Assignment 5

Due: Oct. 20, 2013 at 11:55PM.

RESTRICTION: You should not be using any of the Java language features that we did not cover in class and that are not discussed in chapters 1-8 of the textbook.

WARNING: Start early! Do not wait till the last weekend to complete this code.

Problem 1: Student class

(60 points)

Student class description:
The student type should describe student’s last name, first name, id (letter ‘N’ followed by an 8-digit number), gpa (a real number in the range of 0.0 to 4.0, a whole number of credits completed, and a single letter status indicating full time (‘F’) or part time (‘P’). All students should be full time by default, unless their status is changed.

The operations allowed should be the following:

- Create a new student given the first name, last name, id and status (gpa and number of completed credits both initialized to zero).
- Create a transfer student given the first name, last name, id, status, gpa and number of credits.
- Determine student’s id.
- Determine student’s name.
- Determine student’s status.
- Determine student’s gpa.
- Determine student’s number of credits.
- Change student’s status.
- Update number of completed credits for a student by adding the new number (for example credits completed in the current semester) to the current number of credits. The number of completed credits should never decrease.
- Change student’s gpa (replace the old value of the gpa with a new one).
- Produce a string with student information in the following format:

  **LastName, FirstName id status gpa numberOfCredits**

Driver program:
The driver program is a class with the main method that tests your class. You should make sure that your class can handle the following situations:

1. Create a new student with valid first/last name, id, and status.
2. Create a new student with valid first/last name, id, and invalid status.
3. Create a new student with valid first/last name, invalid id, and valid status.
4. Create a new student with valid first/last name, invalid id, and invalid status.
5. Create a transfer student with valid first/last name, id, status, gpa and number of credits.
6. Create a transfer student with valid first/last name, id, status, invalid gpa and valid number of credits.
7. Create a transfer student with valid first/last name, id, status, gpa and invalid number of credits.
   NOTE: In all of the above cases, you can assume that the first and last names are valid.
8. Change student’s status from part time to full time.
9. Change student’s status from full time to part time.
10. Change student’s status from part/full time to an invalid status.
11. Change student’s number of completed credits by adding a positive integer to the current number.
12. Change student’s number of completed credits by adding a negative integer to the current number (this should not decrease the current number of credits).

After each test you should print the string with student information (using the format described above) for the student that was just modified.

SOMETHING TO THINK ABOUT: Think about how you would represent an entire database of students using a Java class. What operations should that kind of database provide and how (insertions, deletions, queries, modifications of student data)?

Problem 2: ExamGrades class
(40 points)

Write a class that can store a list of students and their exam grades. Your class should contain two arrays:

- an array of 100 student last names,
- an array of 100 student exam grades.

Initially both arrays are empty.
The class should have default constructor that does nothing.
The class should provide the following methods:

- add (... ) - adds new student and exam grade; this will add one entry to each array,
- getNumberOfStudents ( ... ) - returns the current number of students in the list
  NOTE: the maximum number of students is 100,
- sortByName ( ... ) - sorts both arrays, so that they are arranged alphabetically
  NOTE: whenever you move a name in one array, you need to move the grade in the other array
- sortByGrade ( ... ) - sorts both arrays so that they are arranged from highest exam grade to the lowest
  NOTE: whenever you move a grade in one array, you need to move the name in the other array
• print ( ... ) - prints to the screen students’ name and grade one per line (either unsorted, or in the order that results from the last sort method used)

NOTE: if an add method is called after one of the sorts methods is called, the new student should be added to the end of the current list. This will produce a list that is no longer sorted.

WARNING: You need to keep track of the current number of students in the list. When the array is sorted only the used locations in the array should be sorted, not the entire array.

Write a program with main() method that tests the ExamGrades class.

Grading

Does the program compile? If not, you will loose all the points for that problem.

Is the program properly documented? If not, you will lose 1/3 of the points

Proper documentation includes:

• preamble with the name of the author, date of creation and brief description of the program;
• appropriately chosen variable names, i.e., descriptive names;
• comments inside the code describing steps need to be taken to accomplish the goal of the program;
• appropriate formatting, indentation and use of white space to make the code readable;
• brief descriptions of the methods including the meaning of parameters and what the return values are.

Remember that the code is read by humans and it should be easy to read for people who were not involved in its development.

Is the program correct?, i.e., does it produce valid results that follow the specification of the problem every time it is run? If not, you will lose 1/4 of the points for that problem. If the program is not completely correct, you get credit proportional to how well it is developed and how close you got it to the completely correct code.

What and how to submit?

You should submit this assignment using NYU Classes. You should submit a single compressed file that contains Java source code files with solution to problems 1 and 2. The source code file should be named: LAST_Student.java and LAST_Problem1.java (for Problem 1), and LAST_ExamGrades.java and LAST_Problem2.java (for Problem2), in which you replace LAST with your last name. Your compressed file should be named LAST_Assignment5.zip, in which you replace LAST with your last name. Do not submit files with .class extension or any other files generated by Eclipse or other IDE. If you submit the assignment past the deadline, the NYU Classes will not let you upload the file. In this case, email it to Joanna at joannakl@cs.nyu.edu.