Applications & Applets

- Applications
  - main method signature:

```
public static void main (final String [ ] args)
```

    - execution starts with main in *named class*
      - > 1 main () possible (but 1 main () per class)
  - command-line arguments:
    - passed as an array of strings
    - args.length

Applications & Applets

```
import java.util.*;

// file: Stack.java
// usage: java Stack
public class Stack
  {
  Vector v = new Vector ();
  public void push (Object o)
    {
    v.add (o);
    }

  public Object pop ()
    {
    return (v.remove (v.size () - 1));
    }

  public static void main (String [] args)
    {
    final String theContent = "Hello, World";
    Stack s = new Stack ();

    s.push (theContent);
    if ( ! ((String) s.pop ()).equals (theContent))
      {
      System.err.println ("pop failed!");
      System.exit (-1);
      }

    System.exit (0);
    }
  }
```

```
// file: DoSomethingWithAStack.java
// usage: java DoSomethingWithAStack name
class DoSomethingWithAStack
  {
  public static void main (String [] args)
    {
    Stack s = new Stack ();

    s.push (args [0]);
    s.push ("there");
    s.push ("Hello");

    System.out.print (s.pop ());
    System.out.print (" " + s.pop ());
    System.out.println (" " + s.pop ());
    System.exit (0);
    }
  }
```

Applications & Applets

- System.exit (n)
  - returns the value *n* to the calling shell
- System.exec (…);
  - runs another process under the OS
- can't directly access environment, font lists, etc.
  - System.getProperty as partial replacement
  - java.awt.Toolkit

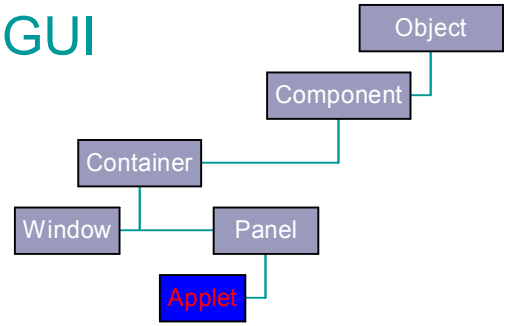
```
import java.awt.Toolkit;

public class X
  {
  public static void main (String [] args)
    {
    if (System.getProperty ("java.version").equals ("1.2.1"))
      Toolkit.getDefaultToolkit ().beep ();
    }
  }
```

Applications & Applets

```
import java.util.*;
import java.io.*;

public class SysIO
  {
  public static void main (String [] args)
    {
    PrintWriter out = new PrintWriter (System.out, true);
    out.println ("Begin.");
    try
      {
      String [] cmdArray =
        { "C:\\WINNT\\SYSTEM32\\CMD.EXE", "/C", "DIR", "/W", "C:\\Temp" };
      Process dir = Runtime.getRuntime ().exec (cmdArray);
      String s;
      BufferedReader r =
        new BufferedReader (new InputStreamReader (dir.getInputStream ()));
      out.println ("Reading...");
      while ((s = r.readLine ()) != null)
        out.println (s);
      out.println ("DONE: " + dir.exitValue ());
      }
    catch (Exception e)
      {
      e.printStackTrace (System.err);
      }
    out.println ("END.");
    }
  }
```

Applications & Applets

- Applets
  - "Executable content"
    - execute only in a supplied *context*
      - browser, appletviewer
    - dynamic
      - applet classes may be downloaded from server as referenced at execution time
  - Designed to "bring a page to life"
    - extends Java.awt.Panel
      - nothing 'special' needed to get a GUI
      - event driven



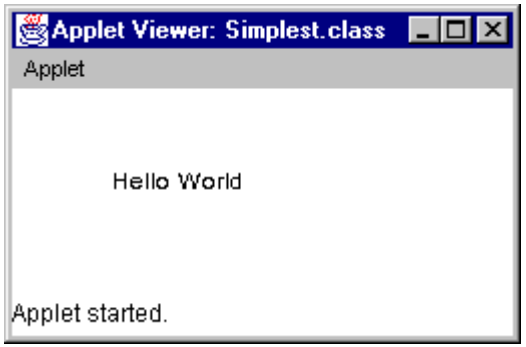
Applications & Applets

- The simplest applet

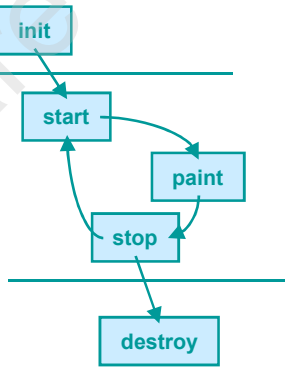
```
import java.awt.*;

public class Simplest extends java.applet.Applet
  {
  public void paint (Graphics g)
    {
    g.drawString ("Hello World", 50, 50);
    }
  }
```

- Embedded within HTML document

```
<APPLET CODE="Simplest.class" WIDTH=250 HEIGHT=100>
</APPLET>
```



Applications & Applets



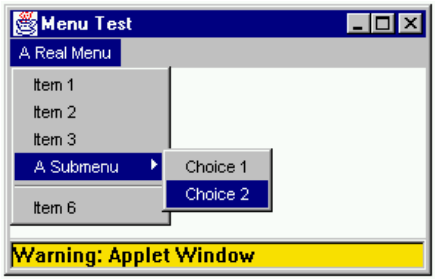
- Required to interact with it's context in a well-defined fashion

method (*ClassName*	**called when…** applet starts	**common use** constructor)
init	applet first starts —after constructor	initializes any data structures that paint will need
destroy	applet exits	reverses the actions of init
paint	window needs refreshing	redraw window's contents
start	browser enters applet window	starts the applet's visible activities (e.g. animation)
stop	browser goes to a new HTML page	stops the applet
getAppletInfo	N/A	returns info about the applet
getParameterInfo	N/A	returns info. about the applet's parameter set

- these methods should be overridden

Applications & Applets

- Applet security policies
  - (untrusted) applets are not allowed to:
    - access the local file system *in any way*
    - connect to any networked computer other than the originating one
    - listen or accept on a local port
    - create an 'untagged' window
    - load dynamic libraries
    - exit or exec
      - Sun asserts *"…exit might cause the context to exit…"*
    - use any of the sun.* packages
    - create a SecurityManager object
    - etc...



- Java 1.1 introduced trusted applets
  - context may lift/modify some of these restrictions

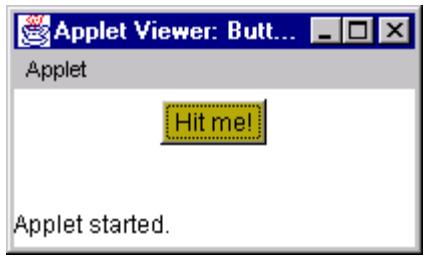
Applications & Applets

- Security is a very delicate issue
  - *"The trouble with applets is that security restrictions are getting tighter and tighter, while what you can do is becoming less and less significant. Every time…another hole, the noose tightens."*
  - *"…stuck in a sandbox."*
  - *"Java is becoming restricted to creating executable window dressing, not (useful) executable content."*

Applications & Applets

- Another (very) simple applet

```
import java.awt.*;
import java.awt.event.*;

public class ButtonTest extends java.applet.Applet
  {
  public void init ()
    {
    Button theButton = new Button ("Hit me!");
    theButton.addActionListener
      (
      new ActionListener ()
        {
        public void actionPerformed (ActionEvent ae)
          {
          Color c = new Color ((float) Math.random (),
                               (float) Math.random (), (float) Math.random ());
          ((Button) ae.getSource ()).setBackground (c);
          }
        }
      );
    add (theButton);
    }
  }
```



Applications & Applets

- getAppletContext ()
  - getApplet
  - getApplets
    - enumerate over all applets *on the same HTML page*
      - if given MAYSCRIPT attribute
  - getAudioClip
  - getImage
  - showDocument
  - showStatus
    - limited use
    - ‘fluff’ messages

Applications & Applets

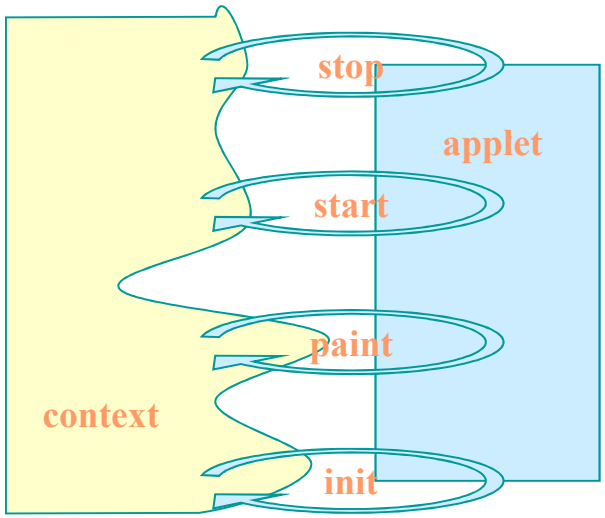
- Applets and JavaScript
  - JavaScript is *not* Java
    - formerly ‘LiveScript’
    - renamed for (dubious) marketing reasons

```
import java.awt.*;

public class Commune extends java.applet.Applet
  {
  public void red ()
    {
    setBackground (Color.red);
    repaint ();
    }

  public void blue ()
    {
    setBackground (Color.blue);
    repaint ();
    }
  }
```

```
<HTML>
<HEAD>
<TITLE>
Communing with JavaScript
</TITLE>

<SCRIPT LANGUAGE="JavaScript">
function red ()
  {
  document.Commune.red ()
  }

function blue ()
  {
  document.Commune.blue ()
  }
</SCRIPT>
</HEAD>

<APPLET CODE = "Commune.class" NAME = "Commune”
        WIDTH = 200 HEIGHT = 200 MAYSCRIPT>
</APPLET>

<FORM>
<INPUT TYPE = "button" VALUE = "Red" onclick = "red()">
<INPUT TYPE = "button" VALUE = "Blue" onclick = "blue()">
</FORM>
</HTML>
```

Applications & Applets

- Application versus applet
  - applications are standalone
    - lose security: access files, other applications, etc.
  - applets require established context:
    - HTML browser or appletviewer
      - "executable content"
    - more structured interaction with context
      - init, paint, etc.
    - tight security



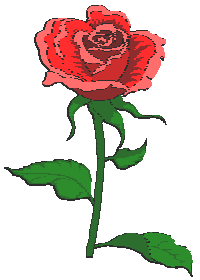
Applications & Applets

- A whole new phylum:
  - servlet
    - server-side Java extension to replace CGI
  - aglet
    - IBM's Java-based personal agent
  - app
    - applications
  - applet
    - executable content
  - Bean
    - embeddable component; ActiveX 'killer' technology
  - EJB
    - enterprise bean

Applications & Applets

- Dual-purpose applets/applications
  - a few ‘gotchas’ can get in the way of fully dual-natured code

```
public class MixedUp extends java.applet.Applet
  {
  public void init ()
    {
    …
    getAppletContext ().showStatus (message);
    …
    }

  public static void main (String [] args)
    {
    …
    new MixedUp ().init ();
    …
    }
  }
```



Applications & Applets

- *AppletStub*, *AppletContext* interfaces provide for dual personality

```
public class AppletFrame extends Frame implements AppletStub, AppletContext
    {
    ... (constructor on next page)
    //AppletStub methods
    public boolean isActive () { return true; }
    public URL getDocumentBase () { return null; }
    public URL getCodeBase () { return null; }
    public STring getParameter (String name) ( return ""; }
    public AppletContext getAppletContext () { return this; }
    public void appletResize (int w, int h) { }

    // AppletContext methods
    public AudioClip getAudioClip (URL url) { return null; }
    public Image getImage (URL url) { return null; }
    public Applet getApplet (String name) { return null; }
    public Enumeration getApplets () ( return null; }
    public void showDocument (URL url) { }
    public void showDocument (URL url, String target) { }
    public void showStatus (String status) { }
    }
```

define a "mini browser"

Applications & Applets

```
public class AppletFrame extends Frame
  implements AppletStub, AppletContext
    {
      AppletFrame (Applet a, int x, int y)
        {
        setTitle (a.getClass ().getName ());
        resize (x, y);
        add ("Center", a);
        a.setStub (this);
        a.init ();
        show ();
        a.start ();
        }
      … (as previous slide)
      }

public class CalculatorAppletApp extends CalculatorApplet
  {
  public static void main (final String [] args)
    {
    new AppletFrame (new CalculatorApplet (), 150, 10);
    }
  }
```