Laboratory Sheet 2

This Lab Sheet contains material based on Lectures 1 – 4 (up to *2 Oct 2013*), and contains the submission information for Laboratory 2 (week 3, *7-11 Oct 2013*).

Be sure to look over the material of Lectures 3 - 4 before Laboratory 2, and bring this lab sheet to your Laboratory session.

You are expected to begin work on laboratory sheets before your scheduled session. This will typically be necessary if you are to make best use of the presence of your tutor during the lab, and to make a satisfactory attempt at the submission exercise(s).

The deadline for submission of the lab exercise via moodle is 24 hours after the end of your scheduled laboratory session in week 3 *(7-11 Oct 2013).*

Of course you may submit work that is incorrect or incomplete. In order to stretch the stronger members of the class, some of the laboratory exercises are quite challenging, and you should not be too discouraged if you cannot complete all of them.

Aims and objectives

- Writing a simple Java program that uses console I/O and various control structures
- Adapting the program to handle certain forms of invalid input

Set up

When you download `Laboratory2.zip` from the moodle JP2 page and unzip it, you will obtain a `Laboratory2` folder, and within it a folder entitled `Submission2`. In this latter folder will be a file `CreditsAndGPAs.java` that contains a skeleton `main()` method for this exercise. In Eclipse, you should switch your workspace to the `Laboratory2` folder, then create a new project entitled `Submission2`; the provided Java source file will become part of this project when you follow steps analogous to those described in Laboratory Sheet 1.

Submission material

Preparatory work for this programming exercise, prior to your scheduled lab session, is expected and essential to enable you to submit a satisfactory attempt.

Your solution to the submission exercise should be developed, as usual, using Eclipse.

Exercise

Write a complete Java program to meet the following specification. Input to the program is a sequence of lines of text from the console (standard input). Each line contains information about a single **student**, namely a **student number** together with details of courses taken. Each course is represented by three fields, namely, the **course code** (a four character identifier), the number of **credits** available for the course (an integer), and the **band** obtained (a one or two character code). The number of courses recorded for a given student can be any number greater than or equal to zero.

An example line of input:

```
0109999  HVF7 20 C3  FRE0 10 CW  SDC5 40 A4  JUI9 20 D1  DFU5 30 C2
```

Assume that no course is duplicated in a student's record. Items of data on the input line are separated by one or more whitespace characters. For each line of input, the program is to generate one line of output to the console (standard output) containing **student number**, total number of **credits** obtained, and **grade point average** to two decimal places (see rules below).

The appropriate output for the above line of input is

```
0109999  110  13.09
```

For your first version of the program, you may assume that the input is valid in all respects - i.e. is correctly formatted, and contains no syntax errors of any kind. But one form of valid input that your program should handle correctly is where a student has obtained zero credits.

Note that each line of console input will generate a corresponding line of output before the next line can be input. The end of the input can be signaled by typing `^Z` (tap Z with the Control key pressed); this simulates an 'end-of-file' condition.

Once you have a working program, consider how it might be extended to cope with invalid input. Run the program with each of the following lines of input, note what happens, and try to understand why.

```
0109999  HVF7 20

0109999  HVF7 X C3

HVF7 20 C3

0109999  HVF7 20 K1
```

Make appropriate changes to the program so that all of these would be identified as invalid, and would cause the program simply to output the message "`Badly formatted input`" to the console (standard output).

Credit and GPA Rules

Credits are awarded for any course taken unless the student is awarded CR (Credit Refused) or CW (Credit Withheld). If credits are awarded for a course, so also are grade points, based on the first letter of the band, on the following scale:

$$A = 16,\quad B = 14,\quad C = 12,\quad D = 10,\quad E = 8,\quad F = 6,\quad G = 2,\quad H = 0.$$

Grade point average is computed by summing over all courses, credits × grade points, and dividing by the total number of credits. So, in the above example, the student has accumulated 110 credits, and the GPA is computed as follows:

$$\text{GPA} = (20 \times 12 \;+\; 40 \times 16 \;+\; 20 \times 10 \;+\; 30 \times 12) \,/\, 110 \;= 13.09.$$

You should use the System.out.printf()[1] method with an appropriate format specifier for floating point numbers to print the GPA to two decimal places. See http://stackoverflow.com/questions/7197078/printf-f-with-only-2-decimal-points for details.

Unit Tests: I have supplied some simple test cases to check that your program is working properly. When you create the Submission2 project, you should see two source code files in the project pane – one is the skeleton Java source code for you to fill in the details. The other file contains the Unit tests to check your program is working properly. In the (likely) event that Eclipse complains about your Unit tests, do the following:
In the package explorer pane (top left): Right click on package > build path > configure build path > libraries > add Library > JUnit. Click Finish. Now all the Eclipse build errors should go away.
To execute Submission2:
right click on CreditsAndGPAs -> run as Java application
To test Submission2:
right click on CreditsAndGPAsTest -> run as JUnit test

A note on Plagiarism / Copying

The solution that you submit for each exercise is supposed to be your own work. Naturally, some discussion of exercises among members of the class is to be expected, and indeed can be quite beneficial to the learning process. However, this is quite different from wholesale copying of someone else's work. Plagiarism of this kind is merely cheating, and is considered to be an act of fraudulence and an offence against University discipline.

When plagiarism / copying is detected in *assessed exercises* (in other courses – there are none of these in JP2), *disciplinary action will be initiated*. The Department has sophisticated software that is routinely used to detect submissions that show a suspicious level of similarity.

This software will also be used on the JP2 submissions, but merely to identify levels of possible over-collaboration. Disciplinary action will not be taken in this context. However, students who appear to be implicated will be spoken to. It is not in anyone's interest for students to be submitting someone else's work – this does not contribute to the learning process. Refer to the on-line Level 2 Class Guide for detailed guidelines on plagiarism.

Submission

You should submit your work before the deadline no matter whether the program is fully working or not. JP2 Ticks are awarded for effort, not necessarily results.

When you are ready to submit via the moodle JP2 page, click on Laboratory 2 Submission. Click 'Add Submission'. Open Windows Explorer and browse to the folder that contains your Java source code ...\Laboratory2\Submission2\ and drag only the single Java file CreditsAndGPAs.java into the drag-n-drop area on the moodle submission page. **Your**

[1] http://docs.oracle.com/javase/6/docs/api/java/io/PrintStream.html#printf%28java.lang.String,%20java.lang.Object...%29

markers only want to read your java file, not your class file. Then click the blue save changes button. Check the .java file is uploaded to the system. Then click submit assignment and fill in the non-plagiarism declaration. Your tutor will inspect your file and return feedback to you via moodle.

Gaining the credits for JP2

Recall that the credit criteria for JP2 include obtaining at least 7 ticks for lab assignments. ***To obtain a tick you must attend the lab and submit the assignment.***