Project Report

Team: Iqbal Sherif, Sarvesh Mathiyazhagan

Additional Feature

We implemented a feature that finds the zip code with the lowest market value per capita and gets the total value of fines for that zip code. The output will look like this:

19103 4560

The first part is the zip code with the lowest market value per capita and the second part is the total dollar value of all the fines for that zip code truncated as an *int*. If there is no fine data for the zip code, the second part will be 0.

This method considers only the zip codes that have population data in the "population.txt" file. The data for market value is obtained from the "properties.csv" file and the data for fines is obtained from the "parking.csv"/"parking.json" file.

To verify that the first part (zip code with lowest market value per capita) works correctly we used a much smaller variant of the properties dataset in which we could manually find the zip code with the lowest market value per capita. For the second part (total values of fines for zip code) we used Excel to filter all fines for the specified zip code and calculated the sum of all the fines.

Use of Data Structures

1. *HashMap<String, Integer>* - We used a *HashMap* to represent the data in the "population.txt" file by mapping the zip code to the population. We chose this data structure rather than making a *List* of custom Population objects because this structure allows for O(1) access time. Since, many of the specified operations require us to access the population for a given zip code, the O(1) access time is much better than having a list and iterating over Population objects and checking if the zip code equals the target zip code and then getting the population value from this object.
2. *double* - This may seem like a trivial choice, but we chose to include it rather than talk about Lists or Sets for the following reason: When computing the Average Market Value and Residential Market Value Per Capita we initially used *int* to compute the total value for the specified zip code and then divided that by the number of properties and population respectively. But we were getting negative values which is not possible. We then realized that the max value for *int* in Java is 2,147,483,647, but total market value for all properties in a zip code far exceeds this limit. By changing the data structure to a *double*, our problem was solved because *doubles* have a max value of 1.7*10^308, which is more than sufficient.
3. *TreeMap<String, Double>* - For the second operation (Total Fines Per Capita) we were required to print the zip code and total fines per capita (ex: 19103 0.0284) for all the zip codes in the "population.txt" file. Our first instinct was to output a *HashMap<String, Double>* that maps zip code to total fines per capita. But since the program specifies that the output must be ascending numerical order we went with *TreeMap,* which stores the keys in sorted order, adhering to the requirements of the project.