Lecture J2: Java Applets

"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.)

"Hello World" as an Applet

HelloWorldApp.java

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

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

What's the same here?

- Execute `% javac HelloWorldApp.java`

  (makes HelloWorldApp.class)

What's different here?

- "`paint`" instead of "`main`" (`main` is in Applet)
- We need a way to view the applet...

Including an Applet in HTML

HelloWorld.html

```
<HTML>
<HEAD>
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
Here is the output of my program:
<APPLET CODE="HelloWorldApp.class" WIDTH=150 HEIGHT=25>
</APPLET>
</BODY>
</HTML>
```

Open HelloWorld.html in Netscape, and voila!

May also use "`% appletviewer HelloWorld.html`" to debug.

Java API

Applications Programmers Interface

- Huge library of classes
- PACKAGE: set of classes

API packages:

- java.lang Language support (types, threads, math…)
- java.io Basic input/output support
- java.util Generic data structures
- java.net URLs, sockets…

Abstract Windows Toolkit (AWT)

- java.awt Graphics

And our friend:

- java.applet Applets

Recursive Graphics Applet

Star.java

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

public class Star extends Applet {
  static final int N = 128;
  static int SZ = 512/(N+N);

  public static void star(int x, int y, int r, Graphics page) {
     int sz = 2*SZ*r-1;
     if (r == 0)
        return;
     page.fillRect(SZ*(x-r), SZ*(y-r), sz, sz);
     star(x-r, y+r, r/2);
     star(x+r, y+r, r/2);
     star(x-r, y-r, r/2);
     star(x+r, y-r, r/2);
  }
  public void paint(Graphics g) {
     star(N, N, N/2, g);
  }
}
```

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

Point.java

```
class Point {
  public double x, y;

  Point(double x, double y) {
    this.x = x; this.y = y;
  }
  public static double dist(Point a, Point b) {
    double dx = a.x - b.x, dy = a.y - b.y;
    return Math.sqrt( dx*dx + dy*dy );
  }
}
```

Data

Constructor (create a new object)

Method

Declare object references and create a point:

```
Point a, b, c;
a = new Point( 0.3, 0.5 );
```

To access the static method dist:

```
double delta = Point.dist(a,b);
```

Another way to do `dist( )`

AnotherPoint.java

```
class Point {
  public double x, y;

  Point(double x, double y) {
    this.x = x; this.y = y;
  }
  public double dist(Point b) {
    double dx = this.x - b.x, dy = this.y - b.y;
    return Math.sqrt( dx*dx + dy*dy );
  }
}
```

To compute the distance from a to b, call the method for either a or b:

```
double delta = a.dist(b);
```

-- or --

```
double delta = b.dist(a);
```

Linked lists

Node.java

```
class Node {
   public Point p;
   public Node next;

   Node(Point pt) {
      p = pt;
      next = null;
   }
}
```

No explicit pointers!!!

(but pointers are everywhere)

```
double delta = a.dist(b);
```

To declare variables:

```
Node s, t, tour = null;
```

To create an object:

```
t = new Node(a);
```

To access the point in the node:

```
p = t.p;
```

To follow a link:

```
s = t.next;
```

Java garbage collection

You don't have to worry about it!

Client examples

TSP.java

```
  ...
  static double tourdist(Node t) {
    Node s;
    double sum = Point.dist(t.p, t.next.p);
    for (s = t.next; s != t; s = s.next) {
      sum += Point.dist(s.p, s.next.p);
    return sum;
  }

  static double delta(Node s, Point p) {
    Point a = s.p, b = s.next.p;
    return Point.dist(a,p)+Point.dist(b,p)-Point.dist(a,b);
  }
...
```

TSP Applet

TSPapplet.java

```
...
for (int i = 0; i < N; i++) {
  p = new Point(Math.random(), Math.random());
  t = new Node(p);
  if (tour == null) { tour = t, tour.next = t; }
  mins = tour;
  min = delta(mins, p);
  for (s = tour.next; s != tour; s = s.next) {
    if ((del = delta(s, p)) < min) {
       mins = s; min = del;
    }
  }
  t.next = mins.next; mins.next = t;
  showStatus("Inserted point " + i + " distance = " + tourdist(tour));
  drawpath(mins.p, t.next.p, g, background);
  drawpath(mins.p, t.p,      g, foreground);
  drawpath(t.p,    t.next.p, g, foreground);
...
}
...
```

Java Summary

C-like language for basic programs

Support for object-oriented programming

- Data type support
- Build levels of abstraction

Libraries

- Language extensions
- Graphics
- Networks

Advanced concepts

- abstract classes, interfaces, threads, exceptions, …

Reference

`java.sun.com`