Project Report

By:
Hannah Wu
Tim Wang
Team-030

5.1 Additional Feature

We created an additional feature calculateFullVacRateAndAveragePropertyValue(String date) that calculates, for each zip code, the number of fully vaccinated people per 1000 capita as int, and the property value per capital as int. The results are displayed in ascending order of the zip code, for a given date. Users are prompted to enter a date of 'YYYY-MM-DD' format.

5.2 Use of Data Structure

First, I used a Hashset of String to store availableDatasets. This is created in Main class and passed to UI class to record what datasets were provided at runtime. Inside UI's constructor, Set<String> availableDatasets is used to help populate values in List<Integer> availableActions, so that if a user choose an action that is not available, an error message will be displayed and user will be reprompted.

I chose Set because I frequently used contains method in UI constructor, and the time complexity for Set contain method is O(1). I used Hashset specifically because it's more efficient. I could've also used List since the number of arguments is small, and the time complexity for List contain method will be very small still.

Second, I used Map<String, String> seenArgs to store the arguments that were provided at runtime, specifically, the name of the argument and its value. This is created and used in Main class to do two things: 1) make sure that no duplicate arguments are provided at runtime, and 2) initialize the file readers with the filenames accordingly for creating a Processor Object.

I chose Map because I need to store two elements: name and value of the arguments, and because it cannot have duplicates and I need to look up and key often. Map provides the most efficient method for this with a time complexity of O(1) for containsKey(), and get(). I used Hashmap because I do not care about order. For this, I did not consider any alternatives.

Third, I used Map<Integer, List<Integer>> to store the results of my custom method calculateFullVacRateAndAveragePropertyValue(). It is used to store for each zipcode, and a given date, the number of fully vaccinated people per 1000 capita, and the property value per capita.

I used TreeMap because I want to store zipcode in ascending order. I used List inside the Map because I have more than one element to store associated with each zip code. I also considered creating a class to store all three, but Treemap with List seems like a more straightforward solution.

5.3 Lessons Learned

We started a project, and push the initial framework to Github for version control, as well as for sharing code. We communicated over Wechat. Specifically, we had an initial call to divide the work. We prioritized that for the Classes assigned to each person, we need to first write the signature of all methods and exposed them to the other person by pushing all code to Github, so that the other person knows how to consume that class and method. For example, Tim provided the method signatures of all Reader classes, so that Hannah knew how to consumes those classes inside Processer. This turns out to be highly effective.

We also communicated throughout the project with major updates. Each person is to let the other person know of when they will work on their tasks, so that we can make sure all changes have been pushed to Github for the person to pull. Overall, it was seamless and pain-free. The debugging process was minimum as each person ensured that their classes and methods are well-written and tested.