Java Review

CSE 373
Data Structures

Java

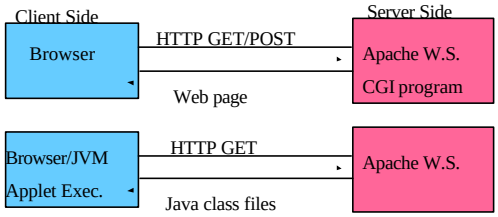
A Programming Language for Web-based Computing with Graphics

Language Features of Java

-- Garbage-collected
-- Support for Object-oriented programming
-- Support for packages
-- Compiles to intermediate code
-- Intermediate code is then interpreted
-- Built-in support for many data structures such as hash tables, vectors

Applets vs CGI

Differences: Java supports client-side processing in Web via "Applets"



More Features of Java

More Differences:

- **Security considerations, because of its web orientation: compulsory "try" and "catch" error handling.**
- **Stronger typing of variables than in, say, Lisp.**
- **Standard graphics API's: the AWT and Swing.**

Influences on Java

C, C++: syntax of arithmetic and boolean expressions, need for safe pointer operations.

Smalltalk, C++, CLOS: Object orientation

Lisp: garbage collection, "bignums"

Java's Web Support

- Applets: Java virtual machine can run in a browser.
- Safe pointers avoid segmentation faults and other dangerous errors.
- Security manager provides that applets don't perform I/O to client hard disk.
- Applets permitted only limited upload communication (to the originating server).
- Standard networking package is provided.

Java's Graphics

- AWT: The Abstract Windowing Toolkit is a package providing standard classes for building GUIs.
- SWING is a higher-level, more complex library of widgets for GUI construction.
- Support for decoding GIF and JPEG image formats is built in.
- Java has used two slightly different event models for user interaction: JDK 1.0 (old) and JDK 1.1 (new).
- Java2D is a more advanced imaging package that's made to work with Java 2.

Java's Threads

- Java programs can contain multiple threads of control.
- This can permit parallel processing.
- This can permit logical separation of the program into code for concurrent processes.
- Threads are especially useful in animation within an applet.
- Run-time scheduling of threads is not completely platform independent.

Example Application

```
class MyFirstApplication {
 public static void main(String[] args) {
   System.out.println("Hello CSE 373");
 }
}
```

To compile on a Unix system, type:

```
javac MyFirstApplication.java
```

Then to run it, type:

```
java MyFirstApplication
```


Example Applet

```
import java.applet.Applet;
import java.awt.Graphics;

public class MyFirstApplet extends Applet {
 public void paint(Graphics g) {
  g.drawString("Hello CSE 373!", 50, 25);
 }
}
```

To compile on a Unix system, type:

```
javac MyFirstApplet.java
```

Then to run it, embed a reference to it in a web page...

Web Page with Applet Tag

```
<HTML><HEAD>
<TITLE> A Simple Program </TITLE>
</HEAD><BODY>
<H1>So here is my first applet:<H2>

<APPLET
     CODE="MyFirstApplet.class"
     WIDTH=150 HEIGHT=25>
</APPLET>

</BODY></HTML>
```

Logistics of Getting Started With Java

- Choose a place to work: Math Sciences Computer Center may be best. (It's CSE 373's designated lab)
- If you need to, download and install J2SDK 1.4 using the link on our syllabus page. Decide whether to use Eclipse or the command-line tools. If you use command-line tools …
- Create a folder for your Java project.
- Choose a text editor to use for editing your source code, e.g., Word, saving as Text File.
- Name each of your Java source files using the name of your public class.
- Under Windows, create files `compile.bat` and `run.bat` to "automate" steps in your edit/compile/run cycle.

Troubleshooting your Java Setup

- Make sure you use the correct paths to the Java compiler and Java runtime executable.
- The compiler expects source files to end in .java and requires that the extension be given in the command:
  `C:\j2sdk1.4.2\bin javac MyApplication.java`
- The Runtime executable requires that you have a .class file to run, and that it is an application, not an applet.
  `C:\j2sdk1.4.2\bin java MyApplication`
- You don't type the .class extension here.
- The source file name must match the name of the public class defined in the file. Capitalization matters. Don't get it wrong.

One Setup for Java Applet Development

- Create a simple web page (HTML file) and put it in the same folder as your applet.
- Start up Netscape Navigator or I.E., and bookmark this page.
- Practice "reloading" the page and the Java applet with your browser. In Netscape, you hold the shift key and click on Reload, which forces the applet to be reloaded as well as the text of the page. Otherwise you will keep getting the (cached) old version of your applet even after updating your .class files.
- You may wish to open another browser window and bring up the Java documentation in that window, keeping it open while you are developing.
- When your applet is completely finished, then you may wish to transfer it and your web page to Dante and publish it on the web. At point, be careful about where you put each file, and the permissions on the files and the folders they are in. Always test your applets on the web before submitting them for grading.

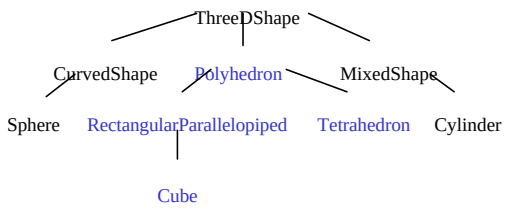
Java Classes and Inheritance

Object: A computational unit consisting of some data elements to which are associated some specific methods for operating on them.

Class: A category of objects having a single formal description in a software system.

Instance of a class: An object created as an element of a class and therefore having all of the member variables and associated methods defined for instances of that class.

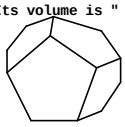
A Class Hierarchy



Class Polyhedron

```
public abstract class Polyhedron {
   protected int nFaces, nVertices, nEdges;
   protected double width, length, height;

   public void describe() {
     System.out.println("This polyhedron has "
     + nFaces + " faces, " + nVertices +
     " vertices, and " + nEdges + " edges. Its volume is "
     + volume());
   }
   public abstract double volume();
}
```



Class RectangularParallelopiped

```
public class RectangularParallelopiped extends
    Polyhedron {
  private double width, length, height;

  // A constructor - It sets protected parent variables:
  RectangularParallelopiped
    (double theWidth, double theLength, double theHeight) {
      width=theWidth; length=theLength; height=theHeight;
    nFaces = 6; nVertices = 8; nEdges = 12;
  }
  // A concrete method for the parent's abstract method:
  public double volume() {
    return width * length * height;
  }
}
```

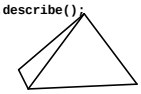
Class Cube

```
public class Cube extends
  RectangularParallelopiped{

  // A constructor that calls its parent constructor:
  Cube(double side) {
    super(side, side, side);
  }
}
// Cube inherits the volume() method of its parent.
```

Class Tetrahedron

```
public class Tetrahedron extends Polyhedron{
  // A constructor:
  Tetrahedron(double theWidth, double theLength, double theHeight) {
    width = theWidth; length = theLength; height = theHeight;
    nFaces = 4; nVertices = 4; nEdges = 6;
  }
  // A different concrete method for the same abstract method:
  public double volume() {
    return (width * length * height) / 6.0;
  }
  // This method overrides the parent's, but calls it, too.
  public void describe() {
    System.out.print("(Tetrahedron) "); super.describe();
  }
}
```

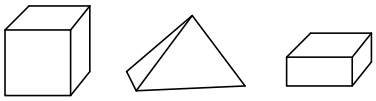


Class Poly (The Application)

```
public class Poly {
  private Polyhedron p1, p2, p3;

    public static void main(String [] argv) {
        Poly thisApp = new Poly();
        thisApp.p1 = new Cube(5);
        thisApp.p2 = new Tetrahedron(10, 8, 7);
        thisApp.p3 = new RectangularParallelopiped(2, 3, 4);

        thisApp.p1.describe();
        thisApp.p2.describe();
        thisApp.p3.describe();
    }
}
```



Output

```
This polyhedron has 6 faces, 8 vertices, and 12 edges.
Its volume is 125.0
(Tetrahedron)This polyhedron has 4 faces, 4 vertices, and 6
edges. Its volume is 93.33333333333333
This polyhedron has 6 faces, 8 vertices, and 12 edges. Its
volume is 24.0
```


Class (Static) Variables and Methods

Class variables (not instance variables) and class methods are shared by all instances of the class.

There is only one copy of a class variable.

```
protected static int numInstances = 0; // a class variable
MyObject() { numInstances++; }         // a constructor

public static void main(String [] argv) {} // a class method
```

Class variables and methods can be used even if there are no instances of the class. Class variables can serve as global variables in a program.

Java Packages

A package is a group of related classes.

Some standard packages are java.awt and java.util
To create a package, put a package declaration at the beginning of each file containing the class definitions that are to belong to the package.

```
package geometry;
public class Dodecahedron { // ...
}
public class Icosahedron { // ...
}
```

Nested packages are permitted. When imported, their class files must be located in subdirectories of their outer package directories.

Scopes of Member Names in Java

public: Accessible inside & outside of its class and subclasses.

private: Accessible only within its class definition.

protected: Accessible within its class definition and those of its descendant classes.

package: Accessible within the same package (possibly from otherwise unrelated classes).

Scopes of Identifiers in Methods

Scope of an identifier introduced within a method defaults to the block containing it.

```
{ for (int i=0; i<10; i++) {
     System.out.println("Now i = " + i);
  }
  doSomething(i);
}
double i = 100.0;
```


Method Inheritance

Suppose public class ChildClass extends ParentClass.

By default, each method of ParentClass is inherited by ChildClass.

An abstract method in ParentClass is one for which no implementation is provided, and that must be overridden by a method of the same name in ChildClass in order to be used.

If ParentClass contains any abstract methods, it must be declared an abstract class, and it cannot be directly instantiated.

If ChildClass overrides a method m(args) of ParentClass, the ParentClass version is still accessible within ChildClass using super.m(args)

A method qualified as final cannot be overridden.

Classes and Types

A type can be considered to be a restriction on the set of values that a variable is permitted to store.

A class can be used as a type.

```
String name = "Washington";
Cube myBlock = new Cube(10);
```

In Java, there are several primitive types that are not classes, e.g.:
int, byte, short, long, float, double, char, boolean.

These permit "lightweight" values to be used, which can be stored and manipulated more efficiently than can *bona fide* objects.

Upcasting and Downcasting

Upcasting occurs when an object of one type is assigned to a variable declared with a supertype of the object. No explicit casting is needed:

```
Polyhedron p = new Cube(5);
```

Downcasting, however, requires an explicit cast:

```
Cube c = (Cube) p;
```

Summary

A class is a formal category of program objects within a software system.

Instances have all the member variables and methods of their class, including those inherited from superclasses.

Subclasses can contain member variables and methods in addition to those inherited.

Inherited methods can be overridden with versions specific to a subclass.

Java provides mechanisms for hiding or not hiding names across the class hierarchy.

Classes are related to types.