[CS209A-23Spring] Assignment 1 (100 points)

> Question Design: Yao Zhao
>
> Code sample & OJ: Yilun Qiu; Test cases: Yunxiang Yan
>
> Git & Code styles: Yunxiang Yan

Deadline: 11:55pm Mar. 29

Grace period: 48 hours. If you missed the deadline, you could still submit within a 48-hour grace period, i.e., before 23:55pm Mar. 31. Yet, you'll get a 40% penalty (i.e., you'll get 60% of your score). Submissions after the grace period will NOT be accepted.

Online Courses from edX Analysis

In this assignment, you will design an `OnlineCoursesAnalyzer` class, which could read an Online Courses dataset from edX and perform various useful analyses. You'll have a chance to use techniques such as `Collections`, `Lambda`, and `Streams`, which we've covered in the lectures.

The `OnlineCoursesAnalyzer` class has one constructor that reads a dataset file from a given path. This class also has 6 other methods that perform data analyses. You'll implement these methods in the `OnlineCoursesAnalyzer` class. Method details are described below.

0. Reading the dataset

```
public OnlineCoursesAnalyzer(String datasetPath)
```

The constructor of `OnlineCoursesAnalyzer` takes the path of the dataset file and reads the data. The dataset is in `csv` format and has the following columns:

- Institution - online course holders
- Course Number - the unique id of each course
- Launch Date - the launch date of each course
- Course Title - the title of each course
- Instructors - the instructors of each course
- Course Subject - the subject of each course
- Year - the last time of each course
- Honor Code Certificates - with (1), without (0).
- Participants (Course Content Accessed) - the number of participants who have accessed the course
- Audited (> 50% Course Content Accessed) - the number of participants who have audited more than 50% of the course
- Certified - Total number of votes
- % Audited - the percent of the audited
- % Certified - the percent of the certified

- % Certified of > 50% Course Content Accessed - the percent of the certified with accessing the course more than 50%
- % Played Video - the percent of playing video
- % Posted in Forum - the percent of posting in forum
- % Grade Higher Than Zero - the percent of grade higher than zero
- Total Course Hours (Thousands) - total course hours(per 1000)
- Median Hours for Certification - median hours for certification
- Median Age - median age of the participants
- % Male - the percent of the male
- % Female - the percent of the female
- % Bachelor's Degree or Higher - the percent of bachelor's degree of higher

Note: For each individual question, if the data cells required for this particular question are empty or ill-formatted, you could simply ignore that entire row for this particular question.

1. Participants count by Institution (10 points)

```
public Map<String, Integer> getPtcpCountByInst()
```

This method returns a `<institution, count>` map, where the key is the institution while the value is the total number of participants who have accessed the courses of the institution.

The map should be sorted by **the alphabetical order of the `institution`**.

2. Participants count by Institution and Course Subject (10 points)

```
public Map<String, Integer> getPtcpCountByInstAndSubject()
```

This method returns a `<institution-course Subject, count>` map, where the key is the string concatenating the Institution and the course Subject (without quotation marks) using '-' while the value is the total number of participants in a course Subject of an institution.

The map should be sorted by **descending order of `count`** (i.e., from most to least participants). If two participants have the same count, then they should be sorted by the alphabetical order of the `institution-course Subject`.

3. Course list by Instructor (20 points)

```
public Map<String, List<List<String>>> getCourseListOfInstructor()
```

An instructor may be responsible for multiple courses, including independently responsible courses and co-developed courses.

This method returns a `<Instructor, [[course1, course2,...],[coursek,coursek+1,...]]>` map, where the key is the name of the instructor (without quotation marks) while the value is a list containing 2-course lists, where **List 0** is the instructor's independently responsible courses, if s/he has no independently responsible courses, this list also needs to be created, but with no elements. **List 1** is the instructor's co-developed courses, if there are no co-developed courses, do the same as **List 0**. Note that the course title (without quotation marks) should be sorted by alphabetical order in the list, and the case of identical names should be treated as the same person.

4. Top courses (20 points)

```
public List<String> getCourses(int topK, String by)
```

This method returns the top K courses (parameter `topK`) by the given criterion (parameter `by`). Specifically,

- `by="hours"`: the results should be courses sorted by **descending order** of `Total Course Hours (Thousands)` (from the longest course to the shortest course).
- `by="participants"`: the results should be courses sorted by **descending order** of the number of the `Participants (Course Content Accessed)` (from the most to the least).

Note that the results should be a list of Course titles. If two courses have the same total Course hours or participants, then they should be sorted by alphabetical order of their titles. The same course title can only occur once in the list.

5. Search courses (20 points)

```
public List<String> searchCourses(String courseSubject, double
percentAudited, double totalCourseHours)
```

This method searches courses based on three criteria:

- `courseSubject`: Fuzzy matching is supported and case insensitive. If the input`courseSubject` is "science", all courses whose course subject includes "science" or "Science" or whatever (case insensitive) meet the criteria.
- `percentAudited`: the percent of the audited should >= `percentAudited`
- `totalCourseHours`: the Total Course Hours (Thousands) should <= `totalCourseHours`

Note that the results should be a list of course titles that meet the given criteria, and sorted by alphabetical order of the titles. The same course title can only occur once in the list.

6. Recommend courses (20 points)

```
public List<String> recommendCourses(int age, int gender, int
isBachelorOrHigher)
```

This method recommends 10 courses based on the following input parameter:

- `age`: age of the user
- `gender`: 0-female, 1-male
- `isBachelorOrHigher`: 0-Not get bachelor degree, 1- Bachelor degree or higher

First, calculate the average `Median Age`, average `% Male`, and average `% Bachelor's Degree or Higher` for each course. Note that `Course Number` is the unique id of each course;

Secondly, the following formula:

$similarity value= (age -average Median Age)^2 + (gender*100 - average Male)^2 + (isBachelorOrHigher*100 - average Bachelor's Degree or Higher)^2$

is used to calculate the similarity between the characteristics of the input user and the characteristics of each course's participants. The higher the similarity, the smaller the value;

Finally, return the top 10 courses with the smallest similarity value.

Note that the results should be a list of course titles. A `Course Number` may correspond to different `course titles`, please return the `course title` with the latest `Launch Date` and the same course title can only occur once in the list. The courses should be sorted by their similarity values. If two courses have the same similarity values, then they should be sorted by alphabetical order of their titles.

Evaluation

- **Code Correctness**: We deploy automatic test cases on the OJ system (https://oj.cse.sustech.edu.cn) to test the correctness of your code. Please submit `OnlineCoursesAnalyzer.java` to OJ. You could use our sample test cases and sample test data to test your code locally before submitting it to OJ.
- **Version Control**: You should use `GitHub` to manage the code changes of your project (see lab 1 for further details on how to use `git`). You should make **at least 2 commits** (2 commits are recommended). Your remote repo on GitHub **should be set to `private` before A1 deadline, so that no one else will see your code.**
- **Coding Style**: You should pay attention to writing readable and maintainable code along the way. See lab 1 for how to use `CheckStyle` for that purpose. **After the deadline of A1**, you should set your GitHub repo to `public`, since we'll check whether any of your commits have reduced `CheckStyle` warnings according to `google_checks.xml`.

OJ Tips

- **Please specify UTF-8 coding when you read the `.csv` file.**
- **Please don't include any Chinese characters in your code comments.**
- **Please using `int` for integer numbers and `double` for floating numbers.**