

## A Brief History of Building GUI's in Java - Part One

---

### Programming Graphical User Interfaces (GUI's) Using Java Swing

---

Lecture notes by Anthony Hornof for CIS 443/543 (10/02/01)

A bit of this material is from Eckstein, Loy, and Wood (1998) *Java Swing*. Bits of code and text are also from Winston and Narasimhan (1998) *On to Java*, 2nd ed.

#### Abstract Window Toolkit (AWT)

- Part of the Java Development Kit (JDK 1.0) from the start (1995?)
- A package that supported some graphics and GUI elements
  - In Java, a package is a group of classes grouped under a common name.
  - Give your code access to classes with "import java.awt.\*" at the top of your source code file.
  - Note: Importing java.awt.\* does automatically also import java.awt.[word].\*
- Supported everything you could do in HTML, plus a little more
- But very limited in terms of its GUI components
- Relied heavily on the runtime platform's native UI components

0

1

## A Brief History of Building GUI's in Java - Part Two

---

#### Swing

- A set of classes that greatly extends AWT
- A subset of the Java Foundation Classes (JFC), which also includes AWT.
- Released in JDK 1.1 (1998)
- Improved with JDK 1.2 (1999)
  - Details on the names are available at <http://java.sun.com/products/jdk/1.2/java2.html>
- You do not load Swing separately. It is part of the Java 2 Platform, Standard Edition.
- More on Swing...

2

## A Brief History of Building GUI's in Java - Part Three

---

#### Swing

- Next-generation GUI toolkit from Sun Microsystems.
- Support trees, tables, tabbed dialogs, tooltips, and other fancy GUI capabilities
- Does not rely on the runtime platform's native UI components
- Takes complete control of the appearance of components
  - Includes pre-built or customizable Pluggable Look-and-Feel (PLAF)
- Distinguishes model and view

3

# Why Swing?

---

Allows you to build GUI's

It's available

- Not the case a year ago

Widely available

Free

Portable

- Solaris is the only officially supported environment, but I would be curious to hear if anyone is using a different JDK 1.2, such as Sun's JDK 1.2 for Windows with Metrowerks CodeWarrior.

Relatively easy to use

You know Java

It's Hot

4

# Swing Resources

---

Sun's web sites

- The Java Tutorial "Creating a GUI with JFC/Swing"
- Java 2 API Specification
  - Packages, All Classes, the complete spec.
- The first two links are the first two links on the on-line version of the Project 2 handout.

Eckstein, Loy, and Wood (1998) *Java Swing*

- Available at UofO bookstore (?) near the required texts for this class

5

## To get started with Swing, follow these steps:

---

1. Log onto a Solaris workstation in 100 Deschutes.
2. Set up your paths  
Described in the "Java Programming Environments for this Class" web page
3. Go to Sun's Swing Tutorial.  
It's the first link on the on-line version of the Project 2 handout.



4. Click right arrow 4 times
  5. Download and run SwingApplication.java
  6. Contact the Prof or GTF if you have any trouble with this.
    - You've now started Project 2
- Questions on Project 2?

6

## Start Project 2 Early

---

The following are not acceptable excuses for a late project:

- "There were no workstations in 100 Deschutes available."
- "Sun's web site crashed"
- "You didn't respond to my email in time."

7

## Code Example: Printing a Vector of Strings Using an Iterator

---

```
Vector v = new Vector ();
... // Fill the vector
Iterator itor = v.iterator();
Object o = null; // used in the loop
while ( itor.hasNext() )
{
    o = itor.next();
    if (o instanceof String)
        System.out.println( (String) o );
}
```

8

## Swing GUI code does not reveal a clear thread of the flow of control

---

You do not write code that you can walk through in your head from beginning to end by just looking at the code.

Remember, GUI's are interactive.

Instead, you do the following:

1. Create a top-level container, typically a frame.
2. Show the frame.
3. Leave it to the application to take care of itself.

9

## The Tiniest Swing Application Possible

---

```
import javax.swing.*;
public class SwingApp {
    public static void main (String argv [ ]) {
        JFrame frame = new JFrame("Tiny Swing
App");
        frame.show();
    }
}
```

## Add functionality by expanding the recipe as follows:

---

1. Create a top-level container.
2. Create components and add them to the top-level container.
3. Tell some components to listen for events.
4. Show the frame.
5. Leave it to the application to take care of itself.
  - The listeners will listen for events and, when they hear an event, will do what you told them to do.

Let's walk through the basic structure of a Swing program.

10

11

# SwingApplication.java

## Walkthrough: General Structure

---

imports at the top

- An old-style swing import is commented out

No “include” statements.

- If there were other classes and the .java files were in the directory, they would be automagically updated and included. (?)

One class in the file

The file is the exact same name as that class

- Compile with “javac SwingApplication.java”
- Run with java SwingApplication

“main” at the end

No clear thread and flow of control.

12

## The imported packages

---

```
import javax.swing.*;
//import com.sun.java.swing.*;
import java.awt.*;
import java.awt.event.*;
```

14

## public interface ActionListener

---

(from Java API)

extends EventListener

The listener interface for receiving action events. The class that is interested in processing an action event implements this interface, and the object created with that class is registered with a component, using the component's addActionListener method. When the action event occurs, that object's actionPerformed method is invoked.

13

## How to add a beep for debugging:

---

```
import java.awt.Toolkit;
Toolkit.getDefaultToolkit().beep();
```

15

## Working with Vectors

---

```
Vector v = new Vector();  
v.addElement(new Song("Jingle Bells", 80));  
(v.firstElement()).print();
```

Java looks for an Object.print()

You must cast the Object as a Song:

```
((Song)v.firstElement()).print();
```

16

## Iterators

---

Problem:

You can't access an element without moving the iterator.

Solution: Use a ListIterator and follow every next() access with a previous().

18

## Other Java Hints

---

Be sure to delete your \*.class files periodically. It doesn't always get the dependencies right.

17

## Events

---

Change in status that can initiate a response from the computer.

Examples:

- Click a mouse button --> Mouse event
- Press a key --> Keyboard event
- Move, hide, click in title bar of window --> Window event

For your program to respond to events

- Define a listener class
- Attach an instance of that class (a listener) to a component

19

## A Listener Class

---

```
public class myWindowListener extends
WindowAdapter {
    public void windowClosing (WindowEvent e)
    {
        System.exit(0); return;
    }
}

// WindowAdapter implements WindowListener
```

20

## A Listener Instance

---

```
public class App {
    public ... main (String argv []) {
        JFrame f = new JFrame("Window");
        f.setSize(300, 100);
        f.addWindowListener(new
        myWindowListener());
        f.show();
    }
}
```

21

## Another Listener Instance

---

```
// from Sun's SwingApplication.java

frame.addWindowListener(new
WindowAdapter() {
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
});

// Anonymous inner class
```

22

## Listeners Create “Delegation”

---

Components delegates responsibility to attached listeners.

Benefits of listeners:

- Refine pre-existing classes, reuse behavior.
- Can be attached to and deleted from components dynamically.
- Keep track of state, such as number of times an event occurred (as in the Sun demo).

23

# The Model-View-Controller (MVC) Architecture

---

Model: The data

View: The display of the data

Controller: How the UI reacts to events

Example: scroll bar

- Model: min, max, current position.
- View: What it looks like.
- Controller: Drag and move, click on the ends.

Benefits: Increased flexibility and reuse.

Where are they in your Java Swing classes?

24

# Swing Simplifies the MVC Architecture

---

Just “Model-View” or “Component and UI Delegate”

Model: Info about the component’s state

UI Delegate: How to draw the component and how to react to events.

Example: JTable

- Model: TableModel
  - JTable(data, columnNames), getModel(), getValueAt(i, j)
- UI Delegate: Handle all GUI responsibilities, including displaying and handling events
  - addMouseListener()

25

# Layout Managers

---

Organize your components within a logical hierarchy of containers.

Nest components into subsets and manage the organization and arrangement of each subset individually.

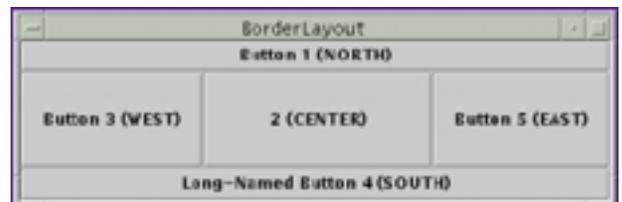
Layout managers are responsible for two tasks:

- Arrange the components in a container
- Calculate the sizes of containers

26

# BorderLayout

---



From Sun's Java Tutorial

Components can only be added to one of five regions.

The components are often other containers, such as JPanels

27