ORF 201

COMPUTER METHODS FOR PROBLEM SOLVING

Lecture 6

Functions and Graphics

(C) Princeton University

Hints on Java Applets

Edit **`/u/yourname/.login`** and set **`umask`** to **`022`**.

Copy **`/u/orf201/lab1/.rhosts`** to **`/u/yourname`**.

Create a directory **`/u/yourname/public_html/JAVA/myutil`**.

Copy all files in **`/u/orf201/public_html/JAVA/myutil`** to your **`myutil`** directory.

Do with **`ccj`** as you did above with **`myutil`**.

Use **`chmod`** to set permissions on **`/u/yourname`**, **`/u/yourname/public_html`**, and *every directory thereunder* to **`drwxr-xr-x`**.

Use **`chmod`** to set permissions on every file in **`/u/yourname/public_html`** and *in every directory thereunder* to **`-rw-r--r--`**.

Check **`CLASSPATH`** by typing **`echo $CLASSPATH`**. If it doesn't start out with **`.:/u/yourname/public_html/JAVA`**, edit **`.cshrc`** to correct the problem.

Edit **`clock.html`**, **`gravity.html`**, etc., and set **`codebase`** to "**`../..`**".

Square Roots - Revisited

```
import myutil.*;
public class Sqrt2
{
    public static void main(String[] args)
    {
        int i;
        double x,r;
        while (true) {
            x = Console.in.readDouble("Enter x: ");
            if (x<=0) return;
            r = sqrt(x);
            System.out.println(Format.floating(10,5,r));
        }
    }
    static double sqrt(double x)
    {
        double r,s, err;
        r = x;
        do {
            s = x/r;
            r = (r+s)/2;
            err = x-r*r;
            if (err < 0) {err = -err;}
        } while (err > EPSILON);
        return r;
    }
    static final double EPSILON=1.0e-10;
}
```

Use **`main`** to test the method.

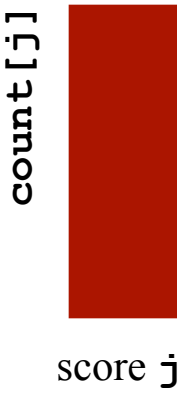
Create a *method* to compute square roots.

Drawing Histograms

Create a class **`Histo`**:

Histogram of grades, therefore, integers between 0 and 100.

```
public class Histo
{
    static final int maxGrade = 100;
    static int[] count = new int[maxGrade+1];
    static int maxcount, mode, median;
}
```



`count[j]` will contain the number of students with score **`j`**.

We need a `main()`

Inside **`Histo`**, put a **`main`**.

```
public static void main(String[] args)
{
    int grade;
    /*******************************************
     * Initialize the array count[] to all zeros.
     *******************************************/
    for (grade=0; grade<=maxGrade; grade++) {
        count[grade] = 0;
    }
    /*******************************************
     * Now read in a list of grades.
     *******************************************/
    int classSize = Console.in.readInt();
    for (int i=0; i<classSize; i++) {
        grade = Console.in.readInt();
        count[grade]++;
    }
    computeModeAndMaxcount();
    computeMedian();
    /*******************************************
     * Draw the histogram
     *******************************************/
    HistoFrame hf = new HistoFrame(maxcount, median,
                                   count, maxGrade);
    hf.resize(600,600);
    hf.show();
}
```

Other Methods in `Histo`

computeModeAndMaxcount()

```
static void computeModeAndMaxcount()
{
    int grade;

    maxcount = 0;
    for (grade=0; grade<=maxGrade; grade++) {
        if (count[grade] > maxcount) {
            maxcount = count[grade];
            mode = grade;
        }
    }
}
```

computeMedian()

```
static void computeMedian()
{
    int grade, n, cum;              // these variables are local

    n = 0;

    for (grade=0; grade<=maxGrade; grade++) {
        n += count[grade];          // count the total number
    }
    cum = 0;
    for (grade=0; grade<=maxGrade; grade++) {
        cum += count[grade]; // count number worse than grade
       if (cum > 0.5*n) break;  // break if more than half
    }
    median = grade;
}
```

Window Frames

The graphics is done in class **`HistoFrame`**, which extends **`Frame`** and contains data, a *constructor*, and a *paint* method.

```
class HistoFrame extends Frame
{
    int maxcount, median, maxGrade;
    int[] count;

    HistoFrame(int maxcount, int median,
               int[] count, int maxGrade)
    {
        this.maxcount = maxcount;
        this.median = median;
        this.count = count;
        this.maxGrade = maxGrade;
    }

    public void paint(Graphics g)
    { /* details on next page */
    }
}
```

Frame

Part of Java. A frame is a graphics window that can be displayed on the console (using **`show()`**).

Constructor

A method having the same name as the class but not having a return type. Called when **`new`** is invoked.

Extends

Means that **`Histoframe`** automatically inherits all of the data and methods in **`Frame`**. For example, **`resize()`** and **`show()`**.

This

Two variables called **`count`**; one is global to class **`HistoFrame`** and one is local to the constructor. Inside the constructor the global variable is referred to as **`this.count`**.

The Paint Method

Class **`Frame`** defines a default **`paint()`** method, which does squat.
In **`HistoFrame`** we can override the default by defining our own **`paint()`**.

```
public void paint (Graphics g)
{
     int grade;
     GL.ginit(g,size().width, size().height, this);
     GL.ortho2(-1, maxGrade+1, -1, maxcount+2.0);
     GL.color(Color.white);
     GL.clear();
     GL.color(Color.black);
     for (grade=0; grade<=maxGrade; grade++) {
          GL.move2(grade, 0.0);
          GL.draw2(grade, count[grade]);
     }

     /* make 'tick' marks every ten*/
     GL.color(Color.green);
     for (grade=0; grade<=maxGrade; grade+=10) {
          GL.move2(grade, 0.0);
          GL.draw2(grade, 0.2);
     }

     /* Highlight the median in red */
     GL.color(Color.red);
     GL.move2(median, 0.0);
     GL.draw2(median, count[median]);
     GL.swapbuffers();
}
```

Methods beginning with **`GL.`** are defined in **`myutil`**.

Final Notes on `Histo`

To have a **`Frame`** we must import the *Abstract Window Toolkit* :

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

And don't forget:

```
import myutil.*;
import ccj.*;
```

Now we're done. See **`Histo.java`** handout.

Drawing Histograms - Fancy Version

Combine **Histo** and **HistoFrame** into one class.
Call it **Histo2**. It must extend **Frame**.

Replace instantiation of **HistoFrame**

```
HistoFrame hf = new HistoFrame(maxcount, median,
                                count, maxGrade);
hf.resize(600,600);
hf.show();
```

with instantiation of **Histo2**

```
Histo2 h = new Histo2();
h.resize(600,600);
h.show();
```

The default constructor could be used since the variables **maxcount**, etc., are global to class **Histo2** and therefore don't need to be propagated.

Move the **paint()** method to class **Histo2** and do away with the constructor altogether.

See **Histo2.java** for details.