Week7 – Presentation 3

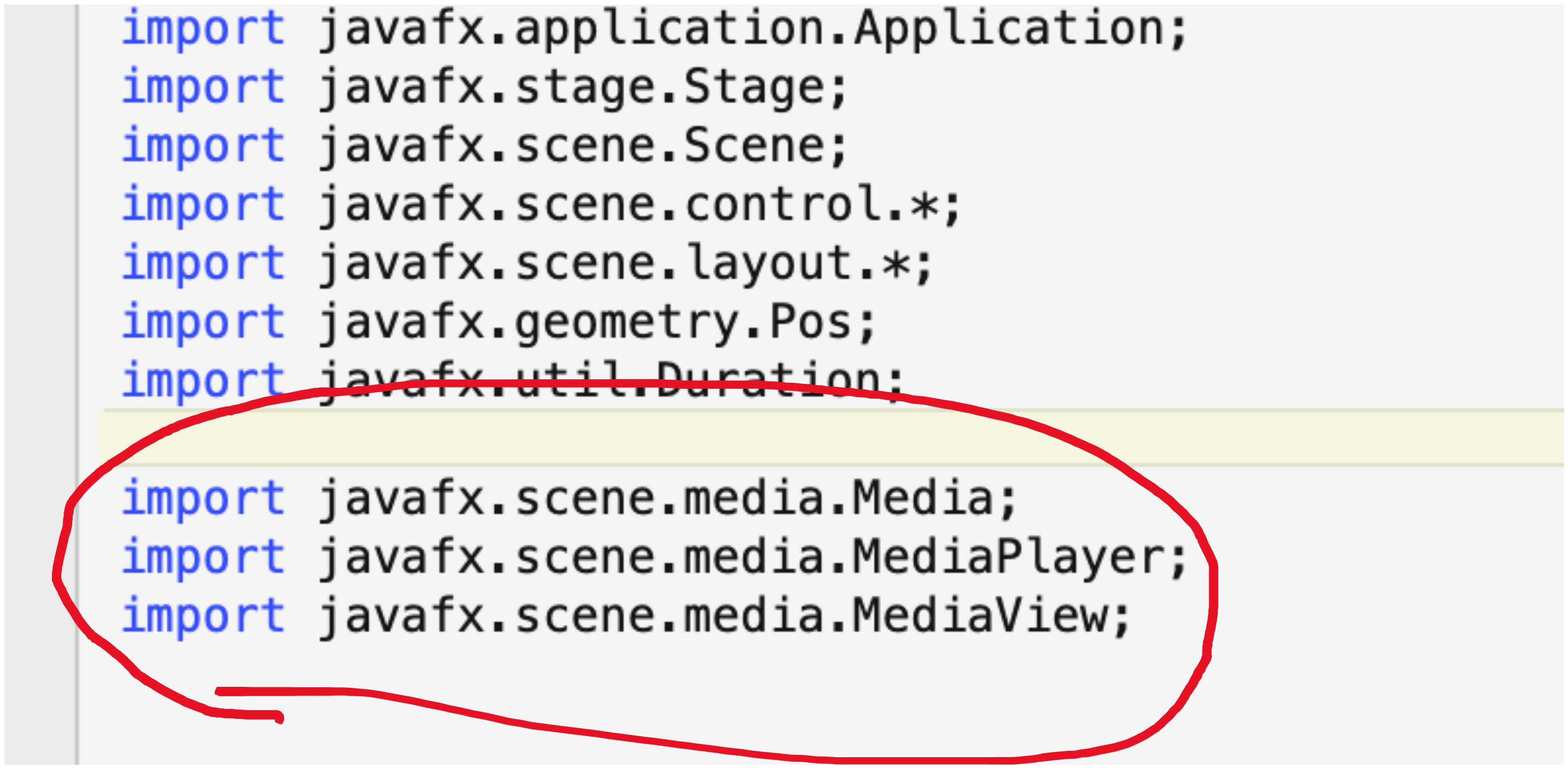
Media

Media

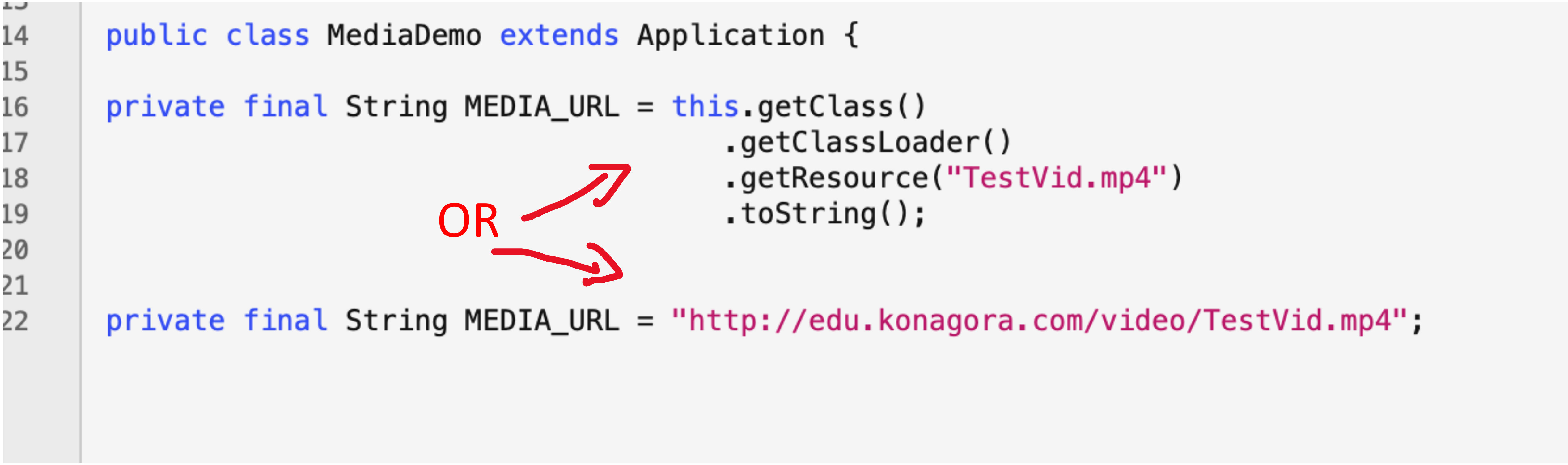
You can also play audio and video in JavaFX. It's not very different from images. With images you have an Image object, and the ImageView that shows it on screen. With audio and video, you have a Media object, you have a MediaView, and between the two you have a MediaPlayer object with controls allowing you to start, pause, stop, rewind and so forth.

Very much like Images

- Media
- MediaPlayer $\longleftarrow$ Controls
- MediaView



```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.*;
import javafx.geometry.Pos;
import javafx.util.Duration;

import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
```



```
public class MediaDemo extends Application {

private final String MEDIA_URL = this.getClass()
                                     .getClassLoader()
                                     .getResource("TestVid.mp4")
                                     .toString();
```

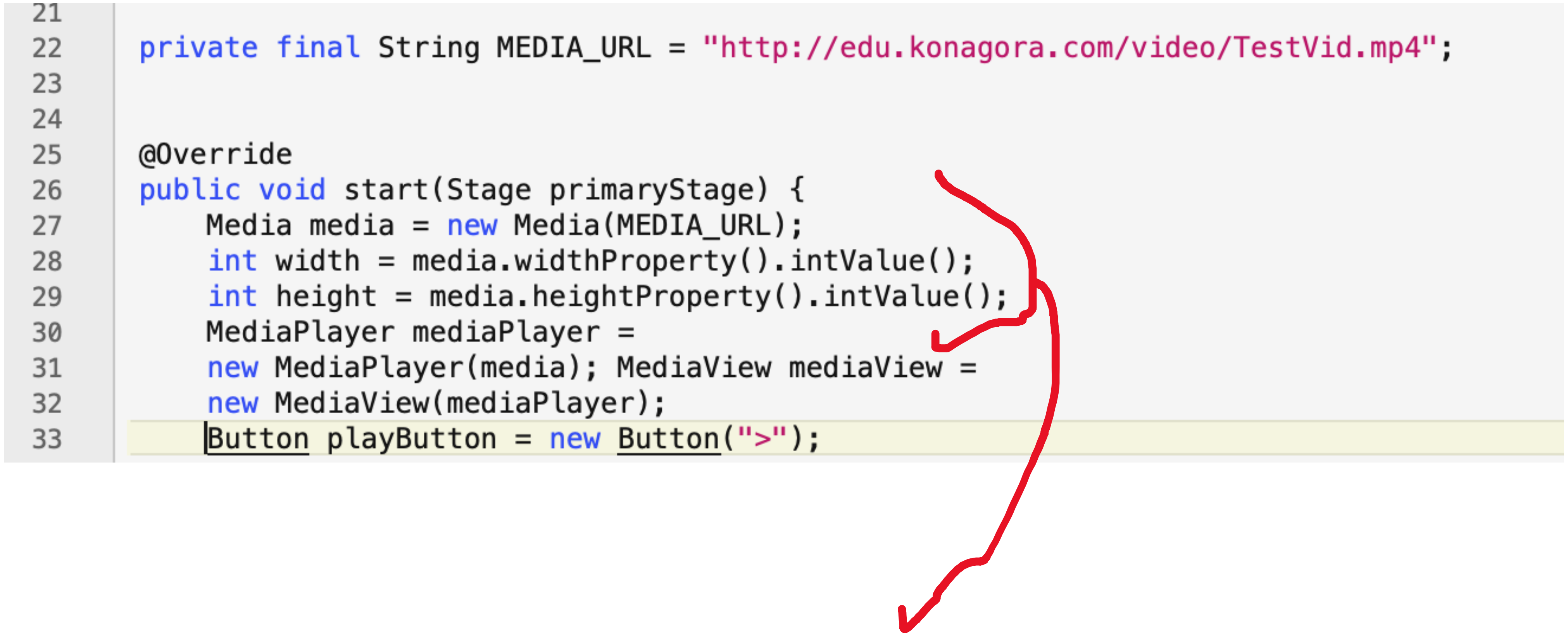
OR

```
private final String MEDIA_URL = "http://edu.konagora.com/video/TestVid.mp4";
```

URL, Path and String

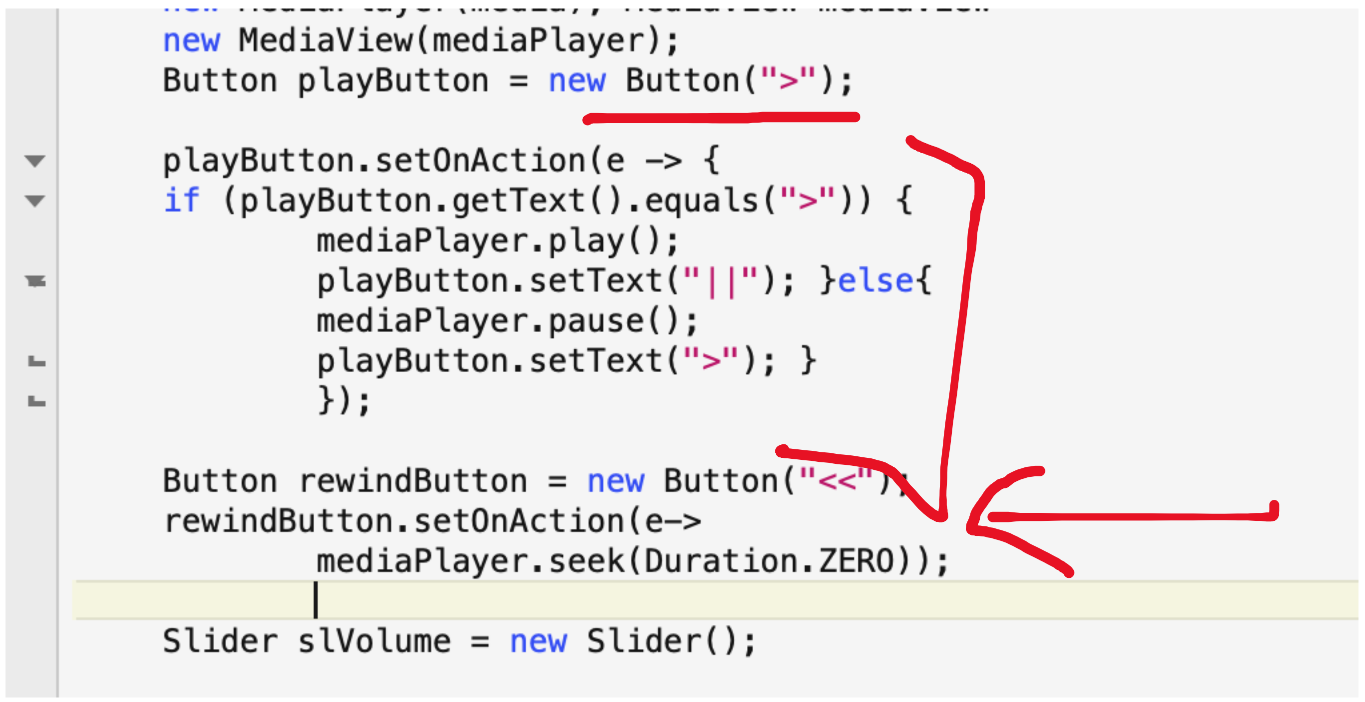
- URL: prefix://path
- PATH: path

A Media (like an Image) can take a URL as argument of a constructor. An URL (like an URI, basically the same thing) is a prefix + a path. If the prefix is "file:" it means that the resource (... name given to anything you can load) is accessible through the file system of your computer (it's not necessarily local, it can be a network disk). It can also be something else such as "http:" to mean that the resource is accessed from a web server through HTTP requests. You usually need to apply toString() to them.



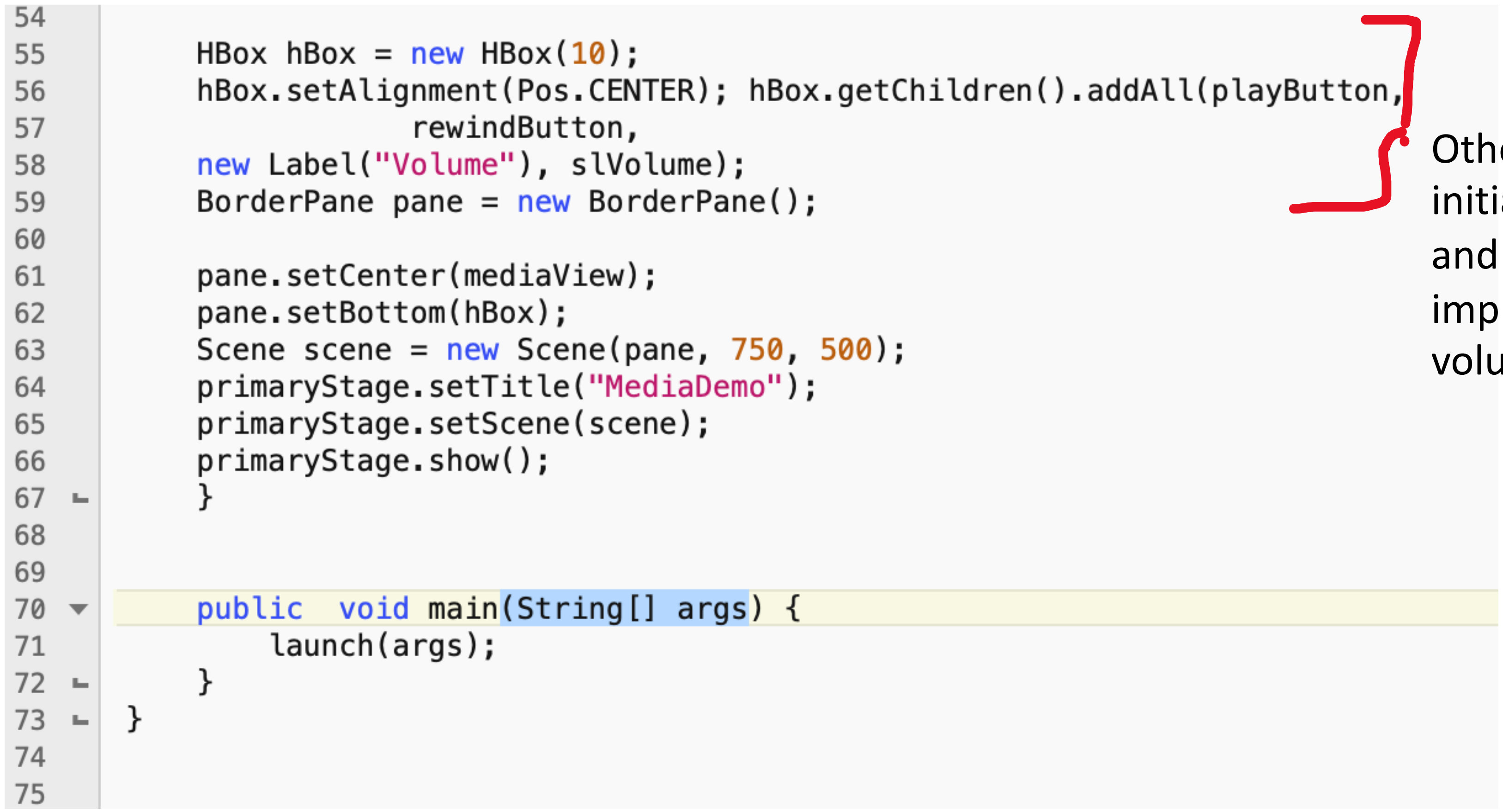
```
private final String MEDIA_URL = "http://edu.konagora.com/video/TestVid.mp4";


@Override
public void start(Stage primaryStage) {
    Media media = new Media(MEDIA_URL);
    int width = media.widthProperty().intValue();
    int height = media.heightProperty().intValue();
    MediaPlayer mediaPlayer =
    new MediaPlayer(media); MediaView mediaView =
    new MediaView(mediaPlayer);
    Button playButton = new Button(">");
```

Once the media is loaded, you associate it with a MediaPlayer, and the MediaPlayer with a MediaView. Controls will execute MediaPlayer methods.



```
    new MediaView(mediaPlayer);
    Button playButton = new Button(">");

    playButton.setOnAction(e -> {
    if (playButton.getText().equals(">")) {
            mediaPlayer.play();
            playButton.setText("||"); }else{
            mediaPlayer.pause();
            playButton.setText(">"); }
            });

    Button rewindButton = new Button("<<");
    rewindButton.setOnAction(e->
            mediaPlayer.seek(Duration.ZERO));

    Slider slVolume = new Slider();
```

The text on the button tells us what is the current state, and whether we should play or pause.
I'm also adding another button for rewinding, and a new widget (Slider) for setting the volume.

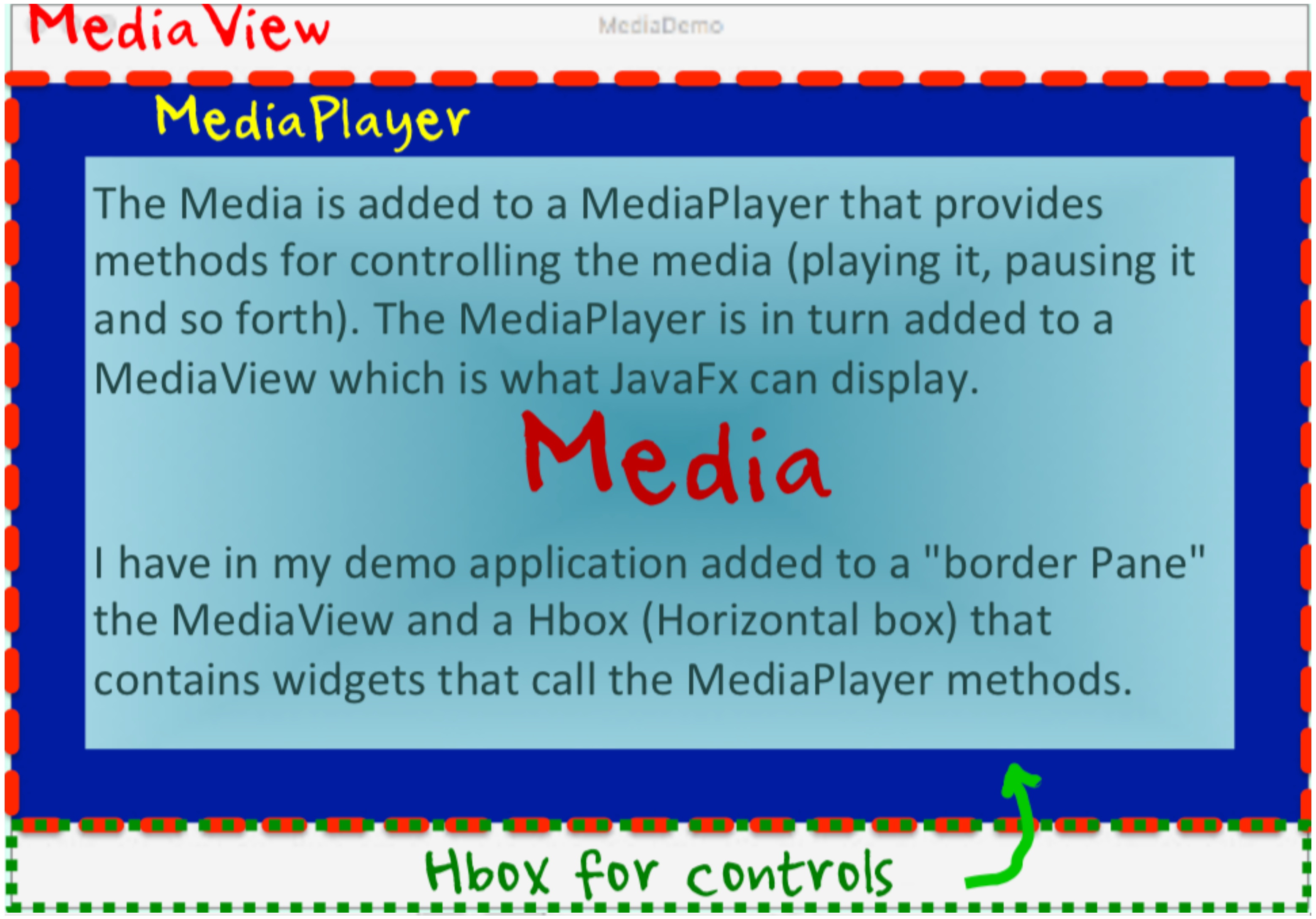


```java
    HBox hBox = new HBox(10);
    hBox.setAlignment(Pos.CENTER); hBox.getChildren().addAll(playButton,
                rewindButton,
    new Label("Volume"), slVolume);
    BorderPane pane = new BorderPane();

    pane.setCenter(mediaView);
    pane.setBottom(hBox);
    Scene scene = new Scene(pane, 750, 500);
    primaryStage.setTitle("MediaDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
    }


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

Other than geometry (size) I give the range and initial value for the slider (0 to 100, initially 50) and "bind" it to the MediaPlayer. There is an implicit ChangeListener behind, to change the volume when the slider moves.

Controls are in a box, everything is added to a BorderPane (that controls placement as top/right/bottom/left and center) and we are ready to go.

This demo and the slides are by Stephane Faroult



The Media is added to a MediaPlayer that provides methods for controlling the media (playing it, pausing it and so forth). The MediaPlayer is in turn added to a MediaView which is what JavaFx can display.

I have in my demo application added to a "border Pane" the MediaView and a Hbox (Horizontal box) that contains widgets that call the MediaPlayer methods.