STANFORD UNIVERSITY

CS193J: Programming in Java
Winter Quarter 2003

Lecture 15
Advanced Java Topics

Manu Kumar

sneaker@stanford.edu

Handouts

- 3 Handouts for today!
  - #32: Advanced Java 2
  - #33: Advanced Java 3
  - #34: Java Conclusions

Recap

- Last time
  - Guest lecture by George Grigoryev and Pierre Delisle from Sun
- Before that…
  - SAX XML Parsing
    - XMLDotReader example
  - Advanced Java
    - Regular Expressions
    - Assert
  - HW4 – XEdit
  - Java Implementation and Performance
    - Bytecode
    - Optimization Techniques
- Assigned Work Reminder
  - HW 4: XEdit
    - Due before midnight on Wednesday, August 13th, 2003

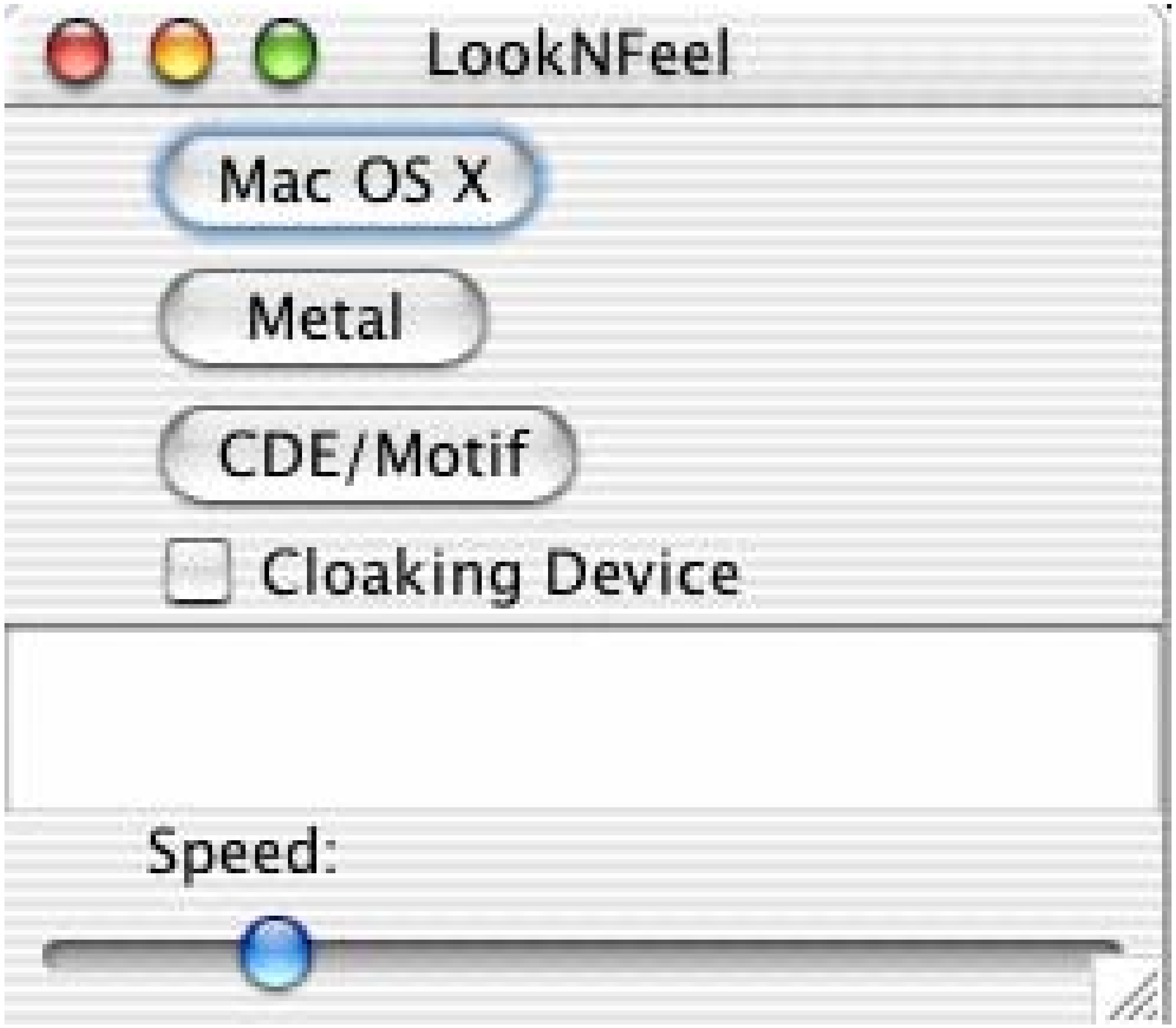
Today

- Today:
  - Advanced Java Topics – very superficial
    - Look and Feel
    - New IO
    - Generics
    - Foreach
    - Java on the client side
      - JWS
    - J2ME/MIDP
    - New 1.4 EventHandler style
    - RMI, JINI, JDBC, Servlets, JSP, Java2D, Java3D
  - Course Evaluations!

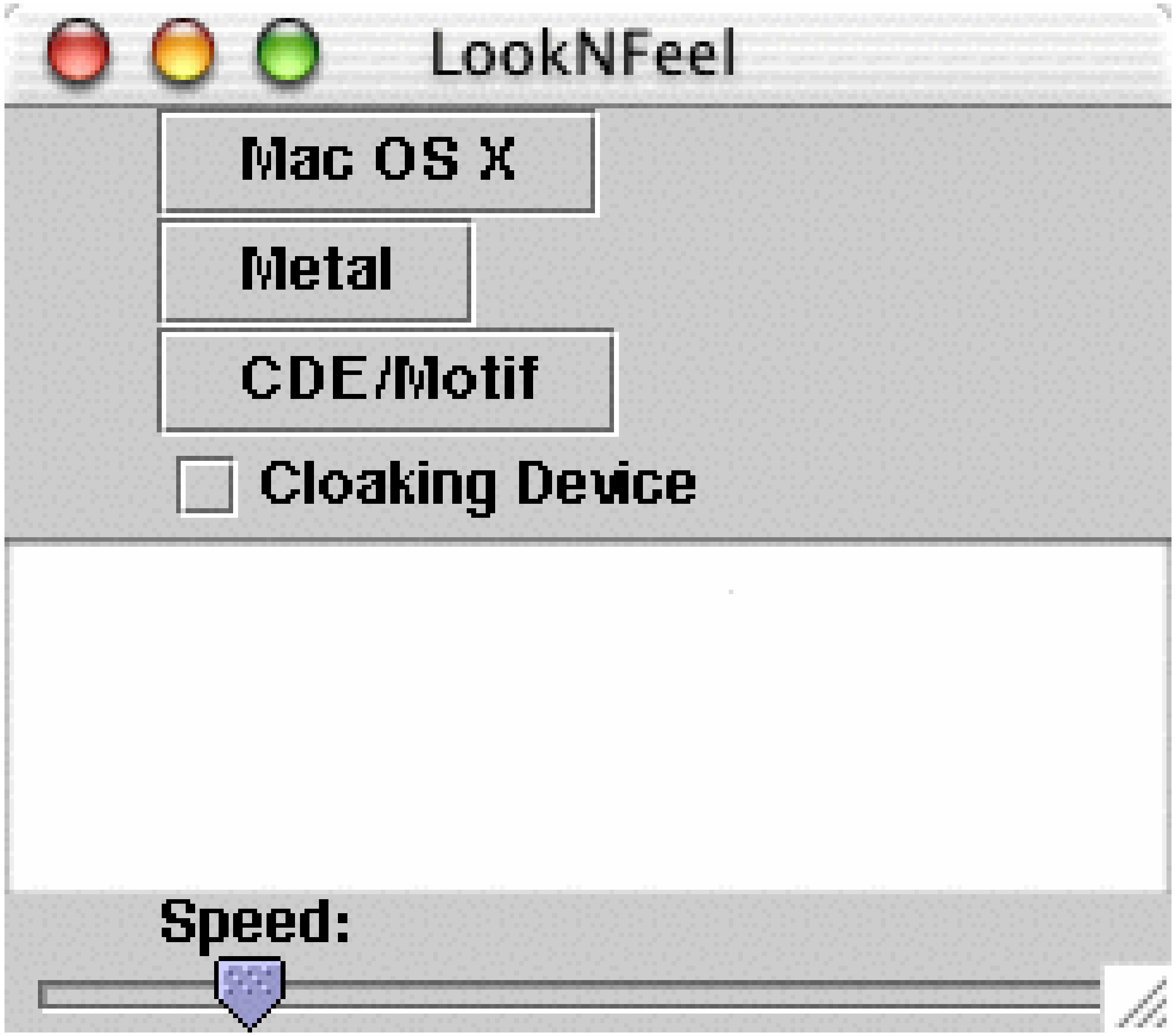
Advanced Java 2 (Handout #32)

- Look and Feel
  - Swing controls can take on different Look N Feel code, to resemble different operating systems.
  - The "metal" look and feel is neutral -- it looks the same on all platforms.
  - By default, a Swing app will use the LnF of the platform where it is running.

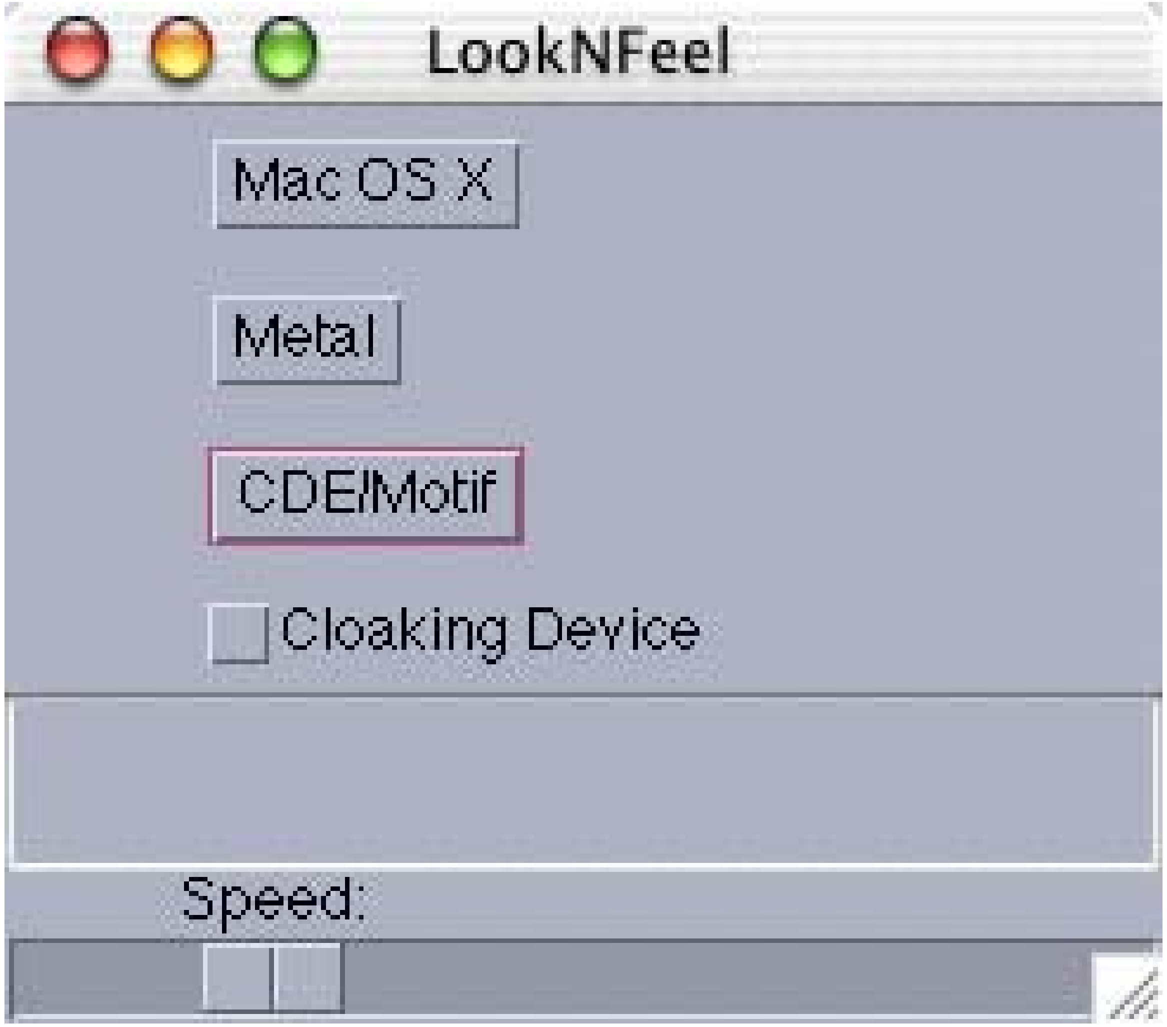
Look and Feel (LnF) – OS X



Look and Feel (LnF) - Metal



Look and Feel (LnF) – Motif / X-windows



Look and Feel Example Code

```
// LookNFeel.java
/*
 Demonstrates changing the look and feel of a Swing app
*/
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;

public class LookNFeel extends JFrame {
	public LookNFeel() {
		super("LookNFeel");

		JComponent content = (JComponent) getContentPane();
		content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));
```

Look and Feel Example Code

```
        // Get a list of the Infs
        UIManager.LookAndFeelInfo[] looks =
UIManager.getInstalledLookAndFeels();

        // Use a hash to map button pointers to Inf class names
        final HashMap map = new HashMap();

  final ActionListener lookListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                // Get the Inf name from the hash
                String look = (String) map.get(e.getSource());
                try {
                // set the Inf
                UIManager.setLookAndFeel(look);

                // Need to do this to change an on-screen window
                SwingUtilities.updateComponentTreeUI(LookNFeel.this);
                }
                catch (Exception ignored) { }
        }
  };
```

Look and Feel Example Code

```
            // For each look, create a button and put an entry
            // in the hashmap button->lnf-class
            for (int i=0; i<looks.length; i++) {
                        JButton button = new JButton(looks[i].getName());
                        button.addActionListener(lookListener);
                        content.add(button);
                        map.put(button, looks[i].getClassName());
            }

            // Put some junk in the window
            content.add(new JCheckBox("Cloaking Device"));
            content.add(new JTextField(10));
            content.add(new JLabel("Speed:"));
            content.add(new JSlider(0, 100, 20));

            pack();
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setVisible(true);

            // Workaround for OSX bug where the content acts
            // like its  minimum size is its preferred size
            //content.setMinimumSize(new Dimension(100, 100));
      }
}
```

NIO (Java 1.4)

- New I/o APIs
  - Introduced in v1.4 provide
  - New features and improved performance in the areas of buffer management, scalable network and file I/O, character-set support, and regular-expression matching
  - The NIO APIs supplement the I/O facilities in the java.io package.
- See
  - http://java.sun.com/j2se/1.4.2/docs/guide/nio/index.html
  - http://developer.java.sun.com/developer/technicalArticles/releases/nio/

NIO Features (Java 1.4)

- The NIO APIs include the following features:
  - Buffers for data of primitive types
  - Character-set encoders and decoders
  - A pattern-matching facility based on Perl-style regular expressions
  - Channels, a new primitive I/O abstraction
  - A file interface that supports locks and memory mapping
  - A multiplexed, non-blocking I/O facility for writing scalable servers

Generics (Java 1.5)

- Compile time types
  - Run time is the same, still checking everytime
  - Just don't need an explicit cast at compile time
    - Cleans up the code and potentially finds compile time errors that may be masked by casting
  - See:
    - http://developer.java.sun.com/developer/technicalArticles/releases/generics/

```
// Suppose Foo responds to the bar() message
ArrayList<Foo> list;
Foo f = ...
list.add(f);...
...
Iterator<Foo> it = list.iterator();
while(it.hasNext()) {
    it.next().bar();        // NOTE: no cast required, it.next() has correct CT type
    ...
}
```

Foreach (Java 1.5)

- Easy way to iterate over collections
  - Does not require an Iterator or index variables
  - Simple syntax

```
String[] strings ...;

for (String s : strings) {
   // use s
}
```

Boxing / Unboxing (Java 1.5)

- Automatic translation between the primitive (int) and it's object form (Integer)
  - Solves the problem that collections can only store pointers to objects and not primitives
- Example

```
ArrayList<Integer> ints;

ints.add(12);      // boxing  12 is converted to new Integer(12)

int val = ints.get(0);        // unboxing: the Integer is
                              // automatically
                              // unboxed into int val
```

VarArgs (Java 1.5)

- Allows you to declare a method that takes a variable number of arguments,
  - Arguments are automatically packed up into an array before being passed to the method
- More discussion about new Java 1.5 features
  - See http://developer.java.sun.com/developer/community/chat/JavaLive/2003/jl0729.html

Advanced Java 3 (Handout #33)

- Sun stewardship
  - Java is controlled by Sun (>7 years now)
    - Not by a non-profit such as W3C
    - Similar to AT&T controlling C/C++
- Vendor support for Java
  - EBM – Everyone But Microsoft
    - IBM, Oracle…
  - Microsoft does not want platform independence that is offered by Java

Java Open Development

- Get a free account on java.sun.com
  - Read the top 25 bugs on the buglist
  - Read the top 25 request for enhancements (RFEs)
  - You can vote on your favorite issues
- Java Community Process (JCP)
  - http://www.jcp.org
  - Discussion of new language features
- Overall, even though Sun officially controls Java, the process of it’s evolution has thus far been pretty open

Java Development Themes

- Backward compatible
  - Old code continues to run as new features are added
- Portable
  - Write Once, Run Anywhere (WORA)
- Large Library
  - More and more off the shelf features
- Elegant/Structured
  - Ass opposed to Perl – quick n’ dirty
- Slow progress
  - Guidance from Sun slow and prudent

Java niches

- Server-side Internet Apps
  - Java very popular here
    - Portable, secure, programmer-efficient
  - "Business Logic" applications using Java and it's JDBC library to connect to databases
    - Usually no GUI
- Custom Applications
  - Custom GUI application as part of a larger custom system
- Client-side Java
  - To implement client interfaces using Java
- Small devices
  - Cell phones, PDAs

Java Platforms

- Java 2 Standard Edition (J2SE)
  - What we cover in this class and more
- Java 2 Micro Edition (J2ME)
  - Intended for small devices
- Java 2 Enterprise Edition (J2EE)
  - Focused on large corporate information technology projects
    - Uses databases, websites, business processes
  - Lot of money spent in this arena
  - J2EE is fairly complex and takes a while to wrap your hands around it
    - Steep learning curve, but potentially big payback as well.

HTML forms are a hack

- But a successful hack
  - All sites use HTML forms
    - Amazon, Yahoo, eBay
  - Huge advantage of compatibility
    - Lowest common denominator
    - HTTP, HTML are standards
- Issues
  - We've gotten so used to HTML forms that we've forgotten how lame they are for good UI design
  - Request-Response paradigm
  - Not as rich as a real UI

Applets

- Popular in the early days of Java
- Allow executable code to be embedded within HTML pages
  - Run in a security "sandbox" in the browser to prevent the applet from doing any damage
    - Signed/unsigned applets
- Issues
  - Performance issues
  - Original applets used AWT
    - Needed Java 1.2 to use Swing
  - Microsoft froze Java support in IE at Java 1.1
    - Sun released Java plug-in, but it's not as automatic (requires initial download)

Jar files

- .jar
  - Archive file that contains directories of .class files and misc. images, sounds and support files.
  - Double-click on the .jar runs the application
    - Works on Windows, Solaris and OS X
  - Must have Java installed first
    - Code does not run in a "sandbox"
  - Good format for distributing a Java application

Java Web Start

- Objective
  - Convenience of an applet, without the problems of running in a browser
- JWS
  - Replacement for applets and jar files
    - http://java.sun.com/products/javawebstart
  - Client installs JWS loader on their machine once
    - Included with JRE installation
  - Vendor packages application as a Jar file
  - Vendor provides link on a website to a JNLP (Java Network Launching Protocol) file which specifies the location of the jar file
  - JWS downloads and caches the jar file and runs the application

Nick's Dice Machine example!

- Demo
  - Running a Jar file
  - Running through Java Web start
- URL
  - http://xenon.stanford.edu/~nick/dice/

JNLP file example

```
<?xml version="1.0" encoding="utf-8"?>
<!-- trying to make a simple, working jnlp for DiceMachine.jar -->
<jnlp
	spec="1.0+"  <!-- can be omitted -->

	<!-- where other things are found -->
	codebase="http://www-cs-students.stanford.edu/~nick/dice/"
	<!-- where the .jnlp file itself lives -->
	href="dice.jnlp"
>

<information>
	<title>DiceMachine</title>
	<vendor>Nick Parlante</vendor>
	<homepage href="http://www-cs-students.stanford.edu/~nick/dice/"/>
	<description kind="one-line">Dice rolling application</description>
	<description kind="short">Dice rolling application that graphs the distribution or rolls. Perfect for the
	game Settlers of Catan.</description>
	<icon href="dice-small.jpeg"/>
	<!-- this allows the app to be run without a net connection -->
	<offline-allowed/>
</information>
<resources>
	<j2se version="1.2+"/>
	<jar href="DiceMachine.jar" main="true" download="eager" />
</resources>
<!-- what's the main class -->
<application-desc main-class="DiceMachine"/>
</jnlp>
```

More JWS

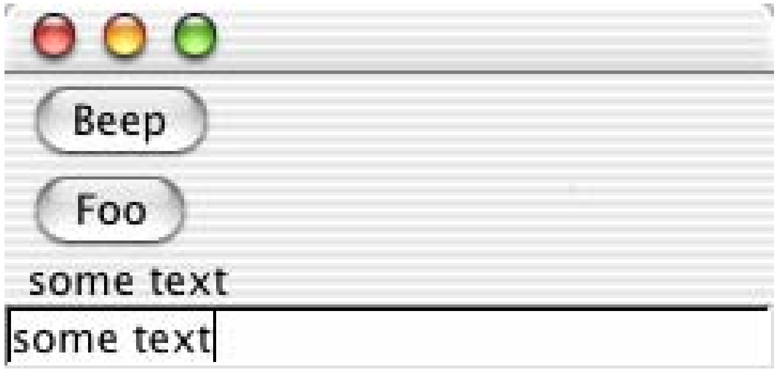
- More JWS
  - Unsigned code runs in a sandbox
  - The client just downloads the .jnlp file which points to enough info for the client to download and run the java code.
  - Can run with or without a net connection once downloaded.
  - Can check for updates automatically
  - The point: You send someone just a URL, and they can just click it to run the program on their machine. Updates can happen automatically.
- Will JWS Catch On?
  - Like Flash catching on -- chicken-and-egg problem that works best if many clients have it pre-installed.
  - This will be hard since Microsoft controls the dominant OS and browser, and Microsoft hates Java
  - Enterprises love it internally -- easy way to distribute and update little custom apps -- just send out the URL

J2ME/MIDP

- Mobile Information Device Profile
  - Allows you to write apps that work on cell phones and PDAs
  - Links
    - http://java.sun.com/j2me
    - http://java.sun.com/products/midp
  - Uses a subset of Java

New 1.4 EventHandler Style

- Removes the need for creating lots of ActionListener objects
  - Instead uses EventHandler.create(…) to specify what object to notify and what message to send
- Idea: Make it easier for a GUI building tool
  - Example BeanBuilder (in development)



EventHandler example

```
// Swing2
/*
 Demonstrates a little use of the EventHandler class.
*/
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import java.beans.*;
public class Swing2 extends JFrame {
	JTextField field;
	JLabel label;


	public void beep() {
		System.out.println("beep!");
	}
```

EventHandler Example

```
public Swing2() {
        JComponent content = (JComponent) getContentPane();
        content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));

        JButton b1 = new JButton("Beep");
        content.add(b1);
        b1.addActionListener(
                // Send msg to: this
                // Message to send: beep
        (ActionListener)EventHandler.create(ActionListener.class, this, "beep")
);


        JButton b2 = new JButton("Foo");
        content.add(b2);
        b2.addActionListener(
        (ActionListener)EventHandler.create(ActionListener.class, this, "foo")
);
// When clicked, this looks for a foo() message, which does not exist

JLabel label = new JLabel("label");
content.add(label);
```

EventHandler Example

```
	field = new JTextField(20);
	content.add(field);

	field.addActionListener(
		// send msg to: label
		// msg to send: setLabel
		// value to send: event.getSource().getText()
		(ActionListener)EventHandler.create(ActionListener.class, label, "text", 
	"source.text")
	);


		pack();
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setVisible(true);

	}

	public static void main(String[] args) {
		new Swing2();
	}
}
```

Java Beans

- Very simple concept
  - Has an empty (default) constructor
  - Has getters and setter methods
- Beans are used as a unit of exchange
  - Module A wants to package information for others to use
    - Setup a "bean" class that uses getters and setters
- Tools designed to work with beans

XML Persistence

- Serialization issue
  - What is the implementation of the class changes
  - Hard to implement backward/forward compatibility
- XML Persistence
  - Only serialize state that is accessible through public get/set methods (the “bean” view of the object)
  - Allows addition of additional getter/setter methods
- Resources
  - http://java.sun.com/j2se/1.4/docs/guide/beans/index.html
  - http://java.sun.com/products/jfc/tsc/articles/persistence/
  - http://java.sun.com/products/jfc/tsc/articles/persistence2/
  - http://java.sun.com/products/jfc/tsc/articles/persistence3/

Java Buzzword Bingo

- RMI
  - Remote Method Invocation
    - For building distributed applications
    - Relies on Serialization of objects to send them over the network
    - Performance slow, but saves lots of network level details
- JINI
  - "Federation" layer allowing little devices to cooperate via networking
    - Example: CD-player since its GUI code to your Palm Pilot
- JDBC
  - Standard layer to interact with a database, send queries, receive results, execute updates

Java Buzzword Bingo

- Java Servlets
  - Replacement for CGI scripts
  - Allows Java code to execute on web server for building web applications
- JSP
  - Java Server Pages
    - Allows mixing of Java code within HTML pages.
    - Compile to a servlet before executing
    - Similar to ASP, PHP etc.
- Java2D, Java3D, Imaging
  - Packages for manipulating graphics and images

Summary!

- Today
  - Advanced Java Topics – very superficial
    - Look and Feel
    - New IO
    - Generics
    - Foreach
    - Java on the client side
      - JWS
    - J2ME/MIDP
    - New 1.4 EventHandler style
    - RMI, JINI, JDBC, Servlets, JSP, Java2D, Java3D
- Assigned Work Reminder
  - HW 4: XEdit
    - Due before midnight on Wednesday, August 13th, 2003

Note!

- No class on Thursday!
  - Today is our final lecture
  - On Thursday, Shankar will have office hours in his office (Gates 252) during regular class time
    - Use these office hours to address any grading questions you have on homeworks!
    - We will NOT entertain any further regrades (on HW1-3) beyond Thursday
- Thank you!