```
        //Instantiating MediaView class
        MediaView mView = new MediaView(player);
        player.play();

         //configure scene
        Scene scene = new Scene(new Pane(mView), 500, 400);
        stage.setScene(scene);  // Setting the scene to stage
        stage.setTitle("Video Example");
        stage.show();
    }
}
```

Now, you can compile and execute the above program from the command prompt. The output of the program is display as shown in figure-9.

Figure 9: Example for video incorporating in javaFX application

The above example is for video including in JavaFX application. In a similar way, you can include your audio also.

☞ Check Your Progress-2

1. Explain the term event. What steps are needed to be followed in order to handle the events?

.....

.....

.....

.....

2. What is the difference between foreground and background events?

.....

.....

3. Give names of animation in JavaFX. Explain any two of them with example.

2.6 SUMMARY

This unit explained about the JavaFX UI Controls, which are used in different perspectives such as creating and linking with CSS, building UI with FXML, event handling, effects, animation and media incorporating with UI. Also, in this unit it is explained that how to create and apply CSS on UI control elements with examples. You have also learnt how to build javaFX UI with FXML. The FXML is a XML based language, and it is used for constructing Java object graphs. FXML offers an appropriate alternative to constructing such graphs in procedural code. You also got knowledge about events handling, and it is used when users interact with UI controls.

This unit explained about giving effects and animation to javaFX UI controls elements. At the end of this unit, you have learnt about the media integration in JavaFX application using API that enables you to integrate audio and video into the applications. The package javafx.scene.media covers all the essential classes for making JavaFX applications. This package contains three main classes such as Media, MediaPlayer and MediaView. The package is used for media playback. Now you are able to work with UI controls.

2.7 SOLUTIONS/ANSWERS TO CHECK YOUR PROGRESS

☞ Check Your Progress 1

1) CSS stands for Cascading Style Sheets which are used for adding style such as fonts, colors, backgrounds and spacing between the paragraphs in HTML documents. It is used to control the presentation of one or more web pages. JavaFX CSS is based on the W3C CSS with some additional features. The objective of JavaFX CSS is to allow developers (who are already at ease with CSS for HTML) to use CSS for customizing JavaFX controls. There are two ways to add CSS to JavaFX application by your own Style Sheet and/or inline Style Sheet.

You can add your own external style sheet to a scene in JavaFX application like the following:

```
Scene sc = new Scene(root, 400, 200);
sc.getStylesheets().add("path/name_of_stylesheet.css");
```

Using inline style, you can set CSS styles for a specific element by setting the CSS properties directly on the element by using the setStyle() method, for example, inline style sheet for a button.

```
Button button = new Button("Reset");
button.setStyle("-fx-background-color: #FFbbbb");
```

For more details, you can refer section 2.2 in this Unit.

2) The difference between an ID and a class selector is that an ID is only used to identify a single element in javaFX application whereas class selector is used to identify more than one element. The IDs are only used when one element in the web page should have a particular style applied to it. The name of class selector is preceded by dot(.) character, while the name of ID is preceded by the hash(#) symbol to select the element.

3) A scene graph is a data structure that denotes the contents of a window. The scene graph consists of nodes that show graphical components in the window, such as buttons, radio buttons and checkbox. A scene graph is represented by package javafx.scene in a JavaFX application. The scene class holds all the contents of a scene graph. Some of the nodes behave as containers that contains other nodes. The GUI in a window is created by building a scene graph. For more details, you may refer section 2.2 of this unit.

4) FXML is an XML-based user interface markup language for defining the user interface of a JavaFX application. It is created by Oracle Corporation. It is another form of designing user interfaces using procedural code. You may refer to section 2.3 of this unit for viewing the example.

☞ Check Your Progress 2

1) An event is an object which describes a state change in a source. It generates whenever a user interacts with UI control elements. It is generated by clicking a mouse, pressing a button, entering a character using the keyboard. It can also generate by without human interaction, such as timer expires, a software or hardware failure etc.

The event handling consists of four stages: target selection, route construction, Event capturing, and bubbling. In the target selection phase, when any user interacts with UI controls then an action occurs, at that time the system decides which node gets focus; this happens in target selection phase. Whenever an event is generated, an event dispatch route is created to handle the events in the stage of route construction. This dispatch route comprises the path from the stage to the node on which the event is generated. The event route is regulated by the event dispatch chain which was formed in the operation of the buildEventDispatchChain() method of the selected event target.

In Event capturing stage, the event is transmitted from the source node. The generated event is traversed all the nodes in the route from top to bottom. If the event filter is registered with any of these nodes which are in the dispatch route then event will be executed otherwise, it will be transferred to the target node. The target node processes the event in that case. When the event is captured and processed by the target node or registered filter, the event starts navigationing all the nodes again from the bottom to the root node. If any of the node in the dispatch chain has handler which is registered for the type of event happened then that handler is called, and it will get executed otherwise the process is returned to the next node up in the route and repeat the process. If a handler does not consume the event, then the root node ultimately receives such event and processing is finished. These happen in the event bubbling phase.

2) Foreground Events are based on the direct interaction of a user (e.g. button is pressed), while Background Events involve the interaction of end-user such as hardware or software failure, timer expiry. Foreground Events are created by a person directly interacting with the graphical components in a GUI. For example, double-clicking on a button, moving the mouse, input character through a keyboard, selecting an item from the menu, scrolling the page, etc. The hardware or software failure, operation completion interruptions are the example of background events.

3) The names of animations are Fade Transition, Fill Transition, Rotate Transition, Scale Transition, Stroke Transition, Translate Transition, Path Transition, Sequential Transition, Pause Transition, Parallel Transition, etc. All these transitions are represented by individual classes in the package javafx.animation.

   Translate transition is used to create movement/transition from one point to another point within a defined duration. It is done by keep changing the translateX and translateY properties of the node at the regular interval. In JavaFX, this animation is represented by the class javafx.animation.TranslateTransition. Using **Rotate transition** feature**,** you can rotate an object. You can set rotation by using 'toAngle' and 'byAngle' method. The 'toAngle' indicates what angle the node should rotate, and 'byAngle' indicates how much it should rotate from the current angle of rotation.

2.8 FURTHER READINGS

- Carl Dea, Gerrit Grunwald, José Pereda, Sean Phillips and Mark Heckler "JavaFX 9 by Example", Apress,2017.
- Sharan, Kishori. Beginning Java 8 APIs, Extensions and Libraries: Swing, JavaFX, JavaScript, JDBC and Network Programming APIs. Apress, 2014.
- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/index.htmlhttps://www.w3.org/Style/CSS/Overview.en.html
- https://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html
- https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm
- https://docs.jboss.org/richfaces/latest_4_5_X/Developer_Guide/en-US/html/chap-Developer_Guide-Skinning_and_theming.html
- https://docs.oracle.com/javafx/2/get_started/fxml_tutorial.htm
- https://docs.oracle.com/javafx/2/api/javafx/fxml/doc-files/introduction_to_fxml.html
- https://docs.oracle.com/javafx/2/events/processing.htm
- https://docs.oracle.com/javafx/2/events/filters.htm
- https://docs.oracle.com/javase/8/javafx/api/javafx/scene/effect/Effect.html
- https://docs.oracle.com/javase/8/javafx/get-started-tutorial/animation.htm
- https://docs.oracle.com/javafx/2/api/javafx/scene/media/package-summary.html