Applets
and
Applications

- Application
- Applet
- HTML for applets
- class Applet
- Inevitable HelloWorld example
- More simple applet examples
- JApplet

- (Practical alternative to Applets – Java Web Start)

Applets and Applications

- Application
  - independent program
  - full access to host machine
    - normal file access (standard security constraints on file ownership)
    - ability to create sockets (arbitrary network connections)
    - (slightly restricted) access to environment variables
  - GUI
    - create Frame (or JFrame) object as principal window
    - may have additional windows for dialogs, alerts etc

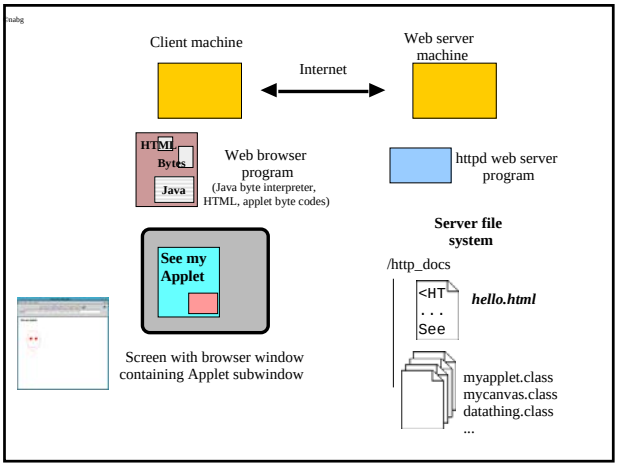
Applets and Applications

- Applet
  - program embedded in web page and run by "browser"
  - restricted access to machine on which it is run
    (restrictions can be varied by sophisticated browser)
    - no access to local files
    - can open network connection only to site from where applet was itself loaded
    - can access some details of enclosing web page
  - GUI
    - use part of browser's page window for main window

Applet

- Applets do have many restrictions
  - but they were the feature that popularised Java

  - ***downloadable executable content***
  - ***security policy easily enforced by browser environment***
  - ***lots more pretty GUI features than you could readily obtain with just HTML+forms***
  - ***real computations, not just checks on data entry & trivial calculations as in Javascript***



- Application
- Applet
- HTML for applets
- class Applet
- Inevitable HelloWorld example
- More simple applet examples
- JApplet

- (Practical alternative to Applets – Java Web Start)

Applet -HTML

- HTML "code" on web page contains tag specifying applet (**<applet ...**
  - somewhat like a **<a href ... >** link, contains name of file with code for applet class, (and dimensions of subwindow required)
  - optionally contains
    - specification of position of window (relative to surrounding text)
    - ...
  - associated with <applet ...> tag may have parameter tags that provide "command line" and "environment" data

Current situation on HTML

- If only to work with
  - Mozilla etc – use "embed" tag
  - IE – use "object" tag
- These (complex) tags have parameters that include directions on how to download and install JVM if it not already in browser
- If to work generally – use Applet tag
  - but then only works if JVM already installed and enabled (if you installed JDK, it will have asked to add JVM plugin to your browser)
- See

http://java.sun.com/docs/books/tutorial/deployment/applet/deployindex.html

IE

- Typical security settings for IE will result in a popup saying some content was disabled, click on task bar for more details
- Then can give permission for Applet to load and run

Applet - loading & starting

- Browser reads HTML text
  - when encounters "**applet**" tag
    - open connection to server
    - fetch byte codes (code for "myapplet.class")
    - byte codes passed to Java interpreter where get validated by class loader
  - when page loading complete
    - browser starts Java interpreter
    - interpreter opens additional connections to fetch other classes, each loaded by class loader
    - applet object starts to run

Using Applets

- As well as writing **.**java files with source, you must prepare a HTML file.
- You have to run via a browser
  - AppletViewer
    - simple, fast to load, ignores all content of HTML file except the applet (best for testing)
  - Browser + Sun plugin
    - current version of Java

HTML for Applets

```
<html> <head>
<title>Applet demo</title> </head>
<body>
See my applet
<applet code="MyApplet.class"
    ...  ...  ....
>
Your browser is Java challenged, modernise!
</applet>
</body> </html>
```

Text between <applet ... > and </applet> displayed if browser doesn't support applets.

HTML for Applets options in <applet

```
<applet code="MyApplet.class"
  width=200 height=120
  align=
 alt="someone has disabled Java in your browser"
 name=
 codebase=
 archive=
>
```

- **align** left, right, bottom, top, texttop, baseline ...
- **alt** (rare, catering for situation of Java aware browser with Java deactivated)

HTML for Applets options in <applet

- name
  - a page can contain more than one applet;
  - the different applets on a particular page can communicate (rare to want this, Horstmann gives example);
  - an applet's "name" is used (by another applet on same page) to get a reference object for communication.

options in <applet

- code and codebase
  - **code** name of file with applet class
  - **codebase**
    - normal to organize directories as follows
      - level 1, the .html page(s)
      - level 2,
        - images subdirectory (any pictures used)
        - applet1 subdirectory
        - applet2 subdirectory
        - ...
    - subdirectory identified by codebase

```
<applet code="MyApplet.class"  codebase="myapplet"
```

myapplet/MyApplet.class

options in <applet

- archive (*more on archives and JAR files later*)
  - pack many classes into an "archive" file
  - saves downloading time,
    - one connection to server, several classes fetched
  - archive argument allows you to specify name(s) of archive file(s)

Parameters for applet

- Parameters are (name, value) pairs; both name and value are strings.
- Uses are generally similar to environment variables (or command line arguments) for Unix programs
  - example, applet that displays pictures and plays back sound; same applet on several pages, with parameters specifying different picture and sound files
  - example, generated HTML page with parameters used to pass current data (eg exchange rate)

Parameters for applet

<applet code="picdisplay.class" width=500 height=300>
<param name="picfile" **value**="images/cat.gif">
<param name="picheight" value="100">
<param name="picwidth" value="150">
you have a java challenged browser, modernise
</applet>

- params placed between **<applet ...>** and **</applet>** tags

- Application
- Applet
- HTML for applets
- class Applet
- Inevitable HelloWorld example
- More simple applet examples
- JApplet

- (Practical alternative to Applets – Java Web Start)

class Applet

class MyApplet extends Applet/JApplet { ... }

An Applet is a
Panel is a
Container is a
Component is an
Object

Or extends JApplet if you want the modern "swing" graphics

Application

class MyApplication *extends Frame* { ... }

An "application" (with a GUI) is something that uses (or, sometimes, *is*) a Frame which is a
Window is a
Container is a
Component is an
Object

Application

- More typically

```
public class MyGui extends Frame/JFrame
  {}
public class MyApplication {
   static MyGui theGUI;
   public static void main(String[] args) {
      …
      theGUI = new MyGui(…);
      …
   }
```

class Applet

- Because an Applet is a "**Component**"
  - it is a basic GUI element in its own right
  - it has a paint() method, a repaint(), an update(), ...
- Because it is also a "**Container**"
  - it can have "subwindows", so can hold Canvases, Checkboxes, Buttons, ...
- And, because it is a "**Panel**", it has a defined way of arranging "subwindows"

Applet - execution control (these may be overridden in subclasses)

class Applet

- An Applet has its own unique functionality
  - init() — first method called when applet being started; typically does things like read parameters, load images, ...
  - start() — called after init() and each time user returns to HTML page with applet; typical use - starting "threads", ...
  - stop() — called each time user leaves HTML page; typical use - suspend "threads"
  - destroy() — free resources (doesn't Java do this? well, sometimes applet does have stuff to clean up) and kill threads

Applet - access "environment"

class Applet

- An Applet has its own unique functionality
  - getAppletContext() — returns an object which allows some communication with browser & HTML page
  - getParameter(String name) — returns "value" of parameter with "name" (or null)
  - getCodeBase(), getDocumentBase() — return URLs of page, etc
  - showStatus(String msg) — puts message in "status window" of browser

Applet - manipulate multimedia data

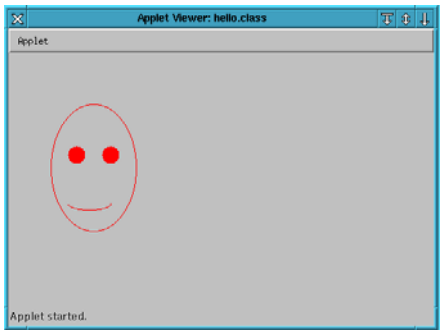
class Applet

- An Applet has its own unique functionality
  - getImage(URL src) — arranges to load an image (eg a .gif file) that is to be displayed in (a subwindow of) Applet
  - getAudioClip(URL src) — arranges to load a sound file
  - play(URL src) — plays sound file
- Image not loaded until try to display, then may get delays.
- Loading an Audio allows finer control, can start sound clip, stop it etc; play() simply plays it, or does nothing if cannot load.
- Multimedia features figure prominently in toy applets.

- Application
- Applet
- HTML for applets
- class Applet
- Inevitable HelloWorld example
- More simple applet examples
- JApplet

- (Practical alternative to Applets – Java Web Start)

The inevitable HelloWorld applet



HelloWorld applet

- All it does is
  - pick a color using <param> argument
  - draw in its “subwindow” within browser
- So,
  - a specialised subclass of Applet
    - has an init() method that reads parameter
    - has a paint(Graphics g) method that draws
    - accepts defaults (do nothing) for start(), stop(), ...

HTML file for Hello Applet

```
<html> <head>
<title>Hello Applet</title>
</head>
<body>
<h1>See my Applet</h1>
<applet code="hello.class" width=500 height=300>
<param name="color" value="red">
You are Java challenged!
</applet>
</body> </html>
```

class hello *extends Applet*

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

public class hello extends Applet {
   Color c = Color.black;
   public void init() {  ... }
   public void paint(Graphics g) { ... }
}
```

class hello *extends Applet*

```
public void init() {
    String colorval = getParameter("color");
    if(colorval == null) return;
    if(colorval.equals("red")) c= Color.red;
    if(colorval.equals("blue")) c= Color.blue;
}
```

class hello *extends Applet*

```
public void paint(Graphics g) {
   g.setColor(c);
   g.drawOval(50, 60, 100, 150);
   g.fillOval(70, 110, 20, 20);
   g.fillOval(110, 110, 20, 20);
   g.drawArc(70, 170, 50, 15, 0, -180);
   }
```

- create hello.html and hello.java
- javac hello.java
- appletviewer hello.html (NetBeans does this to test applet)

or

- communicator, open hello.html

or

- mozilla, open hello.html

Communicator (default browser on Unix) is quite old and has a number of limitations.
Mozilla causes unpleasant colour flickers on Sun Blade workstations

Problems testing applets …

- *Remember – supposed to be opening files across http style network.*
- *This sometimes causes problems when trying to test an applet and just using local files.*
- *Some operations don't work (browser applies security restrictions – so cannot get files).*
- *Tests with local files usually work better if use AppletViewer.*

Applet – constructor?

- Typical class
  - constructor of class does all work to initialize it (so would expect a GUI class to do setup and initialization in constructor)
- Typical Applet class (as in Sun's examples)
  - work is done in init() rather than constructor
- ?

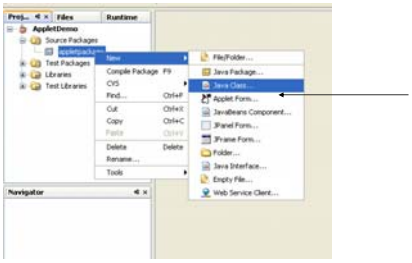
Applet constructor has limitations

- Reason is that some GUI elements cannot be initialized in constructor (most can)
  - Some odd features about setting up Applet environment
  - Environment not set right until constructor completed
  - Some GUI elements require interrogation of environment and cannot be created properly from in constructor
  - (Only those needing an image to be loaded??)

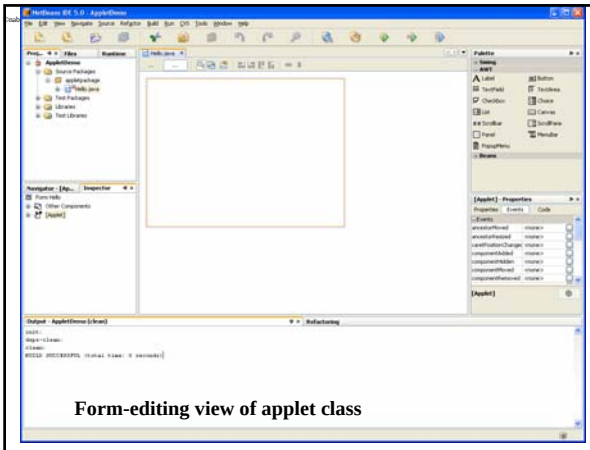
NetBeans version

First chance to meet the GUI builder

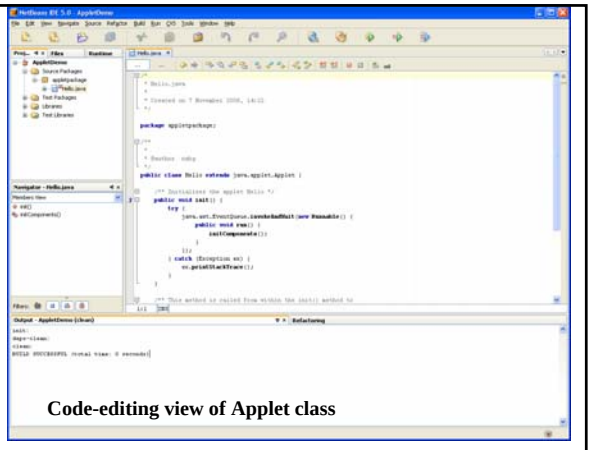
As usual, new Project
new Package
New File/ Applet Form



Pick "Applet Form"



Form-editing view of applet class



Code-editing view of Applet class

GUI editing : form-page/code-page

- (This style is common, if you get to work with Microsoft's Visual Studio you will see very similar operation.
  Actually, Microsoft really lead the development of visual editors starting way back in the late 1980s.)
- Two views – swap back and forth
  - Point and click editor, adding visual components and setting their properties
  - Code editor – see the resulting code

Hello Applet – NetBeans style

- Actually have now finished the GUI editing!
- (It isn't usually that simple.)
- This Applet has no subordinate GUI elements (no buttons, no text input fields etc etc)
- Still have to define the code to pick up the color parameter and to draw the little picture!
- Switch to text view.

Add some lines to "init()" supplied by NetBeans

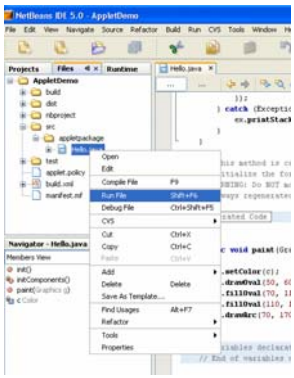
```
public class Hello extends java.applet.Applet {
    private Color c;
    /** Initializes the applet Hello */
    public void init() {
        System.out.println("Trying to get colorval");
        String colorval = getParameter("color");
        System.out.println("Colorval " + colorval);
        if(colorval == null) return;
        if(colorval.equals("red")) c= Color.red;
        if(colorval.equals("blue")) c= Color.blue;

        try {
            java.awt.EventQueue.invokeAndWait(new Runnable() {
                public void run() {
```

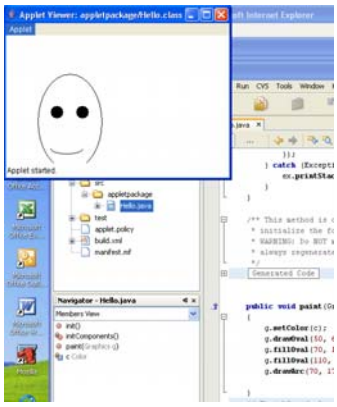
NetBeans will have generated some code in the init() function, add own code to read applet-params and set colour; add colour data member

Add the paint function

```
    public void paint(Graphics g)
    {
        g.setColor(c);
        g.drawOval(50, 60, 100, 150);
        g.fillOval(70, 110, 20, 20);
        g.fillOval(110, 110, 20, 20);
        g.drawArc(70, 170, 50, 15, 0, -180);
    }
```

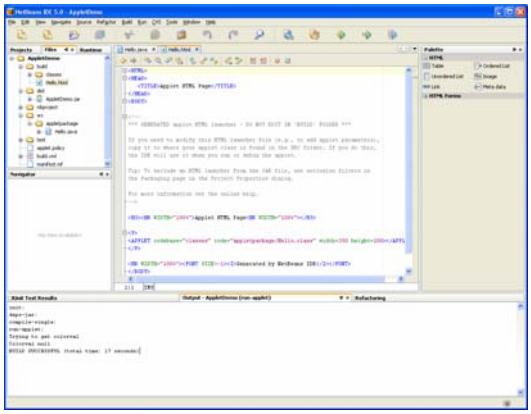
That little blue arrow?
It is a tag that NetBeans adds; it means "this function overrides the definition of the function in the base class"





Generated HTML file

- "Run file" caused NetBeans to generate a HTML file that has request for embedded applet,
  then ran this with AppletViewer
- Generated file is in "build" folder,
  copy and edit it if need parameters

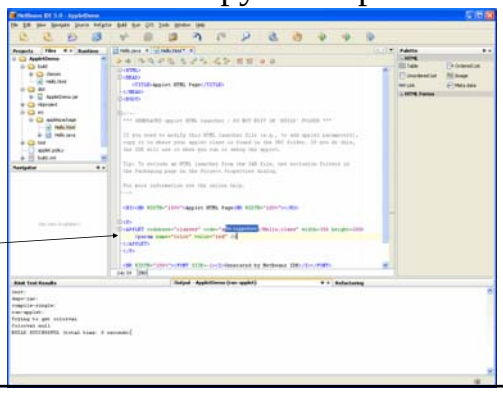


HTML

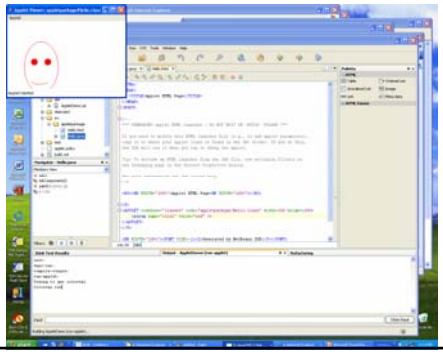
```
<H3><HR WIDTH="100%">Applet HTML Page<HR
    WIDTH="100%"></H3>

<P>
<APPLET codebase="classes" code="appletpackage/Hello.class"
    width=350 height=200></APPLET>
</P>

<HR WIDTH="100%"><FONT SIZE=-1><I>Generated by NetBeans
    IDE</I></FONT>
</BODY>
</HTML>
```

Need to add the param tag specifying a colour

Edit the copy – add params



… and run again

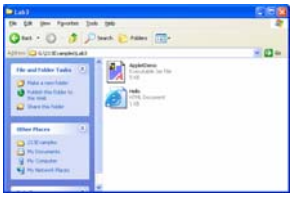


NetBeans applet outside of NetBeans

- Copy the generated .jar file
  - dist/AppletDemo.jar
- Compose a HTML page

HTML

```
<HTML><HEAD>
  <TITLE>My NetBeans Hello</TITLE>
</HEAD>
<BODY>
<H1><HR WIDTH="100%">Applet HTML Page<HR WIDTH="100%"></H1>
<p>My Hello applet will appear below:</p>
<P>
<APPLET archive="AppletDemo.jar"
   code="appletpackage/Hello.class"
   width=350 height=200>
   <param name="color" value="blue" />
</APPLET>
</P>
</BODY></HTML>
```



HTML pages

- Usually, HTML page displaying an applet will have lots of markup, content text, and maybe pictures as well as the applet.
- AppletViewer shows only the applet.
- You have to view page in browser if you want to see all the content.

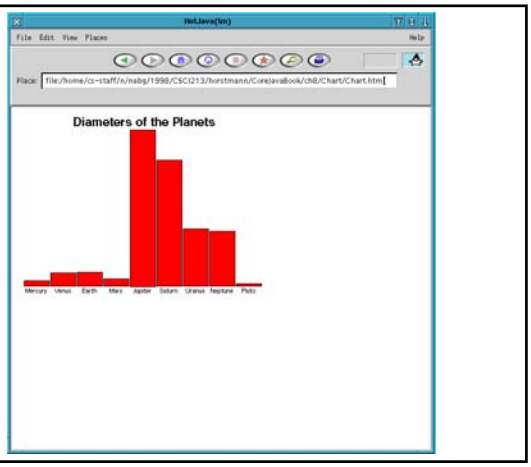
- Application
- Applet
- HTML for applets
- class Applet
- Inevitable HelloWorld example
- More simple applet examples
- JApplet

- (Practical alternative to Applets – Java Web Start)

More Applet examples

Horstmann's example applets

- Following examples from Horstmann, Core Java
  - chart applet
  - bookmark applet



Really nothing much more than the "HelloWorld" example

Horstmann's Chart Applet

- Illustration of use of <param ...>
- Applet
  - draw bar chart
    - named columns
    - relative heights defined by double values
- <param...> tags allow same applet to display different data on different pages
  - could use in generated page with <param ...> tags as output from some program

Minor mods to use standard i/o rather than Horstmann's library

```
<APPLET CODE="Chart.class" WIDTH=400 HEIGHT=300>
<PARAM NAME="title" VALUE="Diameters of the Planets">
<PARAM NAME="values" VALUE="9">
<PARAM NAME="name_1" VALUE="Mercury">
<PARAM NAME="name_2" VALUE="Venus">
...
<PARAM NAME="name_9" VALUE="Pluto">
<PARAM NAME="value_1" VALUE="3100">
<PARAM NAME="value_2" VALUE="7500">
<PARAM NAME="value_3" VALUE="8000">
<PARAM NAME="value_4" VALUE="4200">
...
<PARAM NAME="value_9" VALUE="1430">
</APPLET>
```

Chart.html

Chart.java

```
import java.awt.*;
import java.applet.*;
import java.io.*;
public class Chart extends Applet
{
   public void init()   { ... }
  public void paint(Graphics g) {  ... }
  private double[ ] values;
  private String[ ] names;
  private String title;
}
```

```
public void init()
{  int n =  Integer.parseInt(getParameter("values").trim());
     values = new double[n];
     names = new String[n];
     title = getParameter("title");
     int i;
     for (i = 0; i < n; i++)
     {  String s = getParameter("value_" + (i + 1));
        values[i]
          = Double.valueOf(s.trim()).doubleValue();
        names[i] = getParameter("name_" + (i + 1));
     }
   }
```

Note "standard" approach to dealing with variable number of <param ... > arguments

```
public void paint(Graphics g){
    // find range of values to be plotted
    // select font, display title
    // choose scale for plotting items
   // loop to draw each item, and place item label
}
```

```
public void paint(Graphics g){
     int i;
     int n = Integer.parseInt(getParameter("values").trim());
     double minValue = 0;
     double maxValue = 0;
     for (i = 0; i < values.length; i++)
     {  if (minValue > values[i]) minValue = values[i];
        if (maxValue < values[i]) maxValue = values[i];
     }

     Dimension d = getSize();
     int clientWidth = d.width;
     int clientHeight = d.height;
     int barWidth = clientWidth / n;
```

```
public void paint(Graphics g){
    ...
    int barWidth = clientWidth / n;
    Font titleFont = new Font("Helvetica", Font.BOLD, 20);
    FontMetrics titleFontMetrics  = g.getFontMetrics(titleFont);
    Font labelFont = new Font("Helvetica", Font.PLAIN, 10);
    FontMetrics labelFontMetrics   = g.getFontMetrics(labelFont);
    int titleWidth = titleFontMetrics.stringWidth(title);
    int y = titleFontMetrics.getAscent();
    int x = (clientWidth - titleWidth) / 2;
    g.setFont(titleFont); g.drawString(title, x, y);
    int top = titleFontMetrics.getHeight();
    int bottom = labelFontMetrics.getHeight();
```

```
public void paint(Graphics g){
     ...
    int bottom = labelFontMetrics.getHeight();
    if (maxValue == minValue) return;
    double scale = (clientHeight - top - bottom)
         / (maxValue - minValue);
    y = clientHeight - labelFontMetrics.getDescent();
    g.setFont(labelFont);
    for (i = 0; i < n; i++) {   ... }
    }
```

```
public void paint(Graphics g){
    ...
  for (i = 0; i < n; i++)
  {  int x1 = i * barWidth + 1; int y1 = top;
     int height = (int)(values[i] * scale);
     if (values[i] >= 0)  y1 += (int)((maxValue - values[i]) * scale);
       else  {  y1 += (int)(maxValue * scale); height = -height; }
     g.setColor(Color.red);  g.fillRect(x1, y1, barWidth - 2, height);
     g.setColor(Color.black); g.drawRect(x1, y1, barWidth - 2, height);
     int labelWidth
         = labelFontMetrics.stringWidth(names[i]);
     x = i * barWidth + (barWidth - labelWidth) / 2;
     g.drawString(names[i], x, y);
    }
  }
```

Horstmann's bookmark applet

- Illustrates communication with AppletContext
  - request that AppletContext display a different page (display is in frame, could have used a separate window)
- Multiple HTML files
  - frameset using two frames
  - contents for frames


```
<HTML>  <HEAD>
<TITLE>Bookmark Applet</TITLE>
</HEAD>
<FRAMESET COLS="320,*">
<FRAME NAME="left" SRC="Left.html" MARGINHEIGHT=2
    MARGINWIDTH=2
  SCROLLING = "no" NORESIZE>
<FRAME NAME="right" SRC="Right.html" MARGINHEIGHT=2
    MARGINWIDTH=2
  SCROLLING = "yes" NORESIZE>
</FRAMESET>
</HTML>
```

Bookmark.java (HTML file that defines "frameset" with contents Left.html and Right.html)


```
<HTML>
<TITLE>A Bookmark Applet</TITLE>
<BODY>
Double-click on one of the names in the list box. The corresponding web page
will be displayed in the frame on the right.
<P>
<APPLET CODE="Bookmark.class" WIDTH=290 HEIGHT=300>
<PARAM NAME=link_1 VALUE="http://java.sun.com">
...
<PARAM NAME=link_6 VALUE="http://www.inquiry.com">
<PARAM NAME=link_7 VALUE="http://www.netscape.com">
</APPLET>
</BODY>
</HTML>
```

Left.html (HTML file that contains <applet ... > and associated <param ...>s)


```
<HTML>
<TITLE>
Web pages will be displayed here.
</TITLE>
<BODY>
Double-click on one of the names in the list box to the left. The web page
will be displayed here.
</BODY>
</HTML>
```

Right.html (HTML file that contains initial filler for right hand frame)


```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.net.*;
import java.io.*;
public class Bookmark extends Applet implements ActionListener{
  public void init() {  ... }
  public void actionPerformed(ActionEvent evt)   {    ... }

  private List links = new List(10, false);
}
```

Bookmark.java

- A List GUI element can respond to a double click by reporting an ActionEvent
  - something has to handle that event
  - might as well be the Applet, no need for any other object

```
public void init(){
    setLayout(new BorderLayout());
    add("Center", links);
    links.addActionListener(this);
    int i = 1;
    String s;
    while ((s = getParameter("link_" + i)) != null) {  links.add(s);  i++; }
}
public void actionPerformed(ActionEvent evt) {
    String arg = evt.getActionCommand();
     try
      {  AppletContext context = getAppletContext();
        URL u = new URL((String)arg);
        context.showDocument(u, "right");
      } catch(Exception e) {  showStatus("Error " + e); }
}
```

Horstman's bookmark applet

- Defined only
  - init
    - This created a java.awt.List with data from Applet parmaters
  - actionPerformed
    - This lets applet talk to applet-context (i.e. browser) asking for display of another page
- How did it get to paint the list? No paint() method defined!

Applet.paint()

- Applet is a Panel which is a Container
- Container has a default paint method
  - For each component that I contain, tell contained component to paint itself
- This Applet contains a java.awt.List

```
private List links = new List(10, false);
   …
   add("Center", links);
```

java.awt.List.paint()

- The java.awt.List class defines a paint method
  - Use a scrollable pane, draw scrollbar if necessary
  - Draw each entry (String) on a line in scrollable pane
- So didn't need to redefine paint in the applet itself, the default inherited methods do whatever is necessary.

Chooser Applet

Applet with interactive controls

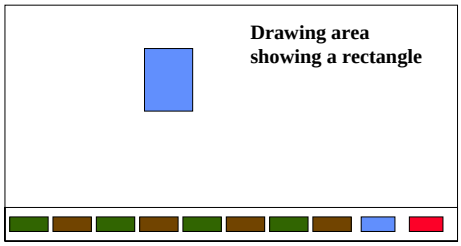
Chooser Applet

- Most GUI interfaces, and many Applets, use standard controls that let user select processing operations.
- Following simple applet illustrates some of these controls and also shows off more of NetBeans GUI Builder

Chooser Applet

- Applet is to show
  - Rectangle or Oval
  - Top/left, width/height adjustable
- Applet is to have
  - drawing area where shape displayed
  - Control area
    - Choice of Rectangle/Oval
    - Labels and text fields for dimensions
    - Action button

Rough view



Controls in panel at bottom

GUI components

- This time Applet isn't simply a single panel used to display data (as it has been in all previous examples)
- As a Panel (a kind of Container) it can hold other GUI components.
- Here, want it to hold
  - A "Canvas" – a region reserved for drawing data object
  - A collection of text input fields (with associated labels)
    - Fields for top, left, width, and height
  - A button which will cause applet to change data and display

GUI classes

- Need to use
  - Standard GUI components
    - Button
    - Label
    - TextField
    - Choice (pop-up list offering "Rectangle"/"Oval")
    - Panel
  - Customized GUI components
    - MyCanvas extends Canvas
  - Layout manager (arranges how components displayed in Applet)

GUI components

- Topic of next main lecture segment!
- For now
  - **Label**
    - Create Label object with String
    - Place in GUI
  - **TextField**
    - Editable text – automatically looks after all keyboard and mouse editing action
    - Can ask a TextField for its current text when needed
  - **Button**
    - Generates action events

GUI components

- GUI components: Label, TextField, Button, and Choice
- Choice
  - Pop-up list, lets you pick one out of a set of String values
  - Create choice object
  - Place in GUI
  - Add option, add option, add option …

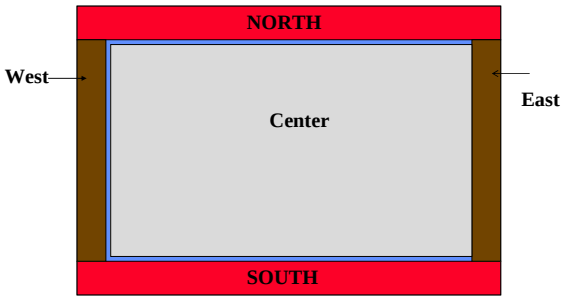
Customized GUI component

- Canvas
  - Reserve an area of GUI for display of application data
  - paint() – uhm, paint what?
- Have to create a subclass of Canvas to have anything useful!
- Class MyCanvas extends Canvas
  - Has link to data object
  - paint() function can arrange for painting of data object

BorderLayout

- Will be dealing with layout managers shortly.
- BorderLayout is convenient for simple applications
  - Divides Applet (or other container) into upto five regions (don't have to use them all)
    - "Center, North, East, South, West"
  - Each region can hold one other component – Canvas, Button, Label

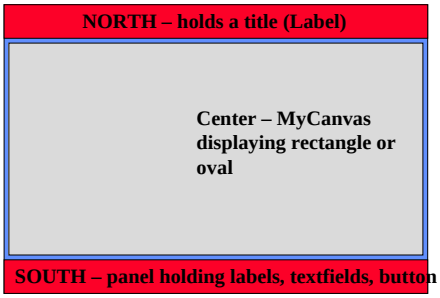
BorderLayout



Extra panels

- BorderLayout allows for one component in each of its regions
- We need several
  - Label and text-field for "top"
  - Label and text-field for "left"
  - …
  - Choice
  - Action button
- Have to use extra "Panel" objects
  - Place Panel in "south" region
  - Add labels and text-fields etc to this panel

BorderLayout as used



Event listening

- Button will generate ActionEvents
- Applet will listen for the ActionEvent
  - Handle it by changing data and arranging for update of display
- Applet creates the button in its init() method, will add itself as the action-listener once button created.

Event listening – the usual …

- GUI interface
- Component that generates events (Button)
- Defined class that will implement appropriate listener so that it can handle the event (Applet class will implement ActionListener)
- Initialization code will link listener to event source
- You should be getting the idea of event listening by now

Classes

- Data
  - Owns some Shape
  - Does
    - Gets told to draw its shape or change its shape
- MyCanvas
  - GUI component – occupies space in GUI where draw data
  - Simply owns link to data and passes on paint requests
- ChooserApplet

MyCanvas

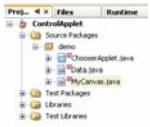
```
public class MyCanvas extends Canvas {
   private Data myData;
   public MyCanvas(Data d) {
      myData = d;
   }

   public void paint(Graphics g) { myData.paint(g); }
}
```

Data

```
public class Data {
   private Rectangle myShape;
   private boolean oval;
   public Data() {
      myShape = new Rectangle(40,80,100,50);
      oval = false;
   }
   public void makeRectangle(int top, int left, int width, int height)  {
      myShape = new Rectangle(top, left, width, height);
      oval = false;
   }
   public void makeOval(int top, int left, int width, int height)  {
      myShape = new Rectangle(top, left, width, height);
      oval = true;
   }
   public void paint(Graphics g)  {
      if(oval)
         g.fillOval(myShape.y, myShape.x, myShape.width, myShape.height);
      else
         g.fillRect(myShape.y, myShape.x, myShape.width, myShape.height);
   }
```

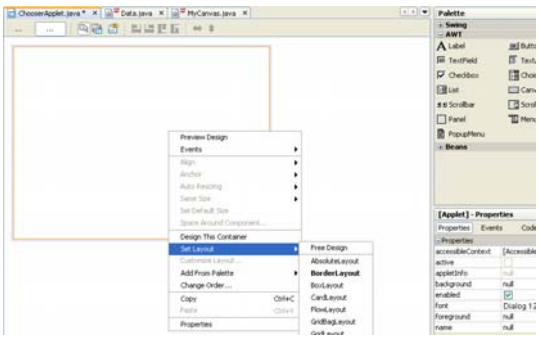
NetBeans

- New Project
  - New Package
    - New Java class MyCanvas
    - New Java class Data
    - New Applet Form ChooserApplet



Applet

- Create the applet class
- In “form” view, click the outline and select setLayout,
  choose BorderLayout



Add parts

- Label at top
  - Palette/AWT
    - Drag Label into top of outlined applet panel

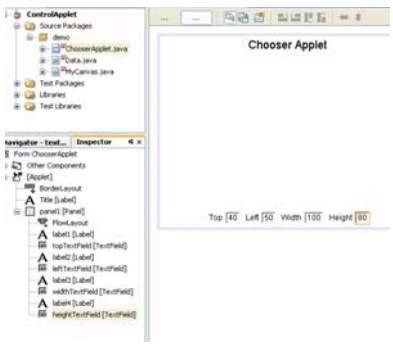
Fix up properties

- Select label
- Go to properties subwindow
  - Change Font (larger size, …)
  - Change Text (“Chooser Applet”)
  - Change Alignment (center)

  - Also change Name of variable

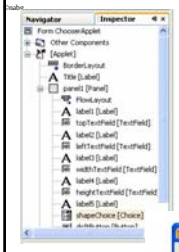
“South”

- From AWT add Panel to South
- Then add to the panel:
  - Pair (Label, TextField) for each of inputs
    - Fix variable names to make them meaningful

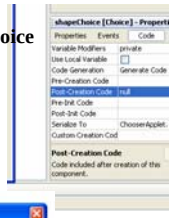


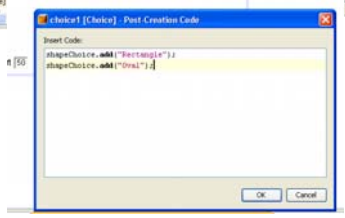
Adding code

- Properties box of component also has a “code” tab;
  - Can specify extra code to be inserted before code that adds component
  - Can specify different specialized code for adding a component
  - Can specify extra code to be inserted after code that adds component
- For “Choice” component –
  - Have to add the Choice object
  - Then in “Post creation code” need to add the Strings



Added "Choice" to Panel
Changed name to shapeChoice
Used properties panel to set font etc;
Then switched to "Code" section of properties
Added code





ActionButton

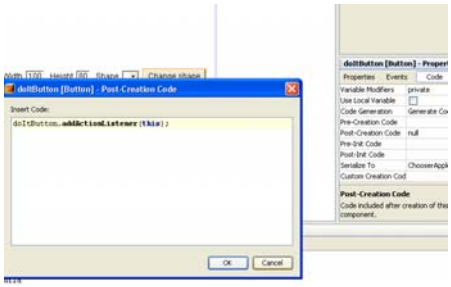
- Similarly add Button
- But now want something special – a listener will be attached to this button

- It is going to be the Applet
- Need extra code – something other than routine; "addActionListener(this)"

Adding code

- Here need extra code after –
  - Add actionListener
  - Specify "this" (i.e. the applet) as the action listener

- Note – will result in code that is erroneous – we haven't yet specified that the applet is to implement ActionListener interface; will fix shortly



MyCanvas

- Main central region of Applet is to show a MyCanvas where will draw the data.

- AWT palette doesn't have "MyCanvas"

- AWT palette does have Canvas!
  - Add a Canvas
  - Use code properties to customize creation code

Switch to code view

- Generated code includes an initComponents function
  - Shown collapsed, with a “do not Edit” warning
- Open collapsed view
  - See errors
    - addActionListener(this) when ChooserApplet is not an ActionListener
    - new MyCanvas(theData) when theData not defined

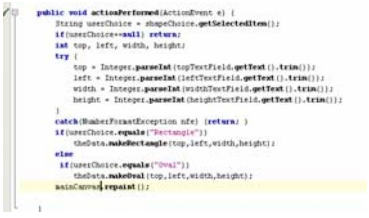
Fix-ups

- Add implements ActionListener to class
  - Fix imports
  - Then tell NetBeans to implement all declared methods
  - Get an empty actionPerformed() function added
- Declare a private data member of class Data and create instance in init() function (before call to initComponents())

Implement actionPerformed

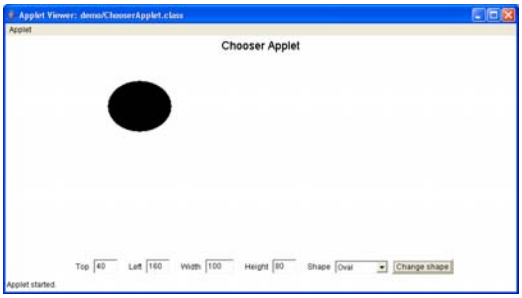
- When Button pressed
  - Check that one of items in choice has been chosen
    - If none chosen, do nothing
  - Read data values from each of TextFields and convert to integers
    - If any input conversion errors, do nothing
  - Invoke operation on data object to set new shape and dimensions
  - Request update of “My Canvas” part of display

Action performed …



Run the Applet

- Run file ChooserApplet
- Will need to resize appletviewer window



Event listening again

- So, typical event listening application
  - Button in GUI is source of ActionEvent events
  - ChooserApplet is a listener for action events
    - implements ActionListener
  - Listener object registered with event source
    - doItButton.addActionListener(this);
  - actionPerformed() function does some work when event occurs

Other events?

- Choice
  - Potentially source of "ItemEvent"s
    - Each time you change a selection you get Deselected then Selected events
- TextField
  - Potentially source of "TextEvent"s
    - Each change to one of number fields will result in event
- NOT listening for either TextEvents or ItemEvents

Usually only monitor action buttons

- Don't need to know that user is changing the Item selection
- Don't need to know details of each change made to text input fields
- Leave these controls to "look after themselves" while user is entering data
- Wait for user to complete input; user will then activate action button.

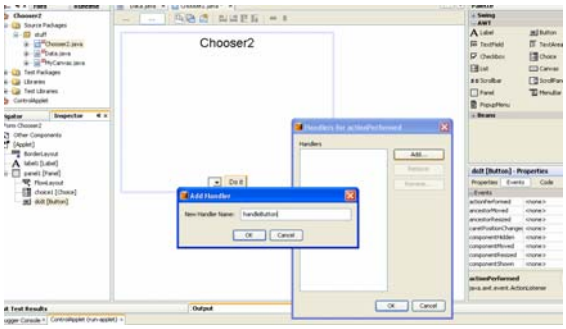
NetBeans auto-generation of event handling code

Event handling

- Normal
  - Make one of your classes implement appropriate listener
  - Add instance of class as listener to event source
- NetBeans has an option that automates part of code but uses a different approach

Chooser2 applet

- Simplified version of Chooser
  - I've just added a Choice and an Button to panel
  - I've selected "Events" from the properties panel associated with the Button
    - Picked Action events
  - Dialog (not very clear) is asking me to enter a name for a function that will be added to enclosing applet that will handle the action-event



Generated code … 1

```
/** Initializes the applet Chooser2 */
public void init() {
    try {
        java.awt.EventQueue.invokeAndWait(new Runnable() {
            public void run() {
                initComponents();
            }
        });
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}

/** This method is called from within the init() method to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
Generated Code

private void handleButton(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}
```

How does handleButton() get called? Magic!

- Well actually, look inside the initComponents and all will be revealed (sort of)

```
private void initComponents() {
    label1 = new java.awt.Label();
    panel1 = new java.awt.Panel();
    choice1 = new java.awt.Choice();
    choice1.add("Rectangle");
    choice1.add("Oval");
    doIt = new java.awt.Button();

    setLayout(new java.awt.BorderLayout());

    label1.setAlignment(java.awt.Label.CENTER);
    label1.setFont(new java.awt.Font("Arial", 0, 24));
    label1.setText("Chooser2");
    add(label1, java.awt.BorderLayout.NORTH);

    panel1.add(choice1);

    doIt.setLabel("Do it");
    doIt.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            handleButton(evt);
        }
    });

    panel1.add(doIt);

    add(panel1, java.awt.BorderLayout.SOUTH);

}// </editor-fold>
```

Anonymous inner class code --- yuk

```
doIt.setLabel("Do it");
doIt.addActionListener(new
   java.awt.event.ActionListener() {
        public void actionPerformed(
              java.awt.event.ActionEvent evt) {
           handleButton(evt);
        }
     });
```

Inner class code

- That stuff
  - Defines a new class with an actionPerformed method
  - This actionPerformed method calls the handleButton function in the enclosing Applet class
  - Creates an instance of this class and attaches it to the button
- Depends on obscure properties of inner classes in Java

Auto-code

- Personally, I don't like such code
  - I find inner classes and their conventions an unnecessary and confusing addition to Java
  - This approach forces the applet to be the listener for the button

Model-View-Control

- These applications have three aspects
  - Model (Data)
    - The data being manipulated
  - View
    - The GUI – views of application data, input fields for use by user
  - Control
    - Monitor the inputs (listen for those events that really need to handle)
    - Read and validate data inputs
    - Invoke processing operation

Model-View-Control : 3 separate aspects

- These are different aspects of application
- Best handled by three separate classes (or, usually, clusters of classes)
  - Model – main class owning all application data and performing application actions
  - View – main GUI class
  - Control – best placed in another class
- Main program creates model, GUI, and control object and links them up

NetBeans auto-coder: View=Control

- Auto-code approach forces Applet to be both view and control in application
  - Maybe OK in simple cases
  - Not a good idea in more advanced applications

NetBeans GUI editing

Is it worth it?

NetBeans GUI editing

- Is it worth it?
- For simple examples like this …
- No
  - It is easier to write the code for the GUI than fiddle with all those dialogs – especially the dialogs for adding custom code like “Post create” etc

NetBeans GUI editing : for complex GUIs

- With more complex GUI interfaces, the GUI editor approach is definitely advantageous.
- GUI editor may not help much (may even hinder) development of simple applications – but use it anyway to see how it works
- GUI editor very useful later with complex GUIs

Applet restrictions

Sandbox environment

- Java programs run with “security manager” objects
  - all calls to functions are interpreted, so have opportunity to hook in extra checking at time of call;
  - classes get tagged with information (‘class loader’ & ‘security manger’) when loaded; if there is a security manager, then this will be used to vet function calls;
  - security manager loaded for applets is by default very restrictive

No peeking by applets

- Applets cannot
  - access directories or files on machine that hosts browser
  - cannot “read pixels” from the screen
  - cannot open socket connections to machine other than one from which applet downloaded
  - …

Easing the restrictions

- Restrictions prevent things like a simple spreadsheet applet (you couldn't store your data)
- So more elaborate security system introduced.
- Applets – get security certificates (mechanism that allows browser to verify that applet really is from source specified)
- Applets that have such security certificates can negotiate with user (via dialogs)
  - can applet have permission to access files?
  - can applet open socket to xxx.yyy.zzz.aaa

Permissions for applets

- Allows applets to be used for more than simple demos and games

JApplet

- If you want to use swing classes (fancy things like JTable) then you must use JApplet rather than Applet
- One problem!
  - threads and swing!

Swing thread issue & JApplet

- Swing library is unchecked single threaded code
  - unchecked means no locks
- Swing code should only be executed by event handling thread
- init() function of Applet (JApplet) is not executed by event handling thread

invokeLater etc

- Strictly, you cannot build a JApplet interface in an init() method that adds JTextFields, JButtons etc
- Instead you have to use more elaborate code that creates tasks that will be invoked later by the event handling thread
- Most simple JApplet examples don't seem to do this
  - they work
  - almost always
  - (if you create the display in the constructor they might always work – provided you don't use images etc)

JApplet – you have been warned

- Cannot rely on simple JApplet interface building to always work
- See Sun JApplet examples for complex code that does interface building correctly

NetBeans and JApplets

- Naturally, NetBeans can generate the messy JApplet set up code
- So OK, to build swing style applets (JApplets) if using NetBeans

Applets? WebStart?

Applets

- Not used much in practice§
  - Permission system improved scope
  - But still rather impractical – hassles with JVMs in browsers, slow download times etc

§
Actually, I've noticed usage increasing in last couple of years. Numerous sites (most of newspapers for example) appear to load an Applet that tracks your usage and measures download times, sending the data back to web site – this Applet doesn't have any GUI.
SOLS, WebCT Vista – applet dependent.

WebStart

- Alternative technology for downloadable executables
  - Code (.class files, .jar archives) cached on client machine in directory managed by WebStart system
  - Link in HTML page to WebStart program
    - start a JVM on client machine
    - first task – check that cached version of code is up to date, if necessary download revised classes
    - run Java program separately from browser

Webstart -2

- Certification etc – can verify that really running the correct code, not something interposed by hacker
- Restrictions – can define access rights for a webstart program
- Full GUI, unrestricted by browser
- Combines
  - applet advantage of downloadable secure code
  - application advantages