IT 226 Project 1: Data Integration

This is a **group** project**.** Only **ONE submission** per project is required (anyone of the team members). All members of the group will get the same grade on the project.

Please Do Mention Your Team Members ULID and NAME On Reggienet when submitting the project.

In this project, you will write a program that combines data from one or more grade books from various courses. This combined data will be stored suitably, and can be "sliced" in several ways to export required data. You will provide a simple text-based interface to add or retrieve data.

Data

Three grade books from three courses have been provided to you. These contain simulated data, but in a format acceptable to Reggienet. Each file is basically an exported version of an Excel spreadsheet. The data format is a .csv file (comma separated values). They are all text files, so open them in Notepad to see their contents.

1. Each line represents one row of a spreadsheet which is all the data for one student in that course. Within a row, all columns are separated by commas.
    a. Column data may contain several words separated by spaces, but commas are reserved only as separators.
    b. Occasionally some column data may legitimately contain commas. In this case, the contents of that column will be enclosed in double quotes.
2. The first line of each file contains the column headings. Some special column headings that will be important for you will be:
    a. "User ID" or "Student Id": these will be ULIDs and will be unique for each student.
    b. If a column heading contains the string "grade" (case insensitive), then the contents of that column are letter grades. Note that this may show up as "grade", "Grade", "Letter grade", etc.
    c. If a column heading contains the string "name" (case insensitive), then the contents of that column are the first, last or complete names of students.
    d. All other columns contain individual grades for each assignment (as a number out of 100) or the cumulative total (also as a number out of 100).
3. The second line onwards contains the data, one student per line. The data for a student is arranged in the same order as the column headings.

The provided data has "three dimensions": student (identified by Id), semester and year (identified as "Fall" and "2005"), and course (identified as course number "IT 226"). When you are provided a file by the user, you are also provided explicitly which semester, year and course this file corresponds to.

What to do

1. Write classes that act as a repository of data read from all the provided files, such that:
    a. It is possible to extract data for a single student across courses and semesters.
    b. It is possible to extract data for a single semester, across students and courses.
    c. It is possible to extract data for a single course, across students and semesters.
2. To organize the data as efficiently as possible (in space as well as to complete the above operations), you have the following data structures at your disposal (you are not expected to create your own data structure):
    a. ArrayList
    b. LinkedList
    c. TreeSet
    d. TreeMap
    e. Regular arrays
3. Write a class that reads the data in the csv format, extracts data from it and using the column headings, organizes it. Lastly it must add it to the existing repository of data.
4. Provide a simple text-based interface that presents a keyboard-key driven menu to the user and allows the following operations:
    a. Add data ('a' or 'A'):
        i. Take file name from the user (*course-semester-year.csv*).
        ii. Read the provided file (if it exists), extract the data and add it to the repository along with the course number, semester and year data.
        iii. Print the number of students whose data it just read, and how many students already existed in the repository.
    b. Save data for a student ('s' or 'S'):
        i. Take student ID (e.g. "ashesh"). Note that the student ID may have numbers.
        ii. Take the name of the file where data must be exported/saved.
        iii. Find all data from the repository pertaining to this student, and export it in csv format with the following columns:
            1. Column headings: "Student Id", "Course", "Semester", "Year", "Assignment name", "Points"
    c. Return number of students who got a specific grade in a specific course in a specific semester ('g' or 'G'):
        i. Take the course number (e.g. 437). The user must have the option of skipping this input by typing "none".
        ii. Take the semester and year from the user (e.g. "Fall and "2005"). The user must have the option of skipping this input by typing "none".
        iii. If both inputs are skipped, the program should return to the menu without doing anything.
        iv. Return an array of integers. The array must be of length 5. The first position should store the number of A's, the next one the number of 'B's and so on.

1. If the course number and semester/year were specified, it should contain data only for that course during that semester/year.
2. If the course number is missing, it should contain data for all the courses during that semester/year.
3. If the semester/year is missing, it should contain data for the given course across all semesters.

d. Exit the program ('e' or 'E').

Please make sure to check that your program does not crash with invalid input (i.e. a file name that does not exist, etc.). You can assume that the provided file will always be a csv file in the format explained above.

Help

1. Look at the Scanner class to see how you can read a line and separate it using the ',' as your delimiter.
2. Look at the documentation of the different Collection classes.
3. Some student IDs that repeat across some of the provided files: 3xr9yx, 1wvs78, u439zz5. Open these files in excel to verify the data for these students.

Grading

The following will be roughly the grading rubric:

1. Parsing the files correctly (35 points).
2. Organizing the data (30 points).
3. Various operations Text interface and saving to files (45 points).

What to submit

Each group should submit the following:

1. All the source code files (*.java). Please do not submit the entire Eclipse project folder. If you put classes in packages, please submit using the appropriate folder structure.
2. Each group will submit a group evaluation form.
3. Each member must submit a peer evaluation form.

All submissions must be made on Reggienet. No email submissions allowed!