JavaFX Basics

Dr. Salma A. Mahmood

- Course Syllabus :
  - JavaFX Basics
  - Event-Driven Programming and Animations.
  - JavaFX Controls and Multimedia.
  - Advanced GUI Programming.

Introduction

- *JavaFX is an excellent tool for learning object-oriented programming, it is a new framework for developing Java GUI programs.*
- *JavaFX provides a powerful, streamlined, flexible framework that simplifies the creation of modern, visually exciting GUIs.*
- *JavaFX as a platform for developing rich Internet applications replace both AWT and Swing tools.*
- When Java was introduced, the GUI classes were bundled in a library known as the ***Abstract Windows Toolkit (AWT)***.

Tool	Advantage	Disadvantage
AWT ***Abstract Windows Toolkit*** : it is a library contains bundled GUI classes.	➢ is fine for developing simple graphical user interface.	➢ AWT not suitable for developing comprehensive GUI projects. ➢ AWT is prone to platform-specific bugs.
The AWT user-interface components replaced by Swing components		
Swing components : a more robust, versatile, and flexible GUI library.	➢ Swing components are painted directly on canvases using Java code. ➢ Swing components depend less on the target platform. ➢ Swing use less of the native GUI resources. ➢ Swing is designed for developing desktop GUI applications.	➢
Swing is now replaced by a completely new GUI platform known as JavaFX.		
JavaFX incorporates modern GUI technologies to enable you to develop rich Internet applications	• **A Rich Internet Application (RIA)** is a Web application designed to deliver the same features and functions normally associated with desktop applications. • A JavaFX application can run seamlessly on a desktop and from a Web browser.	➢

	• JavaFX provides a multi-touch support for touch-enabled devices such as tablets and smart phones. • JavaFX has a built-in 2D, 3D, animation support, video and audio playback, and runs as a stand-alone application or from a browser. • JavaFX offers a more streamlined, easier-to-use, updated approach. • JavaFX also greatly simplifies the rendering of objects because it handles repainting automatically.	

Program Structure

```
package javafx_program_name;
```

package name is same as project name you give to your application and same of main class name , expect the class name begin with capital letter.

```
import javafx.application.*;
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
```

GUI Package must be opened before use their classes.

```
public class Javafx_program_name  extends Application {

    public static void main( String[] args) {
            System.out.println("launching javafx application");
            aunch( args);
            } // end of main method

    @ Override
    public void init() {
            System.out.println("Inside the init() method.");
            } // end of init()

    @ Override
    public void start(Stage myStage ){
            System.out.print("inside the start() method");
            FlowPane root = new FlowPane();
            Scene myscene = new Scene(root, 300,200);
            myStage.setTitle("JavaFX Skeleton");
            myStage.setScene( myscene);
            myStage.show();
            } // end of start()

    @Override
    public void stop(){
                    System.out.print("inside the stop() method");
            } // end of stop()

    }  // end of application
```

The Application Class and the Life-cycle Methods

- A JavaFX application must be a subclass of the Application class, which is packaged in **javafx.application**. your application class will **extends** Application.
- The abstract javafx.application.Application class defines the essential framework for writing JavaFX programs. Every JavaFX program is defined in a class that extends javafx.application.Application.
- The Application class defines **three life-cycle methods** that your application can override. These are called init( ), start( ), and stop( ), and are shown here, in the order in which they are called:

```
void init( )
abstract void start(Stage primaryStage)
void stop( )
```

JavaFX Application life cycle

The init( ) method is called when the application begins execution. It is used to perform various initializations, it *cannot* be used to create a stage or build a scene. If no initializations are required, this method need not be overridden because an empty, default version is provided.

The start( ) method is called after **init( )**. This is where your application begins and it *can* be used to construct and set the scene. Notice that it is passed a reference to a **Stage** object. This is the stage provided by the run-time system and is the **primary stage**. (You can also create other stages, but you won't need to for simple applications.) Notice that this method is abstract. Thus, it must be overridden by your application.

The stop( ) method is called when your application is terminated, It is here that you can handle any cleanup or shutdown chores. In cases in which no such actions are needed, an empty, default version is provided.

Launching a JavaFX Application

- To start a free-standing JavaFX application, you must call the **launch( )** method defined by **Application**. It is defined in main methods.

```
public static void main(String[] args) {

  System.out.println("Launching JavaFX application.");

  // Start the JavaFX application by calling launch().
  launch(args);
}
```

- When called, **launch( )** causes the application to be constructed, followed by calls to **init( )** and **start( )**.
- The **launch( )** method will not return until after the application has terminated.
- Because it is the same format in all JavaFX application , you can ignore the main method.

Compiling and Running a JavaFX Program

One important advantage of JavaFX is that the same program can be run in a variety of different execution environments. For example, you can run a JavaFX program as a stand-alone desktop application or as a Java Web Start application. However, different ancillary files may be needed in some cases, for example, an HTML file or a Java Network Launch Protocol (JNLP) file.
The easiest way to compile a JavaFX application is to use an Integrated Development Environment (IDE) that fully supports JavaFX programming, such as NetBeans.

The Application Execution

There are two types of threads to execute JavaFX application :

The main thread (the launcher thread): to call the application's **constructor**, **the init( )** method. So, they can't be used to construct a stage or scene.
Application thread: to call **the start( )** method, any changes to the GUI currently displayed, execute events that sent to the program, **stop() method**.