Princeton University
Department of Operations Research
and Financial Engineering

ORF 201
Computer Methods in Problem Solving

Lab 3: An Analog Clock

Due Sunday, Feb 20, 11:59 pm

---

1 Privacy vs. Publicity

By default, CIT sets up every student account at Princeton with a high level of security attached to it. That is, files you create are, by default, readable only by you. Similarly, directories you create can only be accessed by you. That is why in last weeks lab you had to go to extreme lengths, with the `chmod` command, to specifically make the files in your `public_html` directory visible to the outside world.

There is a simple change that you can do that will make visibility be the default. In the file `.login`, which is in your home directory, there is a line

```
umask 077
```

Changing the 077 to 022 will make all files and directories that you create visible to anyone with an account on the sesamest or arizona machines. The upside is that you won't have to be constantly running `chmod` to give read permission.

The downside is that files you want to keep private must be explicitly hidden. For example, there might be email messages that you save that you don't want your ex-lover to see (should you be so lucky to have an ex). You'll have to make sure they are hidden. A simple way to accomplish this is to create a directory, called say `Private`, in your home directory, protect that directory by typing

```
chmod 700 Private
```

and then save every private file in that directory (or some subdirectory thereof). Without any further action from you, they will be automatically protected because `Private` is.

Even though there is a certain amount of unwanted exposure, we strongly suggest that you set umask to 022 at least while you are in ORF 201.

2 CLASSPATH

To further set up your environment to do JAVA (and fix a few small things), type the following command:

```
source /u/orf201/bin/fixer_up
```

It is essential for what follows. (Note. The file `fixer_up` is a *batch* file, which in UNIX is called a *shell script*. It is a plain text file and you are welcome to look at it either by `cat`'ing it or loading it into xemacs. It is short. Feel free to ponder, if you wish, what it does.)

Using xemacs, edit your `.cshrc` file and look for the lines that refer to `CLASSPATH`. If you did the above command, you should find such lines. Now, go to the end of the file and add the following two lines:

```
setenv CLASSPATH /u/orf201/public_html/JAVA:$CLASSPATH
setenv CLASSPATH .:/u/yourname/public_html/JAVA:$CLASSPATH
```

Here, of course, `yourname` refers to your login id. Before saving the file add an empty line to the end of it and then save it. You can do this by simply hitting ¡Enter¿ at the end of the second line above.

To make these changes to `.cshrc` take effect, you need to open up new shell windows. To be sure about it, close all currently open windows and then open up some new ones before proceeding with the following steps.

3 Getting Started

For this assignment, you will write a Java applet to draw an analog clock. First, you need to create some directories and copy some files. Begin with this:

```
mkdir -p public_html/JAVA/ORF201/clock
```

Then copy the following files into the clock directory:

```
cd public_html/JAVA/ORF201/clock
cp /u/orf201/public_html/JAVA/ORF201/clock/index.html .
cp /u/orf201/public_html/JAVA/ORF201/clock/clock.html .
cp /u/orf201/public_html/JAVA/ORF201/clock/Clock.java .
```

Note that there is a space before the last "dot" on each line.

Another note. It is tedious to type those last three lines, each of which is similar to the one before and none of which can tolerate even a single typo. Fortunately there are convenient typing shortcuts that make such tasks easy. For example, after entering the first line correctly you can use the up arrow key on the keyboard to bring it back. You can then edit the line using standard xemacs commands. When you've got what you want just hit the *Enter* key to execute the second command. The last command can then be done similarly. Your TA will be able to give you more real-time help with command line editing.

The Java code is in the file Clock.java. But it is seriously broken. In fact, at the moment all it does is paint some lines, circles, and polygons. Your job, of course, is to make it do what it is supposed to do: draw a clock.

4 Compiling

First, however, let's see what it currently does. To do that, we need to compile the code. To compile, type the following command:

```
javac Clock.java
```

If I didn't screw anything up, this step should not produce any error messages. You will however get a warning message—something about using a *deprecated API*. That message is okay and should be ignored. You will be seeing it a lot. Now, type the following command:

```
appletviewer clock.html
```

An applet window should pop up with the broken clock in it.

5 Coding

To fix the broken clock, fire up xemacs with Clock.java as the file to edit:

```
xemacs Clock.java
```

Then, in the xemacs window, search for the string "void paint". The next line contains an open brace. The code between this brace and the matching closing brace is where all of the painting takes place. If you look through this part of the code you will see the statements that draw the lines and circles that you saw when you ran the applet. This is the stuff you have to modify. Here are some things you might need to know:

- The hour, minute, and second are stored in the integer variables hour, min, and sec, respectively.
- The number $\pi = 3.14159...$ is stored in Math.PI.
- Trigonometric functions are referred to as Math.sin(), Math.cos(), etc. They each take a double as input and return a double as output.
- GL.color() sets the color for subsequent drawing. The argument must be Color.something, where something can be any of several common color names.
- GL.ortho2(xmin,xmax,ymin,ymax) establishes minimum and maximum x and y coordinates for the drawing window. The lower left corner is (xmin, ymin) and the upper right corner is (xmax, ymax).
- GL.move2(x,y) moves the drawing "pen" to coordinates (x,y).
- GL.draw2(x,y) draws from the current pen position to the position given by coordinates (x,y).
- GL.circf(x,y,r) draws a circle of radius r at (x,y) and fills it with the current color.
- The three functions, GL.pmv2(), GL.pdr2(), and GL.pclos(), are used to paint polygons. GL.pmv2(x,y) picks up the pen and moves it to (x,y). GL.pdr2(x,y) connects the pen from its current position to the one given by (x,y). GL.pclos() connects the current pen position with the first pen position, given by GL.pmv2(), to create a closed polygon. The polygon is filled with the current color.
- The last line before the closing brace must be GL.swapbuffers(). We'll discuss buffer swapping at a later date.

6 Going Public

Fire up netscape and go to the following address:

```
http://www.princeton.edu/~yourname/JAVA/ORF201/clock/clock.html
```

If your code works (and permissions are set correctly), you should see the clock in the browser. If you see nothing, chances are your permissions are set incorrectly. You need to have `drwxr-xr-x` on each of the following directories:

```
/u/yourname
/u/yourname/public_html
/u/yourname/public_html/JAVA
/u/yourname/public_html/JAVA/ORF201
/u/yourname/public_html/JAVA/ORF201/clock
```

In addition, you need to have `rw-r--r--` on each of the following files:

```
/u/yourname/public_html/JAVA/ORF201/clock/index.html
/u/yourname/public_html/JAVA/ORF201/clock/clock.html
/u/yourname/public_html/JAVA/ORF201/clock/Clock.class
```

You can check these things with `ls -l` and you can fix anything that is amiss with `chmod`, just like you did in Lab 2.

7 Minimum Requirements

At a minimum:

1. Your clock must work.
2. It must have three hands: hour, minute, second.
3. It must have tick marks for the 60 minutes in an hour and distinct tick marks for the 12 hours in a day.