Lecture J1: Java

Vivace (Seattle)

Java is...

Enthusiasts will tell you Java is...

- Simple
- Object oriented
- Machine-independent
- Multi-threaded
- Garbage collecting
- Secure
- The language of the Web

Skeptics will reply that it is…

- Complex
- Slow
- Designed for toasters
- Not finished yet

Make up your own mind!

Java perspective

Similar to C in many ways

- types: `int, float, double, char` …
- control: `for, if, while, do, switch` …

Modern primitives

- Unicode: 16-bit “universal” character set

Higher level of abstraction than C

- Object oriented
- No pointers (just references to objects)
- Inheritance
- Language support for:
  - interfaces, threads, exceptions
- System support for:
  - Software libraries, security, graphics, network

How to learn Java

This is the language of the web, so learn it on-line!

- http://www.javasoft.com/
- http://java.sun.com/docs/books/tutorial/
- http://java.sun.com/jdk/

Start with existing code, read code, read docs

Experiment by making small changes

- Add functionality progressively
- (This is how you’ll do your next assignment.)

"Hello World" in C

hello.c

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("hello, world\n");
}
```

Unix

```
% gcc hello.c
% a.out
hello, world
%
```

Compile and execute

- `gcc` compile code into machine language
- `a.out` load and execute the program

+ Advantage: good comilers make efficient programs
- Disadvantage: translated code is machine-dependent

"Hello World" in Java

hello.java

```
class Hello {
  public static void main(String args[]) {
    System.out.println("hello, world\n");
  }
}
```

Unix

```
% javac hello.java
% java Hello
hello, world
%
```

Translate and interpret the program

- `javac` translate the code into byte stream
- `java` interprets the byte stream (`Hello.class`)

+ Advantage: machine-independent
- Disadvantage: interpretation is slower than running compiled code

(Good native code java compilers are inevitable.)

Byte code example

```
% od -c Hello.class
0000000 312 376 272 276  \0 003  \0   -  \0     \b  \0 024 007  \0 023
0000020 007  \0 032 007  \0 033 007  \0 034  \n  \0 004  \0  \t  \t  \0
0000040 005  \0  \n  \n  \0 003  \0 013  \f  \0 017  \0  \f  \f  \0 036
0000060  \0 026  \f  \0 037  \0  \r 001  \0 003   (   )   V 001  \0 025
0000100   (   L   j   a   v   a   /   l   a   n   g   /   S   t   r   i
0000120   n   g   ;   )   V 001  \0 026   (   [   L   j   a   v   a   /
0000140   l   a   n   g   /   S   t   r   i   n   g   ;   )   V 001  \0
0000160 006   <   i   n   i   t   > 001  \0 004   C   o   d
0000200  \r   C   o   n   s   t   a   n   t   V   a   l   u
0000220  \n   E   x   c   e   p   t   i   o   n   s
0000240   l   l   o 001  \0  \r   H   e   l   l   o
0000260   l   d   ! 001  \0 017   L   i   n   e   N   u   m
0000300   T   a   b   l   e 001  \0 025   L   j   a   v   a   /   i   o
0000320   /   P   r   i   n   t   S   t   r   e   a   m   ; 001
0000340   L   o   c   a   l   V   a   r   i   a   b   l   e   s
0000360  \n   S   o   u   r   c   e   F   i   l   e 001  \0
0000400   l   l   o   .   j   a   v   a 001  \0 023   j
0000420   i   o   /   P   r   i   n   t   S   t   r   e
0000440 020   j   a   v   a   /   l   a   n   g   /   O
0000460   t 001  \0 020   j   a   v   a   /   l   a   n
0000500   s   t   e   m 001  \0 004   m   a   i   n 001
0000520   t 001  \0 007   p   r   i   n   t   l   n  \0
0000540 004  \0  \0  \0  \0  \0 002  \0  \t  \0 035  \0 016  \0 001  \0
0000560 020  \0  \0  \0   %  \0 002  \0 001  \0  \0  \0  \t 262  \0 007
0000600 022 001 266  \0  \b 261  \0  \0  \0 001  \0 025  \0  \0  \0  \n
0000620  \0 002  \0  \0  \0 003  \0  \b  \0 002  \0  \0  \0 017  \0  \f
0000640  \0 001  \0 020  \0  \0  \0 035  \0 001  \0 001  \0  \0  \0 005
0000660   * 267  \0 006 261  \0  \0  \0 001  \0 025  \0  \0  \0 006  \0
0000700 001  \0  \0  \0 001  \0 001  \0 030  \0  \0  \0 002  \0 031
```

Enough here to decompile the code

Check out:
www.ahpah.com

Object-oriented programming

A different way of thinking about programming

- Some people would say that learning C has already ruined you.

Direct support of abstract data type (ADT)

- Encapsulate functions that manipulate data
- Build levels of abstraction

CLASS

- Data type description
- Functions that operate on data (METHODS)
- A program is a sequence of classes

OBJECT

- "instance" of a class
- An object to a class in Java is as a variable to a type in C

METHOD

- instantiate objects
- invoke other methods

Class Implementation

Rational.java

```
class Rational {
  int p, q;

  Rational(int p, int q) {
    this.p = p; this.q = q;
  }
  public String toString() {
    return Integer.toString(p) + "/" + Integer.toString(q);
  }
  void mul(Rational x) {
    p *= x.p; q *= x.q;
  }
  void add(Rational x) {
    p = p*x.q + q*x.p;
    q *= x.q;
  }
}
```

Data

Constructor (create a new object)

Methods

Example Rational client

e.java

```
class eCalc {
  public static void main(String[] args) {
    int j;
    Rational e = new Rational(1, 1);
    Rational t = new Rational(1, 1);
    for (j = 0; j < 6; j++) {
       Rational r = new Rational(1, j+1);
       t.mul(r);
       e.add(t);
       System.out.println(e + " " + t);
    }
}
```

References

Statement `t.mul(r)` invokes the method `mul` for object `t`

- Essentially a function call
- The object `r` is passed as an argument

Inheritance

RationalA.java

```
class RationalA extends Rational {
  RationalA(int p, int q) {
    super(p,q);
  }
  private static int gcd(int m, int n) {
    if (n == 0) return m;
    return gcd(n, m%n);
  }
  private static void reduce(Rational x) {
    int t = gcd(x.p, x.q);
    x.p /= t; x.q /= t;
  }
  void add(Rational x) {
    p = this.q*x.q + x.p*this.q;
    this.q *= x.q;
    reduce(this);
  }
}
```

Why java?

Many of the ideas were around in older languages

- C++, Modula-3, Smalltalk, Ada, ...

What's new?

- Free, easy to install
- Works on most computers
- Support for large-scale software engineering
- Growing library of software available
- Internet
- Graphics
- Security

What's holding us back?

- Legacy code
- Libraries are complex
- Slow?
- Standards?