Python application for a School

Prateek Kumar Singh

my_school.py is a Python application for a school. It reads data from files. The project is to implement the required functionalities in OO style with three classes, School, Students, and Courses.The program reads from a file specified in command line.

After creating a School object, it's read_scores(filename) method is called to load data from a text file specified in the command line. The data, student scores, is stored in a 2D array.

scores.txt

```
34       C081   C082   C083   C084
S2123    99.5   75     85     62
S1909    100    83.2   45     TBA
S2025     x     92     67     52
```

Above is an example file that stores student score data. Data fields are separated by spaces and new lines. The first row contains course IDs and the first column contains student IDs. The first field in the data, the top left corner, shows the number of rows and the number of columns in one integer. For example '34', the first digit 3 means there are 3 students in this table. The second digit 4 means there are 4 courses.

The table stores every student's final results in those courses. Results are all strictly integers. A result '-1' means not enrolled in that course. A '0' means the student did enrol but failed to receive any mark. No mark can go beyond 100, except '888', which means the student is enrolled but the result is not available yet for some reason. Such a course is NOT counted for result average. Characters in scores.txt will be treated as -1, except 'TBA', which means enrolled but result not available as yet, same as '888'.

The program shows usage if no file is supplied. Otherwise it shows the students and list the student with the highest average score and display on the command line **exactly** as below:

```
> Python my_school.py
[Usage:] Python my_school.py <scores file>

> Python my_school.py scores.txt
      | C081 | C082 | C083 | C084
------|------|------|------|------
S2123 |  99  |  75  |  85  |  62
S2025 |      |  92  |  67  |  52
S1909 | 100  |  83  |  45  |
------|------|------|------|------
3 students, 4 courses, the top student is S2123, average 80
```

The program supports two types of courses. One is Compulsory Course. Compulsory courses may have different credit points. Another type is Elective Course. All elective courses have the same credit point. By default that is 6 points. No compulsory course can be changed into an elective. No elective can change type.

The program can read one more file which stores the information of courses (see example below). Info includes course ID, course title (all titles are in one word), type of the course and credit points. C1, C2 are both compulsory courses. E means an elective course. All courses available in the school appear in this file and in the first file (student scores file). There are no duplicate or redundant courses.

```
courses.txt

C081   Mathematics    C1   12
C082   Science        C1   12
C083   English        C2   24
C084   Technologies   E     6
```

The program can print a course summary on screen and save that summary into a file named as **course_report.txt**. Given the above `courses.txt`, the program output looks like below. The content of `course_report.txt` are also the same, except the last line. Course names in the second column use * to indicate a compulsory course and **–** to indicate an elective course. The fourth column is the number of students enrolled in that course. The fifth column is the average score of the course.

```
> Python my_school.py scores.txt courses.txt

CID     Name          Pt. Enl.  Avg.
----------------------------------
C081  * Mathematics   12   2     99
C082  * Science       12   3     83
C083  * English       24   3     65
C084  - Technologies   6   2     57
----------------------------------
The worse performing course is C084 with an average 57

courses_report.txt generated!
```

The program can support two types of Students, full time students (FT) and part time students (PT). A full time student is required to enrol in at least 3 compulsory courses. A part time student is required to enrol in at least 2 compulsory courses.

Student information can be read from a file from command line as another argument. That file stores information about students, that includes student ID, name (no space between first name and last name, but an underscore) and study mode (FT or PT). All students appear in this file as well as in the first file (student results file). There is no duplicate records or empty records. See the example below.

`students.txt`

```
S2123  Sue_Vaneer    FT
S2025  Robin_Smith   FT
S1909  Barry_Banks   PT
```

The program can print a report of students on screen and store that report in a text file named as **student_report.txt**. Given the above **`students.txt`**, your program output should look like below. The content of **`student_report.txt`** should be the same, just without the last line.

```
> Python my_school.py scores.txt courses.txt students.txt

SID    Name         Mode  CrPt   GPA
-------------------------------------
S2123  Sue_Vaneer    FT   54     3.55
S2025  Robin_Smith   FT   42 !   2.42
S1909  Barry_Banks   PT   24 !   2.0

student_report.txt generated!
```

In the Above report, The fourth column is the total credit points that the student has completed. For example Sue Vaneer, she has done all four courses, so she earned 12 + 12 + 24 + 6 = 54 credit points. The fifth column is the GPA. So that for Sue is (4x12+3x12+4x24+2x6)/54=3.55

If the student did not meet the minimum requirement, then a **!** will appear next to that figure.

Additionally, students are listed in descending order of GPA.

The course report and student report are both accumulative, meaning a new report will not overwrite the existing ones but be placed on the top of the file. The latest report is always on the top. In addition, the date and time when the report was generated are also saved in the text files, just above the report, in the format `"%d/%m/%Y %H:%M"`