CST8221 – JAP
Lab 1 – Basic Java GUI Application

Objectives

This is a Show-me-the-GUI lab exercise. The purpose of the lab is to give you a hand-on experience of how to write a basic Graphical User Interface (GUI) based Java application using the Java GUI API. You will say “Hello!” both to Swing and JavaFX API.

The Nature of Things

To this point, you have mainly written programs that take input from the keyboard, interpret it, and then display the result in a plain text on a console screen. There is not anything wrong with that (see the quote at end of the lab) but this is not what most of the users expect now from desktop, mobile, or web applications. The users of those applications expect to see a GUI based applications. The concept behind the GUI is very simple: a nice picture is painted on the screen (more colors, better) which consist of small pieces representing GUI components such as icons, buttons, labels, text fields, scrolls and so on. When the user attacks the picture with the available hardware tools such as mouse, keyboard, stylus, game controllers, greasy fingers, and so on (avoid hummers – they do not work well on computers), the attacked GUI component generates an complaint (called event). Then the event is send to the program code behind the GUI and it must react to the event. Usually, the reaction is to repaint the GUI in some way. Simple like that!

In other words, in order to create a GUI we need painting tools, or much better, pre-painted components we can arrange on some kind of canvas (or window), and some mechanism to capture and react to the user interactions with the interface. Fortunately, every modern operating platform (Windows, MacOS, Linux, Solaris, etc.) provides convenient set of pre-painted containers, components, and event generating and handling mechanisms. They are part of the so called graphical environment (often called windowing system) of the operating platform.

Since we use some programming language to implement the application GUI, this programming language must provide a convenient API (library) that gives access to the operating platform graphical environment. In Java this API is called Swing which is part of the Java Foundation Classes (JFC) API. There is also another GUI API in Java – AWT (Abstract Window Toolkit). Swing is not a complete replacement for AWT – it builds on top of the AWT architecture providing much more capable GUI components. It also uses the event handling mechanism provided by AWT. There is a third GUI API currently available: JavaFX. JavaFX provides a powerful, flexible approach that simplifies the creation of visually pleasant GUIs. As such, JavaFX has clearly been positioned as the platform of the future. It is expected to replace Swing in the near future.

In this lab you will try four similar basic GUI applications which illustrate the basic concepts and principle of how to build a Java GUI application.

Tasks

Download the ***CST8221_Lab1_code.zip*** file from Blackboard. Extract the contents. Provided for you are nine different code examples (four basic and three modifications) which implement the same basic GUI. The first three basic examples use Swing GUI API, which is one of the main topics of the course. The forth example uses JavaFX (read the second note below before working with this example). The fifth example shows how to embed Swing GUI into JavaFX GUI. The sixth example demonstrates how to build a JavaFX GUI using FXML and CSS. Compile and run the examples one by one and examine carefully the code. Try to identify the components involved in the GUI, and try to identify the code segments involved in processing the events generated by the user actions - mouse clicks or keyboard entries. The programs are very simple yet they contain all the basic elements of any complex GUI. A top-level window (that is, a window that is not contained inside another window) is usually called a **frame**. The frame is the main container in which all other components are placed and arranged. Any desktop GUI application must have a frame in Java. The Swing library class which represents frame is called ***JFrame*** (be careful, there is a class called ***Frame*** which belongs to AWT API). Actually, ***JFrame*** extends ***Frame***.

In JavaFX the top-level window is called Stage.

Try to understand as much as possible how the programs work. Consult the Java documentation to find out what the different classes and methods are implementing.

These examples will be referred often in lecture time, so make an effort to understand their inner working.

Note 1 (Swing): When you compile and run the lab examples which contain graphics files(example 2,2R) please read the corresponding java file comments before you run the examples with Eclipse or NetBeans.

Note 2 (JavaFX): To run the JavaFX GUI examples you need Java 8 installed on your computers. Eclipse by default will not run the JavaFX examples. After you create a project and copy the corresponding files, you have to open the project Properties and go to Java Build Path > Libraries > click *JRE System Library* > click *Access rules* > click Edit button > click Add button > in *Add Access Rule* select **Resolution:** *Accessible*, type **Rule Pattern**: javafx/** > OK

Note 3 (Lambda Expressions): To run the examples that contain lambda expressions (indicated with a postfix L) you need Java 1.8.XX installed on your computers. If you want to compile and run the using an IDE, you will need the latest Eclipse or NetBeans 8.1.x

Before the lab
Enjoy Java.

During the lab
Ask questions

Before leaving the lab
Sign the attendance sheet.

After the lab
Remember what you have learned. I will discuss it in lecture time.

Submission
No submission is required for this lab.

Marks
No marks are allocated for this lab, but do not forget that the joy of learning is priceless.

What other people say about GUI:
"Beauty in things exists merely in the mind which contemplates them." David Hume, 1742