Lab 3

(30 pts) Due March 25 at midnight

For this lab, if you choose, you may work with a partner. You may also choose to work alone. No more than 2 people per group. Note that you must BOTH WORK ON ALL THE PROBLEMS.

If you choose to work with a partner, only one of you needs to turn in the lab, but both names must be on the lab (of course).

Part 1:

Note: AS A RULE all fields (properties) must be private, and no methods or fields should be static unless otherwise specified.

1. (8 pts) Create a class for a watch. It should have private hour, minute, and seconds as fields. It should have 4 constructors: one that takes no fields and sets the time to 12:00:00, one that takes only the hour field as an input and sets the minutes and seconds to 0, one that takes the hour and the minute fields as input parameters and sets the seconds to 0, and then one that sets all 3 fields. There should be a "Set" method that sets the hour, the minute, and the seconds field. There should be a toString() method that returns a string that represents the current time.

   Finally, you should create a "main" class that has a main method in it and creates a watch object. Test your 4 constructors. Test your methods.

Dialog Boxes:

If you want to input data by having the user type something in, use a dialog box. To use a dialog box:

First at the very top of the class file,
import javax.swing.JOptionPane;

Then in the method, create a string variable that will hold the input from the dialog box:
String svar = JOptionPane.showInputDialog("Enter your info");
And finally, you'll have to convert the svar to a double:
double xvar = Double.parseDouble(svar);

2. (10 pts) Write a class for a Student. Each student has a bunch of fields: the student's name, their lab scores (an array of 5 ints, initialized to 0), their test scores (an array of 3 ints, initialized to 0, and their grade so far (initialized to an F - we're pessimistic to start with). The fields should also include the number of labs completed so far (initially set to 0) and the number of tests taken so far (initially set to 0).

   The constructor for the Student class should have as an input parameter the student's name, and it should set the lab scores, the test scores, the current grade, the number of labs completed, and the number of tests completed. All should be private

   The toString method should print out the student's name, the lab scores so far, the test scores so

far, and the current grade

There also needs to be a method calcGrade() that calculates the current grade by finding the average of the lab scores so far (meaning, if the user has only completed 2 labs, the lab array might look like this: [84,92,0,0,0] and the number of labs completed would be 2, so the average would be 88) and multiplies that by .45, then finds the average test score (in the same way) and multiplies that by .55, and then returns a char - a letter grade as follows: if the total score is >= 90, the returned char is an 'A', >=80 gets a 'B', >=70 gets a 'C', >= 60 gets a 'D', and otherwise the char returned is an F. Should this method be public or private?

Then you need a method that adds a lab grade by using a **dialog box** (see above) to ask the user for the latest lab grade. The latest lab grade is added to the lab array, the number of labs completed is updated, and the current grade is updated (How?)

Finally, you need a method that adds a test grade, also using a dialog box, adding the latest test grade to the test array, updating the number of tests taken, and updating the current grade.

In a separate class in the same workspace have a main function. Create a student object. Test the class by continually updating the student's scores and printing out the student object.

3. (12 pts) I have a nike app. It keeps track of the miles I run daily, the seven-day average (starting at Sunday each week), my average mph for each run, and my seven-day average for mph. It also tells me whether I've run farther this week than last week (so it keeps track of how many miles I ran last week) and whether I ran, on average, faster this week than last week (again, so it keeps track of how fast I ran last week (and, of course, it has my name).
    a. Write a class for a nike app. Include the proper fields for each of the above properties. The fields should be private.
    b. Write 3 constructors: one that sets all fields to 0 (or arrays of 0); one that sets the miles for the week to different values (e.g., {7.2,5.9,6.1,6.0,5.4,8.3,0.0}) and the rest of the fields to zero; and one that sets the miles for the week and the array of mph for week and the rest of the fields to zero.
    c. Now write some methods:
        i. "Get" functions for each of the fields (also known as "accessors"). Since each field is private, Each of these functions should return a field's value (e.g., the getLastWeeksMileage should return the field holding last week's average mileage). For the array fields, return one string printing out each day of the week's value)
        ii. One that "resets" at the end of the week, calculating the average number of miles run this week and setting the past week's average to that value, then zeroing out this week's average miles, and calculating the average mph and setting last week's average mph to that, then zeroing out the daily mph.
        iii. A method that looks at how far I've run so far this week and tells me whether I've run farther this week than last week

iv. A method that looks at how fast I've run on average this week and tells me whether I'm running faster this week or last week.
v. A method that allows you to input your mileage for a particular day (this method will take one input parameter – the day of the week, in the form of a number, with 0 = Sunday, 1 = Monday, etc. For this function you will want to be able to input data using a **dialog box** (see above).
vi. A method that allows you to input your average mph for a particular day (this method will be very similar to the above method).

d. Now create a main class (a separate class with a main method in it). In it test the constructors by creating different nike app objects with different constructors. Test the different methods (you'll have to test the methods that allow you to "reset" and allow you to enter data before you can test the methods that compare this week to last week).

Part 2:

When you have completed this, begin working on the OCR Project.