CSC 551: Web Programming

Spring 2004

Java Overview

- Design goals & features
  - ➢ platform independence, portable, secure, simple, object-oriented, …
- Programming models
  - ➢ applications vs. applets vs. servlets
  - ➢ intro to applets
    - – libraries, comments, classes, inheritance
    - – applet tag in HTML
    - – applet parameters

Java

Java was developed at Sun Microsystems, 1995

- originally designed for small, embedded systems in electronic appliances
- initial attempts used C++, but frustration at limitations/pitfalls

recall: C++ = C + OOP features
the desire for backward compatibility led to the retention of many bad features

desired features (from the Java white paper):

simple	object-oriented	robust
platform independent	architecture neutral	portable
dynamic	interpreted	high-performance
distributed	multi-threaded	secure

note: these are desirable features for any modern language
thus, Java has become very popular, especially when Internet related
also, Sun distributes free compilers (JDK) and open source

Language features

simple

- syntax is based on C++ (familiarity → easier transition for programmers)
- removed many confusing and/or rarely-used features
  e.g., explicit pointers, operator overloading, automatic coercions
- added memory management (reference count/garbage collection hybrid)

object-oriented

- OOP facilities similar C++, all methods are dynamically bound
- pure OOP – everything is a class, no independent functions*

robust

- lack of pointers and memory management avoids many headaches/errors
- libraries of useful, tested classes increases level of abstraction
  - arrays & strings are ADTs, well-defined interfaces

Language features (cont.)

platform independence

- want to be able to run Java code on multiple platforms
- neutrality is achieved by mixing compilation & interpretation
  1. Java programs are translated into *byte code* by a Java compiler
     - byte code is a generic machine code
  2. byte code is then executed by an interpreter (Java Virtual Machine)
     - must have a byte code interpreter for each hardware platform
- an Applet is a special form of Java application
  - byte code is downloaded with page, JVM is embedded in browser

architecture-neutral

- no implementation dependent features (e.g., size of primitive types is set)

portable

- byte code will run on any version of the Java Virtual Machine (JVM)

high-performance

- faster than traditional interpretation since byte code is "close" to native code
- still somewhat slower than a compiled language (e.g., C++)

Language features (cont.)

distributed

- extensive libraries for coping with TCP/IP protocols like HTTP & FTP
- Java applications can access remote URL's the same as local files

multi-threaded

- a *thread* is like a separate program, executing concurrently
- can write Java programs that deal with many tasks at once by defining multiple threads (same shared memory, but semi-independent execution)
- threads are important for multi-media, Web applications

secure

- Java applications do not have direct access to memory locations
  - memory accesses are virtual, mapped by JVM to physical locations
  - downloaded applets cannot open, read, or write local files
- JVM also verifies authenticity of classes as they are loaded
- *Sun claim: execution model enables virus-free*, tamper-free* systems*

Java programming models

Java *applications* are stand-alone programs

- must be compiled into Java byte code by Java compiler, then distributed
- executed by an interpreter (Java Virtual Machine)

Java *applets* provide for client-side programming

- compiled into Java byte code, then *downloaded as part of a Web page*
- executed by the JVM *embedded within the Web browser*

- unlike JavaScript, Java is full-featured with extensive library support
- Java and its APIs have become industry standards
  - the language definition is controlled by Sun, ensures compatibility
  - Applications Programming Interfaces standardize the behavior of useful classes and libraries of routines

Java *servlets* provide similar capabilities on the server-side

- alternative to CGI programs, more fully integrated into Web server

Java applets

important point: Java applets & applications look different!

- if you want to define a stand-alone application, make an application
  requires `public static void main` function, similar to C++ main
- if you want to embed the code in a Web page, make an applet
  requires `public void paint`, `public void init`, ...
- can define dual-purpose programs, but tricky

as with JavaScript, security is central

- when a Java applet is downloaded, the bytecode verifier of the JVM verifies to see if it contains bytecodes that open, read, write to local disk
- a Java applet can open a new window but they have Java logo to prevent them from being disguised as system window (e.g., to steal passwords)
- a Java applet is not allowed to connect back to other servers except the host
- this secure execution environment is called *sand box model*

First Java applet

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

/**
 * This class displays "Hello world!" on the applet window.
 */
public class HelloWorld extends Applet
{
  public void paint(Graphics g)
  {
      g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
  }
}
```

libraries: Java provides extensive library support in the form of classes

- libraries are loaded using import (similar to `#include` in C++)
  `java.awt`: contains Abstract Window Toolkit (for GUI classes & routines)
  `java.applet`: contains the applet class definition

comments: `//` and `/* */` work the same as in C++

- also have `/** */` which denote documentation comments (can be used to generate docs)

First Java applet

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

/**
 * This class displays "Hello world!" on the applet window.
 */
public class HelloWorld extends Applet
{
  public void paint(Graphics g)
  {
      g.drawString("Hello world!", 10, 10); // writes starting 10 pixels over & down
  }
}
```

class definitions in Java

- similar to C++ (but no semi-colon at end)
  - can contain instance variables (data fields) & methods(member functions)
  - precede class & method definitions with *public* to make available to all programs
- there are no stand-alone functions in Java*
- must be stored in a file of same name with .java extension
  - e.g., `HelloWorld.java`

First Java applet

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

/**
 * This class displays "Hello world!" on the applet window.
 */
public class HelloWorld extends Applet
{
  public void paint(Graphics g)
  {
      g.drawString("Hello world!", 10, 10);  // writes starting 10 pixels over & down
  }
}
```

all applets inherit from the Applet class (in java.applet)

default methods include:

- `init()`: called when page is loaded to create/initialize variables
  - *by default, does nothing*
- `paint(Graphics g)`: called to draw (after init) or redraw (after being obscured)
  - *here, the paint method is overridden to display text on the applet window*

Embedding an applet in HTML

to include an applet in a Web page, use either

- APPLET tag (deprecated)

  CODE specifies applet name, HEIGHT and WIDTH specify window size

  text between the APPLET tags is displayed if unable to execute (e.g., Java not enabled)

- OBJECT tag

  preferred for HTML 4, but not universally supported

```
<html>
<!-- Dave Reed    hello1.html    3/20/04 -->

<head>
  <title>Hello World Page</title>
</head>

<body>

<p>
  <applet code="HelloWorld.class" height=100 width=100>
    You must use a Java-enabled browser to view this applet.
  </applet>
</p>

</body>
</html>
```

view page in browser

HTML & applets

```
<html>
<!-- Dave Reed    hello2.html    3/20/04 -->

<head>
  <title>Hello World Page</title>
</head>

<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld.class" height=200 width=200>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>
</p>

</body>
</html>
```

view page in browser

an applet can be embedded within HTML elements just like any other element

useful for formatting and layout

Parameters in HTML

```
<html>
<!-- Dave Reed    hello3.html      3/20/04 -->

<head>
<title>Hello World Page</title>
</head>

<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld1.class" height=35 width=300>
  <param name="name" value="Chris">
  <param name="age" value=20>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>
</p>

</body>
</html>
```

view page in browser

can specify parameters to the APPLET when it is embedded in HTML

- each parameter must have its own PARAM tag inside the APPLET element
- specifies parameter name and value

Applet parameters

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

/**
 * This class displays a message based on parameters.
 */
public class HelloWorld1 extends Applet
{
    public void paint(Graphics g)
    {
        String userName = getParameter("name");
        int userAge = Integer.parseInt(getParameter("age"));

        String message1 = "Hello " + userName + ".";
        String message2 = "On your next birthday, you will be " +
                          (userAge+1) + " years old.";

        g.drawString(message1, 10, 10);
        g.drawString(message2, 10, 30);
    }
}
```

can access parameters passed in from the HTML document

`getParameter` accesses the value of the parameter (must know its name)

- if the parameter represents a number, must `parseInt` or `parseFloat`