Student Marks Processing

In this lab, you will write a program to process the marks of the students enrolled in a subject. The marks are stored in a text file that has the format shown below.

```
Code|CSEPRG
Title|Programming
FieldCount|3
F|20|10|70
StudentCount|26
S|18447565|VANDERGRAFF     |T  | 74.0 | 42.5 | 57.0 |
S|18907347|MARS            |V  | 80.5 | 60.0 | 72.0 |
S|18981993|FENNEL          |WG | 77.5 | 55.0 | 69.0 |
S|18983070|ECHOLLS         |L  | 69.0 | 40.0 | 71.0 |
S|18930300|KANE            |DS | 65.0 | 47.5 | 80.0 |
S|18915430|KANE            |L  |  0.0 |  0.0 | 34.0 |
S|18917104|NAVARRO         |EW | 58.0 | 35.0 | 66.5 |
S|18928929|CASABLANCAS     |D  | 50.5 | 57.5 | 64.0 |
S|18982917|CASABLANCAS     |C  | 88.5 | 62.5 | 87.5 |
S|18971724|COOK            |J  | 61.0 | 37.5 | 63.5 |
S|18982863|MANNING         |M  | 51.5 | 52.5 | 79.0 |
S|18928875|GANT            |C  | 66.5 | 32.5 | 43.0 |
S|18927829|HAMILTON        |Y  | 69.0 | 35.0 | 48.0 |
S|18963759|TOOMBS          |F  | 83.0 | 60.0 | 74.0 |
S|18917386|DAMATO          |L  | 56.5 | 70.0 | 73.0 |
S|18961542|LAMB            |D  | 87.0 | 42.5 | 74.0 |
S|18130217|WEIDMAN         |C  | 25.0 | 57.5 | 86.0 |
S|18966580|CLEMMONS        |V  | 46.5 | 32.5 | 77.5 |
S|18934516|VAN LOWE        |V  | 71.0 | 17.5 | 74.5 |
S|18933190|MCCORMACK       |C  | 71.5 | 77.5 | 70.5 |
S|18930122|POMEROY         |S  | 57.0 | 57.5 | 78.0 |
S|18940511|SINCLAIR        |M  | 75.5 | 65.0 | 60.5 |
S|18927590|GOODMAN         |W  | 89.5 | 70.0 | 84.5 |
S|18529509|MORAN           |C  | 51.0 | 30.0 | 77.5 |
S|18965595|GRIFFITH        |H  | 76.0 | 50.0 | 49.0 |
S|18933014|FITZPATRICK     |M  | 51.0 | 32.5 | 55.5 |
```

- The first line, beginning with Code, identifies the subject code.
- The second line, beginning with Title, identifies the name of the subject.
- The third line, beginning with FieldCount, identifies the number of assessment components (or assessment tasks) used in the subject.
- The fourth line, beginning with F, contains the percentage weight for each assessment component (e.g. the first component in the example is worth 20/100, the second 10/100 and the third 70/100).
- The fifth line, beginning with StudentCount, identifies how many students' marks the file contains.
- All the remaining lines each identify a student (and start with an S). A student's line has their student number, their family name, their initials, and then a mark for each assessment component. Note all component marks are out of 100.
- The tokens in each line are separated by the vertical bar ('|') symbol.

Class Student and Class Subject

A class called Student with attributes and methods for marks processing is provided for you. The class is shown in the diagram below.

```
                              Student

- String studentNumber
- String familyName
- String initials
- double[] marks

+ Student(String studentNumber, String familyName,
               String initials, double[] marks)
+ boolean setMark(int markNumber, double mark)
+ double getMark(int markNumber)
+ String getName()
+ String getInitials()
+ String getStudentNumber()
+ String toString()
```

A class called Subject captures relevant information for a Subject, including an array of Student objects and a separate array of final marks. The partially completed Subject class diagram is shown below. It is designed only to suit the purpose of this lab.

```
                              Subject

- String code
- String title
- int fieldCount
- String fieldWeights // string with tokens separated by |
                      // e.g. F|20|10|70
- int studentCount
- Student[] students
- int[] finalMarks

+ Subject(String code, String title, int fieldCount, String fieldWeights,
              int studentCount, Student[] students)
+ Subject loadData(Scanner in)  // this is a class method
+ void display()
- calculateFinalMarks()
+ writeReport(PrintWriter)
```

Files for you to copy from LMS:

- Student.java
- Subject.java
- CSEPRG (a text file)
- Several test programs: TestDisplay.java, TestLoadData.java, TestWriteReport. java and TestWriteSubjecToFile. java

All the provided files, except Subject.java, are complete.

Question 1– Define the loadData Method

You may assume the text input file is correctly formatted. Use the StringTokenizer class and methods from the Integer and Double classes (such as Integer.parseInt()) to process the input file.

The overall aim of this method is to read the data file, create the Subject instance and return it. The logic for this task is as follows:

```
Get subject code
Get title
Get field count
Get field weights

Get student count
Create an array to store Student isntances
For each student:
   Get the student number
   Get the family name
   Get the initials
   Get the marks and construct an array of marks
   Create the Student
   Add the student to the array of Students

Create the subject
Return it
```

There are lots of things you need to do. It is best to do it **incrementally**. That is, write some code and test it, then add some more code and test it, etc., until the work is finished.

a. You will do a lot of testing, and you will need a method to display the details of a subject. So this is your first task: Define method display.

It should display a subject like this:

```
code: ...
title: ...
field count: ...
field weights: ... (a string)
student count: ...
students:
details of the first student, including the marks
details of second students
...
```

(Note: You do not have to include the final marks of the subject in the display.)

How to test the display method: Run the provided TestDisplay class, and observe the output.

Discussion: The TestDisplay class (i) creates a subject with no students and display the subject, and (ii) creates a subject with two students and display the subject.

b. (Read and create the subject with no students)

In the `loadData` method, write enough code to read the subject code, title, fieldCount, fieldWeights (which we model simply as a string).

Then create a subject with those details and with no students. Return this subject.

How to test: Run the provided test program TestLoadData.

c. (Create the subject with one student)

Add more code to `loadData` method to skip over student count line, and reads the detail of first student. Create a student based on these details, including the marks. Then create a subject which has this student as the only student of the subject.

How to test: Run the provided test program TestLoadData again. This time, we expect to see the same output as for the previous test, except that it also includes the details of a student (the first student in our data file).

d. (Create the subject with all the students in the data file)

Modify the loadData method to read the student count, add a loop to read all the students, create Student instances, and add them to the students array. Create a subject which contains all the students, and return the subject.

This is our final version of loadData.

How to test: Run the provided test program TestLoadData again. Check the output against the data file.

Question 2– Calculate the Final marks

Write the calculateFinalmark method of the Subject class. This method calculates the final mark for each student and store it in array finalMarks. The marks are rounded to the nearest integer value. The marks in array finalMarks must correspond to the students in array students.

How to test:

(i) In method calculateFinalMarks itself, add some code to display the final marks on the screen.

(ii) In method writeReport, add a statement to call calculateFinalMarks.

(iii) Run test program TestWriteReport, which should display the final marks on the screen.

(iv) Check the output by calculating some of the final marks and compare them to the output.

(v) Once the test is done, comment out the output statements in method calculateFinalMarks.

Question 3– Write Report

Complete the writeReport method of the Subject class.

The written report should contain information as follows for the example input file for the subject CSEPRG:

- The highest mark
- The lowest mark
- The average mark
- A histogram of the subject CSEPRG is:

```
 0- 9:
10-19:
20-29:*
30-39:
40-49:*
50-59:****
60-69:**********
70-79:********
80-89:**
90-99:
  100:
```

The histogram shows the number of students with a final mark in the given range. For example, the following line from the histogram:

50-59:****

means four students received marks from 50 to 59 inclusive.

Hint: The most general technique to produce the histogram is to first define the following arrays that specify the boundaries and the labels for the groups:

```
int[] lowers = {0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
int[] uppers = {9, 19, 29, 39, 49, 59, 69, 79, 89, 99, 100};
String[] labels =
             {" 0- 9", "10-19", "20-29", "30-39", "40-49",
              "50-59", "60-69", "70-79", "80-89", "90-99", " 100"};
```

Question 4– Optional

Add functionality to your program Subject.java to output all the Subject information to a file.

How to test: Run TestWriteToFile program and inspect the file CSEPRG-COPY.txt

NOTE: In case you want to try the split method of the String class, the regular expression you will need is `"\\|"` .

■