Seminar Paper

JavaFX: History, Concepts, Nutshell Examples

WU Vienna University of Economics and Business

Business Information Systems Seminar – 4167

Author:

Elise Landman, h1551237

Submitted on

June 3rd, 2020

Advisor:

Univ. Prof. Mag. Dr. Rony Flatscher

Declaration of Authorship

I assure:

to have individually written, to not have used any other sources or tools than referenced and to not have used any other unauthorized tools for the writing of this report.

to never have submitted this report topic to an advisor neither in this, nor in any foreign country.

that this report matches the report reviewed by the advisor.

Date: June 3rd, 2020 Signature: Elise Landman

[illegible]

Abstract

First released in 2008 by Sun Microsystems, JavaFX built upon AWT and Swing, two tools for building impressive Graphical User Interfaces (GUI) with the programming language Java. JavaFX also supports the famous “write once, run anywhere” paradigm, making it even simpler for developers to create GUI apps for desktops as well as mobile computers without the need of separately writing in different code for each platform. Therefore, JavaFX can be easily learned and adopted by beginners and advanced developers on either Windows, Mac OS or Linux and enables development of apps for desktops and mobile devices. This report focuses on the history of JavaFX, includes a quick start guide to start developing and demonstrates some nutshell examples and code snippets for serving as development references.

Table of Content

1. JavaFX – Introduction and General Overview

JavaFX is a software platform for building powerful Graphical User Interfaces (GUI) based on the programming language Java. These can include desktop applications and Rich Internet Applications (RIA) which can be executed on various different devices.

The JavaFX framework makes it easy to create next-generation and high-performance client-side applications while taking advantage of modern Graphic Processing Units (GPU) and providing a user-friendly interface. It enables users to easily apply animations with graphics and user-interface controls (Taman, 2015).

The standard toolkit supports Microsoft Windows Vista, 7, 8 and 10, Linux and MacOS for desktops. JavaFX can also be run on mobile devices with various operating systems (OS) such as Android, Windows mobile devices and Apple iOS devices (iPhone and iPad) and even on embedded devices such as Raspberry Pis (Wikpedia.org, 2020).

The next following chapters will lead to through the history, the underlying architecture, a quick start guide and some development examples of JavaFX.

2. History of JavaFX

2.1. The Initial Idea and Life Before JavaFX

The roots of JavaFX date back to the mid-90s, where it all started when Sun Microsystems released the general-purpose programming language Java. It followed the famous “Write Once, Run Anywhere (WORA)” principle, which promised an implementation on all popular platforms and operating systems (Wikipedia, 2020).

Shortly after the Java release, Sun Microsystems created a Java library which would simplify the development of desktop applications and its graphical user-interfaces (GUI). This library was originally (and still is) called Abstract Window Toolkit (AWT). It allows for creation of GUI components and event handling of those components (Wikipedia, 2020). Soon afterwards, Sun Microsystems released AWT’s successor Swing, which provided the user with even more powerful GUI components (JavaTpoint, 2020). In 2008, JavaFX was introduced and additionally supported more rich GUI components with increased advanced look and feels (Educba, 2020).

The following sections will highlight the detailed characteristics of AWT, Swing and JavaFX.

2.1.1. Abstract Window Toolkit (AWT)

AWT was first released with Java in 1995 by Sun Microsystems and is the standard user-interface widget toolkit for developing GUI applications in the programming language Java (Wikipedia.org, 2020). The AWT library can be imported into the Java environment through the command `java.awt` and its functions will become accessible. The package now allows creation and design of GUIs within the Java environment.

The example in figure 1 below shows the few lines of code that are necessary for creating a basic window user-interface as in figure 2 (JavaTpoint.com, 2020).

```java
import java.awt.*;
class First extends Frame{
First(){
Button b=new Button("click me");
b.setBounds(30,100,80,30);
add(b);
setSize(300,300);
setLayout(null);
setVisible(true);
}
public static void main(String args[]){
First f=new First();
}}
```

Figure 1 AWT basic window code
(JavaTpoint.com, 2020)

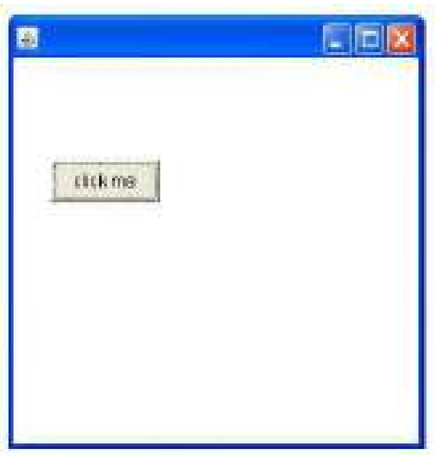


Figure 2 AWT basic window in Microsoft Windows XP (JavaTpoint.com, 2020)

One of the main characteristics of AWT is its components being platform dependent and thus being heavyweight (JavaTpoint, 2020). This can be observed with the development of user-interfaces, where AWT makes direct use of the underlying operating system's (OS) native graphical appearance. Therefore, when creating a checkbox, it will look differently when running the application on Microsoft Windows than on MacOS, even though it was written with the exact same lines of code. The different interfaces on each OS can be favorable for some development use-cases, but can also be a disadvantage when the goal is to develop a program which should constantly retain the exact same interface on multiple systems (Taman, 2015).

AWT became part of the well-known Java Foundation Classes (JFC), the application programming interface (API) for graphical implementations with Java (Wikipedia.org, 2020).

2.1.2. Swing

In 1996, AWT was succeeded by Java Swing which was developed to support a much more sophisticated collection of GUI components. One of the major features introduced in Swing was its platform-independency, therefore now allowing users to apply pluggable "look and feel" interfaces (JavaTpoint, 2020). Swing components were not anymore implemented by operating system-specific code but were written in their own Java code, making the framework lightweight. A standardized Java Swing look and feel "Ocean Look and Feel" is shown in figure 3 (Wikipedia.org, 2020).

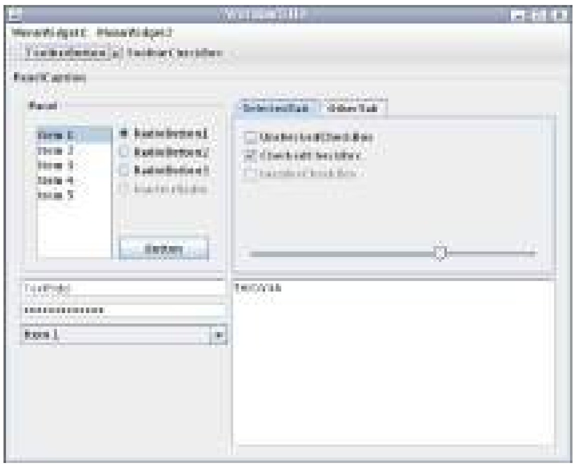


Figure 3 Swift basic widgets with the standard "Ocean Look and Feel"

Additionally, several new features were added such as f. e. tabbed panels, scroll panes, trees, tables, and lists (Yap, 2003). These could now share the same appearance on every OS.

Java Swing is included in the JFC (Wikipedia.org, 2020).

2.1.3. JavaFX

12 years after the initial release of Java Swing, Sun Microsystems released JavaFX in December 2008. With the intention of being Swings successor and replacement, JavaFX was to become the standard GUI library for the Java Development Kit (JDK) (Wikpedia.org, 2020). Oracle (which acquired Sun Microsystems in April of 2009) states in the JavaFX FAQ, that Swing will remain part of Java SE "for the foreseeable future", although intended to be replaced (Oracle, 2020).

The main difference between JavaFX and its predecessors is the representation of the GUI data structures as a scene graph format. The data structure of a scene graph is often implemented for vector-based editing of graphics, as is the case for example in modern video games. This underlying framework allows a description of the graphical interfaces in the commonly used XML and CSS formats, which both enable a much higher variety and flexibility in the appearance of the final program's GUI (Wikpedia.org, 2020).

JavaFX targeted multiple features: one of the primary goals was to be accessible on multiple devices, also following the Java "write once, run anywhere" paradigm (Taman, 2015). This intention can also be derived from the following quote from 2009 by Param Singh, former senior director of Java marketing at Sun Microsystems:

> "Our vision of providing a programming model that spans across multiple screens is one of the core fundamental changes versus development in the past." (Lau, 2009)

Although JavaFX on itself does not support mobile devices, Sun Microsystems released JavaFX Mobile with JavaFX version 1.1 (see 2.2 Timeline – Past Releases and Versioning) in February 2009 (Wikpedia.org, 2020).

About a year after the acquisition by Oracle, the JavaFX framework was made part of the Oracle JDK. The Oracle JDK is Oracle's officially supported Java SE version (Oracle, 2020).

A JDK is a Software Development Kit (SDK) used by developers to create Java programs which can be executed by the Java Virtual Machine (JVM) and the Java Runtime Environment (JRE). There is often confusion when distinguishing the frameworks, although there is a clear difference: the JDK is a package of tools for *developing* software based on Java and the JRE is a package of tools for *running* the Java code (Tyson, 2020).

There are multiple JDKs from various contributors and vendors available for development with Java. Oracle offers its own JDK with new version releases every three years and licensed under the Oracle Binary Code License Agreement (Baeldung, 2019). As announced by Oracle in April 2019, this license is free to use for personal development but includes constraints when it comes to the commercial use. The commercial development should be supported by an additional commercial license through a Java SE subscription model (Java, 2019). OpenJDK is the open source version released under the GNU General Public License (GNU GPL) version 2 and will have new version releases every six months (Baeldung, 2019). While the Oracle JDK is completely developed by Oracle, the OpenJDK has various contributors to it including Oracle itself, OpenJDK, the Java Community and other companies like IBM, Apple, SAP and many more (Baeldung, 2019). As of 2020, the Oracle JDK and OpenJDK are the most common implementations used on the Java development market. Nevertheless, there are many other JDK implementations by various vendors, as for example Azul Zulu, Eclipse OpenJg, Amazon Corretto, etc. (Baeldung, 2019). The JVM Ecosystem Report 2020 study conducted by Snyk highlights which JDKs are most used among Java developers, as shown in figure 4 (Vermeer, 2020).

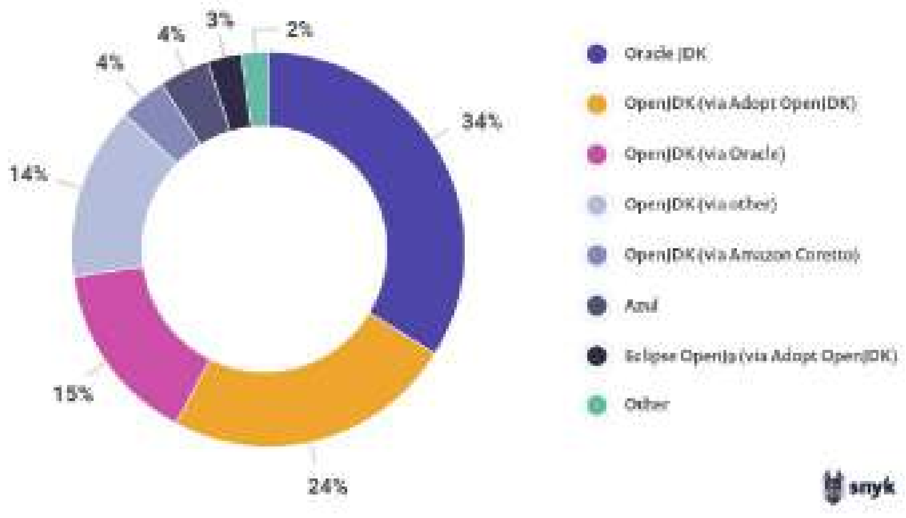


Figure 4 Snyk JVM Ecosystem Report 2020: JavaSDK Market

With the release of the JDK 11 in September 2018, Oracle announced that JavaFX would be separated from the JDK and downloadable as a standalone module (Smith, 2018).

As of April 2020, the latest version of JavaFX is version 14. In the following passage, a brief timeline of past JavaFX versions and their main features will be highlighted.

2.2. Timeline - Past Releases and Versioning

The following table lists the major JavaFX releases (minor features listed in the original release notes without significant changes have been excluded for purposes of simplified visualization). Significant implementations and feature changes have been highlighted in bold. (Wikpedia.org, 2020)

Date	JavaFX Version	Release Notes	Released by
Dec 2008	1.0	initial release	Sun Microsystems
Feb 2009	1.1 (Franca)	• **included JavaFX for mobile**	Sun Microsystems
Jun 2009	1.2 (Marina)	• **Beta support for Linux** and Solaris • Built-in controls and layouts • Skinnable CSS controls • Built-in chart widgets • JavaFX I/O management, masking differences between desktop and mobile devices • Speed improvements • Windows Mobile Runtime with Sun Java Wireless Client	Oracle Corporation

Apr 2010	1.3 (Soma)	• Performance improvements • Support of additional platforms • Improved support for user interface controls	Oracle Corporation
Aug 2010	1.3.1	• Quick startup time of JavaFX application • Custom progress bar for application startup	Oracle Corporation
Oct 2011	2.0 (Presidio) ; beta released in May 2011	• A new set of Java APIs opening JavaFX capabilities to all Java developers, without the need for them to learn a new scripting language. Java FX Script support was dropped permanently. • Support for high performance lazy binding, binding expressions, bound sequence expressions, and partial bind re-evaluation. • **Dropping support for JavaFX Mobile.** • **Oracle announcing its intent to open-source JavaFX.** • JavaFX runtime turning to be platform-specific, utilizing system capabilities, as video codec available on the system; instead of implementing only one cross-platform runtime as with JavaFX 1.x.	Oracle Corporation
Apr 2012	2.1	• **First official version for OS X (desktop only)** • H.264/MPEG-4 AVC and Advanced Audio Coding support • CoolType text • UI enhancements including combo box controls, charts (stacked chart), and menu bars • Webview component now allows JavaScript to make calls to Java methods	Oracle Corporation

Aug 2012	2.2	• **Linux support** (including plugin and webstart) • Canvas • New controls: Color Picker, Pagination • HTTP Live Streaming support • Touch events and gestures • Image manipulation API • Native Packaging	Oracle Corporation
Mar 2014	8 ; now with same numbering as the Java versioning	• **Support for 3D graphics** • Sensor support • MathML support, with JavaFX 8 Update 192 • Printing and rich text support • Generic dialog templates via inclusion of ControlsFX to replace JOptionPane as of JavaFX 8u40	Oracle Corporation
	9	• JEP 253: Prepare JavaFX UI Controls and CSS APIs for Modularization	Oracle Corporation
Sep 2018	11	• **First release that was decoupled from the JDK** • MathML support, with JavaFX 11 • FX Robot API	
Mar 2019	12	• Multiple bug fixes • Reintroduced JFR Pulse Logger • Support mouse forward/back buttons for scenegraph • Implement Accelerated composition for WebView etc.	
Sep 2019	13	• Multiple bug fixes • HTTPS for downloading all build dependencies • Supports static build for macosx • Add support for e-paper displays etc.	
Mar 2020	14	See chapter 2.3. Recent Release and Features	

Upcoming (as of May 2020)	15 ; early-access builds available		

As can be derived from the above table, with the release of version 11, Oracle decoupled JavaFX from the JDK making it a standalone module and clear the way for new contributors (Krill, 2018). As of May 2020, JavaFX version 15 is the upcoming new version of JavaFX.

2.3. Recent Release and Features

As can be derived from the table in chapter 2.2, the latest version is JavaFX version 14 released in March 2020. The JavaFX module still comes separated from the JDK and includes some of the following bug fixes, security improvements and feature enhancements (Vos, 2020):

Bug fixes:

- Duplicate symbols when building static libraries
- Bindings class gives a lot of unneeded 'select-binding' log messages
- Dialog's preferred size no longer accommodates multi-line strings
- Remove use of deprecated finalize method from JPEGImageLoader
- JavaFX: poor printing quality for Region nodes

Security fixes:

- Improve XSLT processing
- Improved internal validations
- Better formatting for numbers

Feature enhancements:

- TableSkinUtils should not contain actual code implementation
- Add tabSize property to Text and TextFlow
- Support HTTP/2 in WebView
- Add property to disable Monocle cursor
- Port Linux glass drag source (DND) to use gtk instead of gdk

Above listed are only a chosen amount of updates – for the complete listing, please see the official JavaFX 14 release notes.

3. JavaFX Concepts

The following section goes through the first-time installation steps of Java and JavaFX. For complexity reduction, this tutorial will be focusing on the installation on a Windows desktop operating system only. JavaFX will be demonstrated using the IntelliJ IDEA integrated development environment (IDE), since this is to date the most used IDE among developer community, as shown in the JVM Ecosystem Report 2020 study conducted by Snyk (Vermeer, 2020).

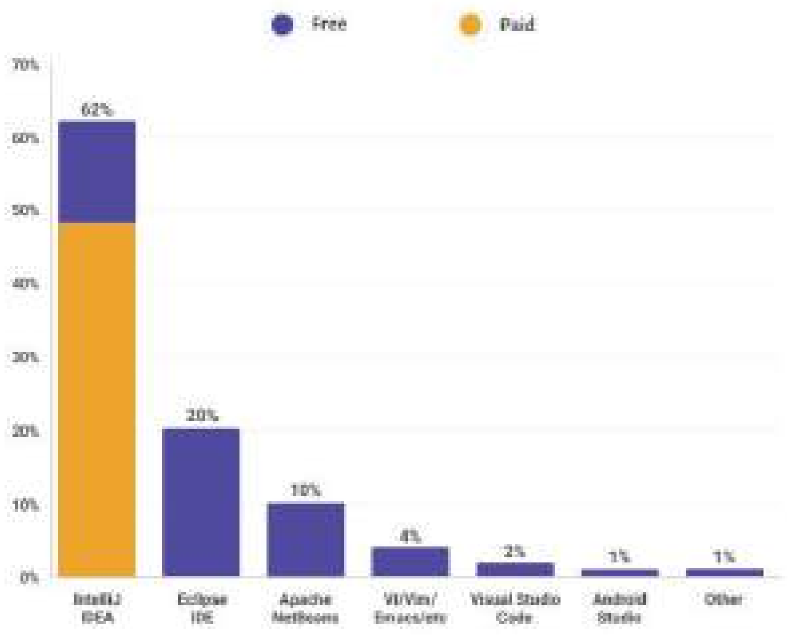


Figure 5 JVM Ecosystem Report 2020: Main IDE

The software and application versions demonstrated in this report include: OpenJDK version 14, JavaFX version 14 and the IntelliJ IDEA Community Edition version 2020.1 on a Windows 10 desktop computer.

3.1. Environment and Installation Prerequisites

JavaFX can only be used with the Java programming language. Thus, a JRE must be installed beforehand (Java.com, 2020). A JRE and its necessary components can be acquired through the installation of a JDK. As of April 2020, JavaFX 14 requires at least JDK version 11 to be installed (OpenJFX.io, 2020).

For the purpose of this paper, the open-source JDK "Open JDK" will be considered. The version used in this demonstration is the current most up to date OpenJDK version 14 and can be downloaded on the official website here http://jdk.java.net/14/ (accessed: 18 May 2020). Since JavaFX is distributed as a standalone module, it requires its additional separate installation. JavaFX 14 used in this demonstration can be downloaded via the Gluon webpage here https://gluonhq.com/products/javafx/ (accessed: 18 May 2020).

After downloading both components, the following folders are included and must be unzipped: *jdk-14.0.1* and *javafx-sdk-14*. These folders should be moved to any suitable practical desktop location, since the IDE will point to their folders and thus will depend on their location.

There are multiple environments in which JavaFX applications can be developed. An simple and user-friendly environment is through an IDE. As already described in the beginning of the section, this demonstration will be focusing on the development with JavaFX through the IntelliJ IDEA Community Edition version 2020.1 IDE. IntelliJ IDEA can be downloaded from the official homepage here https://www.jetbrains.com/idea/ (accessed: 18 May 2020).

After the successful installation of IntelliJ IDEA, it must be configured as described and shown in the following steps:

1. Open IntelliJ IDEA
2. Create a JavaFX project and provide a name for the project

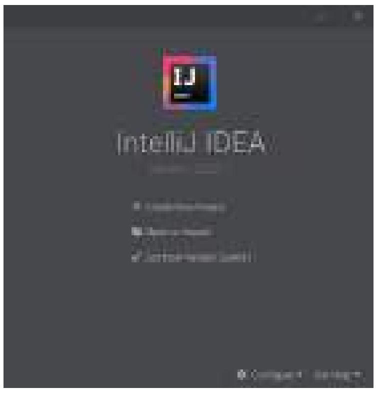
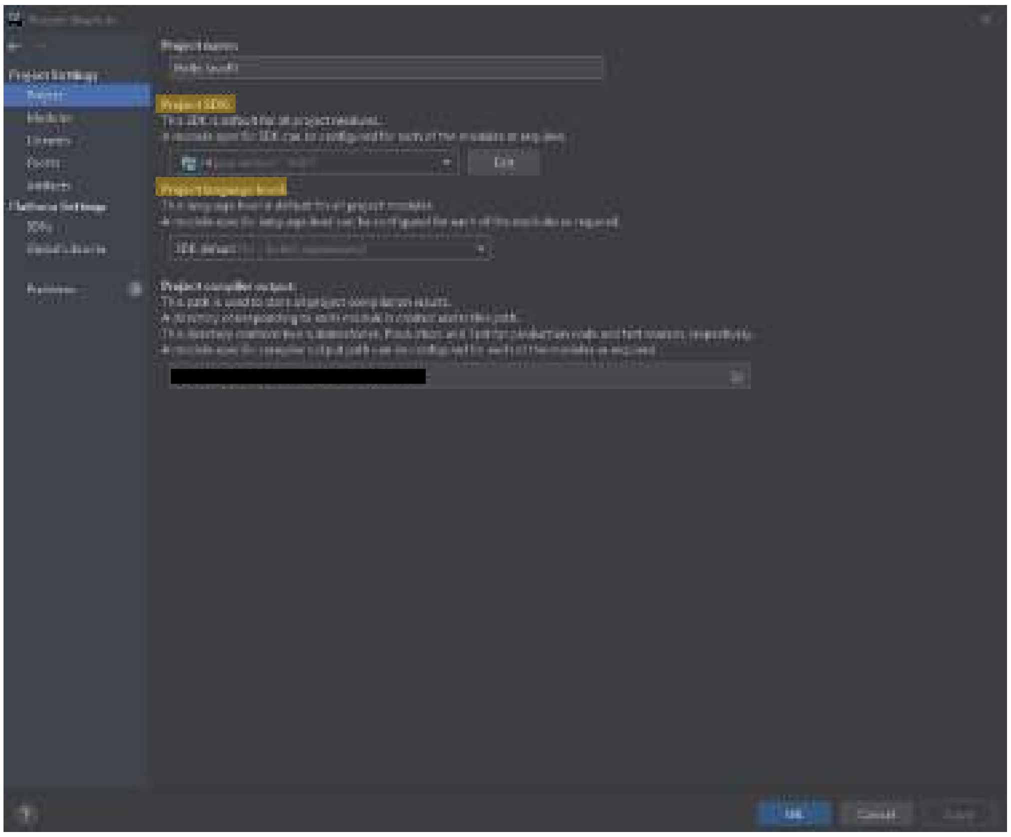


Figure 6 IntelliJ IDEA Home (JetBrains, 2020)

3. Set the project JDK: go to File>Project Structure>Project and under “Project SDK” click add SDK>JDK… and browse for the downloaded JDK folder *jdk-14.0.1* (or any preferred other JDK version that has been downloaded). This links the desired JDK to the current project. Set the “Project language level” to “SDK default”. Click “Apply” and close the dialog.



[redacted]

Figure 7 IntelliJ set project SDK (JetBrains, 2020)

4. Add a project library: go to File>Project Structure>Libraries and by clicking “**+”** browse for the …\lib folder of the JavaFX 14 SDK *javafx-sdk-14*. This links the JavaFX library to the project. Click “Apply” and close the dialog.

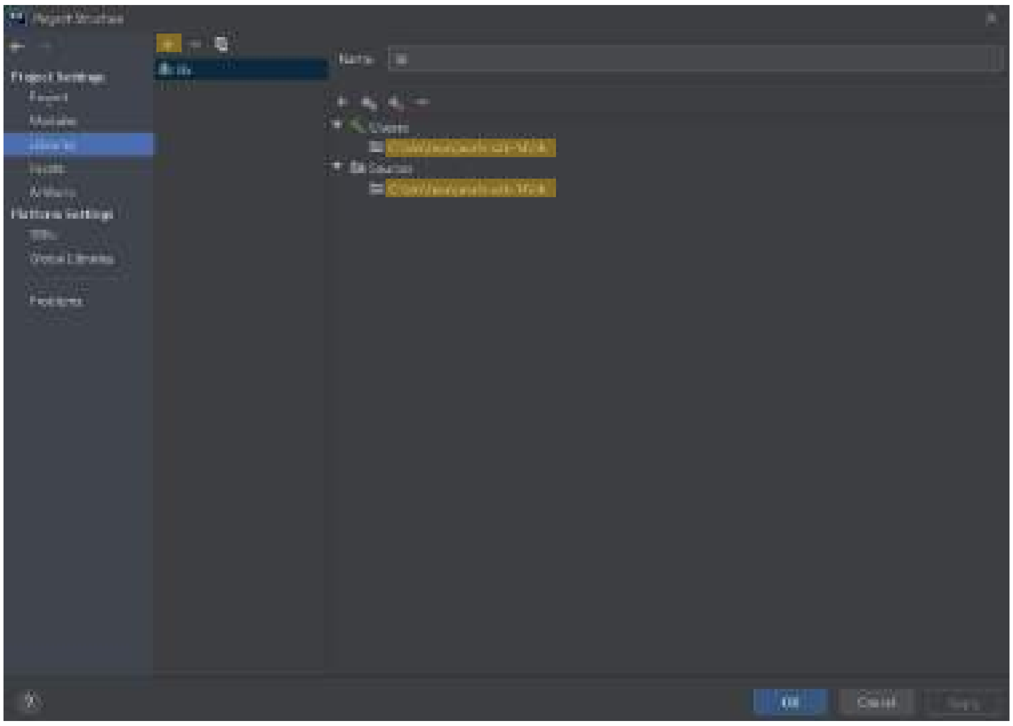


Figure 8 IntelliJ set library SDK (JetBrains, 2020)

5. Add VM options: go to Run>Edit Configurations and paste in these VM options:
   ```
   --module-path %PATH_TO_FX%
   --add-modules javafx.controls,javafx.fxml
   ```
   Replace %PATH_TO_FX% by the full location path of the …\lib folder of the JavaFX SDK *javafx-sdk-14*. Click “Apply” and close the dialog.

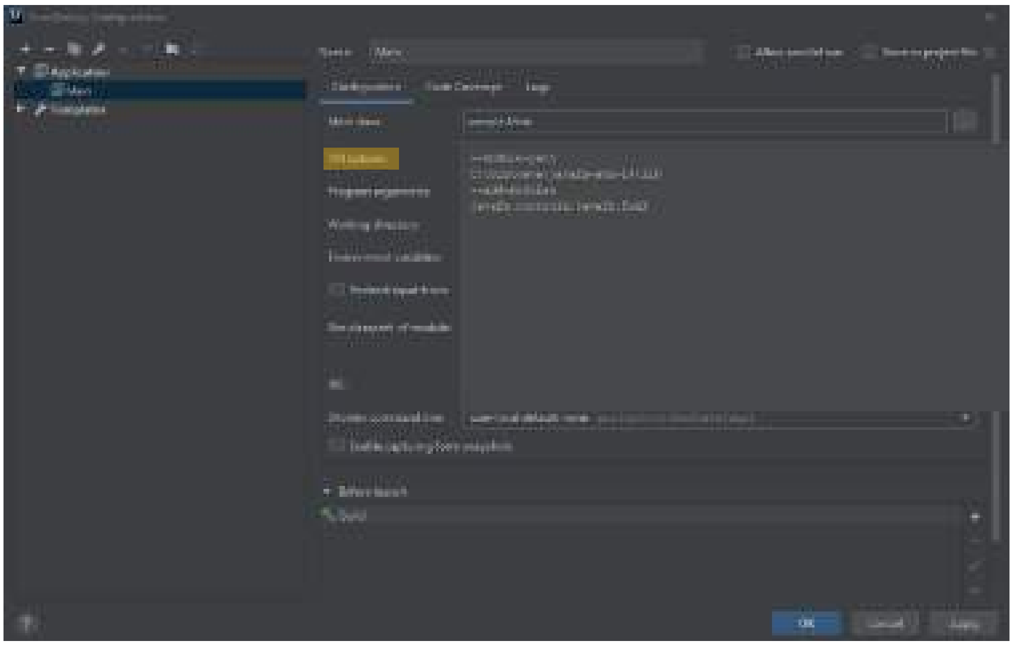


Figure 9 IntelliJ set VM options (JetBrains, 2020)

The project contains a sample JavaFX file which can be executed with Run>Run. Then the *sample.fxml Hello World* window as in figure 10 should appear.

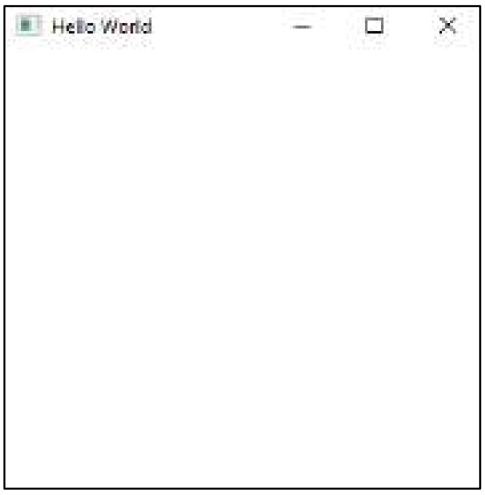


Figure 10 JavaFX sample.fxml window (JetBrains, 2020)

3.2. The JavaFX Architecture

One of the core features of the JavaFX underlying architecture is its focus on scene graph structures for the administration and logical representation of individual components in the GUI. A scene graph is a graphical visualization in a hierarchical tree structure where its nodes represent all visual elements of the application's user interface. Scene graphs are a data structure often used in vector-based graphics editing and modern computer games. For example, a game scene could contain a knight and a horse, whereas the knight could be considered as an extension to the horse. Thus, the scene graph would consist of a horse node with a knight node attached to it (Wikipedia, 2020).

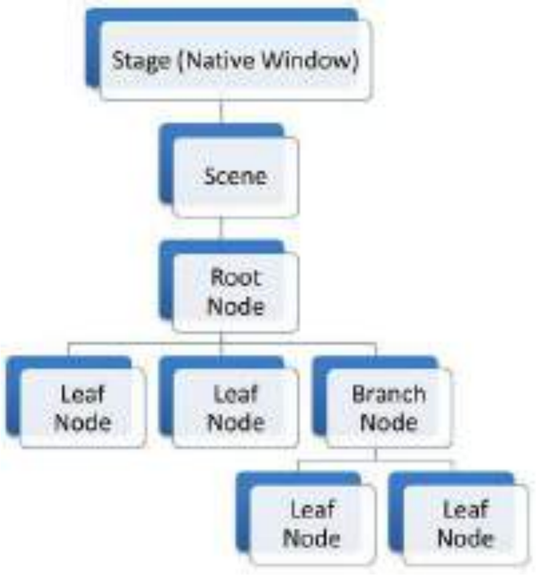


Figure 11 Scene Graph Representation (FXdocs.Github, 2019)

The above figure shows a simple example of such a tree structure. The stage is at the top of the architecture and is the JavaFX representation of the native OS window. A scene is attached to the stage node, which is the initial container for the JavaFX scene graph. The elements of the JavaFX scene graph are then represented as nodes (FXdocs.Github, 2019). Each node in the JavaFX scene graph has its own ID, style class and bounding volume. Except for the root node, every other node has a parent and zero or more child nodes (Oracle, 2014). Branch and leaf nodes are differentiated as such that leaf nodes don't have and child nodes.

A principle that applies to scene graphs is that the properties of a parent node are shared with the child node. The effect of this principle can be seen when applying a transformation or an event to a parent node, which will then automatically also be applied to its children (FXdocs.Github, 2019). Nodes are mostly 2- or 3D shapes, images, media, embedded web browsers, text, UI controls like f. e. buttons, charts, etc. Effects are objects which are able to change the appearance of scene graph nodes as f. e. shadows, blurs or color adjustments.

States transform the visual state of contents f. e. by changing the positioning and orientation of nodes or also by applying visual effects (Oracle, 2014).

3.3. Scene Builder

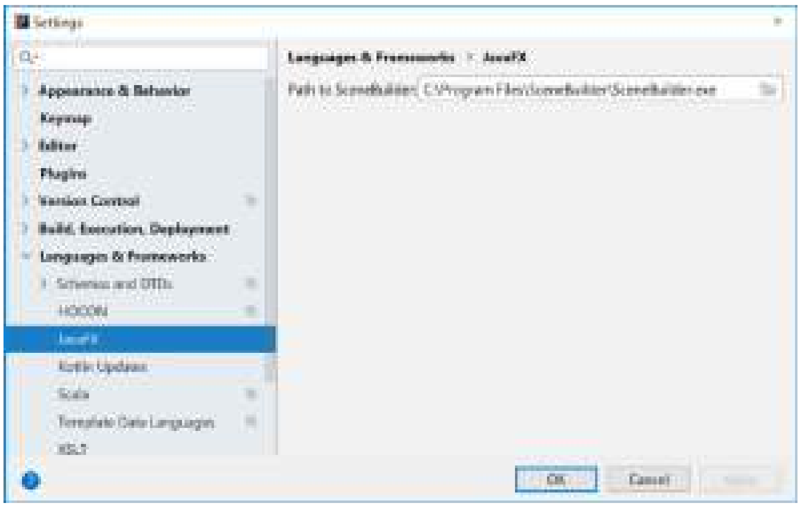
A powerful tool which results from the scene graph framework of JavaFX is Scene Builder. It was originally distributed by Oracle, which stopped its distribution with the release of a Java 8 update. Nevertheless, it has remained available under the OpenJFX project and is now distributed and updated by Gluon (Schulz, 2015).

Scene Builder allows the user to easily design a JavaFX GUI through a simple drag and drop interface. This allows a quick creation and design of user interfaces without the need of source code writing. The underlying scene graph structure is then automatically generated into an .fxml file while the GUI is built and modified. Additionally, the user can directly visualize the future GUI and its layout without the need of code compiling beforehand, which is a very time saving feature. The Scene Builder software displays the future GUI in its standard look and feel. This can be adapted simply by combining with a .css style defining file (Oracle, 2011).

Since IntelliJ IDEA also supports the configuration and implementation of Scene Builder, the next steps will demonstrate the prerequisites for its first usage. The same software, application and OS versions are applied as described in chapter 3.1. For the installation of the JDK, JavaFX and IntelliJ IDEA, please refer to chapter 3.1.

Scene Builder must first be installed. It is distributed by Gluon and can be downloaded here https://gluonhq.com/products/scene-builder/ (accessed: 18 May 2020). After the download and successful installation, the Scene Builder software must be linked to the IntelliJ IDE. This can be done through the following steps:

1. In IntelliJ IDEA go to File>Settings
2. Select “Languages and Frameworks” and select “JavaFX”
3. Browse for the installation location of the *SceneBuilder.exe* file and click “Ok”
4. Click “Apply” for saving the changes and close the dialog box



Path to SceneBuilder: C:\Program Files\SceneBuilder\SceneBuilder.exe

Figure 12 Changing the Scene Builder Location in IntelliJ IDEA (JavaMonkey, 2019)

After creating a new or opening an existing JavaFX project in IntelliJ IDEA, one can start using Scene Builder by right-clicking on any .fxml file and selecting "Open in Scene Builder".

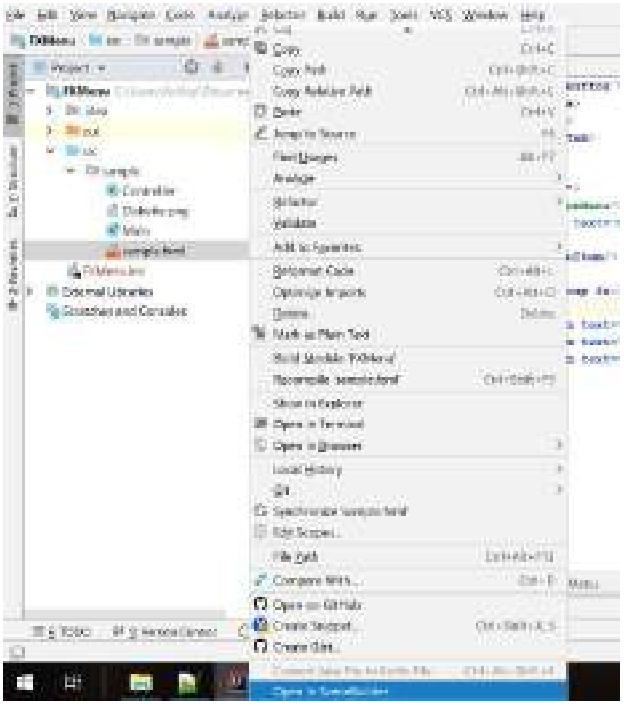


Figure 13 Open .fxml Filr in Scene Builder through IntelliJ IDEA (JavaMonkey, 2019)

After opening the .fxml file in Scene Builder, the controls are mostly self-explaining, due to their very user-friendly interface. The next steps will demonstrate how a button can be added and some text can be changed when the button is clicked.

On the bottom of the Scene Builder page, there are two tabs "Text" and "Scene Builder". The "Text" tab is where the .fxml code is displayed as text, and "Scene Builder" displays it in a user-friendly interface. Click on the "Scene Builder" tab to start the creation of the GUI. As shown in figure 14, the size of the window can be changed through the “AnchorPane” Layout tab on the right side.

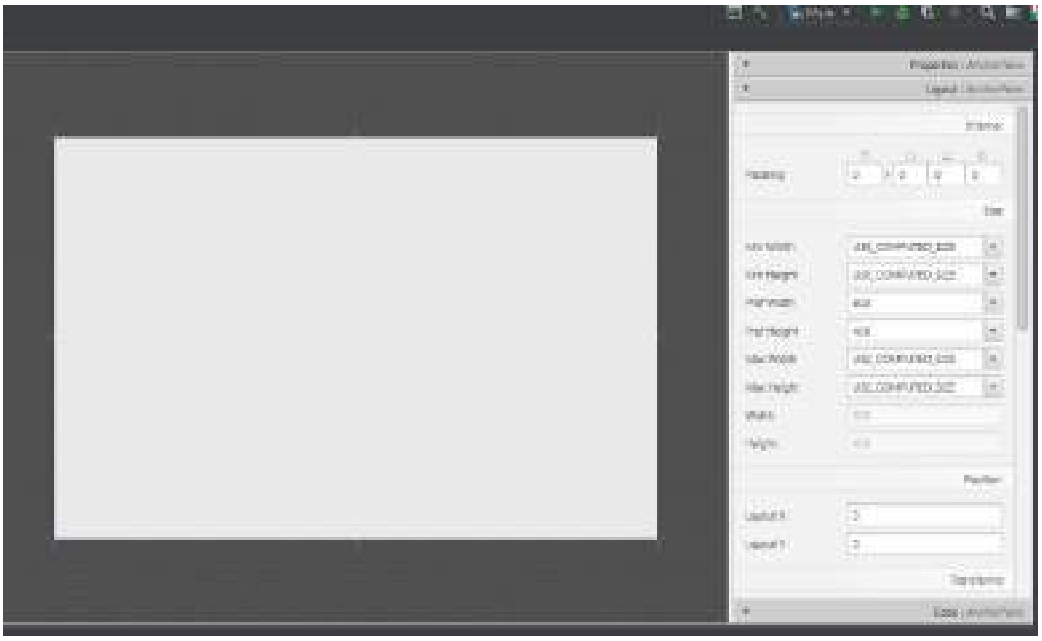


Figure 14 Scene Builder: Change Window Size (Shehanka, 2018)

As shown in figure 15, on the left pane one can select “Components” f. e. a button which can be placed within the window and adjusted as needed.

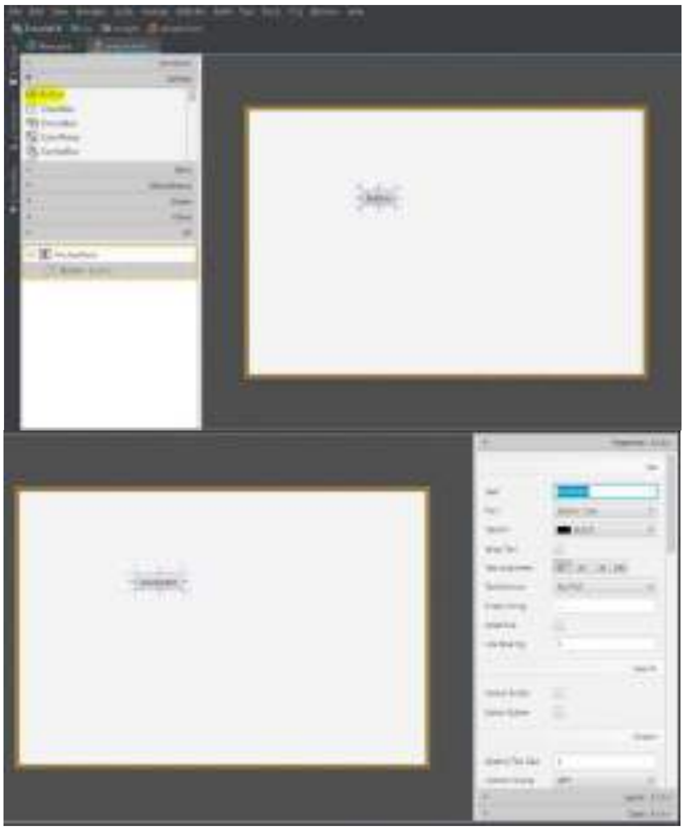


Figure 15 Scene Builder: Add and Adjust Buttons (Shehanka, 2018)

On the right pane, the text displayed on the button and its font, size, color and alignment can be changed. On the left-side, add a “Text” component, place it onto the window and type a “0” (or any other text).

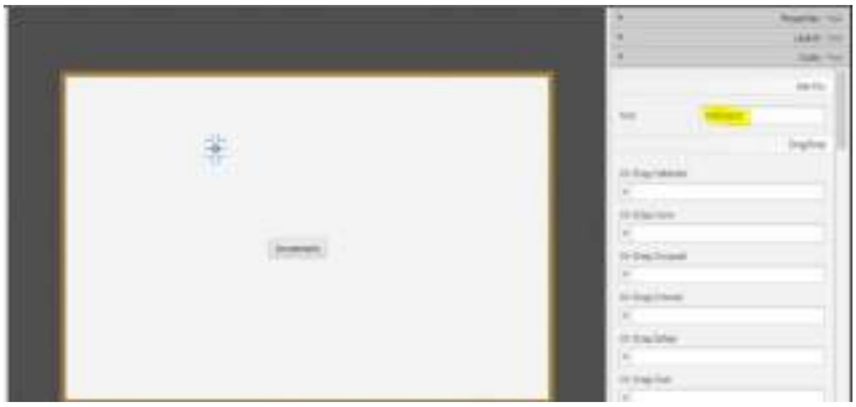
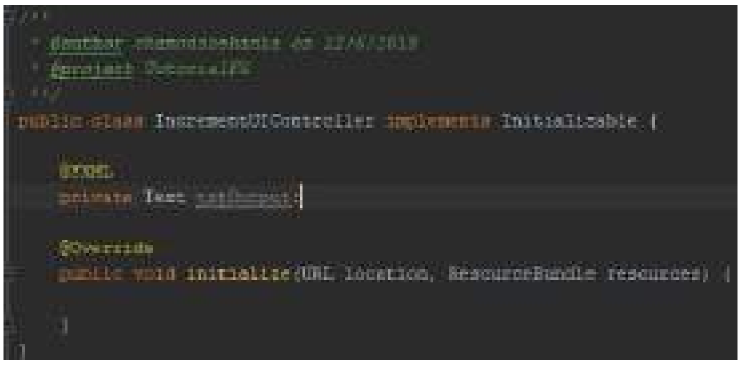


Figure 16 Scene Builder: Add and Adjust Text (Shehanka, 2018)

On the right-side, open the “Code: Text” panel and set the variable name (fx:id) to Text Component and then to “txtOutput”.

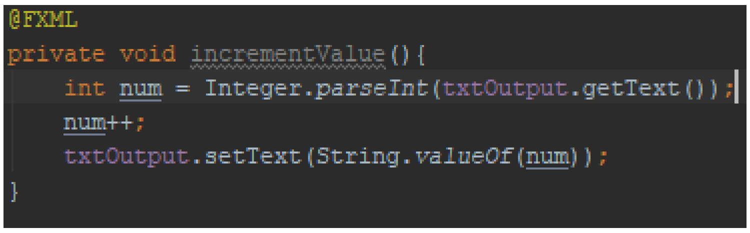
Now switch to the “Text” tab on the bottom. As shown in figure 16, under the public “IncrementalUIController” class, the text component has to be declared. This can be done through adding the following variable and annotating with @FXML and “private Text txtOutput;”.



```
/**
 * @author chamodshehanka on 12/8/2018
 * @project TutorialFX
 **/
public class IncrementUIController implements Initializable {

    @FXML
    private Text txtOutput;

    @Override
    public void initialize(URL location, ResourceBundle resources) {

    }
}
```

Figure 17 Scene Builder: Add a JavaFX Component (Shehanka, 2018)

Furthermore, a function must be written to define what happens when a user clicks on the button. This is done through the following lines of code, shown in figure 17. A function “incrementValue” is defined which converts the text “0” to an integer type value and adds +1 to it. The values will then be reconverted back to a string type and displayed instead of the previous text. It is important to always add @FXML to the coding, so Scene Builder can retrieve its components.



```
@FXML
private void incrementValue(){
    int num = Integer.parseInt(txtOutput.getText());
    num++;
    txtOutput.setText(String.valueOf(num));
}
```

Figure 18 Scene Builder: Add Onclick Event Function (Shehanka, 2018)

Switch back to the “Scene Builder” tab. The last step is to link the onclick event function to the button. Select the button and on the right pane, open the “Code: Button” tab. In the “On Action” drop down menu, choose “incrementValue”, since this is the name of the previously created function.

To run the .fxml file go to the “Main.java” and change the FXMLLoader value to IcrementUI.fxml. Then click on the run ▶ icon on the top in the toolbar tab. The previously created window should pop up and each time the button is clicked, it should increase its displayed value to +1.

4. Developing with JavaFX

4.1. For Desktops with Windows

Development with JavaFX on Windows can be done through several different ways and with different IDEs. Independent of the IDE, the prerequisite for developing with JavaFX is the installation of a JDK and the additional installation of JavaFX itself (see chapter 3.1. for the detailed step by step installation description).

Various IDEs support the development of JavaFX applications for which the following are listed on the OpenJFX webpage:

1. IntelliJ IDEA
2. NetBeans
3. Eclipse

For details on the installation of JavaFX with IntelliJ IDEA please refer to chapter 3.1. For installation steps of the other IDEs and alternatives, please refer to the official OpenJFX documentation https://openjfx.io/openjfx-docs (accessed: 18 May 2020).

4.2. For Desktops with Other OS (MacOS, Linux, etc.)

As of May 2020, the JDK version 14 is officially supported for Windows, MacOS and Linux. This also applies to JavaFX version 14. The same installation prerequisites apply as for Windows desktops and the steps listed in chapter 3.1. should also be followed by Mac or Linux users.

For more detailed installation steps of the required components, IDEs and other alternatives for MacOS and Linux, please refer to the official OpenJFX documentation https://openjfx.io/openjfx-docs (accessed: 18 May 2020).

4.3. For Mobile Devices

For development o of applications for mobile devices, Sun Microsystems first released JavaFX Mobile as part of JavaFX 1.1 in February 2009 (Wikpedia.org, 2020). With the release, Sun Microsystems provided a unified model for the development of RIAs for even more different platforms while using only one language and one set of APIs (Burnette, 2009). Mobile devices include mobile phones, smartphones and tablets running on various operating systems including Android, iOS and Windows. Shortly after the first release, Sun Microsystems partnered with different vendors including Sony Ericsson and LG Electronics, carriers like

Orange and Sprint, and with various independent software vendors (ISV). Sony Ericsson's former executive vice president and chief creation officer Rikko Sakaguchi stated:

> "We see JavaFX as a natural fit to our mobile software platform strategy to enable developers […] to create superior, innovative, expressive mobile applications and services. Sony Ericsson expects that JavaFX will have a great impact on the mobile content ecosystem and plan to bring JavaFX to a significant part of our product portfolio." (Taft, 2009)

Oracle dropped its official support for JavaFX Mobile with the release of JavaFX version 2.0 in October 2011 (Wikpedia.org, 2020). Since the separation of JavaFX from the JDK, starting with JavaFX 11, the JavaFX Mobile project has been to date continued by Gluon under the name Gluon Mobile. Gluon Mobile supports the development of UI for both Android and iOS devices. The written code can be rolled out on both operating systems without the need of recoding (Gluon, 2020). Gluon Mobile is offered as a plugin for various IDEs. In the following section, the installation and first steps for development of JavaFX on mobile devices with Gluon Mobile are explained briefly. The same software, application and OS versions are applied as described in chapter 3.1. For the installation of the JDK, JavaFX and IntelliJ IDEA, please refer to chapter 3.1. Gluon Mobile plugin version 2.7.0 is used in the following demonstration.

Gluon Mobile can be downloaded either as a free version or on a paid subscription basis (Gluon, 2020). For this demonstration, the steps are described using the free version. The Gluon Mobile plugin can be directly downloaded within IntelliJ IDEA, or separately downloaded via this link https://plugins.jetbrains.com/plugin/7864-gluon-plugin (accessed: 18 May 2020).

The steps listed below must be followed to download the plugin directly within IntelliJ IDEA IDE (Gluon, 2018):

1. In IntelliJ IDEA, click File>Settings
2. On the left, select "Plugins"
3. On the top, choose the "Marketplace" tab
4. Search for "Gluon Plugin" and click "Install"
5. You may need to restart IntelliJ IDEA

After successfully installing the Gluon Mobile plugin, a Gluon project can be created. The following steps describe how a sample Gluon project can be started:

1. In IntelliJ, click File>New>Project
2. On the left, select "Gluon"

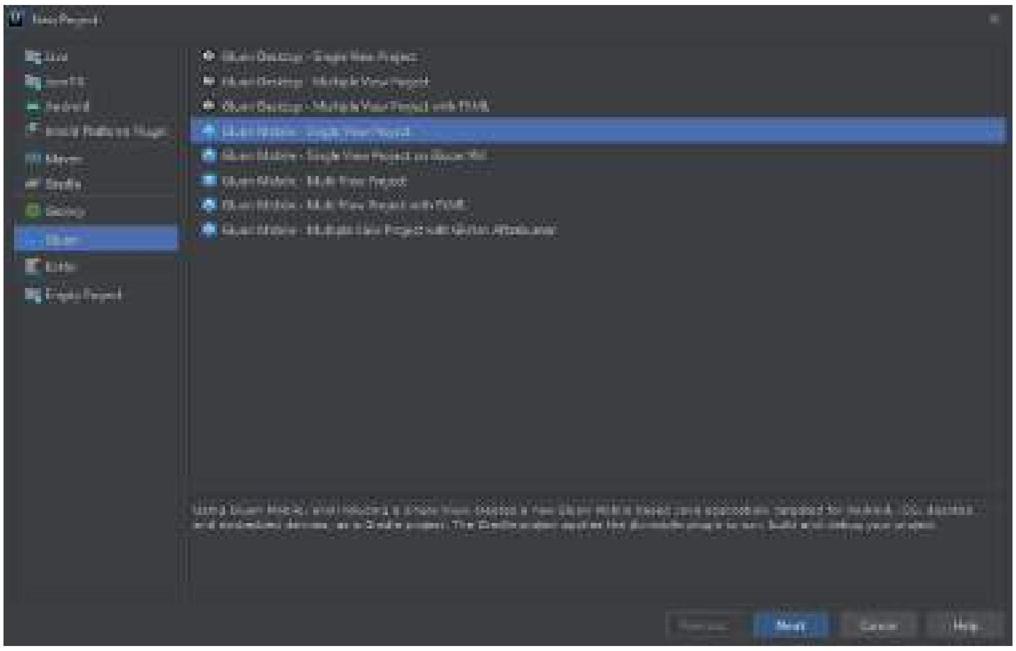

New Project
Java
JavaFX
Android
IntelliJ Platform Plugin
Maven
Gradle
Groovy
Gluon
Kotlin
Empty Project
Gluon Desktop - Single View Project
Gluon Desktop - Multiple View Project
Gluon Desktop - Multiple View Project with FXML
Gluon Mobile - Single View Project
Gluon Mobile - Single View Project on Gluon VM
Gluon Mobile - Multi View Project
Gluon Mobile - Multi View Project with FXML
Gluon Mobile - Multiple View Project with Glisten Afterburner
Previous
Next
Cancel
Help

Figure 19 IntelliJ: create new Gluon Mobile Project (JetBrains, 2020)

3. First-time users will be prompted to register with Gluon by entering their email address. A license key can also be entered in case the paid version of Gluon Mobile is preferred.
4. On the next window, the package name and main class name can be chosen. These will be entered by default by Gluon. It also lets the user select the platforms for which the app is intended to be developed for.

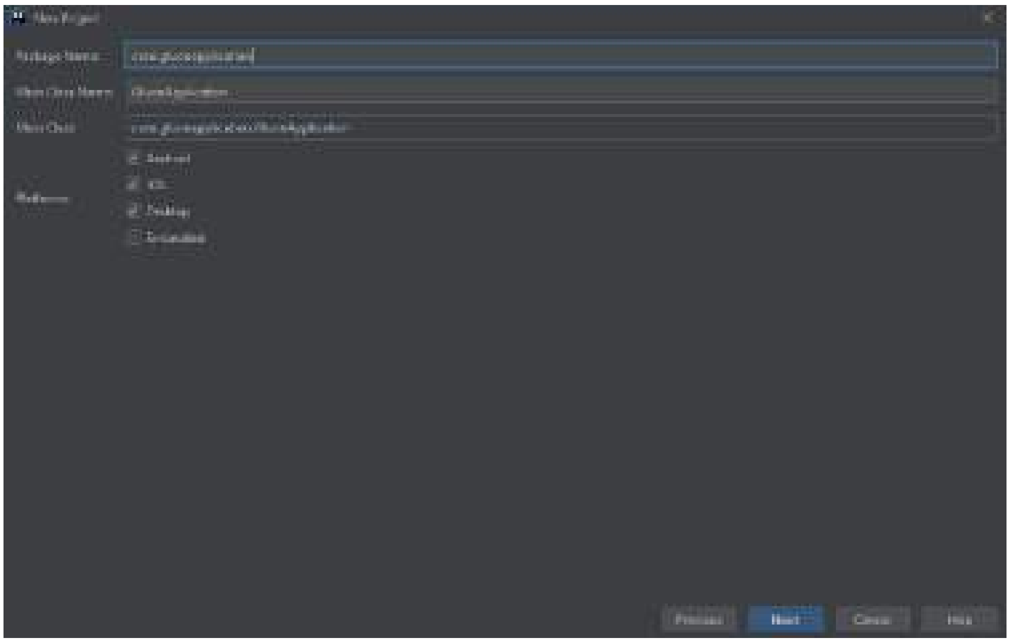
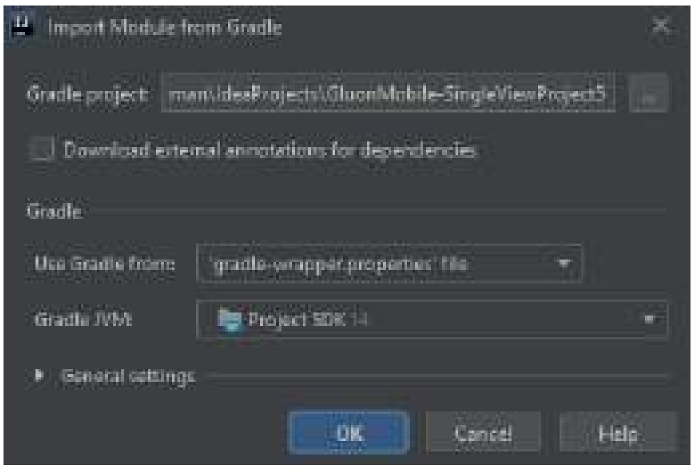


Figure 20 IntelliJ: chose Platforms for Gluon Mobile Project

5. Choose a valid installation of a JDK
6. Choose a name and a location for the project and click “Finish”.

IntelliJ IDEA will now prompt for importing a Gradle project. Gradle is a tool for automating builds and is most often implemented with languages such as Java, Groovy or Scala. Gradle's configuration allows automated running of compilers, tests or the creation of code documentation (MacMurray, 2018).


Import Module from Gradle

Gradle project: man\IdeaProjects\GluonMobile-SingleViewProject5

Download external annotations for dependencies

Gradle

Use Gradle from: 'gradle-wrapper.properties' file

Gradle JVM: Project SDK 14

Figure 21 IntelliJ: Gluon Mobile Project Import Gradle Module

Under "Use Gradle from" choose " 'gradle-wrapper.properties' file". Under "Gradle JVM" select the desired JDK. Click "Ok" and the new project is imported and opened afterwards. Now a JavaFX application can be easily developed for mobile devices.

4.4. For Raspberry Pi

Java applications can also be run on embedded mobile devices such as Raspberry Pi single-board computers. Raspbian OS natively includes Java and since version 2019-06-20 it includes OpenJDK version 11 (Raspbian, 2020). This allows Raspbian to run Java applications, but since JavaFX is not included in the JDK, it must be installed separately. One easy way to install JavaFX is through the Bellsoft LibericaJDK which already includes JavaFX and has a dedicated JDK release for Raspberry Pi devices. The supported devices listed by Bellsoft include Raspberry Pi Model 2 and 3 with ARM v7, ARMv8 or Intel Atom processors (Bellsoft, 2020).

The following steps demonstrate how the LibericaJDK can be installed on a Raspberry Pi (Webtechie, 2020). As of May 2020, the most actual LibericaJDK version is 13.0.2+9 and will be installed in the following steps.

1. On the Raspberry Pi, download LibericaJDK with the following commands:
   ```
   $ cd /home/pi
   $ wget https://download.bell-sw.com/java/13.0.2+9/bellsoft-jdk13.0.2+9-linux-arm32-vfp-hflt.deb
   ```

```
$ sudo apt-get install ./bellsoft-jdk13.0.2+9-linux-arm32-vfp-hflt.deb
$ sudo update-alternatives --config javac
$ sudo update-alternatives --config java
```

2. Check if the installation was successful by running the following command:

   ```
   $ java –version
   ```

   If the installation was completed correctly, the output should look as following:

   ```
   openjdk version "13-BellSoft" 2019-09-17
   OpenJDK Runtime Environment (build 13-BellSoft+33)
   OpenJDK Server VM (build 13-BellSoft+33, mixed mode)
   ```

For the installation of any alternative LibericaJDK version, please replace the links from the commands in step 1 marked in bold with any official download link from the Bellsoft website. After successful installation, the Raspberry Pi now supports running JavaFX applications.

5. Getting Started with Development

The following chapters will focus on hands-on development with JavaFX. The first part will demonstrate and go through the steps for the implementation of a classic “Hello Word” application. The second chapter will provide a collection of code snippets which are intended to be used as references when developing with JavaFX. These demonstrations, the same software, application and OS versions are applied as described in chapter 3.1. For the installation of the JDK, JavaFX and IntelliJ IDEA, please refer to chapter 3.1.

5.1. First “Hello World” JavaFX Program

This demonstration starts after a JavaFX project has been successfully created in IntelliJ IDEA. Please refer to chapter 3.1. for details on how such a project must be set up.

After the initialization of a new JavaFX project and configuration of the environment settings, the project should contain three sample files:

- *Main.java*
- *Controller.java*
- *sample.fxml*

The *sample.fmxl* file already contains the coding for a very basic JavaFX window which will be adapted to a more sophisticated version in the next steps (JetBrains, 2020):

1. Changing the default controller name: in the *Controller.java* file, highlight the `Controller` class name and hold `Shift+F6`. This opens a new Refractor Rename window, which lets the user rename a variable and will then automatically rename it on each project file it appears in. Rename the `Controller` to `HelloWorldController` and then click “Refractor”.

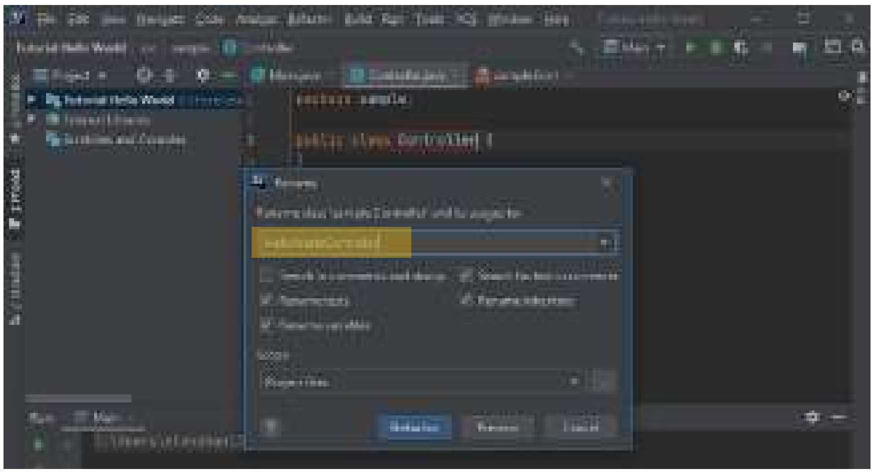
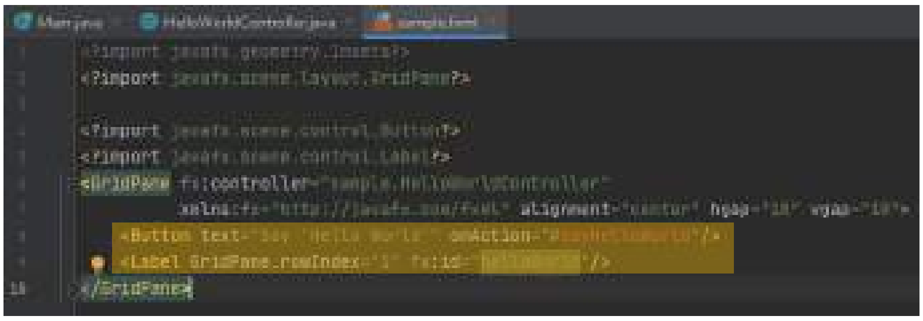


Figure 22 Refractor Controller Name (JetBrains, 2020)

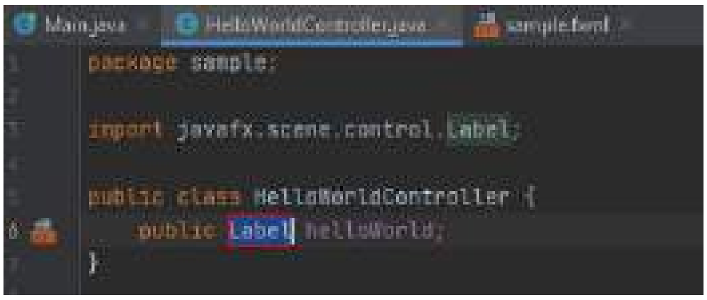
2. Add a button: if the button is clicked, a “Hello Word!” message should appear in the application. In the *sample.fxml* file add the following code within the `<GridPane>` tag:
   ```
   <Button text="Say 'Hello World'" onAction="#sayHelloWorld"/>
   <Label GridPane.rowIndex="1" fx:id="helloWorld"/>
   ```
   In the first line, a button is defined which holds the text “Say Hello Word” and which performs the action “sayHelloWorld” when it is clicked. The “sayHelloWorld” action will be defined in the next step.



```
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<GridPane fx:controller="sample.HelloWorldController"
          xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
    <Button text="Say 'Hello World'" onAction="#sayHelloWorld"/>
    <Label GridPane.rowIndex="1" fx:id="helloWorld"/>
</GridPane>
```

Figure 23 Add "Say Hello Word" Button (JetBrains, 2020)

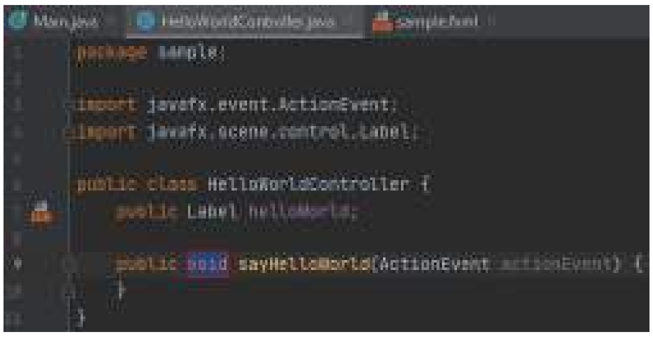
3. Create the message field: in *sample.fxml* highlight `helloWorld` after fx:id in the `label` tag and hold `Alt+Enter`. Select “Create Field ‘helloWorld’”. Now IntelliJ IDEA will switch to the *HelloWorldController.java* file where a new import statement `import javafx.scene.control.Label;` will have been generated. Additionally, a `helloWorld` label field will have been declared. Press `Enter` to apply and exit the refactoring mode.



```
package sample;

import javafx.scene.control.Label;

public class HelloWorldController {
    public Label helloWorld;
}
```

Figure 24 Create 'HelloWorld' Field (JetBrains, 2020)

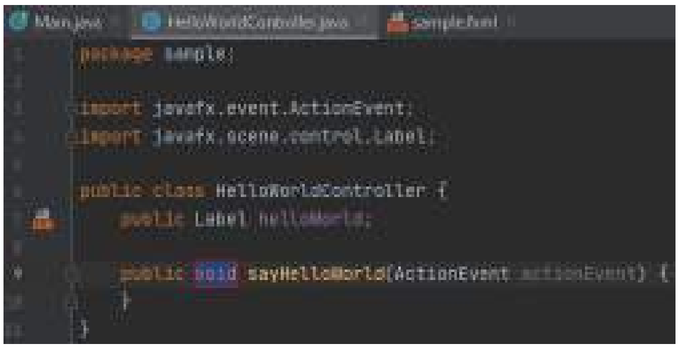
4. Create “sayHelloWorld” method: in sample.fxml highlight `sayHelloWorld` after `onAction` in the `button` tag. Hold `Alt+Enter` and select “Create Method 'void sayHelloWorld(ActionEvent)'”. IntelliJ IDEA now automatically adds the declared method in the *HelloWordCorntroller.java* file.



```
package sample;

import javafx.event.ActionEvent;
import javafx.scene.control.Label;

public class HelloWorldController {
    public Label helloWorld;

    public void sayHelloWorld(ActionEvent actionEvent) {
    }
}
```

Figure 25 Create „sayHelloWorld“ Action Event Method (JetBrains, 2020)

5. Set text for the label: in the HelloWorldController.java file, paste the following code line after the sayHelloWorld method:
   `helloWorld.setText("Hello World!");`



```
package sample;

import javafx.event.ActionEvent;
import javafx.scene.control.Label;

public class HelloWorldController {
    public Label helloWorld;

    public void sayHelloWorld(ActionEvent actionEvent) {
    }
}
```

Figure 26 Set Text for Hello World Label (JetBrains, 2020)

The basic "Hello World" application is now ready to be executed: click Run>Run and the window as shown in figure 27 should appear. When the button is clicked, the "Hello Word!" text should appear below it.

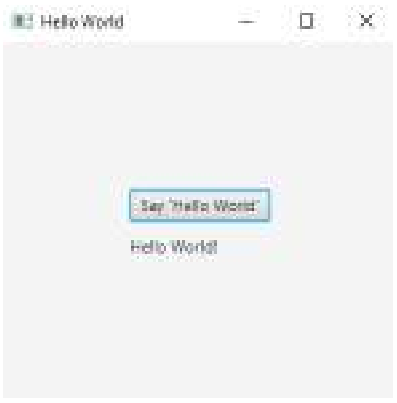
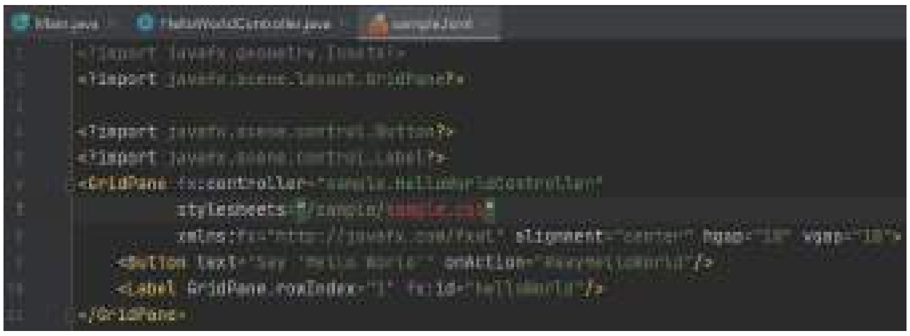


Figure 27 Basic Hello World Window with Button (JetBrains, 2020)

The next steps will demonstrate how a CSS styling can be applied to the interface of the window.

1. Reference to CSS-Stylesheet: in the *sample.fxml* file add the following code after `fx:controller` in the `GridPane` tag:
   `stylesheets="/sample/sample.css"`



```
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<GridPane fx:controller="sample.HelloWorldController"
          stylesheets="/sample/sample.css"
          xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
    <Button text="Say 'Hello World'" onAction="#sayHelloWorld"/>
    <Label GridPane.rowIndex="1" fx:id="helloWorld"/>
</GridPane>
```

Figure 28 Add CSS-Stylesheet Reference (JetBrains, 2020)

2. In *sample.fxml*, highlight `sample.css` and hold `Alt+Enter`. Select "Create File sample.css" and a new .css file will be opened in a new tab.
3. In the sample.css file add the following lines of code:

```
.root {
-fx-background-color: bisque;
}
.label {
-fx-font-size: 20;
}
```

The final "Hello World" application is now ready to be executed: click Run>Run and the window as shown in figure 29 should appear. The background color is changed depending on the defined background color in the sample.css file. Additionally, the font size has been set to 20.

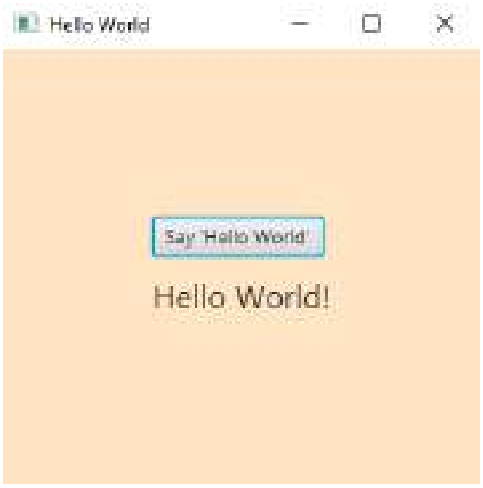


Figure 29 Hello World Window with Button and CSS Style (JetBrains, 2020)

5.2. Various Development Examples

The following section will highlight some code-snippets for the implementation of a basic user-interface as shown in figure 30.

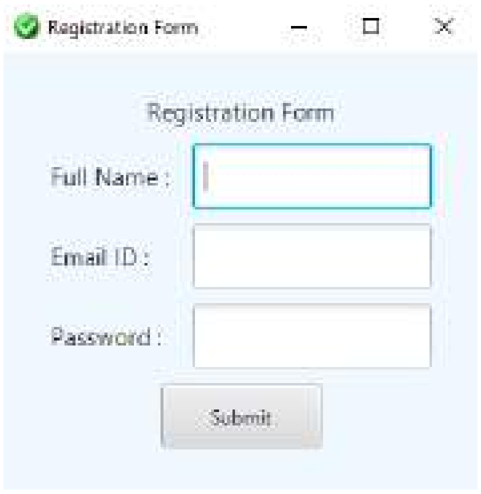


Figure 30 Example of Basic JavaFX Registration Form (JetBrains, 2020)

5.2.1. General-Purpose Text Fields

A general-purpose text field can be used for collecting any type of user input. In figure 30 this is represented by the “Full Name” and the “Email ID” text field. The following code snippet can be applied in an *.fxml* file for the creation of such a text field:

```
<Label text="Full Name : "
      GridPane.columnIndex="0" GridPane.rowIndex="1" >
</Label>
<TextField prefHeight="40"
      GridPane.columnIndex="1" GridPane.rowIndex="1"/>
```

The `Label` tag is used for defining the text field and the displayed text can be adjusted with the `text` parameter. The `GridPane.columnIndex` and `.rowIndex` parameters defines the location of the displayed text.

The `TextField` tag is used for adding the user input field. `prefHeight` defines its height and `GridPane.columnIndex` and `.rowIndex` its location.

5.2.2. Password Text Fields

A password text field has a unique characteristic which hides the input characters when typed in.

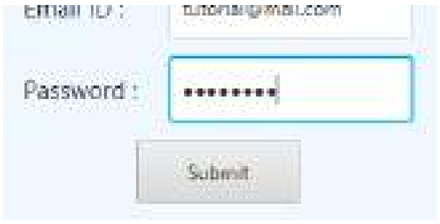


Figure 31 JavaFX Password Text Field (JetBrains, 2020)

This is one of the aspects which make it a suitable field for password input, since it provides security for the user entering it. The following code snippet can be used in an *.fxml* file for the creation of a password text field.

```
<Label text="Password : "
      GridPane.columnIndex="0" GridPane.rowIndex="3" >
</Label>
<PasswordField prefHeight="40"
      GridPane.columnIndex="1" GridPane.rowIndex="3"/>
```

The password text fields differ from general-purpose text fields in such that the PasswordField tag must be used.

5.2.3. Buttons

A button in JavaFX can be added and its appearance adjusted in an *.fxml* file with the following lines of code:

```
<Button text="Submit"
      prefWidth="100" prefHeight="40"
      GridPane.columnIndex="0" GridPane.rowIndex="4"
      GridPane.columnSpan="2" GridPane.rowSpan="1"
      GridPane.halignment="CENTER"
      onAction="#ButtonClickEvent">
</Button>
```

The `Button` tag is used for defining the button and the text displayed on the button can be adjusted with the `text` parameter. `prefWidth` and `prefHeight` define the size of the button. `GridPane.columnSpan` and `.rowSpan` define the location of the button and `.halignment` its alignment. The `onAction` parameter can be implemented for defining what event should be triggered as soon as the button is clicked.

5.2.4. Application Window Icon

The icon of the application window can be adjusted in the *main.java* file with the following line of code. This will change the icon at the top of the window and the icon for the app displayed in the task bar.

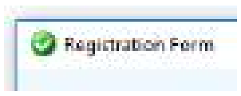


Figure 32 Custom Application Window Icon in JavaFX (JetBrains, 2020)

```
Stage.getIcons().add(new Image(getClass().getResourceAsStream("icon.png")));
```

The file icon.png must be placed in the same directory as the main.java file. Alternatively, the file location address can be inserted instead.

Summary and Conclusion

As has been highlighted in the last chapters, JavaFX is a framework based on the Java programming language which lets users develop rich internet applications and sophisticated graphical user-interfaces. JavaFX applications can be easily and fast implemented on all major popular platforms, operating systems and device available on the market, including desktops with Windows, MacOS and Linux, mobile devices running on Android, IOS or Windows and even embedded devices such as Raspberry Pi. (Wikpedia.org, 2020). It requires little to no programming experience thanks to the Scene Builder integration, which enables users to create graphical user-interfaces through a simple and user-friendly visual drag-and-drop tool (Gluon, 2020). Once the JavaFX code is written, it can easily be deployed on various platforms, without the need of code adaptation to individual platform-dependent characteristics. To conclude, JavaFX is a great starting point for users that want to get familiar in app-interface development with Java.

Bibliography

Baeldung, 2019. *Differences Between Oracle JDK and OpenJDK.* [Online]
Available at: https://www.baeldung.com/oracle-jdk-vs-openjdk
[Accessed 10 May 2020].

Bellsoft, 2020. *Liberica JDK version 13.0.2 for Embedded.* [Online]
Available at: https://bell-sw.com/pages/java-13.0.2-for-Embedded/
[Accessed 18 May 2020].

Burnette, E., 2009. *Java gets a mobile makeover with JavaFX Mobile.* [Online]
Available at: https://www.zdnet.com/article/java-gets-a-mobile-makeover-with-javafx-mobile/
[Accessed 10 May 2020].

Educba, 2020. *Java Swing vs JavaFX.* [Online]
Available at: https://www.educba.com/java-swing-vs-java-fx/
[Accessed 18 May 2020].

FXdocs.Github, 2019. *FXdocs Github.* [Online]
Available at: https://fxdocs.github.io/docs/index.html
[Accessed 29 April 2020].

Gluon, 2018. *Gluon Mobile Documentation.* [Online]
Available at: https://docs.gluonhq.com/charm/5.0.1/#intellij-plugin
[Accessed 11 May 2020].

Gluon, 2020. *Buy Gluon Mobile.* [Online]
Available at: https://gluonhq.com/products/mobile/buy/
[Accessed 11 May 2020].

Gluon, 2020. *Gluon Mobile.* [Online]
Available at: https://gluonhq.com/products/mobile/
[Accessed 11 May 2020].

Java.com, 2020. *General Information on JavaFX.* [Online]
Available at: https://www.java.com/en/download/faq/javafx.xml
[Accessed 15 April 2020].

Java, 2019. *Important Oracle Java License Update.* [Online]
Available at: https://java.com/en/download/release_notice.jsp
[Accessed 10 May 2020].

JavaMonkey, 2019. *Noble Code Monkeys.* [Online]
Available at: https://noblecodemonkeys.com/javafx-scenebuilder-with-intellij-and-netbeans/
[Accessed 29 April 2020].

JavaTpoint.com, 2020. *Java AWT Tutorial.* [Online]
Available at: https://www.javatpoint.com/java-awt
[Accessed 30 March 2020].

JavaTpoint, 2020. *Java Swing Tutorial.* [Online]
Available at: https://www.javatpoint.com/java-swing
[Accessed 2020 May 2020].

JetBrains, 2020. *Develop a basic JavaFX application.* [Online]
Available at: https://www.jetbrains.com/help/idea/developing-a-javafx-application-

examples.html
[Accessed 18 May 2020].

JetBrains, 2020. *IntelliJ IDEA Version 2020.1,* s.l.: JetBrains s.r.o..

Krill, P., 2018. *JavaFX will be removed from the Java JDK.* [Online]
Available at: https://www.infoworld.com/article/3261066/javafx-will-be-removed-from-the-java-jdk.html
[Accessed 11 May 2020].

Lau, K., 2009. *Javaworld.com.* [Online]
Available at: https://www.javaworld.com/article/2077998/javafx-mobile-released.html
[Accessed 4 April 2020].

MacMurray, A., 2018. *A beginners guide to Gradle.* [Online]
Available at: https://medium.com/@andrewMacmurray/a-beginners-guide-to-gradle-26212ddcafa8
[Accessed 11 May 2020].

OpenJFX.io, 2020. *Getting Started with JavaFX.* [Online]
Available at: https://openjfx.io/openjfx-docs/
[Accessed 15 April 2020].

Oracle, 2011. *JavaFX Scene Builder.* [Online]
Available at: https://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-info-2157684.html
[Accessed 11 May 2020].

Oracle, 2014. *Docs.Oracle.* [Online]
Available at: https://docs.oracle.com/javase/8/javafx/JFXST.pdf
[Accessed 29 April 2020].

Oracle, 2020. *Java SE Downloads.* [Online]
Available at: https://www.oracle.com/java/technologies/javase-downloads.html
[Accessed 10 May 2020].

Oracle, 2020. *JavaFX FAQ.* [Online]
Available at: https://www.oracle.com/technetwork/java/javafx/overview/faq-1446554.html#6
[Accessed 4 April 2020].

Raspbian, 2020. *Release Notes.* [Online]
Available at: http://downloads.raspberrypi.org/raspbian/release_notes.txt
[Accessed 18 May 2020].

Schulz, B., 2015. *Bye Bye JavaFX Scene Builder, Welcome Gluon Scene Builder 8.0.0.* [Online]
Available at: https://dzone.com/articles/bye-bye-javafx-scene-builder
[Accessed 11 May 2020].

Shehanka, C., 2018. *JavaFX step by step Part 2— UI design with Scene Builder.* [Online]
Available at: https://blog.usejournal.com/javafx-step-by-step-part-2-ui-design-with-scene-builder-4dc8473b3c2c
[Accessed 29 April 2020].

Smith, D., 2018. *Oracle Blog.* [Online]
Available at: https://blogs.oracle.com/java-platform-group/the-future-of-javafx-and-other-java-

client-roadmap-updates
[Accessed 4 April 2020].

Taft, D. K., 2009. *Sun Launches JavaFX Mobile.* [Online]
Available at: https://www.eweek.com/development/sun-launches-javafx-mobile
[Accessed 10 May 2020].

Taman, M., 2015. *JavaFX Essentials.* Birmingham, UK.: Packt Publishing Ltd.

Tyson, M., 2020. *Javaworld.com.* [Online]
Available at: https://www.javaworld.com/article/3296360/what-is-the-jdk-introduction-to-the-java-development-kit.html
[Accessed 4 April 2020].

Vermeer, B., 2020. *JVM Ecosystem Report 2020.* [Online]
Available at: https://snyk.io/wp-content/uploads/jvm_2020.pdf
[Accessed 11 May 2020].

Vos, J., 2020. *Release Notes for JavaFX 14 ; Github.com.* [Online]
Available at: https://github.com/openjdk/jfx/blob/jfx14/doc-files/release-notes-14.md
[Accessed 4 April 2020].

Webtechie, 2020. *Installing Java and JavaFX on the Raspberry Pi.* [Online]
Available at: https://webtechie.be/post/2020-04-08-installing-java-and-javafx-on-raspberry-pi/
[Accessed 18 May 2020].

Wikipedia.org, 2020. *Abstract Window Toolkit.* [Online]
Available at:
https://en.wikipedia.org/w/index.php?title=Abstract_Window_Toolkit&oldid=931019423
[Accessed 30 March 2020].

Wikipedia.org, 2020. *Swing (Java).* [Online]
Available at: https://de.wikipedia.org/w/index.php?title=Swing_(Java)&oldid=196092289
[Accessed 30 March 2020].

Wikipedia, 2020. *Java (programming language).* [Online]
Available at:
https://en.wikipedia.org/w/index.php?title=Java_(programming_language)&oldid=956349384
[Accessed 18 May 2020].

Wikipedia, 2020. *Raspbian.* [Online]
Available at: https://en.wikipedia.org/w/index.php?title=Raspbian&oldid=951519874
[Accessed 18 May 2020].

Wikipedia, 2020. *Scene graph.* [Online]
Available at: https://en.wikipedia.org/w/index.php?title=Scene_graph&oldid=953884868
[Accessed 18 May 2020].

Wikpedia.org, 2020. *JavaFX.* [Online]
Available at: https://en.wikipedia.org/w/index.php?title=JavaFX&oldid=947700994
[Accessed 30 March 2020].

Yap, C., 2003. *Java Swing Tutorial. New York University.* [Online]
Available at: https://cs.nyu.edu/~yap/classes/visual/03s/lect/l7/
[Accessed 30 March 2020].

List of Figures