Student name : Stanimir Belomelski

Homework10 : JAVAFX

History of JAVAFX

JavaFX was developed by Chris Oliver at SeeBeyond and it was called Form Follows Function or F3.

F3 was a Java scripting language for easily developing GUI applications. It offered declarative syntax, static typing, type inference, data binding, animation, 2D graphics, and Swing components.[1]

Later the company was bought by Sun Microsystems and the so-called language F3 was renamed to JavaFX in 2007.Oracle acquired Sun Microsystems in 2010 and they made JavaFX to be open sourced in 2013. JavaFX became very popular because you can develop Rich Internet Applications. By using Java Technology, these applications have a browser penetration rate of 76%.[2]

The first version of JavaFX was released in 2008. The current release for JavaFX is version 8.0. The version number jumped from 2.2 to 8.0. From Java 8, the version numbers of Java SE and JavaFX will be the same. The major versions for Java SE and JavaFX will be released at the same time as well. We should use JavaFX because 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 **Swing**.

The Java Features are :

1. JavaFX is written in Java (no need to learn a new language). We need to use only Java and these JavaFX applications are also platform independent. [3]
2. Support CSS for skinning. By using CSS we can improve the design of your application with a simple knowledge of CSS. [3]
3. Support FXML: a XML-based declarative language to define the structure of the user interface separated from the application code. [3]
4. Swing interoperability: You can use Swing UI in JavaFX application.  You can update the existing Swing applications with JavaFX features like embedded web content and rich graphics media. [3]
5. WebView: for embedding HTML contents. [3]
6. 2D/3D Graphics : JavaFX provides classes for **2d** and **3d** graphics. [3]
7. Media: audio (mp3, wav, aiff), video (flv) and image: We can use broad aspect of audio and video files to develop our apps.
8. **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)[2]
9. **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**.[2]
10. **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.[2]

The advantages of JAVAFX are as we say : Using FXML, JavaFX Scene Builder, CSS support, JavaFX media support, HTML content, and Animation.[4]

The disadvantages with JavaFX are:

1.JavaFX has a considerably smaller community support compared to Swing[5]

2.Startup time of an application is too long compared to Swing applications.[5]

B. Packages in Java are:  a mechanism to encapsulate a group of classes, sub packages and interfaces.[6] The advantages of packages in Java are[6] :

- Preventing naming conflicts: for example there can be two classes with name Employee in two packages.

- Making searching/locating and usage of classes, interfaces, enumerations and annotations easier

- Providing controlled access: protected and default have package level access control. A protected member is accessible by classes in the same package and its subclasses. A default member (without any access specifier) is accessible by classes in the same package only.

- Packages can be considered as data encapsulation (or data-hiding).

We have two types of packages: **Built in Packages** and **User-defined Packages**

**Built in Packages -** These packages consist of a large number of classes which are a part of Java **API**. Some of the commonly used built-in packages are:

* **java.lang:**Contains language support classes
* **java.io:**Contains classed for supporting input / output operations.
* **java.util:**Contains utility classes which implement data structures like Linked List, Dictionary and support ; for Date / Time operations.
* **java.applet:**Contains classes for creating Applets.
* **java.awt:**Contain classes for implementing the components for graphical user interfaces (like button , ;menus etc).
* **java.net:**Contain classes for supporting networking operations.

**User-defined packages**  
These are the packages that are defined by the user. First we create a directory **myPackage** (name should be same as the name of the package). Then create the **MyClass** inside the directory with the first statement being the **package names**.

C)JAR file in Java is Java Archive file. It's a file format based on the popular ZIP file format and is used for aggregating many files into one. Although JAR can be used as a general archiving tool, the primary motivation for its development was so that Java applets and their requisite components (.class files, images and sounds) can be downloaded to a browser in a single HTTP transaction, rather than opening a new connection for each piece. We have to use JAR file because improves the speed with which an applet can be loaded onto a web page and begin functioning. The JAR format also supports compression, which reduces the size of the file and improves download time still further. Additionally, individual entries in a JAR file may be digitally signed by the applet author to authenticate their origin.[7]

How to create JAR file : for example we have

public static void main(String[] args) {

     System.out.println("Hello World!");

We can do this with *[javac](https://www.baeldung.com/javac)* from the command line:

|  |  |
| --- | --- |
|  | javac com/helloWorld/jar/\*.java  **To create the jar file, we are going to use the jar command.**  To use the jar command to create a jar file, we need to use the c option to indicate that we're creating a file and the f option to specify the file:  jar cf JarExample.jar com/helloWorld/jar/\*.class  **Setting a Manifest File** : The jar command provides functionality for providing our own manifest information  Main-Class: com.helloWorld.jar.JarExample  **It's important that we end our manifest file with a** **newline**. |
|  |  |

jar cfm JarExample.jar com/helloWorld/jar/example\_manifest.txt com/ helloWorld /jar/\*.class

d) At the heart of GitHub is Git, an open source project started by [Linux creator Linus Torvalds](https://beta.techcrunch.com/2012/04/19/an-interview-with-millenium-technology-prize-finalist-linus-torvalds/). Matthew McCullough, a trainer at GitHub, explains that Git, like other version control systems, manages and stores revisions of projects.[8] Although it’s mostly used for code, McCullough says Git could be used to manage any other type of file, such as Word documents or Final Cut projects. Think of it as a filing system for every draft of a document. When a developer makes changes, those changes are made directly to the central repository. With distributed version control systems like Git, if you want to make a change to a project you copy the whole repository to your own system. You make your changes on your local copy, then you “check in” the changes to the central server. McCullough says this encourages the sharing of more granular changes since you don’t have to connect to the server every time you make a change. GitHub is a Git repository hosting service, but it adds many of its own features. We should use GitHub because of fork, pull request and merge – are what make GitHub so powerful.

The flagship functionality of GitHub is “forking” – copying a repository from one user’s account to another. This enables you to take a project that you don’t have write access to and modify it under your own account.

Add project into GitHub account

1)At GitHub open your account.

2)Create new Repository and create a name

3)Go to the project folder and right click and press Get Bash here

4)Type git init (to initialize)

5)Type git add . (add all the files from the folder you choose for the project)

6)Type git omit -m “my JAVAFX project”

7)Type git status (to make sure the status of the project files are there)

8) We have to copy the repository URL

9)Type git remote add origin and Paste the copied URL

10)Type git push -f origin master (to push the files)