Additional Feature

The additional feature that we incorporated was the total market value for a zip code divided by the number of individuals in that zip code divided by the number of full vaccinations in that location. This uses property, covid, and population information files to process and compute these values, retrieving the cumulative property values, full vaccination numbers, and population numbers to calculate our value. To test whether it was working, we calculated some of the values for a few zip codes manually and saw that the results were approximately the same. We also examined some of the other zip code results and determined that they appeared reasonable based on our expectations, producing results like 0.42, which appears to be realistic.

Data Structures

In the main method, we used a HashSet known as argsTraversed to keep track of the command line arguments and to check for potential repeats in the file type. When adding to argsTraversed, if the file type is already present in the set, then an exception is thrown that there is a repeat for a particular file type. The reason a set was used is because a set doesn't contain duplicate elements, so this is a good way to check for duplicate file types. HashSet has a O(1) or constant lookup and insert time. This makes it fast when checking for duplicates. An ArrayList would be slower as in the worst case it would have to iterate through the entire structure or would have O(n) lookup time.

Starting in main, a HashMap called data was used to return and store the values after reading from files. Each returned HashMap could be passed subsequently on reading calls to input the data from previous calls and to add to the HashMap. After all the file reading was done, then the HashMap, along with the instantiated Logger and argsTraversed Set were passed to UserInterface to instantiate an instance, ui. HashMap was used here as the unique zip codes are the keys which map to Area objects. Lookup is O(1), allowing us to retrieve a zip code's associated information very quickly. A Tree data structure would be slower to set up as it would need to be organized. ArrayList would also be slower to search for a particular zip code with O(n) time.

Finally, we used a LinkedList in the readRow function in CSVReader to have an expanding list of rows from a csv file specified in the command line arguments. One reason for this is that we don't know beforehand how many lines the input file will contain. After all the rows from the file are added, then the LinkedList is converted to an Array of Strings and returned. A LinkedList works well for adding to the end of the data structure, with a time complexity of O(1). LinkedList is better than a regular Array because we can append to it and grow it

easily. Arrays, conversely have fixed size. LinkedList also works better in this use case than an ArrayList as appending to an array would be amortized O(1) but it would be slower as the underlying Array would have to be copied to a larger one intermittently.

Lessons Learned

We used Eclipse as our editor and set up the project there. Initially there were some difficulties getting the packages standardized and running properly in the environment. It took some setting of the build path and run configurations to get this working properly. We used git to track our changes and GitHub to share the project with group members. For communication Slack was mainly used along with Google Meet for group meetings where we discussed progress and next steps. We initially divided up the work and then met through Google Meet for progress reports and next steps. We also would message through Slack between meetings to update each other on our progress and to share information and iteratively discuss what part of the project needed improvements.

Zayd: I faced difficulties early on with the execution of the program. When I had used premade projects before, the project was prepackaged and easy to run on my IDE, however, here I had to learn to troubleshoot and understand my partners project set up and find ways to make it work more flexibly. The challenge for me was collaborating and working together on a project through tools like git and GitHub and learning how to make it work on my system. At first, this was challenging and took a lot of trial and error, however, once I had a process it was easier to quickly fix the libraries in the build path and output folder. So, there was time required up front to get the IDE working but after I better understood the project structure it became easier to troubleshoot. I also thought it was a helpful exercise to work to collaborate with my partner and ask him questions to understand his code and why he structured it that way.

Krithik: I mainly faced issues with the set-up of the program as well, since the autograder was set to work on a different JDK that I was used to, which took a lot of time to troubleshoot. Eclipse is also difficult to integrate with Git, so in the future I will consider other tools like IntelliJ as well. I will also more carefully examine the defined requirements in the project description before I start coding and designing, to avoid wasted time debugging later.