CS 1301 - Homework 12

Due before Thursday, November 13, 5:30 *pm* on Vista (WebCT). You can resubmit your work up until the deadline.

1. Airplanes fly through a sector that is monitored by the Air Traffic Control (ATC). All planes have an ID which is composed of two upper-case alphabetic characters followed by four digits (you will NOT be responsible for enforcing this, you'll assume it is entered correctly). An Airplane has an exact location in 3-d space (x,y,z). An Airplane has the ability to tell you how far it is from another plane (called *separation*) and how far it is vertically from another plane (called *vertical* separation). At any point in time the ATC has a group of planes that are in the sector. Airspace rules require that all planes maintain a separation of at least 2500 ft. and maintain a vertical separation of at least 2000 ft. A warning is issued for each pair of planes that violates either of these rules.

   You will write a system that reads in the locations of all the planes in the sector at some point in time. Then you will display the separation and vertical separation (and any warnings) of all pairs of planes. Your output will look like this:

   ```
   VK4434, JW8823, sep=4024, vsep = 2383
   VK4434, UR8439, sep=634, vsep =345,  ***sep warning, ***vsep warning
   ...
   JW8823, GL0912, sep=2324, vsep = 2183,  ***sep warning

   JW8823, GL0912, sep=2679, vsep = 1274,  ***vsep warning
   ```

   Notes:

   1. Remember, try to separate the problem from any natural objects that are present in the system. In this case, we will write a Plane class which represents a plane and an ATC class whose main method will solve the problem.

   2. Create a **Plane** class. This class will have exactly three methods: getID, getSeparation(p : Plane), getVerticalSeparation(p : Plane) and a constructor that takes the x,y,z location of the plane and the ID as input. In other words, some sample code may look like this:

      ```
      Plane p1  = new Plane( "CN2392", 10.4, 66.2, 8832.4 );
      Plane p2  = new Plane( "MT9232", 88.4, 7234.2, 3332.7 );

      double separation = p1.getSeparation( p2 );

      // which would give the same result as p2.getSeparation( p1 )
      ```

   3. *Vertical Separation* is simply the absolute difference between the *y* coordinates of two planes. *Separation* is the distance between two planes in 3-d space: (Source: http://en.wikipedia.org/)

      For two 3D points, $P = (p_x, p_y, p_z)$ and $Q = (q_x, q_y, q_z)$, the distance is computed as

      $$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}.$$

   4. Create an **ATC** class to solve the problem. Try to write a simple main with static methods to help you.

   5. The ATC main needs to represent an arbitrary number of planes. How do you model this?

   6. In the ATC class, finding all combinations of planes will probably require a bit of thought. Hint: suppose you have 4 people in a room (A,B,C,D), write down all possible pairs of people. Now, how would you do that in code?

7. Testing the system is challenging. Choose locations of planes so that you test thoroughly. Here is an online calculator you can use to calculate distances in 3-d:

   http://www.analyzemath.com/Geometry_calculators/distance_midpoint_3D.html

2. This problem is similar to HW 10, Problem 5. You will write a class, **Payroll** which will manage the hours worked for each employee for each of the seven days of the week. Requirements:

   a. A private variable that tells how many employees there are. This property is read-only.
   b. A private 2-d array to hold all hours worked for each employee. Provide a getter and setter. You must not set the hours for an employee unless the hours are greater than zero.
   c. A private 1-d array to hold the hourly salary of each of the employees. This property is read-write.
   d. A constructor that accepts a parameter that tells how many employees there are. Hint: the constructor should create the array of hours and the array of salaries.
   e. A method, **getEmpHours** that accepts an index for an employee and returns the number of hours worked for the week.
   f. A method, **getEmpPay** that accepts an index for an employee and returns their pay for the week. Any hours over 40 are paid at time-and-a-half. Hint: you can use the **getHours** method.
   g. A method, **getDayHours** that accepts an index for a day and returns the total number of hours worked (for all employees) for the day.
   h. A static method **printPayrollReport** (used by main) which prints a payroll report similar to that shown below. You can add some methods to the class if you want to, to help you with this report. Or, you can just hard code it with the methods that are required.

Payroll Report

Number of Employees: 4
Total Hours Worked for Week: 135
Total Pay for Week: $1795.00

Employee	Total Hours	O. Hours	Rate	Pay
0	30	0	12.00	360.00
1	45	5	10.00	475.00
2	40	0	20.00	800.00
3	20	0	8.00	160.00

Total Hours Worked:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
25	25	20	25	20	0	20

   i. A main that tests your class and calls the printPayrollReport method.

EXTRA CREDIT

   j. A method, **getDayPay** that accepts an index for a day and returns the total pay (for all employees) for the day. Note, you must figure out if an employee is in "overtime" for the day or part of the day. For instance, if you ask this method for the pay on Thursday, and one employee has worked 12 hours on Mon, Tues, and Wed and 8 hours on Thursday, then 4 hours of his Thursday pay is at overtime rate.