Java GUI Programming
AWT/SWING - GUI

GUI APPLICATION CLASS

ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

Java GUI frameworks. What to choose? Swing, SWT, AWT, SwingX, JGoodies, JavaFX, Apache Pivot?

AWT (java.awt)

Is the foundation of swing, it performs well but is lacking in advanced components. If you intend to create rich applications, AWT is probably not the way to go. However for smaller GUI applications that doesn't require rich user interfaces. This might suit perfectly as it's a tried and proven framework.

Swing (javax.swing)

Based on AWT as previously stated. In its infancy it was regarded as slow and buggy and caused IBM to create SWT for Eclipse. However with Java 6, Swing became the framework of choice for building new applications. Swing has a lot of rich components but are still lacking in some areas. One example being that there isn't a full featured **TreeTable** component which can do sorting and **filtering/searching**.

Java GUI frameworks. What to choose? Swing, SWT, AWT, SwingX, JGoodies, JavaFX, Apache Pivot?

SWT

Created by IBM for Eclipse, they seemed to think that Swing was not suited for Eclipse at the time. By itself is pretty low-level, and it uses the platform's native widgets through JNI. **It is not related to Swing and AWT at all.** Their API is however somewhat clunky and not intuitive. They do have some advanced component's like a TreeTable. (but i don't think they support sorting and filtering out of the box). SWT uses some native bindings and the rant on the internet is that this framework should not be used in today's projects.

JavaFX (javafx)

The latest flagship of Java/Oracle. promising to be the facto standard in developing rich desktop or web applications.

Java GUI frameworks. What to choose? Swing, SWT, AWT, SwingX, JGoodies, JavaFX, Apache Pivot?

Apache Pivot
It renders UI using Java2D, thus minimizing the impact of (IMO, bloated) legacies of Swing and AWT. It's main focus seems to be on RIA (Rich internet applications), but it seems it can also be applied to desktop applications. And as a personal comment, Looks very interesting! I Especially like that it's an apache project.

https://cwiki.apache.org/PIVOT/frequently-asked-questions-faq.html

Qt Jambi (Good for Desktop Applications, Qt package Cross-language, C/C++, Python, Java)
A java wrapper to the native qt library which is written in **c/c++**. Very powerful, widely used and accepted. Has a lot of GUI components and a easy to use API.

http://qt-jambi.org/

Java GUI Applications

Web page GUI applications (applets): going obsolete because of security issue but with rich programming examples that can be converted to other applications. We will use applets to show the program examples for AWT and Swing because of its easiness to create an application.

Desktop (JFC, Javafx): Main focus in this course.

Responsive Web Page Design (Javafx 2.0/3.0): good direction (at current time) to move on, but lack of educational materials.

AWT Packages

AWT is huge! It consists of **12** packages (Swing is even bigger, with **18** packages as of JDK 1.7!). Fortunately, only 2 packages - java.awt and java.awt.event - are commonly-used.

The java.awt package contains the core AWT graphics classes: (Structural Modeling, like HTML)

- GUI **Component** classes (such as Button, TextField, and Label),
- GUI **Container** classes (such as Frame, Panel, Dialog and ScrollPane),
- **Layout** managers (such as FlowLayout, BorderLayout and GridLayout),
- Custom graphics classes (such as Graphics, Color and Font).

The java.awt.event package supports event handling: (Behavioral Modeling, like Javascript)

- **Event** classes (such as ActionEvent, MouseEvent, KeyEvent and WindowEvent),
- **Event Listener** Interfaces (such as ActionListener, MouseListener, KeyListener and WindowListener),
- **Event Listener Adapter** classes (such as MouseAdapter, KeyAdapter, and WindowAdapter).

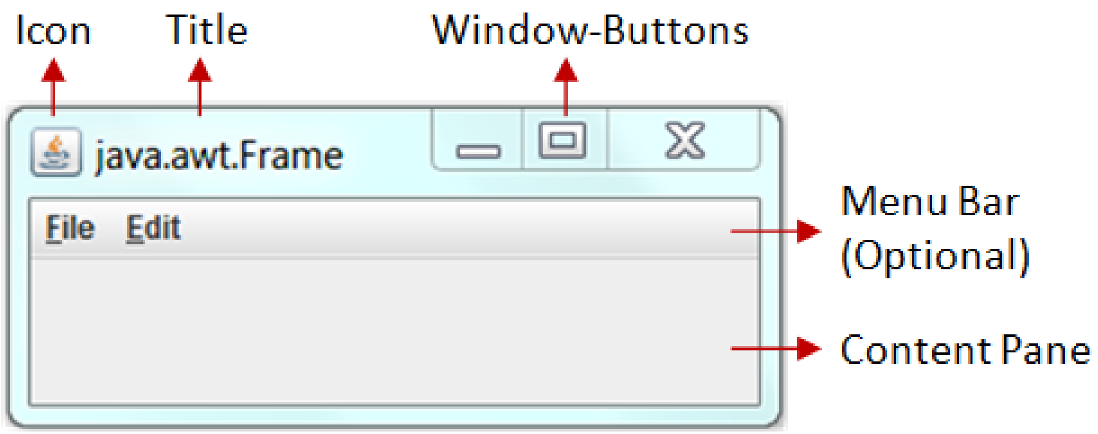
AWT provides a platform-independent and device-independent interface to develop graphic programs that runs on all platforms, such as Windows, Mac, and Linux.

AWT Container Classes

Top-Level Containers: Frame, Dialog and Applet

Each GUI program has a ***top-level container***. The commonly-used top-level containers in AWT are **`Frame`**, **`Dialog`** and **`Applet`**:

- A `Frame` provides the "main window" for the GUI application, which has a title bar (containing an icon, a title, the minimize, maximize/restore-down and close buttons), an optional menu bar, and the content display area. To write a GUI program, we typically start with a subclass extending from **`java.awt.Frame`** to inherit the main window as follows:

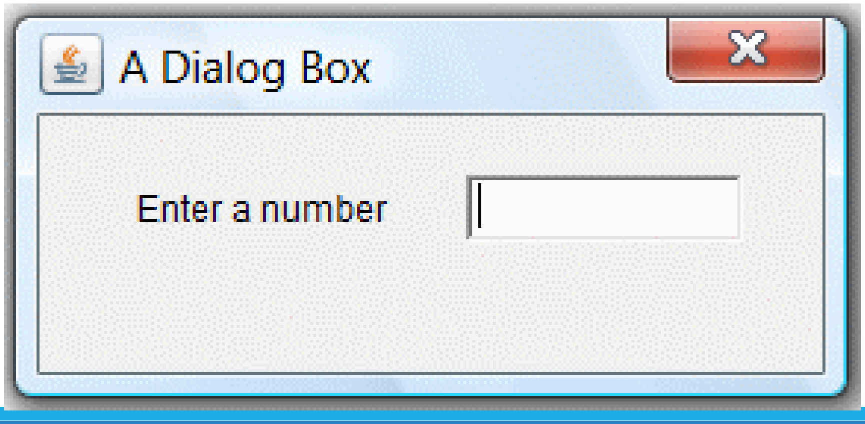


GUI Application Using Frame

```java
import java.awt.Frame;  // Using Frame class in package java.awt

// A GUI program is written as a subclass of Frame - the top-level container
// This subclass inherits all properties from Frame, e.g., title, icon, buttons, content-pane
public class MyGUIProgram extends Frame {

   // Constructor to setup the GUI components
   public MyGUIProgram() { ...... }

   // Other methods
   ......
   ......

   // The entry main() method
   public static void main(String[] args) {
      // Invoke the constructor (to setup the GUI) by allocating an instance
      new MyGUIProgram();
   }
}
```

GUI Application Using Dialog/Applet

- An AWT Dialog is a "**pop-up window**" used for interacting with the users. A Dialog has a title-bar (containing an icon, a title and a close button) and a content display area, as illustrated.
- An AWT Applet (in package java.applet) is the top-level container for an applet, which is a Java program running inside a **browser**. Applet will be discussed in the later chapter.



Javafx Application Class

Application class from which JavaFX applications extend.

Life-cycle

The entry point for JavaFX applications is the Application class. The JavaFX runtime does the following, in order, whenever an application is launched:

1. Constructs an instance of the specified Application class
2. Calls the init() method
3. Calls the start(javafx.stage.Stage) method
4. Waits for the application to finish, which happens when either of the following occur:
    - the application calls Platform.exit()
    - the last window has been closed and the implicitExit attribute on Platform is true
5. Calls the stop() method

FirstEx

The example shows a text in the middle of the application's window.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.stage.Stage;
```

The essential JavaFX classes, collections, and properties reside in the javafx package.

```
public class FirstEx extends Application {
```

Application is the main class of a JavaFX program.

FirstEx

```
@Override
public void start(Stage stage) {

   initUI(stage);
}
```

The Application's start() method is overridden. The start() method is the main entry point to the JavaFX program. It receives a Stage as its only parameter. (Stage is the main application window or area.) The user interface is built in the initUI() method.

```
StackPane root = new StackPane();
```

StackPane is a container used for organizing nodes. It uses a simple layout manager that places its content nodes in a back-to-front single stack. In our case, we only want to center a single node.

FirstEx

Scene scene = new Scene(root, 300, 250);
Scene is the container for all content in a scene graph. It takes a root node as its first parameter. The StackPane is a root node in this scene graph. The next two parameters specify the width and the height of the scene.

Label lbl = new Label("Simple JavaFX application.");
lbl.setFont(Font.font("Serif", FontWeight.NORMAL, 20));
A Label control is created and its font is set with the setFont() method. Label is a non-editable text control.

root.getChildren().add(lbl);
The label control is added to the StackPane. The getChildren() method returns the list of children of a pane.

FirstEx

```
stage.setTitle("Simple application");
```
The setTitle() method of a Stage sets a title for the main window.

```
stage.setScene(scene);
```
The scene is added to the stage with the setScene() method.

```
stage.show();
```
The show() method shows the window on the screen.

```
public static void main(String[] args) {
   launch(args);
}
```
The traditional main() method is not needed. It is only used as a fallback for situations in which JavaFX launching is not working.

Demo Program:
(download GUI_basic.zip for file structure, put files into appropriate directory)

Go BlueJ!!!