cis111-2023-3-project02

v2023-12-18

Content

Info

Submission

Due: You summit two places:

- 2023-12-27T22:30 via YULearn
- 2023-12-28T22:30 via CodeRunner

Warning.

- Submit *only* your `ProcessGradingData.java` *before due date* via *YULearn*.

Evaluation criteria

- This project will be evaluated *automatically*. Therefore, you should be careful in your submission.

  For example, the file name should be exactly `ProcessGradingData.java`.

- Make sure that your code passes all the tests in `Grading_jUnit_Test`.
- We may use some additional test in the evaluation.

Project

Description

In this project you are expected to

- parse student grades data in csv format
- create an array of `Student`
- define how to check equivalence of `Student`s.
- report the overall performance of the students.

Files

- `ProcessGradingData` processes student data. This the *only file* that
    - you make changes.
    - you submit via YULearn
- `Student` is the student object (do not change, do not submit).
- `Grading_jUnit_Test` is for testing (do not change, do not submit). If you want to add some more test, make a new version of it so that the original stays unchanged.
- `StudentUsage` is a helper, which contains examples of `Student` object.

1. First things first

- Make sure that in `ProcessGradingData`
    - your student number is in `MY_ID`.
    - your name is in `MY_NAME` and
    - your lastname is in `MY_LASTNAME` and
- `ProcessGradingData` is the only file you are allowed to make any changes. If you change other files, auto grading may fail.
- Submit only `ProcessGradingData.java` via YULearn. If you submit other files, auto grader may fail.
- If auto grader fails, you get 0 as your project grade.

2. `equals` method

Modify `equals`

```java
public static boolean equals(Student stdA, Student stdB)
```

in `ProcessGradingData` so that it returns

- `true` if given two `Student` instances are equal, that is, the corresponding elements such as `studentID` or `name` have equivalent contents.
- `false` otherwise.

3. `getWeightedGrade` method

Modify `getWeightedGrade`

```java
public static double getWeightedGrade(double[] weight, Student student)
```

in `ProcessGradingData` so that it returns the weighted grade of the student.

$$w = w_m g_m + w_h g_h + w_l g_l$$

where w_m, w_h, w_l are the weights, and g_m, g_h, g_l are the grades of midterm, homework and labs, respectively.

For example, if weights are [0.40, 0.10, 0.50] and grades of a student are [10, 20, 30], then weighted grade is 21.

4. `getStudentData` method

Modify `getStudentData` method

```java
public static Student[] getStudentData(String csv)
```

in `ProcessGradingData` so that it parses student data, and returns an array of `Student` instances.

Student data

Student data is in csv format.

A row corresponds to a student record. Each student record has six fields, namely,

- student number
- name
- lastname
- midterm grade (over 100)
- homework grade (over 100)
- lab grade (over 100).

The first three rows have metadata:

- The very first row is used for column titles.
- The second row is used to define the weights. Student number for this raw is set to be `-1`.
- The third row is used for the maximum possible grades. Student number for this raw is set to be `-2`. This is not currently used. You assume that the maximun possible grades are 100 for all grades.
- Student data starts at the fourth row.

The following is an example of a csv data, where there is only one student in it. A more complex example is `csvStudentDataA` in `Grading_jUnit_Test`.

```java
String csvStudentDataB = "studentID,name,lastName,midterm,homework,lab\n"//
            + "-1,-weight,in percent,35,20,45\n" // percentages
            + "-2,-full,point,100,100,100\n" // test for full points
            + "1007,Ccc,Cccc,40,20,30\n" // test for random points
;
```

5. `getReport` method

Modify `getReport`

```Java
public static String[] getReport(Student[] arrStudent)
```

in `ProcessGradingData` so that it returns a `String` array, where each entry is the student number and weighted grade of that student.

For example, `getReport` should return

```Java
String[] expected = { //
                "1001,50.0"//
                , "1002,10.0"//
                , "1003,0.0"//
                , "1004,40.0"//
                , "1005,10.0"//
                , "1006,50.0"//
                , "1007,2.1"//
};
```

when the student data read is

```Java
String csvStudentDataA = "studentID,name,lastName,midterm,homework,lab\n"//
            + "-1,-weight,in percent,40,10,50\n" // percentages
            + "-2,-full,point,100,100,100\n" // test for full points
            + "1001,Aaa,Aaaa,50,50,50\n" // test for 50% points
            + "1002,Bbb,Bbbb,10,10,10\n" // test for 10% points
            + "1003,Ccc,Cccc,0,0,0\n" // test for 0% points
            + "1004,Only,midterm,100,0,0\n" // test for midterm only
            + "1005,Only,homework,0,100,0\n" // test for homework only
            + "1006,Only,final,0,0,100\n" // test for final only
            + "1007,Ddd,Dddd,1,2,3" // test for random points
;
```