Today's Lecture

More Java:

1. Linked Structures

2. Object Finalization

3. Applets

4. `java.util`

5. `main` from Sun's Hotjava browser

Java: Linked Structures

- Java's classes correspond to C's `structs`

- Class instantiation (object construction) corresponds to allocating space for a C `struct`

- Dynamic structures are built by *chaining* objects

- Java's runtime *garbage collector* reclaims an object when it's no longer in use

Java: Linked Lists


```
/* IntList.java */

class IntList {
    private IntList rest = null;
    private int the_int;
    private int len = 0;

    public IntList(int i) {
        the_int = i;
        len = 1;
    }

    public IntList(int i,IntList l) {
        rest = l;
        the_int = i;
        len = 1 + l.length();
    }

    public int head() {
        return the_int;
    }

    public IntList tail() {
        return rest;
    }

    public int length() {
        return len;
    }
}

/* end of IntList.java */
```


Java: Trees


```
/* IntTree.java */

class IntTree {
    private IntTree left = null;
    private IntTree right = null;
    private int the_int;
    private int size = 0;
    private int depth = 0;


    public IntTree(int i) {
        the_int = i;
        size = 1;
        depth = 1;
    }

    public IntTree(int i,IntTree l,IntTree r) {
        the_int = i;
        left = l;
        right = r;
        size = l.size() + r.size() + 1;
        depth = Math.max(l.depth(),r.depth) + 1;
    }

    public IntTree left() {
        return left;
    }

    public IntTree right() {
        return right;
    }

    public int value() {
        return the_int;
```

```
    }

    public int depth() {
        return depth;
    }

    public int size() {
        return size;
    }
}

/* end of IntTree.java */
```


Object Finalization

- Recall, a *constructor* method initializes an object
- Similarly, a *finalizer* method may be used to *finalize* an object
- The name `finalize` is reserved for a class' finalization method
- `finalize` always has return-type `void` and no parameters
- An example from Java's `FileOutputStream` class:

```
/* Closes the stream when 'this' is garbage collected
 * Checks the file descriptor first to make sure that
 * it is not already closed.
 */

    protected void finalize() throws IOException {
        if (fd != null) close();
    }
```


Object Finalization (cont'd)

- If an object has a finalizer, that method is invoked before the object is reclaimed (GC'd).
- A Java program may exit before all outstanding objects have been collected (and hence finalized).
- Java *usually* frees outstanding resources before exiting.
- A finalizer method may "resurrect" an object – this delays the actually freeing of the object until the next GC.
- Exceptions that escape out of finalizers are ignored by the system. Why?
- The programmer must explicitly *chain* any superclass finalizers.

Object Finalization Example


```
/* FinalizedIntList.java */

class FinalizedIntList {
    public static int approxNumberOfInstances = 0;

    private FinalizedIntList rest = null;
    private int the_int;
    private int len = 0;

    public FinalizedIntList(int i) {
        approxNumberOfInstances++;
        the_int = i;
        len = 1;
    }

    public FinalizedIntList(int i,FinalizedIntList l) {
        approxNumberOfInstances++;
        rest = l;
        the_int = i;
        len = 1 + l.length();
    }

    public int head() { return the_int; }

    public FinalizedIntList tail() { return rest; }

    public int length() { return len; }

    public void finalize() {
        approxNumberOfInstances--;
    }
}

/* end of FinalizedIntList.java */
```

Applets

To date, we've only dealt with *stand-alone* Java applications

Code for network transmission and execution (in a browser) is accomplished via Java *applets*

A special HTML *tag* (`APPLET`) is used to insert an applet into an HTML document

A Java-capable browser will fetch and execute the compiled classes (*i.e.* bytecodes) corresponding to the referred-to applets

An applet may request "real estate" in the browser's display for output and user input

An applet is permitted limited communication with other network hosts (*e.g.*, it may communicate with servers on the originating host)

Subsequent lectures will cover details of Java graphics, events, and networking

Applets: `HelloWorld.java`

An applet requires HTML to reference it

```
<APPLET CODE="HelloWorld.class" WIDTH=100 HEIGHT=50> </APPLET>
```

---

and a subclass of the Java-supplied Applet class to implement it:

```
public class HelloWorld extends Applet {
    public void init() {
        resize(150,25);
    }

    public void paint(Graphics g) {
        g.drawString("Hello World",50,25);
    }
}
```

Note: applets do not have a `main` *entry point*

Important Applet Methods

Methods for creating, initiating and destroying an applet:

`void init()`

`init` is called once, when the applet is first created. The derived applet subclass should override this method and perform any initialization required by the new applet.

`void destroy()`

`destroy` is called once, when the applet is no longer necessary. The derived applet subclass may override this method to perform any necessary clean up.

`void start()`

`start` is called when the applet is visited. The derived applet may override this method to perform actions associated with a visit, *e.g.* resume an animation, *etc.*

`void stop()`

`stop` is called when the applet leaves the screen (*i.e.* browser display). The derived applet may override this method to perform actions associated with the applet becoming (temporarily) inaccessible, *e.g.* suspend an animation, *etc.*

Important Applet Methods (cont'd)

Methods for reflecting changes to the applet drawing area:

`void repaint()`

`repaint` should be called when the applet needs to be scheduled for repainting. This method should not be overridden. Invocation of `repaint()` will result in a subsequent invocation of `update` by the system.

`void update(Graphics g)`

`update` is the method called by the system when the screen needs to be updated. This method clears the applet drawing area and calls the `paint` method. In special cases the derived class may need to override this method.

`void paint(Graphics g)`

`paint` is called whenever the applet drawing area needs to be refreshed. The derived applet should (almost always) override this method to provide it's graphical content.

Applets: Scribble.java


```
/* Scribble.java */

import java.applet.*;
import java.awt.*;

public class Scribble extends Applet {
    private int last_x = 0;
    private int last_y = 0;
    private Color cur_color = Color.black;

    private Button clear_button;
    private Choice color_choices;

    public void init() {
        this.setBackground(Color.white);

        clear_button = new Button("Clear");
        clear_button.setForeground(Color.black);
        clear_button.setBackground(Color.lightGray);
        this.add(clear_button);

        color_choices = new Choice();
        color_choices.addItem("black");
        color_choices.addItem("red");
        color_choices.addItem("yellow");
        color_choices.addItem("green");
        color_choices.setForeground(Color.black);
        color_choices.setBackground(Color.lightGray);
        this.add(new Label("Color"));
        this.add(color_choices);
    }
```



```
    public boolean mouseDown(Event e,int x,int y) {
        last_x = x; last_y = y;
        return true;
    }

    public boolean mouseDrag(Event e,int x,int y) {
        Graphics g = this.getGraphics();
        g.setColor(cur_color);
        g.drawLine(last_x,last_y,x,y);
        last_x=x;
        last_y=y;
        return true;
    }

    public boolean action(Event e,Object a) {
        if (e.target == clear_button) {
            Graphics g = this.getGraphics();
            Rectangle r = this.bounds();
            g.setColor(this.getBackground());
            g.fillRect(r.x,r.y,r.width,r.height);
            return true;
        }
        else if (e.target == color_choices) {
            if (a.equals("black")) cur_color = Color.black;
            else if (a.equals("red")) cur_color = Color.red;
            else if (a.equals("yellow")) cur_color = Color.yellow;
            else if (a.equals("green")) cur_color = Color.green;
            return true;
        }
        else return super.action(e,a);
    }
}

/* end of Scribble.java */
```


Java Classes

Sun currently supplies these Java class packages:

`java.applet` – classes that handle applets

`java.awt` – classes for the Abstract Windowing Toolkit

`java.io` – classes for File IO

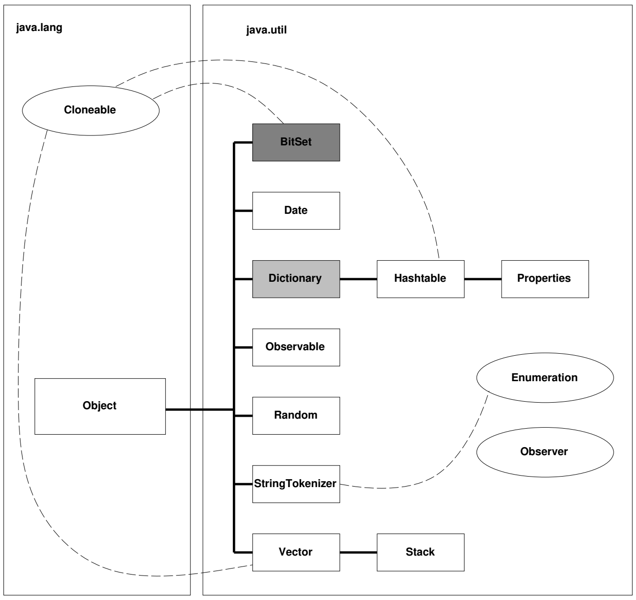
`java.lang` – support for java datatypes and control (threads)

`java.net` – networking support (sockets, URLs)

`java.tools` – classes for debugging, documentation

`java.util` – utility classes, building blocks

java.util



java.util: class BitSet


```
public final class BitSet extends Object implements Cloneable {
    // constructors
    public BitSet();
    public BitSet(int nbits);
    // public instance Methods
    public void and(BitSet set);
    public void clear(int bit);
    public Object clone(); // overrides Object.clone()
    public boolean equals(Object o); // overrides Object.equals()
    public boolean get(int bit);
    public int hashCode(); // overrides Object.hashCode()
    public void or(BitSet set);
    public void set(int bit);
    public int size();
    public string toString(); // overrides Object.toString()
    public void xor(BitSet set);
}
```


java.util: class Date


```
public class Date extends Object {
    // constructors
    public ();
    public (long date);
    public (int year,int month,int date);
    public (int year,int month,int date,int hrs,int min);
    public (int year,int month,int date,int hrs,int min,int sec);
    public (String s);
    // class methods
    public static long UTC(int year,int month,int date,int hrs,int min,int sec);
    public static long parse(String s);
    // public instance methods
    public boolean after(Date when);
    public boolean before(Date when);
    public boolean equals(Object o); // overrides Object.equals()
    public int getDate();
    public int getDay();
    public int getHours();
    public int getMinutes();
    public int getSeconds();
    public long getTime();
    public int getTimezoneOffset();
    public int getYear();
    public int hashCode(); // overrides Object.hashCode()
    public void setDate(int date);
    public void setHours(int hours);
    public void setMinutes(int minutes);
    public void setMonth(int month);
    public void setSeconds(int seconds);
    public void setTime(long time);
    public void setYear(int year);
    public string toGMTString();
    public string toLocaleString();
    public string toString(); // overrides Object.toString()
}
```


java.util: interface Enumeration


```
public abstract interface Enumeration {
    // public instance methods
    public abstract boolean hasMoreElements();
    public abstract Object nextElement()
        throws NoSuchElementException;
}
```


java.util: class Dictionary


```
public abstract class Dictionary extends Object {
    // public constructors
    public Dictionary();
    // public instance methods
    public abstract Enumeration elements();
    public abstract Object get(Object key);
    public abstract boolean isEmpty();
    public abstract Enumeration keys();
    public abstract Object put(Object key,Object value)
        throws NullPointerException;
    public abstract Object remove(Object key);
    public abstract int size();
}
```

```
public class Hashtable extends Dictionary implements Cloneable {
    // public constructors
    public Hashtable(int initialCapacity,float loadFactor)
        throws IllegalArgumentException;
    public Hashtable(int initialCapacity);
    public Hashtable();
    // public instance methods
    public synchronized void clear();
    public synchronized Object clone(); // overrides Object.clone();
    public synchronized boolean contains(Object value)
        throws NullPointerException;
    public synchronized boolean containsKey(Object key);
    public synchronized Enumeration elements(); // Dictionary.elements()
    public synchronized Object get(Object key); // Dictionary.get()
    public boolean isEmpty(); // Dictionary.isEmpty()
    public synchronized Enumeration keys(); // Dictionary.keys()
    public synchronized Object put(Object key,Object value)
        throws NullPointerException; // Dictionary.put()
    public synchronized Object remove(Object key); // Dictionary.remove()
    public int size(); // Dictionary.size()
    public synchronized String toString(); // overrides Object.toString()
    // protected instance methods
    protected void rehash();
}
```



```
public abstract interface Observer {
    // public instance methods
    public abstract void update(Observable o,Object e);
}
```



```
public class Observable extends Object {
    // default constructor: public Observable()
    // public instance methods
    public synchronized void addObserver(Observer o);
    public synchronized int countObservers();
    public synchronized void deleteObserver(Observer o);
    public synchronized void deleteObservers();
    public synchronized boolean hasChanged();
    public void notifyObservers();
    public synchronized void notifyObservers(Object o);
    // protected instance methods
    protected synchronized void clearChanged();
    protected synchronized void setChanged();
}
```



```
public class Properties extends Hashtable {
    // public constructors
    public Properties();
    public Properties(Properties defaults);
    // protected instance variables
    protected Properties defaults;
    // public instance methods
    public String getProperty(String key);
    public String getProperty(String key,String defaultValue);
    public void list(PrintStream out);
    public synchronized void load(InputStream in)
        throws IOException;
    public Enumeration propertyNames();
    public synchronized void save(OutputStream out,String header);
}
```

java.util: class Random


```
public class Random extends Object {
    // public constructors
    public Random();
    public Random(long seed);
    // public instance methods
    public double nextDouble();
    public float nextFloat();
    public synchronized double nextGaussian();
    public int nextInt();
    public long nextLong();
    public synchronized void setSeed(long seed);
}
```


java.util: class StringTokenizer


```
public class StringTokenizer extends Object implements Enumeration {
    // public constructors
    public StringTokenizer(String str,String delim,boolean returnTokens);
    public StringTokenizer(String str,String delim);
    public StringTokenizer(String str);
    // public instance methods
    public int countTokens();
    public boolean hasMoreElements();
    public boolean hasMoreTokens();
    public Object nextElement()
        throws NoSuchElementException;
    public String nextToken()
        throws NoSuchElementException;
    public String nextToken(String delim);
}
```


java.util: class Vector


```
public class Vector extends Object implements Cloneable {
    // public constructors
    public Vector(int initialCapacity,int capacityIncrement);
    public Vector(int initialCapacity);
    public Vector();
    // protected instance variables
    protected int capacityIncrement;
    protected int elementCount;
    protected Object[] elementData;
    // public instance methods
    public final synchronized void addElement(Object obj);
    public final int capacity();
    public synchronized Object clone(); // overrides Object.clone()
    public final boolean contains(Object elem);
    public final synchronized void copyInto(Object[] anArray);
    public final synchronized Object elementAt(int index)
        throws ArrayIndexOutOfBoundsException;
    public final synchronized Enumeration elements();
    public final synchronized void ensureCapacity(int minCapacity);
    public final synchronized Object firstElement()
        throws ArrayIndexOutOfBoundsException;
    public final int indexOf(Object elem);
    public final synchronized int indexOf(Object elem,int index);
    public final synchronized void insertElementAt(Object obj,int index)
        throws ArrayIndexOutOfBoundsException;
    public final boolean isEmpty();
    public final synchronized Object lastElement()
        throws NoSuchElementException;
    public final int lastIndexOf(Object elem);
    public final synchronized int lastIndexOf(Object elem,int index);
    public final synchronized void removeAllElements();
    public final synchronized boolean removeElement(Object obj);
    public final synchronized void removeElementAt(int index)
        throws ArrayIndexOutOfBoundsException;
    public final synchronized void setElementAt(Object obj,int index)
        throws ArrayIndexOutOfBoundsException;
    public final synchronized void setSize(int newSize);
    public final synchronized String toString(); // overrides Object.toString()
    public final synchronized void trimToSize();
}
```


java.util: class Stack


```
public class Stack extends Vector {
    // default constructor: public Stack()
    // public instance methods
    public boolean empty();
    public Object peek()
        throws EmptyStackException;
    public Object pop()
        throws EmptyStackException;
    public Object push(Object item);
    public int search(Object o);
}
```

Browser Source: hotjava.java Excerpts


```
/*
 * @(#)hotjava.java     1.181 95/05/15 Jonathan Payne and Patrick Naughton
 *
 * Copyright (c) 1994 Sun Microsystems, Inc. All Rights Reserved.
 *
 * Permission to use, copy, modify, and distribute this software
 * and its documentation for NON-COMMERCIAL purposes and without
 * fee is hereby granted provided that this copyright notice
 * appears in all copies. Please refer to the file "copyright.html"
 * for further important copyright and licensing information.
 *
 * SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF
 * THE SOFTWARE, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
 * TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 * PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SUN SHALL NOT BE LIABLE FOR
 * ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR
 * DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
 */

package browser;

import awt.*;
import net.*;
import java.lang.*;
import java.io.*;
import java.util.*;
import net.www.html.URL;
import net.www.html.MeteredStream;
import net.www.html.ProgressReport;
import net.www.http.HttpClient;
import net.ftp.FtpClient;
import browser.tools.JavaSearch.SearchWindow;

/**
 * Class hotjava is really an awt.Frame, with code to create and
 * manage all the UI components of hotjava.
 * @version 1.181, 15 May 1995
 * @author Jonathan Payne
 * @author Sami Shaio
 * @author Arthur van Hoff
 */
public class hotjava extends Frame {
    static private boolean      securityConfigured = false;
```



```
    static WRAccessHandler      handler = null;
    static WServer      server = null;
    static Font         labelFont;
    static Font         inputFont;
    static Font         dialogFont;
    static Color        blue = null;
    static SecurityDialog       sDialog = null;
    static ProgressDialog       pDialog = null;
    public static Color anchorColor = null;
    public static Color visitedAnchorColor = null;
    public static Properties    props;
    static int          copiesRunning;
    public static String home;
    public static String dochome;
    public static String version = "1.0 alpha3";
    public static String programName = "HotJava";
    public static String untitledTitle = "Untitled";
    public static URLHistory    history;

    public static int windowWidth  = 641;
    public static int windowHeight = 729;

    StatusWindow        statusWindow;
    ToolBar             toolBar;
    InfoWindow          infoWindow;
    SearchWindow        searchWindow;
    HistoryWindow       histWindow;
    StringDialog        findDialog;

    MenuBar             mbar;
    public FileMenu     fileMenu;
    GoToMenu            gotoMenu;
    protected WRWindow  doc;
    HotList             hlist;
    String              urlSpec;
    String              pageTitle;
    Image               icon;

    public static GifImage fetchGIF(String fromURL) {
        try {
            URL imgUrl = new URL(null, fromURL);
            InputStream iStream = imgUrl.openStream();
            return new GifImage(iStream, null);
        } catch (Exception e) {
            return null;
```



```
        }
    }

    public void go(String u) {
        doc.pushURL(new URL(null, u));
    }

    public void goHome() {
        doc.pushURL(new URL(null, home));
    }

    public void setTitle(String t) {
        if (t == null || t.equals("")) {
            super.setTitle(programName + " " + version);
            pageTitle = "";
        } else {
            super.setTitle(programName+": "+t);
            pageTitle = t;
        }
    }

    public void setURL(String u) {
        infoWindow.urlField.setText(u);
    }


    public void handleQuit() {
        if (--hotjava.copiesRunning > 0) {
            // XXX: should actually be destroying the window and
            // recovering resources...
            unMap();
        } else {
            System.exit(0);
        }
    }

    public hotjava(WServer serv, String args[]) {
        super(serv, true, null, windowWidth, windowHeight, Color.lightGray);
        setInsets(4,0,4,0);
        copiesRunning++;
        super.setTitle(programName);
        urlSpec = args.length > 0 ? args[0] : home;


        // Read properties and set defaults
```



```
        readProperties();

        if (blue == null) {
            blue = Color.black;
            anchorColor = new Color(serv, 0, 0, 192);
            visitedAnchorColor = new Color(serv, 96, 32, 128);
            labelFont = serv.fonts.getFont("Helvetica", Font.BOLD, 14);
            inputFont = serv.fonts.getFont("DialogInput", Font.BOLD, 12);
        }

        dialogFont = serv.fonts.getFont("Dialog", Font.BOLD, 12);
        setDefaultFont(dialogFont);
        doc = new WRWindow(this, "Center");
        infoWindow = new InfoWindow(this);
        doc.setScrollbar(new Scrollbar(doc, "sbar", Scrollbar.VERTICAL, true));
        statusWindow = new StatusWindow(this);
        toolBar = statusWindow.getToolBar();

        mbar = new MenuBar(this);
        fileMenu = new FileMenu(mbar, this);
        new OptionsMenu(mbar, this);
        new NavigateMenu(mbar, this);
        gotoMenu = new GoToMenu(mbar, this);
        new HelpMenu(mbar, this);

        hlist = new HotList(serv, this);
        histWindow = new HistoryWindow(serv, this, doc.docHistory);
        findDialog = new StringDialog(this, hotjava.dialogFont,
                                      "Find", "Find what:",
                                      false, false, "Find", "Cancel",
                                      "Help", "", null);

        // Set the HotJava icon
        // XXX: Note that this needs to be done before the frame is
        // mapped for it to take effect.
        //
        serv.sync();

        try {
            GifImage gif = hotjava.fetchGIF("doc:demo/images/wricon.gif");
            icon = doc.createImage(gif);
            setIconImage(icon);
        } catch (Exception e) {
            System.out.println("Couldn't set icon image");
        }
```

```
        if (sDialog == null) {
            sDialog = new SecurityDialog(this);
        }
        if (pDialog == null) {
            pDialog = new ProgressDialog(this);
        }

        map();

        if (! securityConfigured) {
            sDialog.map();
            sDialog.resize();
        }

        URL current_url = null;
        try {
            current_url = new URL(current_url, urlSpec);
            doc.pushURL(current_url);
        } catch(Exception e) {
            System.out.println("Ignoring error " + e);
        }
    }

    public static void main(String args[]) {
        try {
            File.setReadACL(System.getenv("HOTJAVA_READ_PATH"));
            File.setWriteACL(System.getenv("HOTJAVA_WRITE_PATH"));
        } catch (IOException e) {
            System.out.println("Couldn't initialize security ACLs.");
            return;
        }
        Firewall.readFirewallHosts();
        if (!Firewall.readAccessMode()) {
            securityConfigured = false;
        } else {
            securityConfigured = true;
        }
        try {
            server = new WServer();
        } catch(Exception e) {
            System.out.println("Couldn't open connection to window server");
            return;
        }
        server.start();
```



```
            hotjava wr = new hotjava(server, args);
            handler = new WRAccessHandler(wr);
            File.setAccessErrorHandler(handler);
            Firewall.setHandler(new WRSecurityHandler(wr));
        }

}

class StopButton extends Button {
    hotjava target;

    public StopButton(Container w, hotjava mw) {
        super("Stop", null, w);
        target = mw;
    }

    public StopButton(Image i, Image iP, Image dis, Container w,
                      hotjava mw) {
        super(i, iP, dis, null, w);
        target = mw;
    }

    public void selected(Component c, int pos) {
        target.doc.stopFetch();
    }
}

class HotlistMenuItem extends MenuItem {
    String url;
    hotjava mw;

    public HotlistMenuItem(String title, String url, Menu parent,
                           hotjava mw) {
        super(title, parent);
        this.url = url;
        this.mw = mw;
    }
    public void selected() {
        mw.doc.pushURL(new URL(null, url));
    }
}
```


Next Lecture

Guest: Gene Nelson, AT&T Bell Labs

Software & Intellectual Property