Code Review 2 – Rating Analyzer

The due date is Friday, 6/24/2022 at 7am PT. Submission details later in this document.

Overview

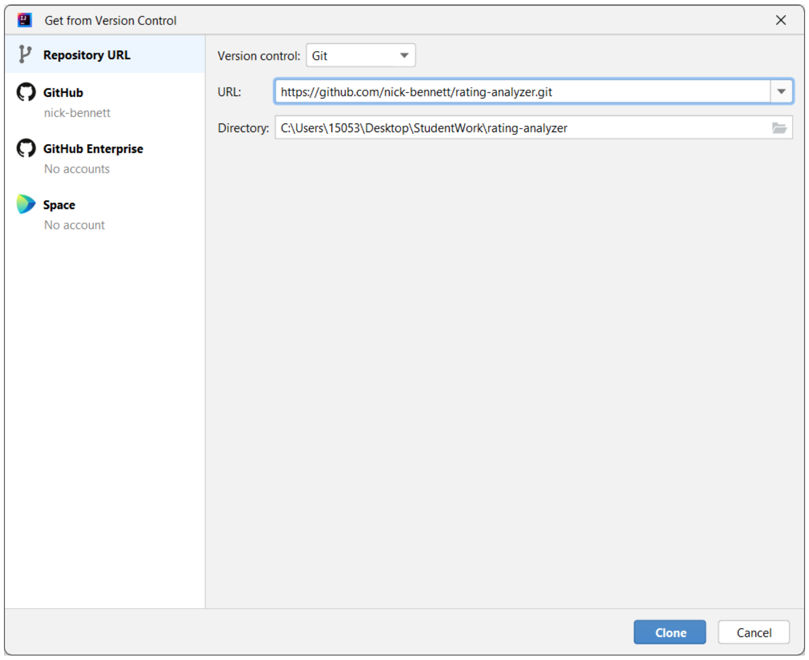
We have provided a specification for a theoretical product: **JAMS** 1.3 – the **Java API for Mathematical Statistics**. The spec consists of an interface definition and API documentation for package `org.stats`. You are in the role of ***provider***, which means you are to implement this specification. (In practical terms, this simply means you will write an implementation class for the `org.stats.RatingAnalyzer` interface.)

Your submission should include a `RatingAnalyzer` implementation class, along with any supporting classes (if any). Your class(es) should (of course) be in a package, e.g., `com.yourname.stats`. **Your solution should also include unit tests.**

In all, the code you write is likely going to involve only two classes: your implementation class and a test class.

Getting Started

All resources needed to get started are in GitHub, at the Git URL https://github.com/nick-bennett/rating-analyzer.git. Clone this repository and create a project in IntelliJ by using the **Get from VCS** button (in the **Welcome** screen), or the **File/New Project from Version Control** menu command in the IntelliJ workspace, and paste the URL above into the **URL** field. (Please make sure the project is being created in a directory that makes sense, so you can find it later!) The screen capture below shows the dialog as it appears on my screen.



Version control: Git

URL: https://github.com/nick-bennett/rating-analyzer.git

Directory: C:\Users\15053\Desktop\StudentWork\rating-analyzer

Alternatively, the contents of the repository (along with these instructions) are in the `rating-analyzer.zip` file in the **Files** tab of the Teams **1 – Java** channel, in the **Code Review/Rating Analyzer** folder.

Implementing the Specification

Read the API docs! Start with the `docs/api/index.html` file in the project (if that isn't obvious); however, for best results, open this file from Windows File Explorer or OS X Finder.

You probably won't find the Javadocs overwhelming, but you **absolutely must read them**. After reading them, proceed with your coding. Your implementation approach is up to you – i.e you can use arrays, collections, the `java.util.stream` package, whatever you want. However, **it must be your own work** – this is an **individual assignment** to verify that you can solve basic algorithm problems and write clean, well-formatted, well-structured Java code that adheres to standard naming conventions and fulfills basic OO principles.

Avoid any temptation to copy code from *Stack Overflow* or other websites. While we don't want to spend time tracking down plagiarized solutions, we do have a code copy detection tool that we can use to readily determine if your code is too similar to code posted publicly elsewhere. Besides, this is a good opportunity to practice the very same things you'll be doing at Amazon, while getting the satisfaction that, "Hey, I did it!"

Implementation Notes

There are several websites that will calculate these statistics, so you can quickly enter in a sample dataset and then use the results as the expected values for assertions in your unit testing. This is a good one: http://www.calculator.net/mean-median-mode-range-calculator.html

Your test class should **only** be coupled to the `RatingAnalyzer` interface, i.e., it should not have any knowledge of your exact implementation class. The static factory method `RatingAnalyzer.newInstance()` reads the `rating-analyzer.properties` file in the `src/main/resources` directory and instantiates your implementation class using a process called *reflection* (the details of which you don't need to be concerned with). Therefore, in your test class, you can simply call this method to get an instance of your analyzer for testing. **Repeat:** This static factory method is already implemented in the interface itself, you **do not** need to code it. **However**, you should read the API docs for it.

Instantiation example (you can use something like this in your test methods):

```
int[] ratings = { 3, 5, 2, 3, 4, 1, 3, 4, 3 };
RatingAnalyzer analyzer = RatingAnalyzer.newInstance(ratings);
```

In your project, you need to ensure that the `rating-analyzer.properties` file can be located by the `RatingAnalyzer.newInstance()` method, which is looking for it in the classpath; by default, any files in the `src/main/resources` and `src/test/resources` directories will be on the classpath during testing.

You will also need to ensure that the contents of the `rating-analyzer.properties` file are correct. Java properties files consist of key-value pairs, separated the equals sign (`=`), a colon (`:`), and/or whitespace. So, if your implementation class is called `MyRatingAnalyzer` (for a somewhat silly example), and it's in the `com.mystuff` package, you would modify line 2 of the `rating-analyzer.properties` file to read:

```
rating.analyzer=com.mystuff.MyRatingAnalyzer
```

Evaluation Criteria

We will run our tests against your code, and they will need to pass. We have been very thorough in our test cases, as you should be in your own testing. We will also take a brief tour of your implementation class(es), to verify that your code meets the criteria set forth in "Implementing the Specification" above. Our test cases will use ratings in the range 1–5, but your implementation should be able to handle any integer values.

Submission Instructions

You must zip up your solution and upload it to MS Teams, *to your private channel's **Files** tab*, by the deadline stated at the top of this document. **No late submissions will be accepted.** The zip file should be named `rating-analyzer-`*`yourname`*`.zip.` Use your common name for the name portion of the filename, e.g., "Pat," "Kuma," "Xander," etc. **No spaces in the filename**, please. You must submit a `.zip` file, **not** the proprietary `.7z` format if you're using 7-Zip.

This is very easy to do. On the filesystem, navigate to the location of your `rating-analyzer` directory, right-click on it → 7-zip → Add to "`rating-analyzer.zip`".
Then simply rename the file to include the "`-yourname`" portion. On OS X, proceed similarly.

Getting Assistance

The instructor will help you with any structural issues you have getting started, but cannot give any implementation advice. We believe this assignment is on the right level of challenging, yet very attainable, **if** you take the time to follow the instructions, read the API docs, and get started earlier than later.

Good luck!