getChildren().addAll(l,c1,c2,c3);
    root.setSpacing(50);
    Scene scene=new Scene(root,800,200);
    primaryStage.setScene(scene);
    primaryStage.setTitle("CheckBox Example");
    primaryStage.show();
}
```

}

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. It can be used in a scenario of multiple choice questions in the quiz where only one option needs to be chosen by the student.

Program on Radio Button:

```
import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.RadioButton;

import javafx.scene.control.ToggleGroup;

import javafx.scene.layout.VBox;

import javafx.scene.control.Label;

import javafx.stage.Stage;

public class RadioP extends Application {


public static void main(String[] args) {

launch(args);

}


@Override

public void start(Stage primaryStage) throws Exception {

    // TODO Auto-generated method stub


    ToggleGroup group = new ToggleGroup();

Label l = new Label("Year:     ");

    RadioButton button1 = new RadioButton("First");

    RadioButton button2 = new RadioButton("Second");
```

```
    RadioButton button3 = new RadioButton("Third");
    RadioButton button4 = new RadioButton("Forth");
    button1.setToggleGroup(group);
    button2.setToggleGroup(group);
    button3.setToggleGroup(group);
    button4.setToggleGroup(group);
    VBox root=new VBox();
    root.setSpacing(10);
    root.getChildren().addAll(l,button1,button2,button3,button4);
    Scene scene=new Scene(root,400,300);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Radio Button Example");
    primaryStage.show();
}
}
```

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.

Program on Scrollbar:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class ScrollBarTest extends Application{
```

```
    @Override
    public void start(Stage primaryStage) throws Exception {
        // TODO Auto-generated method stub
        ScrollBar s = new ScrollBar();
        StackPane root = new StackPane();
        root.getChildren().add(s);
        Scene scene = new Scene(root,300,200);
        primaryStage.setScene(scene);
        primaryStage.setTitle("ScrollBar Example");
        primaryStage.show();


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


}
```

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.

The constructor accepts three arguments: the minimum value, the maximum value, and the initial value of the slider.

Program on Slider:

```
import javafx.application.Application;
import javafx.scene.Scene;
```

```
import javafx.scene.control.Slider;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class SliderTest extends Application{

    @Override
    public void start(Stage primaryStage) throws Exception {
        // TODO Auto-generated method stub
        Slider slider = new Slider(1,100,20);
        StackPane root = new StackPane();
        root.getChildren().add(slider);
        Scene scene = new Scene(root,300,200);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Slider Example");
        primaryStage.show();

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

}
```

combo box :

A combo box is a typical element of a user interface that enables users to choose one of several options. A combo box is helpful when the number of items to show exceeds some limit, because it can add scrolling to the drop down list, unlike a choice box. If the number of items does not exceed a certain limit, developers can decide whether a combo box or a choice box better suits their needs.

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class ComboBoxExam extends Application {
   public static void main(String[] args) {
      launch(args);
   }

   final Button button = new Button ("Send");
   final Label notification = new Label ();
   final TextField subject = new TextField("");
   final TextArea text = new TextArea ("");

   String address = " ";

   @Override
public void start(Stage stage) {
      stage.setTitle("ComboBoxSample");
      Scene scene = new Scene(new Group(), 450, 250);

      final ComboBox c = new ComboBox();
      c.getItems().addAll(
         "Java",
         "Python",
         "C++",
         "C",
         "SQL"
      );



      GridPane grid = new GridPane();
      grid.setVgap(4);
      grid.setHgap(10);
      grid.setPadding(new Insets(5, 5, 5, 5));
      grid.add(new Label("Course: "), 0, 0);
      grid.add(c, 1, 0);
```

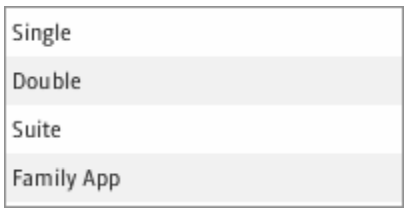
```
        grid.add(text, 0, 2, 4, 1);

          Group root = (Group)scene.getRoot();
        root.getChildren().add(grid);
        stage.setScene(scene);
        stage.show();
    }
}
```

List View:

The ListView class represents a scrollable list of items. Figure 11-1 shows the list of available accommodation types in a hotel reservation system.

Figure 11-1 Simple List View



Program on List View :

```
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.ComboBoxListCell;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class ListViewExam extends Application {

    public static final ObservableList names =
        FXCollections.observableArrayList();
    public static final ObservableList data =
        FXCollections.observableArrayList();

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

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("List View Sample");

        final ListView listView = new ListView(data);
```

```
        listView.setPrefSize(200, 250);
        listView.setEditable(true);

        names.addAll(
            "Red", "Blue", "Green", "Black",
            "White"
        );


           data.add("Color");


        listView.setItems(data);
        listView.setCellFactory(ComboBoxListCell.forListView(names));

        StackPane root = new StackPane();
        root.getChildren().add(listView);
        primaryStage.setScene(new Scene(root, 200, 250));
        primaryStage.show();
    }
}
```

TextArea :

A text area is a multi-line editor where you can enter text. Unlike previous versions, in the latest versions of JavaFX, a TextArea does not allow single lines in it. You can create a text area by instantiating the javafx.scene.control.TextArea class.

Program on TextArea:

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.HBox;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;
public class TextAreaExam extends Application {
  public void start(Stage stage) {
    //Setting the label
    Label label = new Label("Address");
    Font font = Font.font("verdana", FontWeight.BOLD, FontPosture.REGULAR, 12);
    label.setFont(font);
    //Creating a pagination
```

```
    TextArea area = new TextArea();
    //Setting number of pages
    area.setText("Enter your address here ");
    area.setPrefColumnCount(15);
    area.setPrefHeight(120);
    area.setPrefWidth(300);
    //Creating a hbox to hold the pagination
    HBox hbox = new HBox();
    hbox.setSpacing(20);
    hbox.setPadding(new Insets(20, 50, 50, 60));
    hbox.getChildren().addAll(label, area);
    //Setting the stage
    Group root = new Group(hbox);
    Scene scene = new Scene(root, 595, 200, Color.BEIGE);
    stage.setTitle("Text Area");
    stage.setScene(scene);
    stage.show();
  }
  public static void main(String args[]){
    launch(args);
  }
}
```

Video :

```
import java.io.File;
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;
public class AVMain1 extends Application {
  @Override
  public void start(Stage stage) {
    // Passing the video file to the File object
    File videofile = new File("V1.mp4");
    // creating a Media object from the File Object
    Media videomedia = new Media(videofile.toURI().toString());
    // creating a MediaPlayer object from the Media Object
    MediaPlayer mdplayer = new MediaPlayer(videomedia);
    // creating a MediaView object from the MediaPlayer Object
    MediaView viewmedia = new MediaView(mdplayer);
    //setting the fit height and width of the media view
    viewmedia.setFitHeight(455);
    viewmedia.setFitWidth(500);
    // creating video controls using the buttons
```

```
        Button pause = new Button("Pause");
        Button resume = new Button("Resume");
        // creating an HBox
        HBox box = new HBox(20, pause, resume);
        box.setAlignment(Pos.CENTER);
        // function to handle play and pause buttons
        pause.setOnAction(act -> mdplayer.pause());
        resume.setOnAction(act -> mdplayer.play());
        // creating the root
        VBox root = new VBox(20);
        root.setAlignment(Pos.CENTER);
        root.getChildren().addAll(viewmedia, box);
        Scene scene = new Scene(root, 400, 400);
        stage.setScene(scene);
        stage.setTitle("Example of Video in JavaFX");
        stage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```