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 properties in that location. Or the total market value divided by the number of properties multiplied by the number of individuals in that zip code. This uses property and population information files in order to process and compute these values. (How we know it is working correctly).

Data Structures

In the main method, we used a HashSet known as argsTraversed in order 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. Also starting in main, a HashMap called ret 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 are passed to UserInterface to instantiate an instance, ui. (Explain why HashMap was used for the data. currMemo ArrayList in AllZipCodes is another option. Edit: Already go on to describe ArrayList in readRow in CSV Reader) ArrayList was used in the readRow function in CSVReader in order to have an expanding list of rows from a csv file specified it the command line arguments. After all the rows from the file are added, then the ArrayList is converted to an Array of Strings and returned. ArrayList was used instead of a regular array because when we are reading the file we don’t know the total number of rows beforehand so a flexible data structure like an ArrayList is more suitable while the reading is taking place. (Maybe talk about why you would use ArrayList over LinkedList).

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.

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 collaboratively 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 later on once I was familiar with the issues in this project it was faster to troubleshoot. (Maybe optional - I also found it tricky initially all of my partner’s code and problem solving. However, after questioning his methodology, it made more sense) (Not sure also how should we frame who is speaking etc. Maybe I could just write Zayd for my reflections and you can put your name if you have any additional ones you want to include.)