Problem Set 3

Due: 4:30PM, Friday March 1, 2002

Problem 1: Warmup on fluids and pipes [10%]

This is a short question to prepare you for the main problem below. Write a simple Pipe class and a Fluid class as defined below. Also write a main() method in a Test class to:

1. Create an array of 3 Fluids (each having a name and a density). Pick any values.
2. Create a Vector of 3 Pipes (each having a name, length and diameter). Pick any values.
3. Set a pipe roughness that is the same for all pipes. (Static variable.)
4. The main method ends by outputting the data for each fluid and pipe, and the static variables as well.

The Pipe and Fluid classes each just need a constructor, the fields listed above, and one method to print out the data.

The Pipe class also needs a static method to set the roughness parameter, and a static method to print roughness and gravity. It keeps gravity as a single constant across all pipes.

This problem introduces you to arrays, Vectors, static methods and static variables, which are the key concepts in the main problem below. Remember to import java.util.* for the Vector class.

Problem 2: Fluid flow in pipes [90%]

Problem background.

You are modeling a network of pipes through which liquids can flow. The network is drawn below. The pipes are assumed to be fairly straight, so that pressure changes occur only from elevation changes and friction. The angles are exaggerated in the drawing.
Each pipe has a diameter D, a length L, and an elevation change z. All pipes have the same roughness h. Your program will calculate the pressure drops and flow rates at points B, C, D, E and F, given an input flow rate at point A. The characteristics of each section are:

<table>
<thead>
<tr>
<th>Section</th>
<th>Diameter D (m)</th>
<th>Length L (m)</th>
<th>Elevation change z (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>0.5</td>
<td>50</td>
<td>-0.1</td>
</tr>
<tr>
<td>BC</td>
<td>0.4</td>
<td>15</td>
<td>-0.05</td>
</tr>
<tr>
<td>CD</td>
<td>0.3</td>
<td>5</td>
<td>-0.03</td>
</tr>
<tr>
<td>CE</td>
<td>\sqrt{7} \times 10^{-1}</td>
<td>8</td>
<td>-0.03</td>
</tr>
<tr>
<td>BF</td>
<td>0.3</td>
<td>25</td>
<td>+0.05</td>
</tr>
</tbody>
</table>

Sample initial inputs at point A are:

- Pressure at A: 100,000 Pa (Pa = pascal, or Newton/m²)
- Fluid velocity in pipe ($v_A$): 1 m/s

Pipe roughness h = 0.00001 m for all pipes

The characteristics of the 4 fluids are:
### Fluid Properties

<table>
<thead>
<tr>
<th>Fluid</th>
<th>Density $r$, kg/m$^3$</th>
<th>Viscosity $q$, kg/(m·s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Water</td>
<td>$10^3$</td>
<td>$10^{-3}$</td>
</tr>
<tr>
<td>Air</td>
<td>1.2</td>
<td>$1.8 \times 10^{-5}$</td>
</tr>
<tr>
<td>Ethyl alcohol</td>
<td>$0.79 \times 10^3$</td>
<td>$1.2 \times 10^{-3}$</td>
</tr>
<tr>
<td>Methyl alcohol</td>
<td>$0.79 \times 10^3$</td>
<td>$0.62 \times 10^{-3}$</td>
</tr>
</tbody>
</table>

The equation to calculate the pressure change in a pipe is:

$$p_B = p_A - r \cdot g \left( z + \frac{(f \cdot L \cdot v^2)}{(2 \cdot D \cdot g)} \right)$$

The friction factor $f$ depends on the Reynolds number $R$ of the flow and the roughness of the pipe wall. The Reynolds number is:

$$R = \frac{r \cdot v \cdot D}{q}$$

For laminar flow ($R < 2000$, in pipes),

$$f = \frac{64}{R}$$

For turbulent flow ($R > 2000$ in pipes; we ignore 'transition'), there are many models of $f$. The most common model, the Colebrook formula, requires a numerical solution of an equation, typically using Newton's method, which we cover later in 1.00. In this homework, we will use Haaland's formula, which estimates $f$ directly:

$$f = \left(1.81 \log_{10} \left(\frac{6.9}{R} + \left(\frac{h}{3.7D}\right)^{1.11}\right)\right)^{-2}$$

Java doesn't have a log10 method; note that $\log_{10} x = \left(\log_e x\right) \left(\log_{10} e\right) = \left(\ln x\right) (0.4343)$. 

In your network, the cross sectional areas of the pipes entering and leaving each junction are the same, making flow conservation easy to satisfy. Thus, for example, at junction B, the output flow pressure and velocity from AB is equal to the input flow pressures and velocities in BC and BF.

### Program

You must design a program to implement the analysis of a system of pipes using the model presented above. We offer some suggestions on the design here, but you may develop your own design. There are a few required items that you must include; we note them below.

The two major "things" in this problem are pipes and fluids, so they are the logical classes to define. You will also need a class that contains the main() method to launch and manage the objects and methods you write.

Your pipe class (or the closest class to it, if you develop your own design), must contain two static methods that are broadly useful and may be called from other methods or objects to assist calculations in other programs without instantiating a pipe object of your design. These two methods are to compute the friction $f$ and the Reynolds number $R$ in a pipe. Your pipe class should also have appropriate static fields to store the values of several parameters that are the same for all pipes (or even more broadly in the case of gravity).
You should compute the Reynolds number, friction, pressure change, output pressure and flow volume in each pipe. The flow volume is just the pipe cross-sectional area times the velocity. Each pipe should be able to output its own data (including its name). Each pipe should have a field that indicates how it is connected to adjacent pipes, as necessary to do the calculations. You will probably want to keep a reference to adjacent pipe(s) in each pipe object; you should spend a bit of thought on whether storing the upstream, downstream or all adjacent pipes is the best. (The best solution is quite simple.) You may assume that the pipe network is always a branching tree, as shown in the figure.

The fluid class can be quite simple, holding just the parameters of each fluid. There is, of course, an interaction between fluids and pipes, since fluids flow through the pipes. It is probably most natural to have the computational methods associated with pipes rather than fluids, but you are free to design this. Note that Reynolds number and friction are general concepts in fluid mechanics, but there are very different functions for computing them for fluid flows in pipes versus open channels versus plates, etc., so these methods would be viewed by most engineers as being associated with pipes.

The main() method must do the following:

- Define the 4 fluids and their properties. Place the fluids in an array.
- Define the 5 pipes and their properties. Place the pipes in a vector. Think about what order you should put them in. Normally you'd handle the pipes and fluids in the same way for consistency, but we want you to work with both arrays and vectors in this homework. Also, normally you'd read all these parameters from a file, but since we haven't covered files yet, "hardcode" the fluid and pipe parameters in your program. You cannot ask the user to input all this data each time.
- Prompt the user, using JOptionPane for:
  - Fluid type for this analysis
  - Initial pressure at A
  - Initial velocity at A
  - Pipe roughness, for all pipes
- Go through the system of pipes to compute the characteristics of each pipe
- Output the fluid type, and then the data for each pipe

Testing

Test your program with laminar and turbulent flows, with all 4 fluids. For segment AB, the correct outputs are given below (with fewer digits than Java will give you) for an initial pressure of 100,000 N/m², an initial velocity of 1 m/sec, and a roughness of 0.

Fluid type: water
Pipe AB
Reynolds number: 500000.0Friction: 0.0129
Pressure at inlet: 100000.0
Pressure at outlet: 100333.9
Pressure change: 333.9
Volume flow rate: 0.196
Turnin

Turnin Requirements

- Problem 1: Email only. No hardcopy required.
- Problems 2 & 3: Hardcopy and electronic copy of ALL source code (all .java files).
- Place a comment with your name, username, section, TA's name, assignment number, and list of people with whom you have discussed the problem set on ALL files you submit.
- DO NOT turn in electronic or hardcopies of compiled byte code (.class files).

Electronic Turnin

To electronically turn in your problem sets, run Netscape

Then go to the 1.00 web page at:

http://command.mit.edu/1.00Spring02

Click on the “Submit Assignment” button. Be sure to set the Selection Bar to Problem Set 1 or your files may be lost. Finally, go back to the home page and click on the “View” section and be sure that your files were received. If you submit a file twice, the latest version will be graded.

Penalties

- Missing Hardcopy: -10% off problem score if missing hardcopy.
- Missing Electronic Copy: -30% off problem score if missing electronic copy.
- Late Turnin: -20% off problem score if 1 day late. More than 1 day late = NO CREDIT