JAVA PROGRAMMING 1

Summer 2018 - Christian Hur

# Unit 12 Lecture - Introduction to Swing Components

Reading: Chapter 14

# Objectives:

* Understand Swing components
* Use the JFrame class
* Use the JLabel class
* Use a layout manager
* Extend the JFrame class
* Add JTextFields, JButton, and tool tips to a JFrame
* Learn about event-driven programming
* Understand Swing event listeners
* Use the JCheckBox, ButtonGroup, and JComboBox classes

# Introduction

Computer programs usually are more user friendly (and more fun to use) when they contain graphical user interface (GUI) components. **GUI components** are buttons, text fields, and other components with which the user can interact. GUI components are also called ***controls*** or ***widgets***.

As the internet and Web evolved, so does Java. A brief history of Java’s 3 GUI libraries.

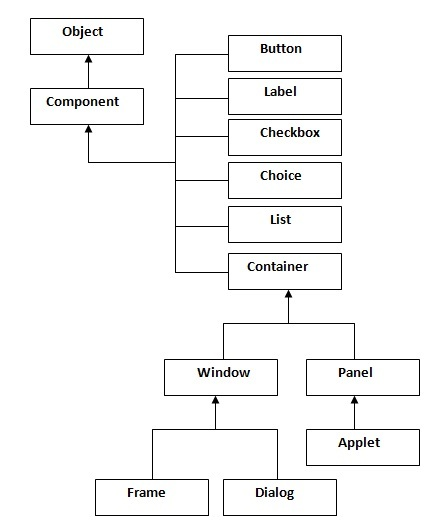
* Abstract Windows Toolkit (AWT)
* Swing
* JavaFX

## Abstract Windows Toolkit (AWT)

In the beginning, the **Abstract Windows Toolkit** (**AWT**) was Java’s first set of GUI library of classes for developing simple graphical user interfaces. AWT is not suitable for developing comprehensive GUI projects. It is prone to platform-specific bugs because the GUI components differ from platform to platform. In other words, the GUI designs are dependent on the local system’s (native) designs. In the AWT, components have simple names, such as ***Frame*** and ***Button***.

Hierarchy of Java AWT classes:

import java.awt.\*;



**Examples:**

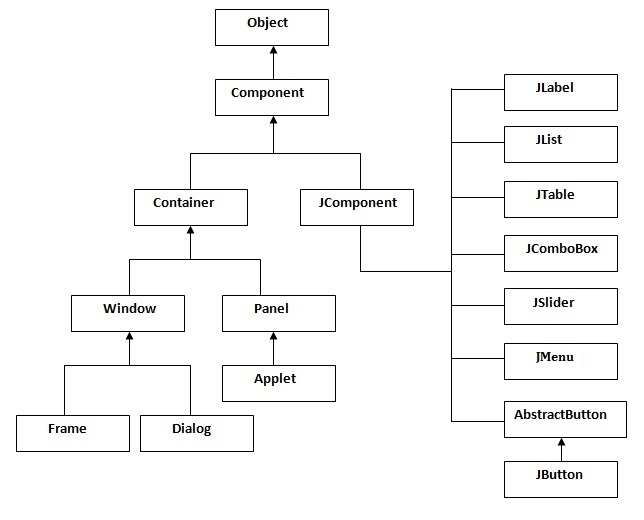
|  |  |
| --- | --- |
| **Extending the Frame class** | **Associating with the Frame class** |
| Button b=new Button("click me");  b.setBounds(30,100,80,30);  add(b);  setSize(300,300);  setLayout(null);  setVisible(true); | Frame f=new Frame();  Button b=new Button("click me");  b.setBounds(30,50,80,30);    f.add(b);  f.setSize(300,300);  f.setLayout(null);  f.setVisible(true); |
| Windows 10 | Windows 7 |
| Notice the buttons look different because they ran on different OS? | |

## The Swing Components

To prevent platform-specific bugs, the AWT user-interface components were replaced with a new library known as **Swing**. The Swing components is a more robust, versatile, and flexible library which allow graphical components to be painted directly on canvases. ***Swing is designed for developing desktop GUI applications, and it doesn’t work well with Web browser***. The Swing components used a ***J*** in front of each new class name such as ***JFrame*** and ***JButton***.

Hierarchy of java swing API

import java.swing.\*;



**Swing Example**:

|  |  |
| --- | --- |
| **Extending the JFrame class** | **Associating with the JFrame class** |
| JButton b=new JButton("click");  b.setBounds(130,100,100, 40);    add(b);  setSize(400,500);  setLayout(null);  setVisible(true); | JFrame f=new JFrame();    JButton b=new JButton("click"); b.setBounds(130,100,100, 40);    f.add(b);  f.setSize(400,300); f.setLayout(null); f.setVisible(true); |
| Windows 10 | Windows 7 |
| Notice the buttons look exactly the same although they ran on different OS? | |

### GUI Comparison between AWT and Swing.

|  |  |
| --- | --- |
| **AWT** | **Swing** |
| Heavy weight - slow processing time  (components uses the resources of system) | Light weight - fast processing time |
| Platform dependent so their look and feel changes according to OS. | Platform independent so their look and feel remains constant throughout. |
| Does not support MVC (Model-View-Controller) | Supports MVC |
| Difference between AWT and Swing | Difference between AWT and Swing |

## JavaFX

Instead of upgrading the Swing components to accommodate RIA, a completely new framework was created and thus the birth of JavaFX!

JavaFX is the new framework replacing the older Swing and AWT libraries for developing cross-platform-rich Internet applications on desktop computers, on hand-held devices (mobile), and on the Web. The purpose of JavaFX is to incorporate modern GUI technologies to develop rich Internet application (RIA) in order to deliver the “same” features and functions normally associated with desktop applications.

***JavaFX applications run seamlessly on a desktop and Web browser***. It also supports multi-touch features for touch-enabled devices (ie. mobile devices, tablets, touch-screen monitors, etc.). In addition, it has a built-in 2D, 3D, animation, video/audio playback.

### Example GUI Buttons: JavaFX vs. Swing

@Override

public void start(Stage arg0) throws Exception {

Button btnOK = new Button("click me");

StackPane pane = new StackPane();

pane.getChildren().add(btnOK);

Scene scene = new Scene(pane, 300, 300);

arg0.setTitle("My FX");

arg0.setScene(scene);

arg0.show();

}

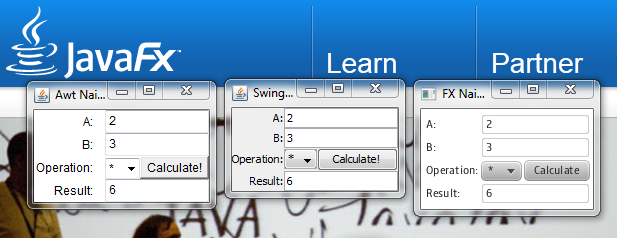
|  |  |
| --- | --- |
| JavaFX | Swing |

### 

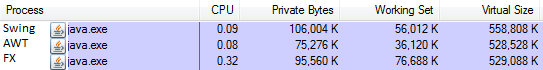
## 

## 

### GUI Toolkits: AWT vs. Swing vs. JavaFX



### Memory Consumption



## The Basic Anatomy of a Swing Program

Almost all Swing components are said to be **lightweight components** because they are written completely in Java and do not have to rely on the local operating system code. This means the components are not “weighed down” by having to interact with the operating system (for example, Windows or Macintosh) in which the application is running.

The only heavyweight components used in Swing are **swing.JFrame, swing.JDialog, swing.JWindow, swing.JApplet, awt.Component, awt.Container,** and **awt.JComponent**.

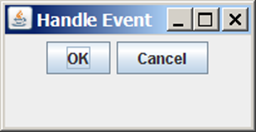
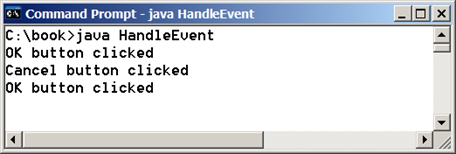
## Procedural vs. Event-Driven Programming

**Procedural Programming** - code is executed in procedural order.

Example: “Your Vote Counts” application uses procedural programming.

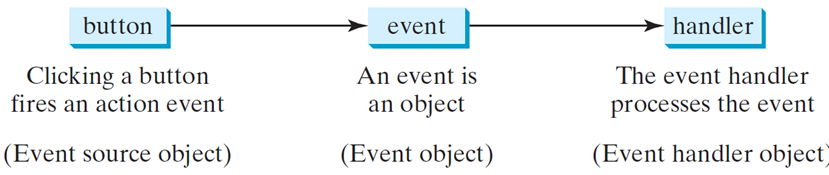
**Event-driven Programming** - code is executed upon activation of events

Example: A GUI application with two buttons waiting for an event to occur.

At a high level, there are three pieces:

1. The component (source object) - WHO
2. The event class (event object) - WHAT
3. The listener interface (event handler) - HOW



Event Information

For a better understanding of how event-driven programming works, let’s first define some important terminologies.

**Event Source Object** (or Source Object, Source Component) is the object (***the WHO***) that is capable of “firing” an event (an action). This is where the action originates.

Examples of source objects are:

* Button
* text field
* Pane
* shape (circle, rectangle, etc.)
* Image
* etc.

**Event Object** - this is the event object (***the WHAT***) that is being fired by the source object. Four main types of event objects:

* ***ActionEvent*** (e.g. click a button, press Enter in a text field, check/uncheck)
* ***MouseEvent*** (e.g. mouse pressed, mouse released, mouse clicked, mouse draggred)
* ***KeyEvent*** (e.g. key pressed, key released, key typed)
* ***WindowEvent*** (e.g. window resized, window moved, window minimized)  
    
  **Event** - a type of signal to the program that something has happened. It is generated by an external user actions (e.g. mouse movements, mouse clicks, keystrokes).

**Event Handler Object** - an object that handles and processes the event (***the HOW***). Not all objects can be handlers for action event. An event hanlder object must meet the following requirements:

1. Object must be an instance of the EventHandler interface: **EventHandler<ActionEvent> handler**
2. The EventHandler object **handler** must be registered with the event source object (e.g. button) using the method: ***source*.setOnAction(handler)**

An event object contains whatever properties are relevant to the event. You can identify the source object (the who) of the event (the what) using the **getSource()** instance method in the **EventObject class**. The subclasses of EventObject deal with special types of events, such as **button actions**, **window events**, **component events**, **mouse movements**, and **keystrokes**.