ADVANCED PROGRAMMING CONCEPTS USING JAVA

(CSX-331)

ASSIGNMENT-2

COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DR. B R AMBEDKAR NATIONAL INSTITUTE OF TECHNOLOGY

JALANDHAR – 144011, PUNJAB (INDIA)

JULY-DECEMBER, 2017

SUBMITTED To:	SUBMITTED BY:
PARAMVIR SINGH	Aakash
Associate Professor	15103031
Department of CSE	G-2,Sem-5

JavaFX

JavaFX is a Java library using which you can develop Rich Internet Applications. By using Java technology, these applications have a browser penetration rate of 76%.

What is JavaFX?

JavaFX is a Java library used to build Rich Internet Applications. The applications written using this library can run consistently across multiple platforms. The applications developed using JavaFX can run on various devices such as Desktop Computers, Mobile Phones, TVs, Tablets, etc.

To develop **GUI Applications** using Java programming language, the programmers rely on libraries such as **Advanced Windowing Toolkit** and **Swings**. After the advent of JavaFX, these Java programmers can now develop GUI applications effectively with rich content.

Need for JavaFX

To develop **Client Side Applications** with rich features, the programmers used to depend on various libraries to add features such as Media, UI controls, Web, 2D and 3D, etc. JavaFX includes all these features in a single library. In addition to these, the developers can also access the existing features of a Java library such as **Swings**.

JavaFX provides a rich set of graphics and media API's and it leverages the modern **Graphical Processing Unit** through hardware accelerated graphics. JavaFX also provides interfaces using which developers can combine graphics animation and UI control.

One can use JavaFX with JVM based technologies such as Java, Groovy and JRuby. If developers opt for JavaFX, there is no need to learn additional technologies, as prior knowledge of any of the above-mentioned technologies will be good enough to develop RIA's using JavaFX.

Features of JavaFX

Following are some of the important features of JavaFX −

- **Written in Java** − The JavaFX library is written in Java and is available for the languages that can be executed on a JVM, which include − **Java, Groovy and JRuby**. These JavaFX applications are also platform independent.
- **FXML** − JavaFX features a language known as FXML, which is a HTML like declarative markup language. The sole purpose of this language is to define a user Interface.
- **Scene Builder** − JavaFX provides an application named Scene Builder. On integrating this application in IDE's such as Eclipse and NetBeans, the users can access a drag and drop design interface, which is used to develop FXML applications (just like Swing Drag & Drop and DreamWeaver Applications).
- **Swing Interoperability** − In a JavaFX application, you can embed Swing content using the **Swing Node** class. Similarly, you can update the existing Swing

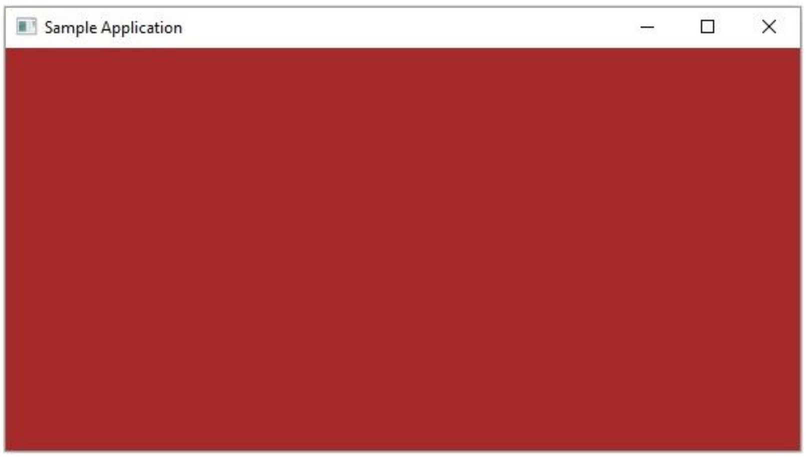
applications with JavaFX features like embedded web content and rich graphics media.

- **Built-in UI controls** – JavaFX library caters UI controls using which we can develop a full-featured application.
- **CSS like Styling** – JavaFX provides a CSS like styling. By using this, you can improve the design of your application with a simple knowledge of CSS.
- **Canvas and Printing API** – JavaFX provides Canvas, an immediate mode style of rendering API. Within the package **javafx.scene.canvas** it holds a set of classes for canvas, using which we can draw directly within an area of the JavaFX scene. JavaFX also provides classes for Printing purposes in the package **javafx.print**.
- **Rich set of API's** – JavaFX library provides a rich set of API's to develop GUI applications, 2D and 3D graphics, etc. This set of API's also includes capabilities of Java platform. Therefore, using this API, you can access the features of Java languages such as Generics, Annotations, Multithreading, and Lambda Expressions. The traditional Java Collections library was enhanced and concepts like observable lists and maps were included in it. Using these, the users can observe the changes in the data models.
- **Integrated Graphics library** – JavaFX provides classes for **2d** and **3d** graphics.
- **Graphics pipeline** – JavaFX supports graphics based on the Hardware-accelerated graphics pipeline known as Prism. When used with a supported Graphic Card or GPU it offers smooth graphics. In case the system does not support graphic card then prism defaults to the software rendering stack.

Example code:

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.stage.Stage;

public class JavafxSample extends Application {

   @Override

   public void start(Stage primaryStage) throws Exception {
      //creating a Group object
      Group group = new Group();

      //Creating a Scene by passing the group object, height and width
      Scene scene = new Scene(group ,600, 300);

      //setting color to the scene
      scene.setFill(Color.BROWN);

      //Setting the title to Stage.
      primaryStage.setTitle("Sample Application");
      //Adding the scene to Stage
      primaryStage.setScene(scene);

      //Displaying the contents of the stage
      primaryStage.show();
   }
   public static void main(String args[]){
      launch(args);
   }
}
```

Output:



Volatile

Volatile keyword in Java is used as an indicator to Java compiler and Thread that do not cache value of this variable and always read it from main memory .

Java volatile keyword cannot be used with method or class and it can only be used with variable. This keyword also guarantees visibility and ordering , after Java 5 write to any volatile variable happens before any read into volatile variable. volatile keyword also prevents compiler or JVM from reordering of code or moving away them from synchronization barrier.

Transient

Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.

Let's take an example, I have declared a class as Student, it has three data members id, name and age. If you serialize the object, all the values will be serialized but I don't want to serialize one value, e.g. age then we can declare the age data member as transient.

```
import java.io.Serializable;
public class Student implements Serializable{
 int id;
 String name;
 transient int age;//Now it will not be serialized
 public Student(int id, String name,int age) {
  this.id = id;
  this.name = name;
  this.age=age;
 }
}
```