COS 126 Lecture 22: Java applets and libraries

Object-oriented programming review

Applets

Java and HTML

appletviewer

Netscape

API: Application Programming Interface

AWT: Abstract Windows Toolkit

Sample class: Cartesian points

A java class is a user-defined data type

```
class Point
   {
      public double x, y;
      Point(double xval, double yval)
         { x = xval; y = yval; }
      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);
         }
   }
```

to declare variables

of type "reference to Point"

```
Point a, b, p;
```

to create a new Point

```
t = new Point(x, y);
```

to access fields

```
double dx = a.x - b.x, dy = a.y - b.y;
```

to access static method "dist"

```
delta = Point.dist(a, p);
```

Alternate design for Point class

Pure object-oriented programming approach

associate "dist" method with point objects

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

to compute dist from a to p

```
      delta = a.dist(p);
```

or

```
      delta = p.dist(a);
```

in implementation,

special keyword "this" refers to

object used to invoke method

When should a method be static?

design decision

[some people have religious beliefs]

Linked structures

No explicit pointers (!)
    but implicit pointers are everywhere

Ex: linked list of nodes containing Points

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

to declare variables
    of type "reference to Node"

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

to create a node containing
    (reference to) Point p:

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

to access point in a given node:

```
a = s.p;
```

to follow a link:

```
b = s.next.p;
```

AUTOMATIC GARBAGE COLLECTION
    since you can't manipulate pointers
    system "knows" when there are no active
        references to an object

Client examples

Object-oriented programming makes it
natural to express computation by
building layers of abstraction

Ex: (from code for TSP assignment)
linked list of Nodes with Points

function to compute total dist

```
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;
  }
```

function to compute distance increase

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

exercise: make Tour itself a class

Writing such code is
possible in C
supported in Java (and C++)

Java API

Applications Programmers Interface

- huge library of classes
- Ref: Gosling, Yellin, Java Team

java.sun.com/products/
jdk/1.1/docs/api/packages.html

PACKAGE: set of classes

API packages

- java.lang
  - container classes: Integer, Double, ...
  - language extensions: String, Class, ...
  - Java Virtual Machine: threads, ...
  - system resources: Math, in, out, ...
- java.io
  - basic I/O support
- java.util
  - generic data structures
  - Random, Time, Date...
- java.net
  - URLs, IP addresses, Sockets, ...

Abstract Windows Toolkit (AWT)

- java.awt
  - Graphics
- java.applet
  - Applet

buyer beware:

- documentation is produced automatically

Using the API

Scan documentation/manuals for what you need

- find a package
- find a class
- object-oriented or static method
- Import to resolve names if necessary

Ex: Two ways to compute random numbers

FUNCTION-ORIENTED (static method)

```
double x = Math.random();
```

OBJECT-ORIENTED:

- control over state
- multiple generators in one program

```
import java.util.Random;
...
Random rand = new Random();
```

rand: OBJECT that can generate random numbers

- instance of Random class
- contains state information, methods

```
double x = rand.nextDouble();
```

Reasons to use API

- basic language extensions
  - (Math, String, IO, ...)
- network and graphics abstractions
  - (abstract window toolkit, applets)

Sample client program

Traveling Salesperson solution (assignment 4)

user-defined resources: Node, Point

system resources: parse, random, print

```
class TSP
{
  // delta and showtour
public static void main(String[] args)
 {
 double min, del;
 int N = Integer.parseInt(args[0]);
 Point p, Node s, t, mins, tour = null;
   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;
   }
   System.out.println("Length "+tourdist(tour));
 }
}
```

is this "fun", yet?

Everyone’s first applet

HelloWorldApp.java:

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

public class HelloWorldApp extends Applet
{
   public void paint(Graphics page)
      {
         page.drawString("Hello world!", 50, 25);
         page.fillRect(50, 30, 85, 4);
      }
}
```

NOTE: "paint" not "main" (main is in Applet)

"javac HelloWorldApp.java"

makes HelloWorldApp.class, as usual

HelloWorld.html:

```
<title>Hello, World</title>
<hr>
<applet
   code=HelloWorldApp.class
   width=577 height=400>
</applet>
<hr>
```

Open "HelloWorld.html" in netscape: voila!

Use "appletviewer HelloWorld.html" to debug

Sample graphics applet

Recursive graphics (ref. prog. assignment 5)

```
import java.applet.Applet;
import java.awt.Graphics;
public class Star extends Applet
  {
    static final int N = 128;
    static int SZ = 512/(N+N);
    static Graphics g;
    public static void star(int x, int y, int r)
      { int sz = 2*SZ*r-1;
        if (r ,= 0) return;
        g.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 window)
      {
        g = window;
        star(N, N, N/2);
      }
  }
```

TSP applet

Add graphics to TSP program in four easy steps

* replace

```
class TSP
```

with

```
import java.applet.Applet;
import java.awt.Graphics;
public class TSPapplet extends Applet
```

* add new method to draw path segments

```
static void drawpath
   (Point p, Point q, Graphics page, Color c)
   { int xp = (int)(512*p.x), yp = (int)(512*p.y);
     int xq = (int)(512*q.x), yp = (int)(512*q.y);
     page.setColor(c);
     page.drawLine(xp, yp, xq, yq);
   }
```

* replace

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

with

```
public void paint(Graphics page)
```

* add code in loop to update path

```
drawpath(mins.p, t.next.p, page, Color.white);
drawpath(mins.p, t.p, page, Color.black);
drawpath(t.p, t.next.p, page, Color.black);
```

Exercise: try this out!

```
javac TSP.java
appletviewer TSP
```

Java summary

C-like language for basic programs

object-oriented programming
- direct data type support
- build layers of abstraction

libraries
- language extensions
- graphics user interface
- networks

advanced concepts
- abstract classes
- interfaces
- Java Virtual Machine
  - threads
  - exceptions
- sockets
- ...

visit

http://java.sun.com

regularly to appreciate scope of effort

serious alternative: C++

Ref: Stroustrup, C++ Programming Language