AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER 8
JAVAFX UI Controls and Multimedia

SUBJECT:OOP-I CODE:3140705	PREPARED BY: ASST.PROF.NENSI KANSAGARA (CSE DEPARTMENT,ACET)	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

Introduction to UI Control

- ★ This part of the tutorial provides you the in-depth knowledge of 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.
- ★ However, 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.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

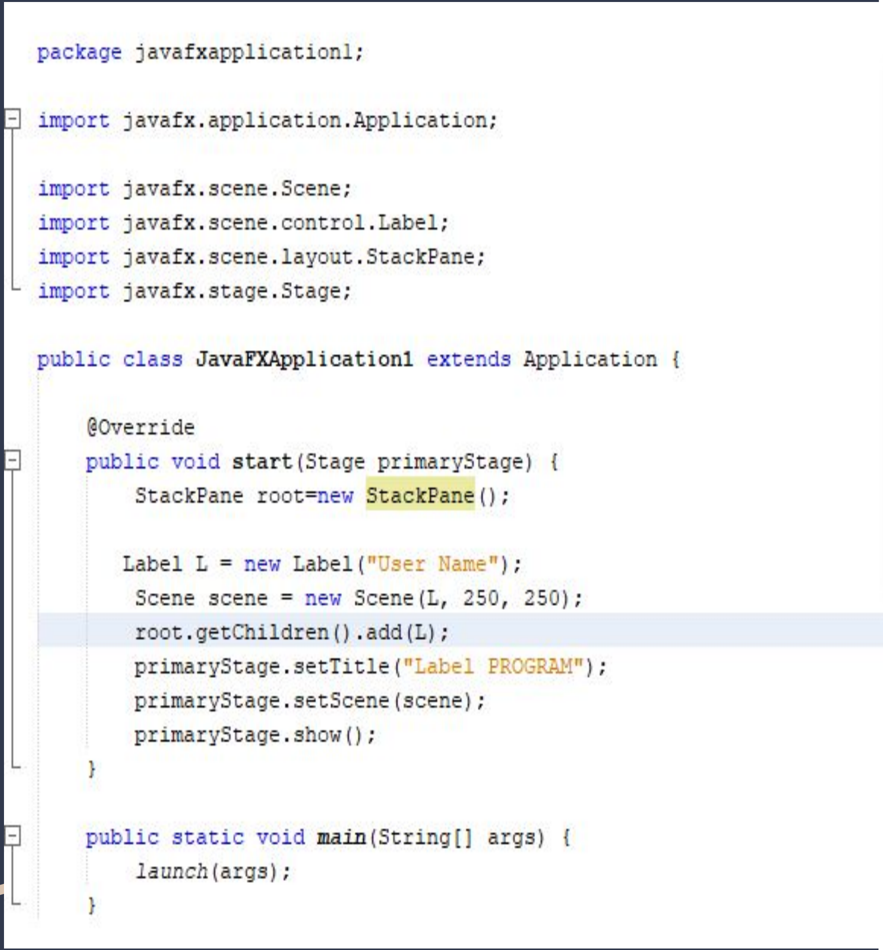
Control	Description
Label	Label is a component that is used to define a simple text on the screen. Typically, a label is placed with the node, it describes.
Button	Button is a component that controls the function of the application. Button class is used to create a labelled button.
RadioButton	The Radio 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.
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).
TextField	Text Field is basically used to get the input from the user in the form of text. javafx.scene.control.TextField represents TextField

Textarea	This control allows the user to enter multiple line text.
ComboBox	This control display the list of items out of which user can select at most one item.
ListView	This control displays the list of items out of which user can select one or multiple items from the list.
Slider	This control is used to display a continuous or discrete range of valid numeric choices and allows the user to interact with the control.

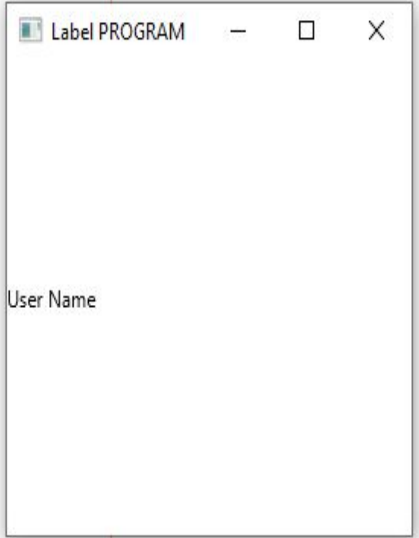
Labeled and Label

- ★ **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



```
package javafxapplication1;

import javafx.application.Application;

import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class JavaFXApplication1 extends Application {

    @Override
    public void start(Stage primaryStage) {
        StackPane root=new StackPane();

       Label L = new Label("User Name");
        Scene scene = new Scene(L, 250, 250);
        root.getChildren().add(L);
        primaryStage.setTitle("Label PROGRAM");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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



```
Label PROGRAM

User Name
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Button

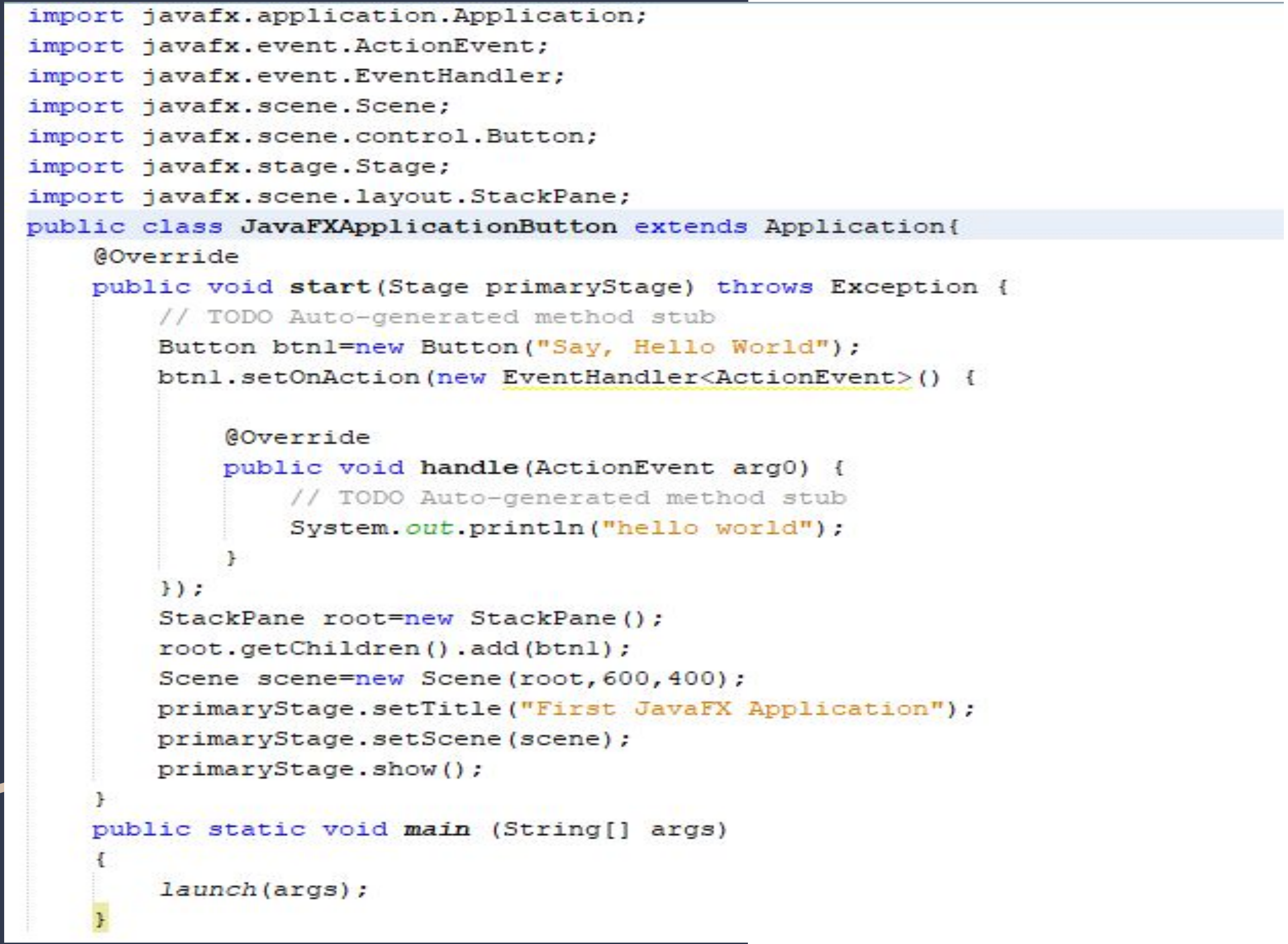
★ JavaFX button control is represented by **javafx.scene.control.Button** class. A button is a component that can control the behaviour of the Application. An event is generated whenever the button gets clicked.

★ How to create a Button?

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

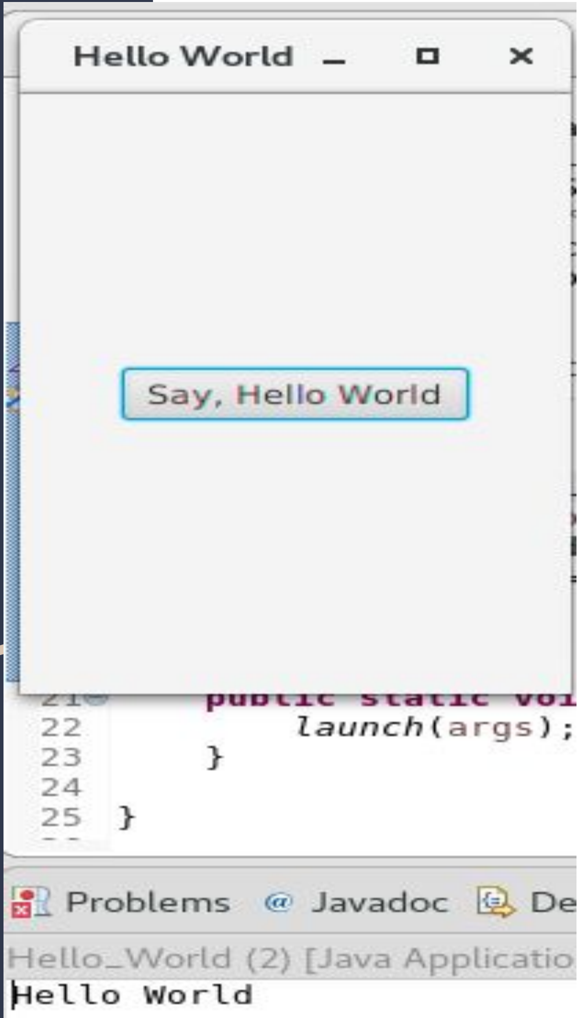
★ SYNTAX:

```
Button btn = new Button("My Button");
```



```java
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import javafx.scene.layout.StackPane;
public class JavaFXApplicationButton extends Application{
    @Override
    public void start(Stage primaryStage) throws Exception {
        // TODO Auto-generated method stub
        Button btn1=new Button("Say, Hello World");
        btn1.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent arg0) {
                // TODO Auto-generated method stub
                System.out.println("hello world");
            }
        });
        StackPane root=new StackPane();
        root.getChildren().add(btn1);
        Scene scene=new Scene(root,600,400);
        primaryStage.setTitle("First JavaFX Application");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main (String[] args)
    {
        launch(args);
    }
```

OutPut:



```
public static voi
        launch(args);
    }

}
```

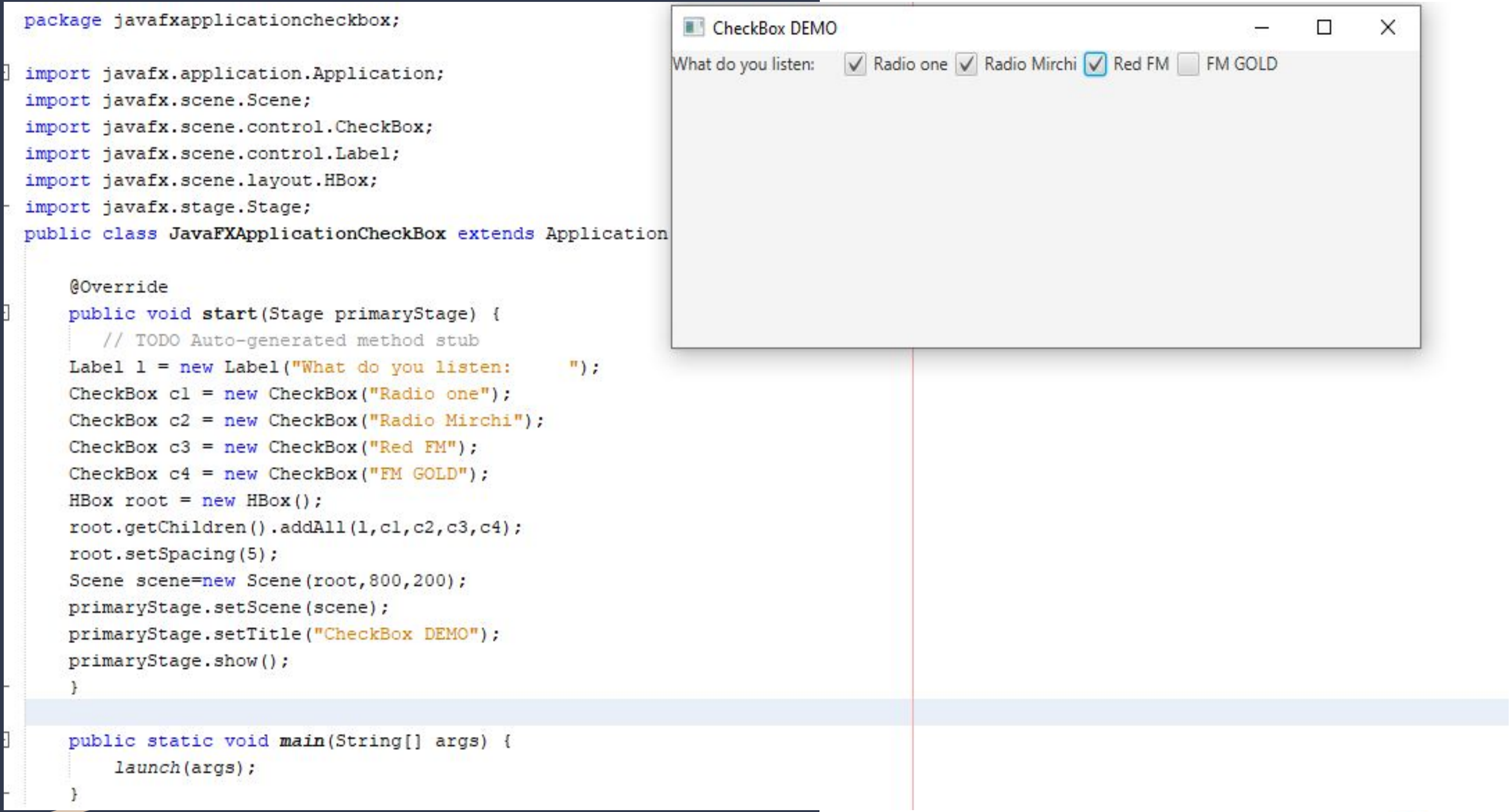
```
Hello_World (2) [Java Applicatio
Hello World
```

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.
- ★ SYNTAX:

```
CheckBox checkbox = new CheckBox("Label Name");
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY


```
package javafxapplicationcheckbox;

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 JavaFXApplicationCheckBox extends Application

    @Override
    public void start(Stage primaryStage) {
        // TODO Auto-generated method stub
    Label l = new Label("What do you listen:      ");
    CheckBox c1 = new CheckBox("Radio one");
    CheckBox c2 = new CheckBox("Radio Mirchi");
    CheckBox c3 = new CheckBox("Red FM");
    CheckBox c4 = new CheckBox("FM GOLD");
    HBox root = new HBox();
    root.getChildren().addAll(l,c1,c2,c3,c4);
    root.setSpacing(5);
    Scene scene=new Scene(root,800,200);
    primaryStage.setScene(scene);
    primaryStage.setTitle("CheckBox DEMO");
    primaryStage.show();
    }

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

CheckBox DEMO

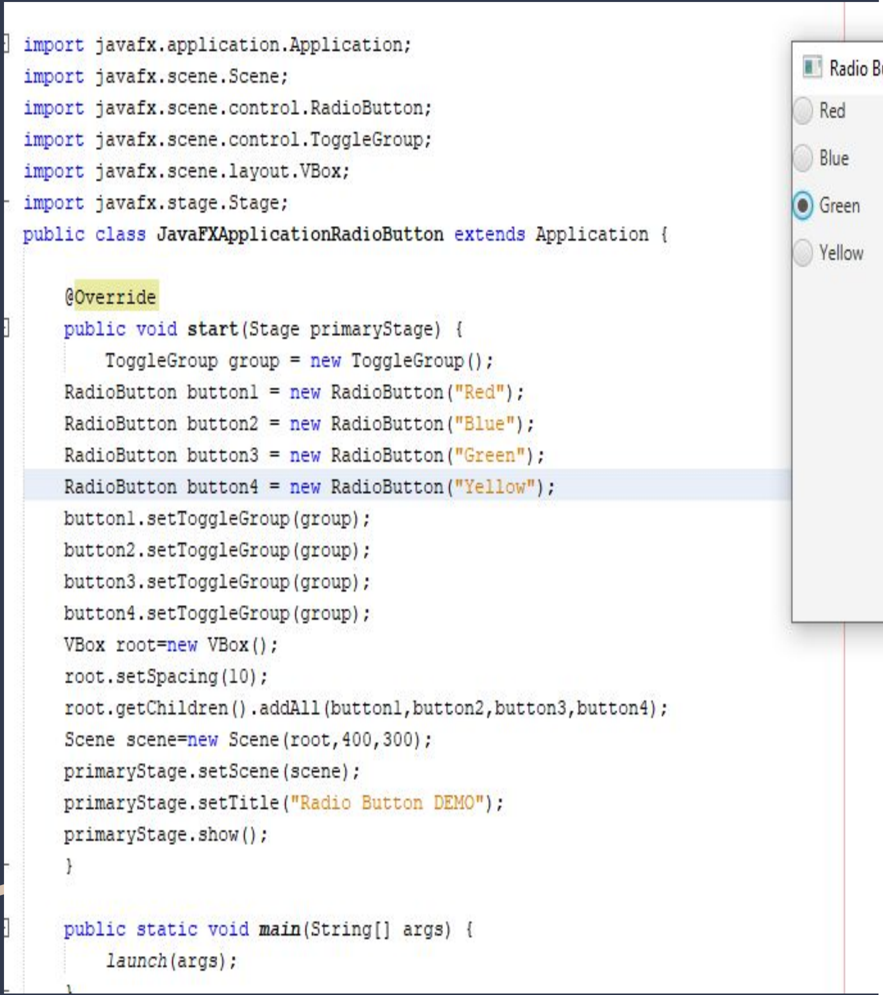
What do you listen: ✓ Radio one ✓ Radio Mirchi ✓ Red FM ☐ FM GOLD

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

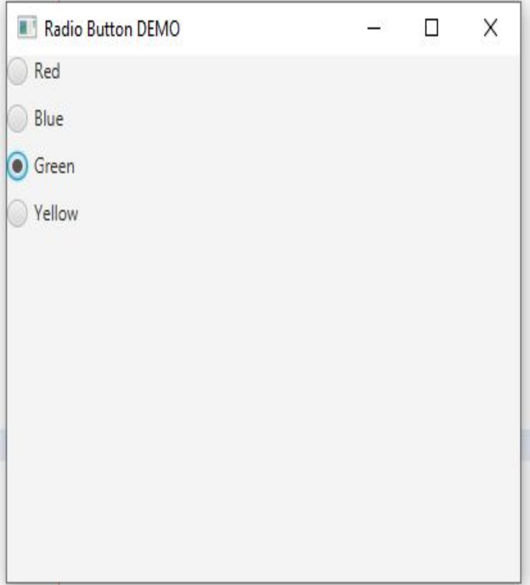
RadioButton

- ★ 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.
- ★ We can group JAVAFX RadioButton instances into a ToggleGroup.A ToggleGroup allows at most one RadioButton to be selected at any time.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



```
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.stage.Stage;
public class JavaFXApplicationRadioButton extends Application {

    @Override
    public void start(Stage primaryStage) {
        ToggleGroup group = new ToggleGroup();
    RadioButton button1 = new RadioButton("Red");
    RadioButton button2 = new RadioButton("Blue");
    RadioButton button3 = new RadioButton("Green");
    RadioButton button4 = new RadioButton("Yellow");
    button1.setToggleGroup(group);
    button2.setToggleGroup(group);
    button3.setToggleGroup(group);
    button4.setToggleGroup(group);
    VBox root=new VBox();
    root.setSpacing(10);
    root.getChildren().addAll(button1,button2,button3,button4);
    Scene scene=new Scene(root,400,300);
    primaryStage.setScene(scene);
    primaryStage.setTitle("Radio Button DEMO");
    primaryStage.show();
    }

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



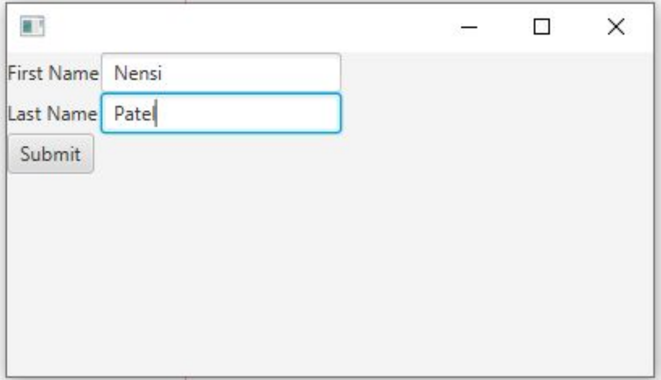
AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

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.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```java
package javafxapplicationgridpane;

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 JavaFXApplicationGridPane extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        Label first_name=new Label("First Name");
        Label last_name=new Label("Last Name");
        TextField tf1=new TextField();
        TextField tf2=new TextField();
        Button Submit=new Button ("Submit");
        GridPane root=new GridPane();
        Scene scene = new Scene(root,400,200);
        root.addRow(0, first_name,tf1);  //addRow(int rowIndex,Node..,Childern)
        root.addRow(1, last_name,tf2);
        root.addRow(2, Submit);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
```

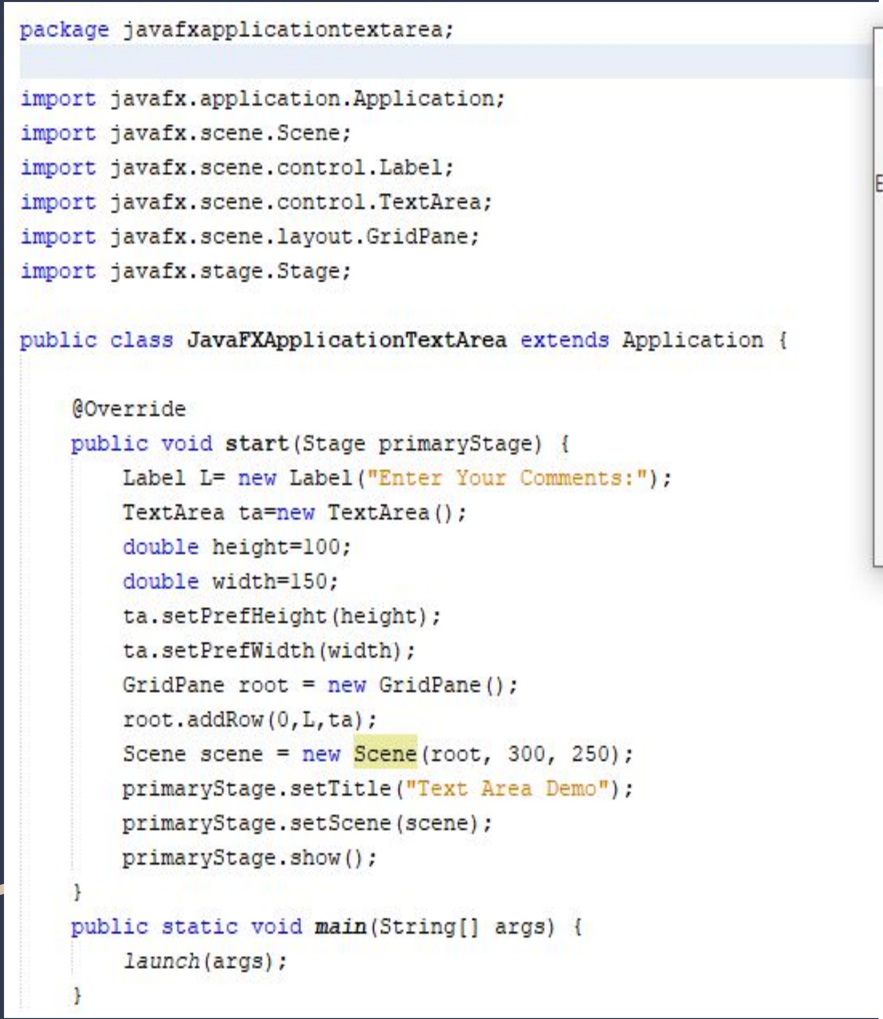


TextArea

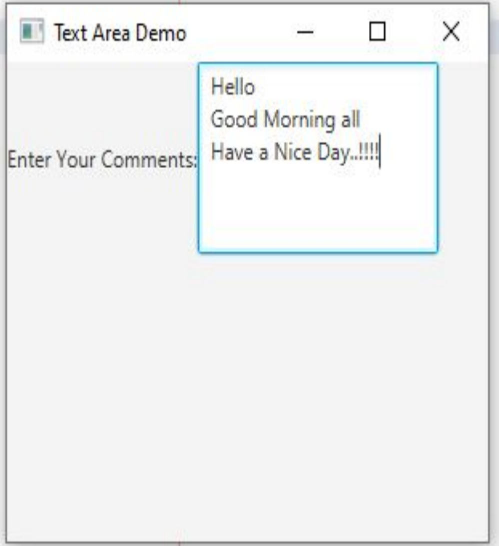
- ★ The TextARea control allows to enter multiline text.The control is represented by the class **javafx.scene.control.TextArea.**
- ★ SYNTAX:

```
TextArea ta=new TextArea()
```

- ★ We can set the size of the TextArea using **setPrefHeight()** and **setPrefWidth()** functions.



```
package javafxapplicationtextarea;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class JavaFXApplicationTextArea extends Application {

    @Override
    public void start(Stage primaryStage) {
        Label L= new Label("Enter Your Comments:");
        TextArea ta=new TextArea();
        double height=100;
        double width=150;
        ta.setPrefHeight(height);
        ta.setPrefWidth(width);
        GridPane root = new GridPane();
        root.addRow(0,L,ta);
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("Text Area Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
```



Text Area Demo

Enter Your Comments:

```
Hello
Good Morning all
Have a Nice Day..!!!!
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Combo Box

- ★ The JavaFX ComboBox control enables users to choose an option from a predefined list of choices, or type in another value if none of the predefined choices matches what the user want to select. The JavaFX ComboBox control is represented by the class javafx.scene.control.ComboBox . This JavaFX ComboBox tutorial will explain how to use the ComboBox class.
- ★ **Creating a ComboBox**
- ★ SYNTAX:

```
ComboBox comboBox = new ComboBox();
```

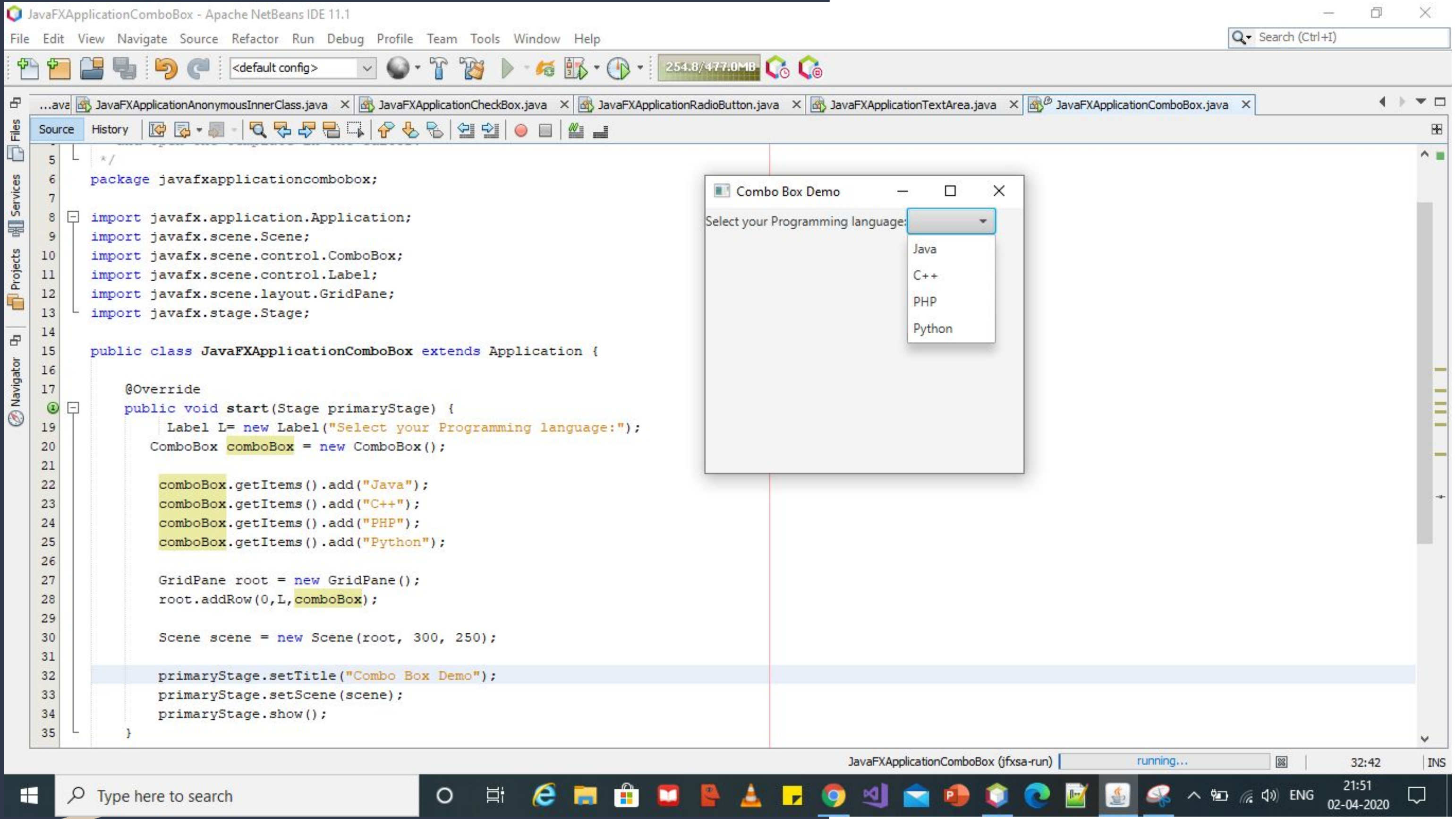
Combo Box

Adding Choices to a ComboBox

You can add choices to a ComboBox by obtaining its item collection and add items to it. Here is an example that adds choices to a JavaFX ComboBox :

```
cb.getItems().add("Choice 1");

cb.getItems().add("Choice 2");

cb.getItems().add("Choice 3");
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY


JavaFXApplicationComboBox - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
JavaFXApplicationAnonymousInnerClass.java
JavaFXApplicationCheckBox.java
JavaFXApplicationRadioButton.java
JavaFXApplicationTextArea.java
JavaFXApplicationComboBox.java
Source History
JavaFXApplicationComboBox (jfxsa-run) running...
32:42 INS

```java
 */
package javafxapplicationcombobox;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class JavaFXApplicationComboBox extends Application {

    @Override
    public void start(Stage primaryStage) {
         Label L= new Label("Select your Programming language:");
      ComboBox comboBox = new ComboBox();

        comboBox.getItems().add("Java");
        comboBox.getItems().add("C++");
        comboBox.getItems().add("PHP");
        comboBox.getItems().add("Python");

        GridPane root = new GridPane();
        root.addRow(0,L,comboBox);

        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle("Combo Box Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

```
Combo Box Demo
Select your Programming language:
Java
C++
PHP
Python
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

ListView

★ The JavaFX ListView control enables users to choose one or more options from a predefined list of choices. The JavaFX ListView control is represented by the class javafx.scene.control.ListView . This JavaFX ListView tutorial will explain how to use the ListView class.

★ SYNTAX:

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

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

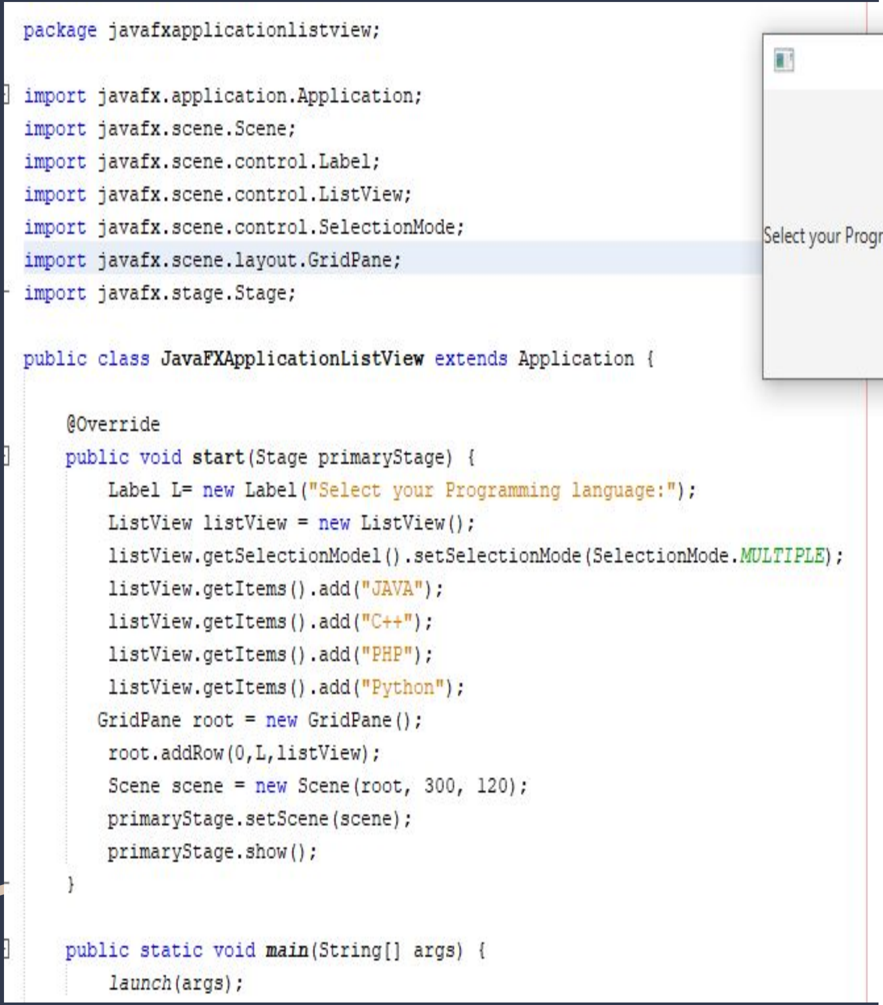
ListView

Adding Items to a ListView

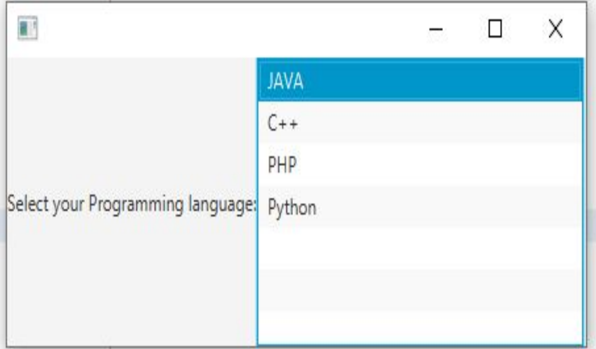
You can add items (options) to a ListView by obtaining its item collection and add items to it. Here is an example that adds items to a JavaFX ListView :

```
listView.getItems().add("Item 1");

listView.getItems().add("Item 2");

 listView.getItems().add("Item 3");
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY



```
package javafxapplicationlistview;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.SelectionMode;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class JavaFXApplicationListView extends Application {

    @Override
    public void start(Stage primaryStage) {
        Label L= new Label("Select your Programming language:");
        ListView listView = new ListView();
        listView.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
        listView.getItems().add("JAVA");
        listView.getItems().add("C++");
        listView.getItems().add("PHP");
        listView.getItems().add("Python");
       GridPane root = new GridPane();
        root.addRow(0,L,listView);
        Scene scene = new Scene(root, 300, 120);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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



Select your Programming language:

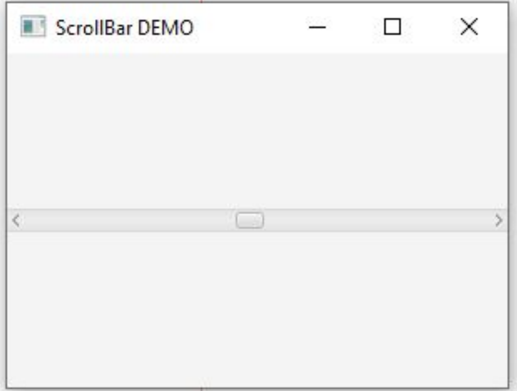
JAVA
C++
PHP
Python

Scrollbar

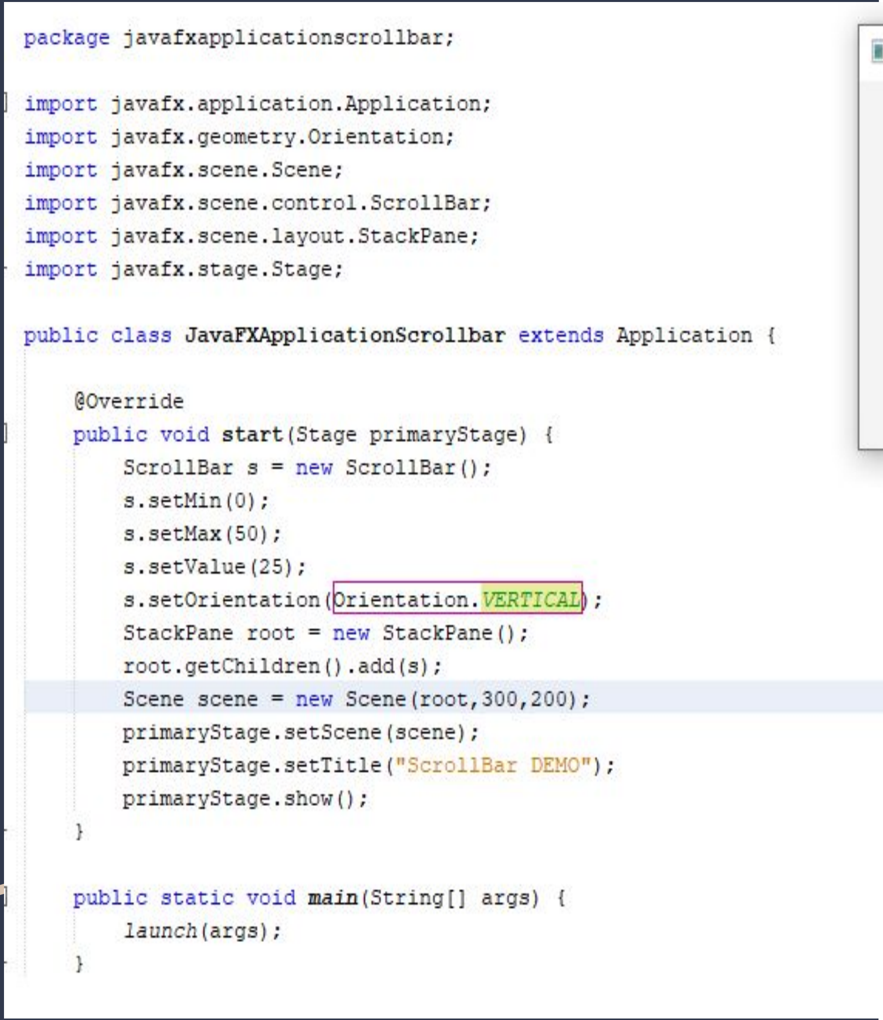
★ JavaFX Scroll Bar is used to provide a scroll bar to the user so that the user can scroll down the application pages. It can be created by instantiating **javafx.scene.control.ScrollBar** class.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

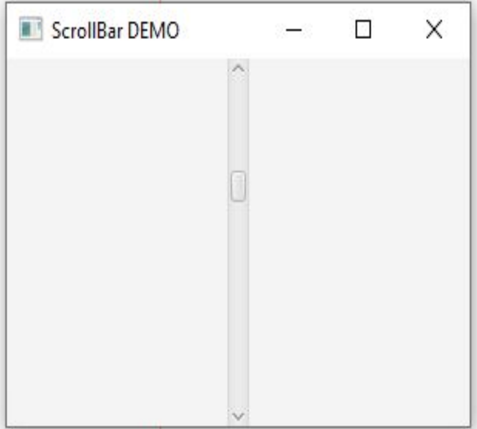
```java
package javafxapplicationscrollbar;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.ScrollBar;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class JavaFXApplicationScrollbar extends Application {

    @Override
    public void start(Stage primaryStage) {
        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 DEMO");
        primaryStage.show();
    }

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

}
```



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

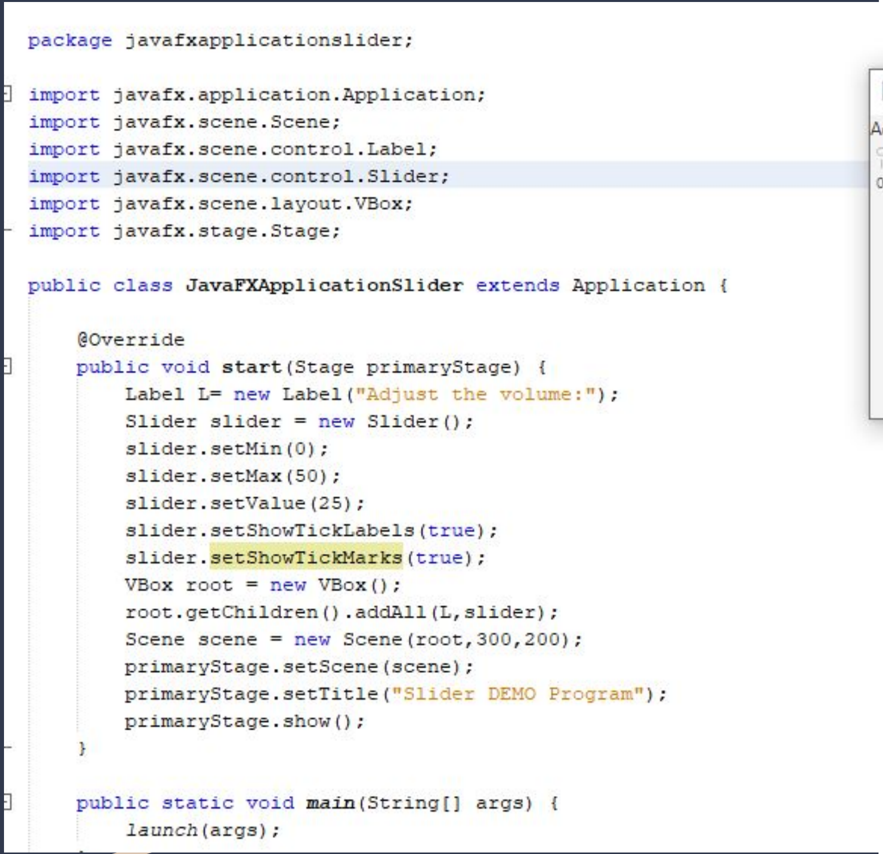


```
package javafxapplicationscrollbar;

import javafx.application.Application;
import javafx.geometry.Orientation;
import javafx.scene.Scene;
import javafx.scene.control.ScrollBar;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class JavaFXApplicationScrollbar extends Application {

    @Override
    public void start(Stage primaryStage) {
        ScrollBar s = new ScrollBar();
        s.setMin(0);
        s.setMax(50);
        s.setValue(25);
        s.setOrientation(Orientation.VERTICAL);
        StackPane root = new StackPane();
        root.getChildren().add(s);
        Scene scene = new Scene(root,300,200);
        primaryStage.setScene(scene);
        primaryStage.setTitle("ScrollBar DEMO");
        primaryStage.show();
    }

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

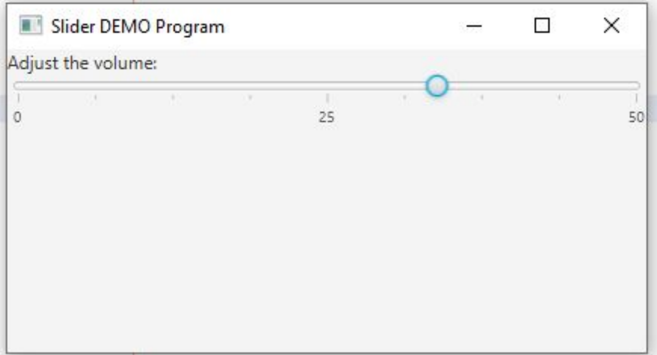


Slider

★ 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.



```
package javafxapplicationslider;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class JavaFXApplicationSlider extends Application {

    @Override
    public void start(Stage primaryStage) {
        Label L= new Label("Adjust the volume:");
        Slider slider = new Slider();
        slider.setMin(0);
        slider.setMax(50);
        slider.setValue(25);
        slider.setShowTickLabels(true);
        slider.setShowTickMarks(true);
        VBox root = new VBox();
        root.getChildren().addAll(L,slider);
        Scene scene = new Scene(root,300,200);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Slider DEMO Program");
        primaryStage.show();
    }

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



Video

★ Playing video in JavaFX is quite simple. We need to use the same API as we have used in the case of playing Audio files. In the case of playing video, we need to use the MediaView node to display the video onto the scene.

★ For this purpose, we need to instantiate the MediaView class by passing the Mediaplayer object into its constructor. Due to the fact that, MediaView is a JavaFX node, we will be able to apply effects to it.

★ In this part of the tutorial, we will discuss the steps involved in playing video media files and some examples regarding this.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Steps to play video files in JavaFX

1. Instantiate the **javafx.scene.media.Media** class by passing the location of the audio file in its constructor. Use the following line of code for this purpose.
   1. Media media = **new** Media("http://path/file_name.mp3");
2. Pass the Media class object to the new instance of **javafx.scene.media.MediaPlayer** object.
   1. Mediaplayer mediaPlayer = **new** MediaPlayer(media);
3. Invoke the MediaPlayer object's play() method when onReady event is triggered.
   1. mediaPlayer.setAutoPlay(**true**);

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

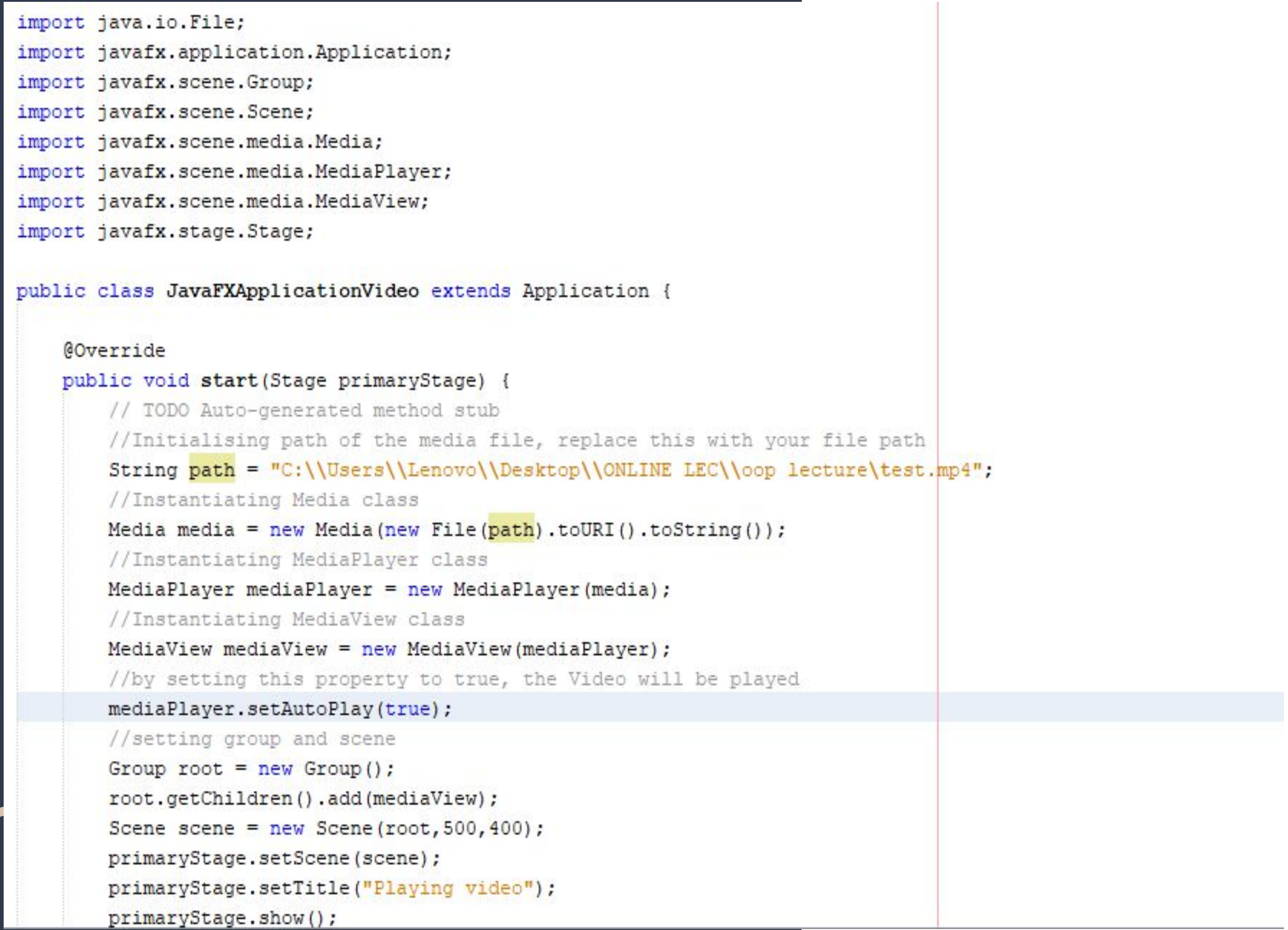
4.Instantiate MediaView class and pass Mediaplayer object into its constructor.

1. MediaView mediaView = **new** MediaView (mediaPlayer)

5.Add the MediaView Node to the Group and configure Scene.

```
Group root = new Group();

    root.getChildren().add(mediaView)

    Scene scene = new Scene(root,600,400);

    primaryStage.setTitle("Playing Video");

    primaryStage.show();
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



```java
import java.io.File;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;

public class JavaFXApplicationVideo extends Application {

    @Override
    public void start(Stage primaryStage) {
        // TODO Auto-generated method stub
        //Initialising path of the media file, replace this with your file path
        String path = "C:\\Users\\Lenovo\\Desktop\\ONLINE LEC\\oop lecture\test.mp4";
        //Instantiating Media class
        Media media = new Media(new File(path).toURI().toString());
        //Instantiating MediaPlayer class
        MediaPlayer mediaPlayer = new MediaPlayer(media);
        //Instantiating MediaView class
        MediaView mediaView = new MediaView(mediaPlayer);
        //by setting this property to true, the Video will be played
        mediaPlayer.setAutoPlay(true);
        //setting group and scene
        Group root = new Group();
        root.getChildren().add(mediaView);
        Scene scene = new Scene(root,500,400);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Playing video");
        primaryStage.show();
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Audio

- ★ We can load the audio files with extensions like **.mp3,.wav** and **.aifff** by using JavaFX Media API. We can also play the audio in HTTP live streaming format. It is the new feature introduced in JavaFX 8 which is also known as HLS.
- ★ Playing audio files in JavaFX is simple. For this purpose, we need to instantiate **javafx.scene.media.Media** class by passing the audio file path in its constructor. The steps required to be followed in order to play audio files are described below.

Audio

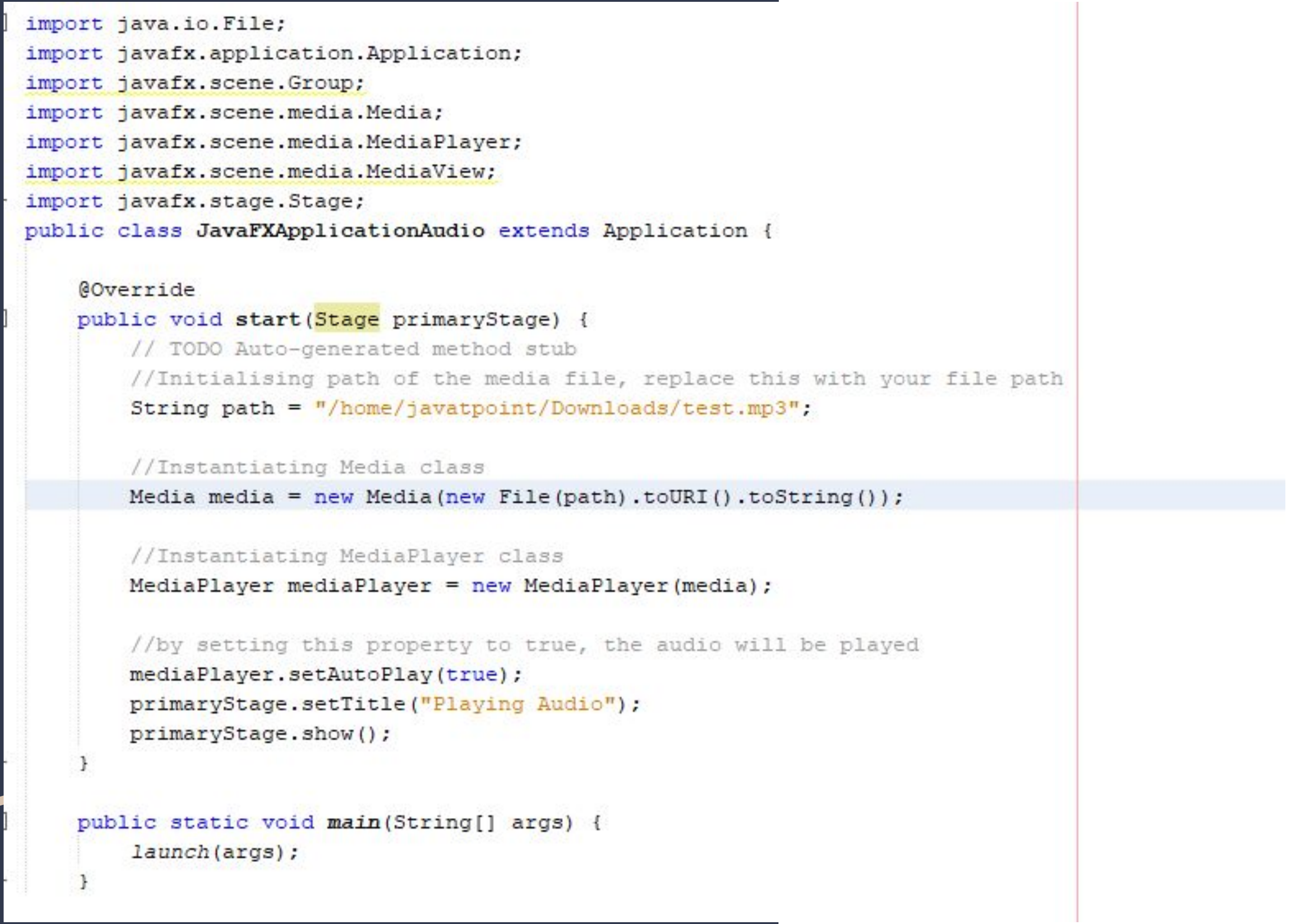
1. Instantiate the **javafx.scene.media.Media** class by passing the location of the audio file in its constructor. Use the following line of code for this purpose.
    1. Media media = **new** Media("http://path/file_name.mp3");
2. Pass the Media class object to the new instance of **javafx.scene.media.MediaPlayer** object.
    1. Mediaplayer mediaPlayer = **new** MediaPlayer(media);
3. Invoke the MediaPlayer object's play() method when onReady event is triggered.
    1. MediaPlayer.setAutoPlay(**true**);

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Audio

★ The Media File can be located on a web server or on the local file system. SetAutoPlay() method is the short-cut for setting the setOnReady() event handler with the lambda expression to handle the event.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



```
import java.io.File;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;
public class JavaFXApplicationAudio extends Application {

    @Override
    public void start(Stage primaryStage) {
        // TODO Auto-generated method stub
        //Initialising path of the media file, replace this with your file path
        String path = "/home/javatpoint/Downloads/test.mp3";

        //Instantiating Media class
        Media media = new Media(new File(path).toURI().toString());

        //Instantiating MediaPlayer class
        MediaPlayer mediaPlayer = new MediaPlayer(media);

        //by setting this property to true, the audio will be played
        mediaPlayer.setAutoPlay(true);
        primaryStage.setTitle("Playing Audio");
        primaryStage.show();
    }

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

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY