Developing a TA App

1 Introduction

In this activity, you will create a Java class that will simulate a grader teaching assistance. A grader teaching assistance sometimes helps the instructor by performing a series of grade-related tasks such as:

a) Reads the students grades from a text file (you don't want the professor to type grade by grade while running the program, right?)
b) Computes the final student grades (A, B, etc.) based on their weighted overall scores (tests, quizzes, labs, etc.)
c) Computes the student performance in relation to the average class score
d) Provides a nice final report for each student found in the text file

2 The overall design

The following sub-sections describe the overall design for this activity.

2.1 The TA App project

Since most of the professors at Pitt teach more than one course per semester, the teachers have more than one grader TA. Each grader TA is responsible for one course. Therefore, you should create a Java Class (the blueprint) of a grader TA. This Java class should have the following features:

a) When created, the grader TA should be given the location of the text file which contains the grades for each single student in the course.
b) When reading the text file, it should create retain all information about the student: the name, e-mail, test scores, lab scores, and quizzes scores.
c) It should keep a list of all the students for later statistics
d) Perform tasks (b), (c), and (d) from Section 1 above.

2.2 The Student Class

Since the information needed by the Java TA class for each student is the same (item (b) above), you should create a second Java Class named Student, which will contain:

a) The student name
b) The student ID
c) A list of their exam scores
d) A list of their lab scores
e) A list of their quiz scores
f) A method that returns the average score of each list above

2.3 The TA_APP program

In order to the TA provide a final grade (letter A, B, etc) to the students, the TA must have two pieces of information: a file containing all the students scores and the grade system (90-100 points is A, etc.).

Your main program shall create a TA object that will take two text file names as inputs in the constructor call: one that points to the location of the students grades and the other that points to the grade system adopted in this course.

This main program will also be responsible to request the TA to provide (printout) all sorts of information about each student and the overall class performance, as discussed later in this document.

3 Code implementation

3.1 The main project file

In NetBeans, create a new Java Project called TaApp. NetBeans will generate the necessary project files including the TaApp.java class, which is automatically opened in the NetBeans main area.

At this point we will not do changes in this file yet. Let's create the other classes required by this project first. Section 3.5 has the details on this Java file (class).

3.2 The Student Class

3.2.1 The Class Fields

In this project create a Student Class. This student class shall have the following fields:

- Student name
- Student e-mail address
- A 1D array of exam grades (3 exams)
- A 1D array of lab grades (5 labs)
- A 1D array of assignments (3 assignments)
- Final grade (e.g., A+, B-, etc.)
- The overall score (e.g., 89.5)

All the above variables shall be global (class fields) and with private access.

3.2.2 Getter and Setter Methods

Since the class fields are private, you must create getter methods for all of them, so that external Java programs can request their values.

Remember a getter method is of the form:

```
public return_type getClassFieldName() {
   return classFieldName;
}
```

Example: For the e-mail address class field, its getter method would be:

```
public String getEmail() {
   return email;
}
```

All the class fields shown in Section 3.2.1 with the exception of the final grade and overall score will be initialized from within the Student class constructor (See Section 3.2.3 below). Due to that these class fields shall have just getter methods, so that external classes can request their value, but not set them after those values have been entered via constructor inputs. The final grade and overall score class fields will have both: getters and setters, since the TA class will set their values later on.

3.2.3 Class Constructor

The Student Class will have only one (non-default) constructor. This constructor will take the input parameters required to initialize the class fields defined in Section 3.2.1, with the exception of the final grade letter field, as discussed in Section 3.2.2.

3.2.4 Class methods

The Student Class shall have no additional methods than the getters and setters discussed above.

3.3 The GradeSystem Class

3.3.1 GradeSystem Class Fields

The GradeSystem Class shall contain the following private fields:

- The file path and name of the Grade System text file.
- A 1D array of the possible grades (such as A+, B-, etc.).
- A 2D array of minimum and maximum score value for each possible grade.

3.3.2 GradeSystem Class Constructor

The GradeSystem shall have just one (non-default) constructor. This constructor shall require the file path and name of the Grade System text file.

This constructor shall:

a) Open and read the file
b) Parse the data into the class fields discussed in Section 3.3.1.

3.3.3 Getter and Setter methods

This class will not have any getter and setter for the class fields.

3.3.4 GradeSystem Methods

The only public method that this class will have is the one that returns the grade (e.g., B+) for a given score (i.e, 85.2) used as input.

```
public String getGradeBasedOnScore(double score)
```

3.4 The TeacherAssistant Class

3.4.1 The Class Fields

The TeacherAssistant Class shall have the following fields:

- A 1D list of Students (this list will contain all the Student Objects created when reading the text file that contains their names and scores)
- A GradeSystem object. The TeacherAssistant Class will use this object to get each student final grade

3.4.2 Getter and Setter methods

This class will not have getters and setters. There are other methods thought, discussed in Section 3.4.4.

3.4.3 Class Constructors

The TeacherAssistant Class will have only one (non-default) constructor. This constructor will take two inputs:

- The score file path and name: students_scores.csv
- The grade scale file path and name (grade_scale.csv)

As soon as a TeacherAssistance object is created in your application, it could go ahead and perform all the necessary activities that the TA would do:

- Create a GradeSystem object based on the file provided as input in its constructor
- Create multiple Student Objects and save them in the 1D array of Students, while parsing the csv file provided as input also.

You can do that by placing calls of those activities inside the constructor, such as:

```
public TeacherAssistant(String scoreFilePathAndName, String gradeSystemFile) {
    Scanner scoreFileScaner = getScoreFileScanner(scoreFilePathAndName);
    gradeSystem = new GradeSystem(gradeSystemFile);
    populateStudentList(scoreFileScaner);
}
```

Where getScoreFileScanner method is something like given below:

```
private Scanner getScoreFileScanner(String scoreFilePathAndName) {
    File file = new File(scoreFilePathAndName);
    Scanner scanner = new Scanner(file);
    return scanner;
}
```

And the populateStudentList method should use the Scanner object to read student-by-student information from the csv file and as it goes create Student objects and add them into the 1D array of students (one of the class fields mentioned in Section 3.4.1.

3.4.4 Public methods

This class shall have 3 public methods (so they can be called by your main program TA_App):

- issueReportForAllStudents()
- issueReportForStudent(student NAME here)
- getClassAverageScore()

Instead of putting in words what those methods should do, please see the next pages printout to figure it out.

Method issueReportForAllStudents() outputs:

```
======================================
Name: student1
e-mail: student1
Exams: 100.0 83.0 71.0
Labs: 18.0 97.0 100.0 40.0 87.0
Assignments: 52.0 96.0 96.0
Final Score: 82.7
Final Grade: B-

======================================
Name: student2
e-mail: student2
Exams: 92.0 87.0 97.0
Labs: 67.0 33.0 29.0 33.0 19.0
Assignments: 89.0 54.0 56.0
Final Score: 83.9
Final Grade: B

======================================
Name: student3
e-mail: student3
Exams: 85.0 85.0 100.0
Labs: 37.0 96.0 83.0 4.0 69.0
Assignments: 63.0 67.0 79.0
Final Score: 84.7
Final Grade: B

======================================
Name: student4
e-mail: student4
Exams: 84.0 92.0 96.0
Labs: 52.0 83.0 35.0 88.0 90.0
Assignments: 59.0 74.0 70.0
Final Score: 86.3
Final Grade: B

======================================
Name: student5
e-mail: student5
Exams: 84.0 84.0 77.0
Labs: 96.0 14.0 50.0 52.0 98.0
Assignments: 73.0 62.0 65.0
Final Score: 78.2
Final Grade: C+
```

==========
Name: student6
e-mail: student6
Exams: 93.0 73.0 77.0
Labs: 90.0 79.0 27.0 61.0 100.0
Assignments: 68.0 92.0 95.0
Final Score: 80.4
Final Grade: B-

==========
Name: student7
e-mail: student7
Exams: 79.0 78.0 99.0
Labs: 79.0 95.0 6.0 7.0 5.0
Assignments: 79.0 56.0 66.0
Final Score: 78.8
Final Grade: C+

==========
Name: student8
e-mail: student8
Exams: 97.0 71.0 72.0
Labs: 15.0 59.0 9.0 99.0 77.0
Assignments: 56.0 99.0 51.0
Final Score: 76.0
Final Grade: C

==========
Name: student9
e-mail: student9
Exams: 76.0 70.0 87.0
Labs: 19.0 76.0 30.0 47.0 92.0
Assignments: 54.0 90.0 85.0
Final Score: 75.0
Final Grade: C

==========
Name: student10
e-mail: student10
Exams: 99.0 87.0 76.0
Labs: 86.0 8.0 60.0 97.0 94.0
Assignments: 74.0 60.0 80.0
Final Score: 83.9
Final Grade: B

==========
Name: student11
e-mail: student11
Exams: 75.0 88.0 97.0
Labs: 54.0 77.0 96.0 41.0 12.0
Assignments: 63.0 84.0 76.0
Final Score: 82.4
Final Grade: B-

==========
Name: student12
e-mail: student12
Exams: 95.0 96.0 79.0
Labs: 15.0 56.0 85.0 88.0 57.0
Assignments: 68.0 53.0 77.0
Final Score: 84.6
Final Grade: B

==========
Name: student13
e-mail: student13
Exams: 86.0 73.0 73.0
Labs: 29.0 54.0 80.0 33.0 52.0
Assignments: 99.0 55.0 66.0
Final Score: 74.2
Final Grade: C

==========
Name: student14
e-mail: student14
Exams: 100.0 74.0 72.0
Labs: 49.0 31.0 65.0 69.0 76.0
Assignments: 85.0 80.0 50.0
Final Score: 78.6
Final Grade: C+

==========
Name: student15
e-mail: student15
Exams: 79.0 88.0 84.0
Labs: 84.0 83.0 2.0 99.0 74.0
Assignments: 56.0 73.0 76.0
Final Score: 80.6
Final Grade: B-

==========
Name: student16
e-mail: student16
Exams: 97.0 80.0 79.0
Labs: 95.0 94.0 88.0 0.0 73.0
Assignments: 64.0 52.0 72.0
Final Score: 81.5
Final Grade: B-

issueReportForStudent(String studentName) outputs

Example:

By calling issueReportForStudent(“student10”) it should print info just for that student:

==========
Name: student10
e-mail: student10
Exams: 99.0 87.0 76.0
Labs: 86.0 8.0 60.0 97.0 94.0
Assignments: 74.0 60.0 80.0
Final Score: 83.9
Final Grade: B

getClassAverageScore() output

Class Average Score: 80.7

3.5 Back to TaApp Class (created on Section 3.1)

This class creates a TeacherAssistant object and gives to this new object (during its construction) the locations of the files containing the students scores and also the grade system that he/she should use to grade the students.

After creating this object, you just as him/her to issue the overall report (for all the students) and also a report for a single student that has called you last night with some questions about his grades.

Finally you ask for the overall class score, so you can tell if the student was above or below the average.

An example of such TaApp class is given below. By running this code you should get the outputs shown in the previous pages.

```
public class TaApp {

   public static void main(String[] args) throws FileNotFoundException {
      new TaApp();
   }

   public TaApp() throws FileNotFoundException {
      String scoreFilePathAndName = "C:/Users/Paulo/Documents/CS0401/labs/students_scores.csv";
      String gradeSystemFile = "C:/Users/Paulo/Documents/CS0401/labs/grade_scale.csv";
      TeacherAssistant TA = new TeacherAssistant(
            scoreFilePathAndName, gradeSystemFile);

      TA.issueReportForAllStudents();
      TA.issueReportForStudent("student1");
      String classAverageScore = TA.getClassAverageScore();
      System.out.println("Class Average Score: " + classAverageScore);
   }

}
```

The Students csv file contents:

```
student1,student1@pitt.edu,100,83,71,18,97,100,40,87,52,96,96
student2,student2@pitt.edu,92,87,97,67,33,29,33,19,89,54,56
student3,student3@pitt.edu,85,85,100,37,96,83,4,69,63,67,79
student4,student4@pitt.edu,84,92,96,52,83,35,88,90,59,74,70
student5,student5@pitt.edu,84,84,77,96,14,50,52,98,73,62,65
student6,student6@pitt.edu,93,73,77,90,79,27,61,100,68,92,95
student7,student7@pitt.edu,79,78,99,79,95,6,7,5,79,56,66
student8,student8@pitt.edu,97,71,72,15,59,9,99,77,56,99,51
student9,student9@pitt.edu,76,70,87,19,76,30,47,92,54,90,85
student10,student10@pitt.edu,99,87,76,86,8,60,97,94,74,60,80
student11,student11@pitt.edu,75,88,97,54,77,96,41,12,63,84,76
student12,student12@pitt.edu,95,96,79,15,56,85,88,57,68,53,77
student13,student13@pitt.edu,86,73,73,29,54,80,33,52,99,55,66
student14,student14@pitt.edu,100,74,72,49,31,65,69,76,85,80,50
student15,student15@pitt.edu,79,88,84,84,83,2,99,74,56,73,76
student16,student16@pitt.edu,97,80,79,95,94,88,0,73,64,52,72
```

The grade system csv file contents:

```
A+,97-100,4
A,93-97,4
A-,90-93,3.7
B+,87-90,3.3
B,83-87,3
B-,80-83,2.7
C+,77-80,2.3
C,73-77,2
C-,70-73,1.7
D+,67-70,1.3
D,63-67,1
D-,60-63,0.7
E/F,0-60,0
```