1 Assignment

Summary: Write code that parses the provided course grade JSON data and then write a collection of utility functions that allow you to filter and summarize the data in various ways.

Fork the starting git repository using the URL below. You'll note that the provided repository doesn't include files for the code that you are supposed to write. Instead, you should create those files and name them appropriately. We do provide a collection of JSON files in the 'data' directory and a Data class that provides some utilities for loading those JSON files into your program.

`https://classroom.github.com/a/YgCMxgZP`

Step 1: Parsing the JSON: Use the Gson library, as demonstrated in lecture on Thursday, to parse the provided JSON. We recommend that you first focus on parsing individual course JSON objects, by writing tests where you parse them from Strings that are hard-coded in your test class (i.e., cut-and-paste one course from one of the JSON files). Once you can successfully parse a single course's data, test parsing a small JSON array of courses (e.g., again cut-and-paste a small collection of courses' JSON wrapped in square brackets).

Step 2: Loading JSON from Files:

Note: Java's libraries for reading and writing files are not so straight-forward, so we've provided you some utilities in a Data class. Please read through this code so that you understand what it is doing, and for a good pattern for creating ArrayLists from Java arrays.

Use the provided Data utilities to create your own utility function(s) that load the provided JSON files and parse the JSON they contain to produce Java objects. Specifically, the methods you create should include one method that can be used to load all of the JSON files into a single ArrayList of your objects. (Be sure to write tests that validate that these functions are working.)

Step 3: Filtering methods Write a collection of utility methods that take a collection of parsed objects and return a new collection that is the subset meeting a given criterion. Your methods should be designed in such a way that you can take the output of one method and pass it in as the input of another method, in any order. You should write at least 6 such methods, including the following:

1. returns all of the courses from a given department or subject
2. returns all of the courses whose instructor's name contains a given String
3. returns all of the courses with a course number in a given range (e.g., $100 - 199$, inclusive)
4. returns all of the courses with a number of students in a given range (e.g., < 50, $20 - 200$, > 400)

The other two methods can be of your choice. Have fun with them. Do something interesting!

Step 4: Aggregation methods Write a collection of methods for aggregating data across a collection of courses (that are potentially a filtered subset of all of the courses, using the above methods). You should write at least the following three methods:

1. returns the total number of students that took the given collection of classes
2. returns the number of students receiving a given range of grades (e.g., B+ to C-)
3. returns an arithmetic mean of the course GPA averages weighted by course enrollment

Together these methods can be used to figure out what fraction of students taking a given selection of courses get given ranges of grades.

Process: In the above, we've provided you 4 progressive steps for breaking down this assignment. Each of these is a small piece of functionality that you can (1) write tests for, (2) implement and get working, and (3) commit and push to github. We expect to see these commits.

Design and Style: As always, use the best design and coding style that you are familiar with. In particular, for this assignment, continue to focus on using good names (you have a lot of freedom in naming in this assignment) and code layout (as directed by Chapter 4 of the text book and section 4 of the Google Java style guide).