Introduction to Computers and Engineering Problem Solving

Spring 2005

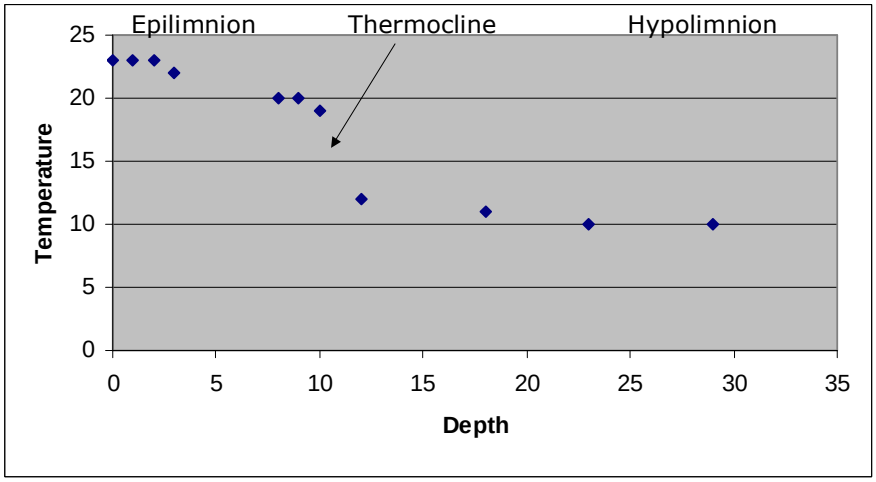
Problem Set 4: Lake Ecology

Due: 12 noon, Session 16

1. Problem statement

Lakes in North America and other temperate zones are often thermally stratified in summer: there is warm, less dense water near the surface and cold, denser water near the bottom, with a sharp delineation between the two layers. The upper layer is called the epilimnion; the lower layer is the hypolimnion; and the boundary between them is the thermocline. There is very little mixing between the layers, so decomposition of organic matter at the bottom of a lake can cause severe depletion of oxygen in the hypolimnion, affecting plant and animal life strongly in that layer.

In the figure below, the thermocline occurs at a depth of approximately 10 to 12 meters, where the temperature decreases rapidly from about 20° C to 10° C.



The thermocline is the point at which the absolute value of the first derivative (dT/dz) is greatest, or where the second derivative (d^2T/dz^2) is zero. It is difficult to estimate the exact depth z; we will define the thermocline to be a range of z values in which there is a rapid change in temperature. In the example above, the thermocline would be at z= 10 m to z=12 m.

2. Problem objective and approach

You have data on water temperature and dissolved oxygen for a set of four lakes, as shown at the end of this problem set. You must represent this data for each lake as an array of Observation objects. You thus need an Observation class.

The Observation class is very simple: it just contains the depth, temperature and oxygen concentration. You may make the Observation class data members public for simplicity. (It is better to make them private and to use get() and set() methods, but we simplify this to save effort.) The Observation class has no methods, other than a constructor.

Each lake has an Observation array, name, area and depth, represented as member data in the Lake class.

You must be able to find the following properties for each lake. You must write three member methods in the Lake class to return:

1. Thermocline: upper and lower z values that define it, and the temperature difference. Your method must return a triplet of values, as an ArrayList. Use simple logic as in the example above to determine the thermocline depth. You may assume there is a clear thermocline; you don't need to handle cases where there is none or it's ambiguous. Assume there is only one observation at each depth. The thermocline may extend over more than a pair of observations, though.
2. Average oxygen concentration above and below the thermocline. Ignore oxygen concentration readings in the thermocline. Your method must return a pair of values as an array. This method must find the thermocline first; don't assume the Thermocline method was run already.
3. Whether oxygen concentration in the hypolimnion is lower than the minimum allowable concentration. This method returns a boolean. Don't assume the Thermocline method was run. (This method is easy; just call the method above and check against the minimum allowable concentration.)

You must also write two static methods in the Lake class. This is perhaps not the natural way to implement this, but we want you to learn how these methods are written and used.

1. Minimum temperature. This must be a static method.
2. Maximum temperature. This must be a static method.

These methods will need arguments.

The Lake class must also have two static data members:

1. Number of lakes. Your constructor(s) must manage this member.
2. Minimum allowable average oxygen concentration in a layer: a constant, 5 mg/L.

Last, the Lake class must have get() methods for the name, area, depth and a static get() method for the number of lakes.

You must also write a LakeTest class that:

1. Creates the four lakes and all their member data. No user input is required; all the data can be 'hardwired' in the program. The Lake objects must be placed in an ArrayList. (See MIT server for a code fragment that creates the Observation data.)
2. For each lake:
    a. Invokes the get() methods and prints out the lake name, area and depth.
    b. Invokes all five computational methods on each lake and writes out their return values. You will have to unpack the return values to print them when they are in an array or ArrayList.
3. Outputs the static data member, number of lakes.

3. Problem data

The data for your four lakes is given below. Temperature is °C; oxygen concentration is in mg/L. Treat all data values as doubles.

Lake Nepessing

Depth	Temperature	Oxygen
0	23	15
1	23	14
2	23	14
3	22	14
8	20	14
9	20	15
10	19	14
12	12	7
18	11	7
23	10	7
29	10	6

Ortonville Lake

Depth	Temperature	Oxygen
0	26	11
1	25	11
2	24	11
4	24	10
5	23	11
8	20	10
9	17	9
11	17	9
13	17	10

Lake Hadley

Depth	Temperature	Oxygen
0	24	12
2	24	12
4	23	13
6	20	10
8	21	11
10	17	9
12	13	5
14	13	4
16	12	4
18	12	4
20	12	3

Fish Lake

Depth	Temperature	Oxygen
0	23	8
3	24	8
5	22	7
8	15	4
10	16	3
13	15	3
15	14	2

The area and depth of each lake is:

Name	Area (km^2)	Depth (m)
Nepessing	2.3	35
Hadley	1.7	24
Ortonville	1.3	15

Name	Area (km^2)	Depth (m)
Fish	1.1	19

We have placed the temperature and oxygen concentration data on the MIT server. You can cut and paste from the data into your program to save some typing effort

Turn In

1. Place a comment with your full name, MIT server username, section, TA name and assignment number at the beginning of all `.java` files in your solution.
2. Place all of the files in your solution in a single zip file.
    a. Do not turn in electronic or printed copies of compiled byte code (`.class` files) or backup source code (`.java~` files)
    b. Do not turn in printed copies of your solution.
3. Submit this single zip file.
4. Your solution is due at noon. Your uploaded files should have a timestamp of no later than noon on the due date.

Penalties

- 30% off if you turn in your problem set after Friday noon but before noon on the following Monday. You have one no-penalty late submission per term for a turn-in after Friday noon and before Monday noon.
- No credit if you turn in your problem set after noon on the following Monday.