A Java Crash Course

Dan Wallach, ***Princeton University***

Outline

- Applet "Hello, World"
  - graphics
  - widgets
- AWT event model
- Multithreaded programming
- Networking
- Utilities and tricks

Starting resources

- If you own only one book...

  Java in a Nutshell, David Flanagan (O'Reilly & Associates)

  `http://www.ora.com/catalog/javanut/examples/` (examples online!)
- "Whenever possible, steal code." [Duff]

  `http://www.developer.com` (formerly *gamelan.com*)

  `http://www.acme.com/java/software/`

  `http://java.sun.com`

Normal Hello World

```
public class Hello {
    public static void main(String args[]) {
        System.out.println("Hello, world.");
    }
}
```

- Put in: `Hello.java`
- Compile with: `javac Hello.java`
  - Creates `Hello.class`
- Run with: `java Hello`

Applet Hello, world #1

```
import java.applet.*;    // Don't forget these import statements!
import java.awt.*;

public class FirstApplet extends Applet {
    // This method displays the applet.
    // The Graphics class is how you do all drawing in Java.
    public void paint(Graphics g) {
        g.drawString("Hello, world.", 25, 50);
    }
}
```

- *paint()* called by system when refresh is necessary
- *Graphics* class has lines, polygons, text, images, etc.

Hello, world #2

```
import java.applet.*;
import java.awt.*;
import java.io.*;

public class HelloWorld2 extends Applet {
    TextArea textarea;

    // Create a text area to send our output
    public void init() {
        textarea = new TextArea(20, 60);
        this.add(textarea);
        Dimension prefsize = textarea.preferredSize();
        this.resize(prefsize.width, prefsize.height);
    }
```

- Make a scrolling text area where you can do terminal-like output

Hello, world #2 (cont.)

```
        public void start() {
            ByteArrayOutputStream os = new ByteArrayOutputStream();
            PrintStream ps = new PrintStream(os);

            try {
                go(ps);
            } catch (Throwable t) {}

            textarea.setText(os.toString());
        }

        public void go(PrintStream ps) {
            // your program goes here
            ps.println("Hello, world.");
        }
    }
```

- text printed after program is done
- *TextArea* widget redraws itself

Applets in HTML

```
<applet
    CODE="ScrollingText.class"
    CODEBASE="http://www.whatever.com/applets/"
    ARCHIVE="http://www.whatever.com/applets/ScrollingText.zip" (Netscape 3)
    WIDTH=500
    HEIGHT=500>

<param name="text" value="Dan & Drew's Excellent Java Class">
<param name="speed" value="5">

<img src="nojava.gif" alt="Oh, you don't have Java.  Sorry.">

</applet>
```

- *codebase*/*archive* tags are optional
- argument-passing through *param* tags

Applet class

- java.applet.Applet
  - you *extend* this for your applet
- java.awt.Panel
- java.awt.Container
  - applet widget can contain other widgets
- java.awt.Component
  - lots of interesting methods here
- java.lang.Object

Basic methods on Applet

- init()
  - called once for your applet
- start()
  - called every time you enter the page
- stop()
  - called every time you leave the page
- destroy()
  - called when your page is discarded

Funky methods on Applet

- AudioClip getAudioClip(URL url)
- Image getImage(URL url)
  - starts asynchronous image loading
- void showDocument(URL url)
  - tells browser to load new document
  - optional second argument for frames
- void showStatus(String msg)
  - writes to browser status line

Applet repainting

- paint()
  - defaults to nothing
- update()
  - clears screen, calls *paint()*
- repaint()
  - passes events to Motif/Win32
  - don't mess with this

Applet event handling

- boolean handleEvent(Event evt)
  - mouse, keyboard, all widget events
  - checks event type, then calls...
- mouseUp() / mouseDown() / keyUp() / keyDown()
- action(Event evt, Object arg)
  - evt.target - specific widget
  - arg - widget-specific result (i.e., new state of a checkbox)

Applet event handling

- Centralized event management
  - add standard buttons, widgets as children of the top-level applet
  - custom *action()* method, checks *evt.target*
- Distributed event management
  - subclass buttons, widgets
  - custom *action()* methods in subclasses

Java and threads

- One lock per object plus one per class
- *synchronized* keyword on a method
- Mesa-style monitors
  - *wait() / notify() / notifyAll()*
  - must be called within a *synchronized* block
- System classes already thread-safe
  - HashTable, OutputStream, AWT, etc.

Thread-safe Message Passing

```
public class SafeBuffer {
  private Object buffer;

  public SafeBuffer() {}

  synchronized public void put(Object o) {
    while(buffer != null) {
      try {
        wait();
      } catch (InterruptedException e) {}
    }
    buffer = o;

    notifyAll();
  }
```

```
  synchronized public Object get() {
    while(buffer == null) {
      try {
        wait();
      } catch(InterruptedException e) {}
    }
    Object tmp = buffer;
    buffer = null;

    notifyAll();
    return tmp;
  }
}
```

Exercise for reader: barrier sync, bounded-buffer queue, etc.

Starting Threads

```
class client implements Runnable {
  private SafeBuffer b;

  public client(SafeBuffer b) {
    this.b = b;
  }

  public void run() {
    String s;
    ...
    s = (String) b.get();
    ...
  }
}
```

```
...

SafeBuffer sb = new SafeBuffer();
new Thread(new client(sb)).start();

sb.put("Hello, world.");

...
```

- Thread constructor takes any object which *implements Runnable*

Networking

- Applet restrictions
  - Same IP address which loaded applet
  - UDP support is flakey
- Using the browser's cache
  - java.net.URL constructor takes normal string argument
  - InputStream toStream()
  - only current way to get SSL support

Networking

- **client**: java.net.Socket
  - constructor takes DNS name, port
  - getInputStream() / getOutputStream()
- **server**: java.net.ServerSocket
  - constructor takes local port number
  - Socket accept()
    - blocks until success -- use multithreading!

Utilities and tricks

- StringBuffer vs. String
  - strings are immutable
  - StringBuffer *append()* is cheaper
- java.util.Hashtable
  - uses *Object.hashCode()* and *Object.equals()*
- java.util.StringTokenizer
  - split string on whitespace / separator chars

Use Javadoc

- Literate programming for Java
  - document as you write code
  - generates pretty, cross-linked HTML

```
/**
 * Creates an absolute URL from the specified protocol,
 * host, port and file.
 * @param protocol the protocol to use
 * @param host the host to connect to
 * @param port the port at that host to connect to
 * @param file the file on that host
 * @exception MalformedURLException If an unknown protocol is
 * found.
 */
public URL(String protocol, String host, int port, String file)
    throws MalformedURLException {
...
```


Useful tools

- Debugging / Development
  - Microsoft Visual J++ / Symantec Visual Café
    - Debuggers integrated with IE / Netscape
  - Kaffe - free JVM with JIT (`http://www.kaffe.org`)
  - Jikes - fast Java compiler from IBM
    `http://www.alphaworks.ibm.com` (?)
  - `cc-mode` for Emacs understands Java
    `ftp://ftp.python.org/pub/emacs/cc-mode.tar.gz`
- When in doubt…
  - `http://www.developer.com/`