Java2 Lab 11
(JavaFx3)

[Experimental Objective]

1. 理解JavaFx的视频，音频播放
2. 通过编写css文件，改变布局和样式

[JavaFx音乐播放器]

参照理论课课件，用javafx，写一个支持音频和视频格式的音乐播放器。

用Media传入视频的Url,可以使用本地文件也可以使用远程的网站，MP3，MP4等格式都支持。

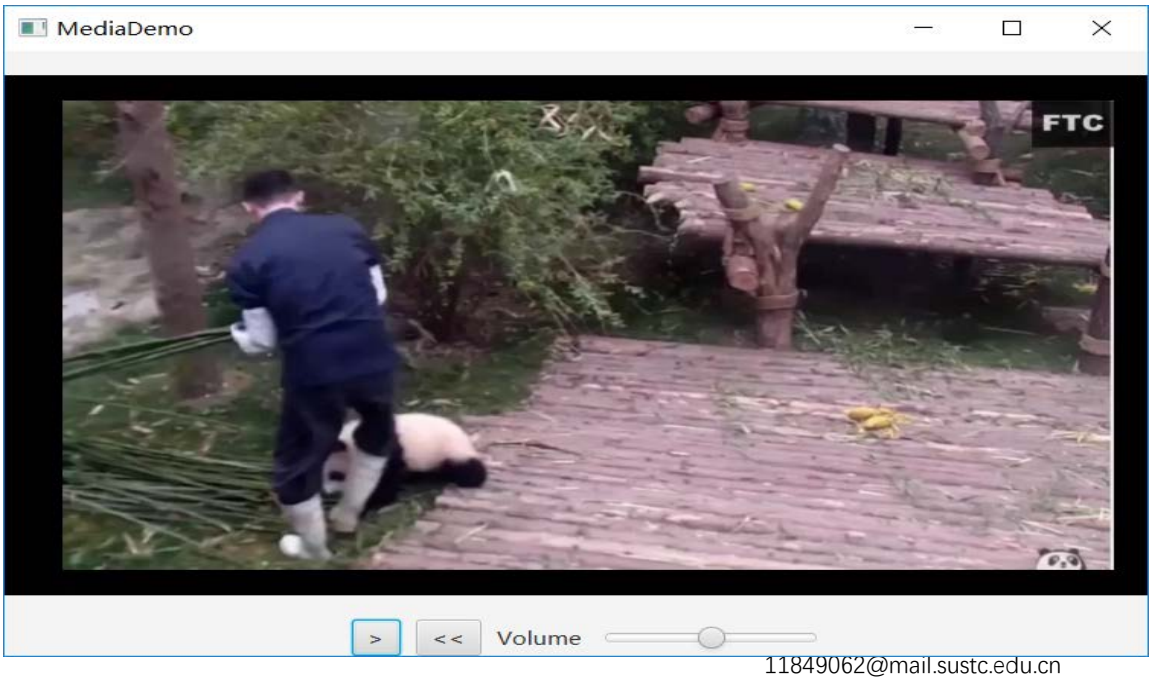
建一个播放的按钮，用>表示，点击它以后激活play()行为，按钮展示内容改为||。如果点击||，会触发pause()暂停播放，并展示为>。

再建一个重新播放按钮<<，点击会触发seek(Duration.ZERO)，返回起始位置重新播放。

最后建一个音量滑动调节窗，根据滑动的距离所占的百分比，调节音量的变化。

```
1.  public class testVideo1 extends Application {
2.
3.      private final String MEDIA_URL =
4.              "http://edu.konagora.com/video/TestVid.mp4";
5.
6.      @Override
7.      public void start(Stage primaryStage) {
8.
9.          Media media = new Media(MEDIA_URL);
10.         int width = media.widthProperty().intValue();
11.         int height = media.heightProperty().intValue();
12.         MediaPlayer mediaPlayer =
13.         new MediaPlayer(media);
14.         MediaView mediaView =
15.         new MediaView(mediaPlayer);
16.         Button playButton = new Button(">");
17.
18.         playButton.setOnAction(e -> {
19.             if (playButton.getText().equals(">")) {
20.             mediaPlayer.play();
21.             playButton.setText("||");
22.             } else {
23.             mediaPlayer.pause();
24.             playButton.setText(">");
25.             }
26.             });
27.             Button rewindButton = new Button("<<");
```

```
28.             rewindButton.setOnAction(e->
29.             mediaPlayer.seek(Duration.ZERO));
30.             Slider slVolume = new Slider();
31.
32.
33.
34.             slVolume.setPrefWidth(150);
35.             slVolume.setMaxWidth(Region.USE_PREF_SIZE);
36.             slVolume.setMinWidth(30);
37.             slVolume.setValue(50);
38.             mediaPlayer.volumeProperty()
39.             .bind(slVolume.valueProperty()
40.             .divide(100));
41.
42.             HBox hBox = new HBox(10);
43.             hBox.setAlignment(Pos.CENTER);
44.             hBox.getChildren().addAll(playButton,
45.             rewindButton,
46.             new Label("Volume"),
47.             slVolume);
48.
49.             BorderPane pane = new BorderPane();
50.
51.             pane.setCenter(mediaView);
52.             pane.setBottom(hBox);
53.             Scene scene = new Scene(pane, 750, 500);
54.             primaryStage.setTitle("MediaDemo");
55.             primaryStage.setScene(scene);
56.             primaryStage.show();
57.     }
58.
59.     public static void main(String[] args) {
60.         launch(args);
61.     }
62. }
```



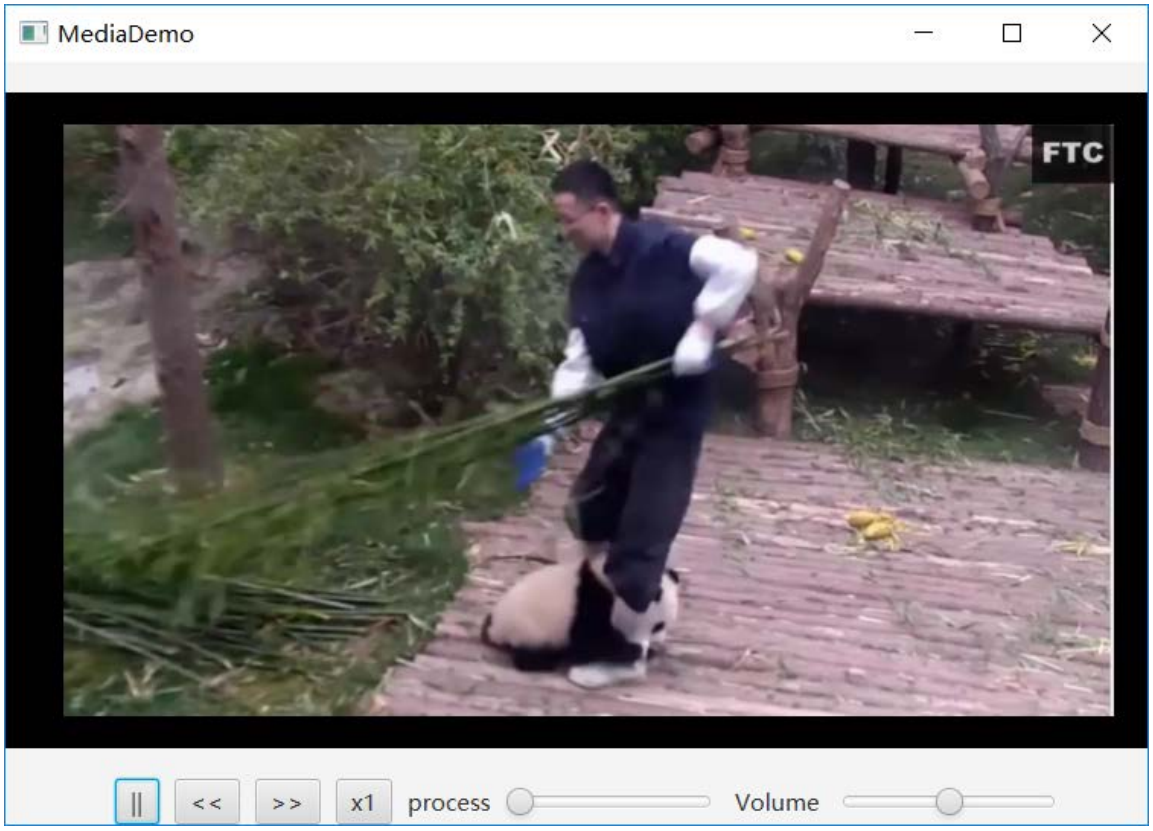
【练习1】

在上述代码的基础上，新加入以下功能：

（1） 仿照重新播放按钮<<，新建一个快速结束按钮>>，点击后能立即跳至视频最末端并结束播放。

（2） 仿照播放暂停按钮>,||,新建一个倍速改变按钮，点击后可以分别支持x1,x2,x4,x6,x8倍速播放，并在按钮中显示出当前播放倍速。

（3） 仿照音量调节按钮，新建一个播放进度条调节按钮，使移动进度条可以改变播放的进度。



[JavaFx中的Css]

为了简单起见，我们将java文件与css文件放在同一个目录下,在java中将scene与css文件绑定:

```
scene.getStylesheets().add(getClass().getResource("application.css").toExternalForm());
```

在 JavaFX 8 中默认的 CSS 风格源码是一个称为 modena.css 文件。该 CSS 文件可以在 JavaFX jar 文件 jfxrt.jar 中找到，它位于 Java 目录 `/jdk1.8.x/jre/lib/ext/jfxrt.jar`。
解压 `jfxrt.jar`，你应该能在 `com/sun/javafx/scene/control/skin/modena/` 目录下找到 `modena.css`。

缺省的样式表总是应用到 JavaFX 应用上。通过添加自定义样式表，我们可以通过覆盖 `modena.css` 中缺省的样式，对样式进行修改。

[css选择器]

选择器在场景图上定位 JavaFX 节点，然后我们可以使用 CSS 样式定义对它们进行样式化。
两种类型的选择器类型是 `id` 和 `class` 。
id 选择器是在场景节点上设置的唯一字符串名称。

类选择器是一个字符串名称，可以作为标记添加到任何 JavaFX 节点。

类选择器与 Java 类的概念无关。类选择器用于对节点进行分组以将它们与一个 CSS 样式定义一起设置样式。

[id选择器]

id 选择器是赋值给节点的唯一字符串名称。当使用 id 选择器时，我们将调用 JavaFX 节点对象上的 setId(String ID)方法来设置其 id。
例如，要定位 id 为 my-button 的 Button 实例，您将调用 setId(“my-button”)方法。
要为按钮设置 id 为 my-button 的样式，您将创建一个用 id 声明的 CSS 样式定义块 selector ＃my-button，如下所示:

```
#my-button {

    -fx-text-fill: rgba(17, 145,  213);

    -fx-border-color:  rgba(255, 255,  255,  .80);

    -fx-border-radius: 8;

    -fx-padding: 6  6  6  6;

}
```

此 CSS 样式块将应用于具有 `my-button` 的唯一标识的按钮。
CSS 选择器名称前面带有＃符号，而 Java 代码中的 id 不使用＃符号。

类选择器

当使用类型选择器时，我们将调用 getStyleClass()。add(String styleClass)方法向节点添加一个选择器。该方法允许我们有多个样式类来对节点进行样式。
getStyleClass()方法返回一个 ObservableList，我们可以添加和删除样式类来动态更新它的外观。
以下代码通过 `getStyleClass().add("num-button")`方法定位样式类包含 `num-button` 的按钮。
以下是用类选择器声明的 CSS 样式定义块 `.num-button` :

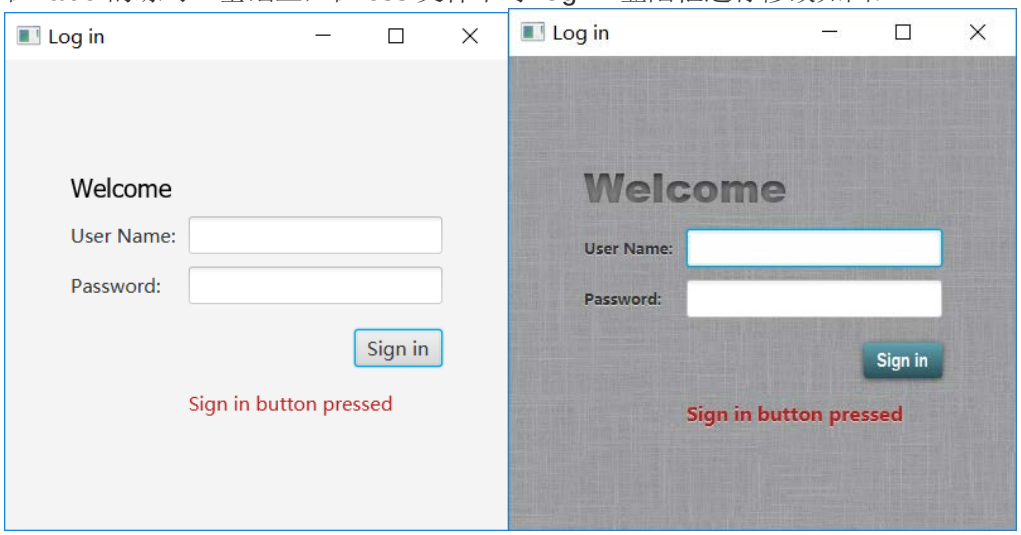
```
.num-button  {

    -fx-background-color: white, rgb(189,218,230), white;
```

```
    -fx-background-radius: 50%;

    -fx-background-insets: 0, 1, 2;

    -fx-font-family:  "Helvetica";
    -fx-text-fill:  black;
    -fx-font-size:  20px;
}
```

此 CSS 样式块将应用于具有样式类 `num-button` 的按钮。
CSS 选择器名称以`.`号为前缀，而 Java 代码中的选择器不使用。符号。

[案例2]

在 lab9 的练习二基础上，在 css 文件中对 log in 登陆框进行修改如图：



要求：背景要求使用如下图片（可以把下图另存为jpg添加到工程中）。修改welcome字体为Arial Narrow加粗，修改user name等label灰色加粗去阴影，点击user name 和password文本框时出现蓝色加粗边框，修改sign in 按钮风格及悬停效果，点击按钮时出现的action提示字体加粗

提示：

- **加载背景图片：**

```
.root {
   -fx-background-image: url("background.jpg");
}
```

- **修改 label 大小：**

```
.label {
   -fx-font-size: 12px;
   -fx-font-weight: bold;
```

```
    -fx-text-fill: #333333;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

- **按钮风格设置：**

```
.button {
    -fx-text-fill: white;
    -fx-font-family: "Arial Narrow";
    -fx-font-weight: bold;
    -fx-background-color: linear-gradient(#61a2b1, #2A5058);
    -fx-effect: dropshadow( three-pass-box , rgba(0,0,0,0.6) , 5, 0.0 , 0 , 1 );
}
```

- **按钮悬停设置：**

```
.button:hover {
    -fx-background-color: linear-gradient(#2A5058, #61a2b1);
}
```

- **设置文本 text:**

(1)java 中设置 id

```
text.setId("welcome-text");
```

(2)css中设置text样式

```
#welcome-text {
    -fx-font-size: 32px;
    -fx-font-family:"Arial Black";
    -fx-fill: #818181;
    -fx-effect: innershadow( three-pass-box , rgba(0,0,0,0.7) , 6,0.0,0,2 );
}
```

- **Action提示加粗：**

(1) java中设置id：

```
actiontarget.setId("actiontarget");
```

(2) css中实现action样式

```
#actiontarget {
    -fx-fill: FIREBRICK;
    -fx-font-weight: bold;
    -fx-effect: dropshadow( gaussian , rgba(255,255,255,0.5) , 0,0,0,1 );
}
```

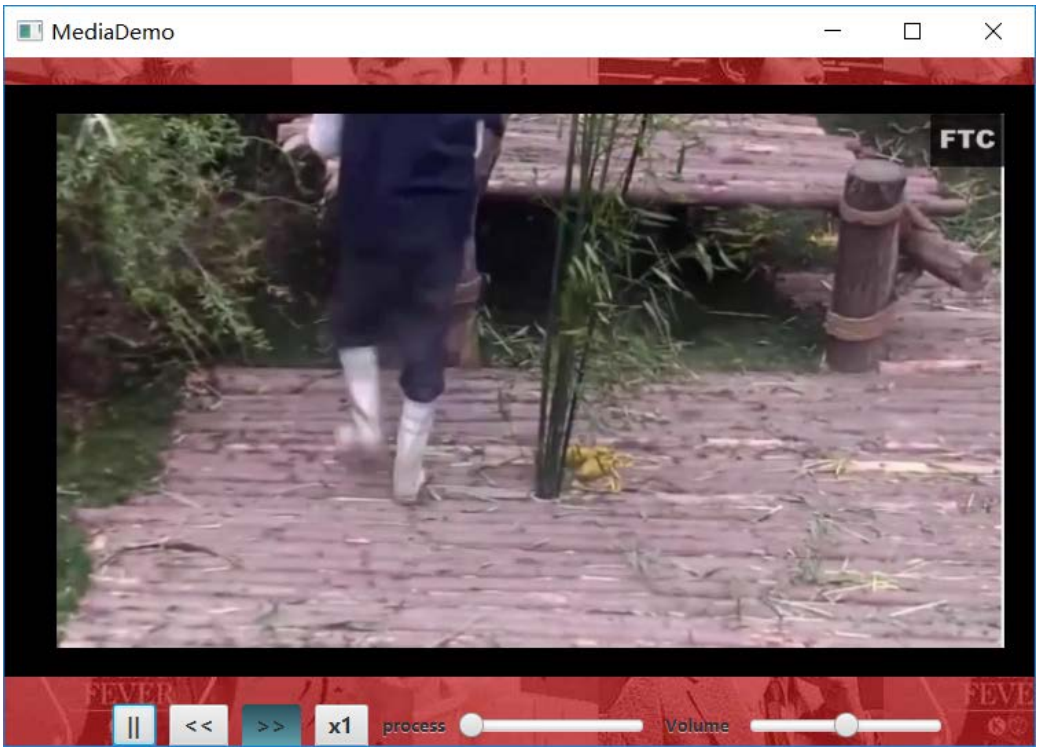
【练习2】

在练习1的基础上，用css修改播放器风格，要求：

（1） 修改播放器背景图片。可以用下面这个图片，也可以自己选择。

（2） 修改所有按钮显示的字体都加粗，标签带阴影加粗

（3） 按钮悬停会有颜色变化



11849062@mail.sustc.edu.cn