Assignment: Team 102 Project Report

Due Date: 28 April 2021

Members: Ramin Khorasany & Daniel Marvin

Additional Feature:

For our additional feature in question 6, we computed the fine per capita times the number of houses in a specified ZIP. This uses each data set. To get the fines per capita, similar to question 2, we needed to use the population file to find the population of a specific ZIP code, and parking file to find the number of fines that were charged in the ZIP code. To get the number of houses in the ZIP code, we needed the properties file. Thus, finding the product of houses in an area code and fines per capita in that same area, we used all three data sets. We know the number is correct (aside from showing the tests) because the number should be quite a bit greater than 0 since we multiplied it, whereas, if you divided it, it would be very close to 0.

Data Structure 1: CSVPropertyReader, CsvViolationReader, JsonVilationReader store their data in ArrayList.

To read in each of the necessary files, we wrote reader classes for each possible file (with the excpetion of PopulationTextReader, which used a different data structure that will be discussed later). We needed to store just a single object per line of data for each file with no other requirements. It was clear that a list would suffice her since we needed no specific associations to any of the elements. Upon choosing between a LinkedList and an Array List, we needed to weigh our options. Since both traverse the whole data set in O(n) time, we needed to find a different reason to pick one over the other. An advantage of LinkedLists over ArrayLists are that one can easily insert new data. An advantage of ArrayLists over LinkedLists are that they do not require as much space since they lack a pointer to the next element. As subsequent usage of these lists would require us to iterate through the whole list regardless, and we would never need to insert new data into them, we chose the ArrayList.

Data Structure 2: PopulationTextReader stores its data in a HashMap.

Unlike the other readers, the PopulationTextReader only had two data elements: a ZIP Code, and its population. Because of this, it made sense to directly store the data in a Map that could easily allow for retrieval of specific elements later on. As we did not care about the order, and cared only about efficiently retrieving data associated with that ZIP Code, a HashMap fit the job perfectly. HashMaps use HashCodes to uniquely associate a key with a value and allow for O(1) retrieval times similar to ArrayLists. This let us retrieve the data associated with the provided ZIP code without even traversing the Map. Because of this, a HashMap made the most sense.

Data Structure 3: TotalFinePerCatipa stores its final data in a TreeMap.

For question 2, we wrote a TotalFinePerCapita class that contained a TreeMap as an attribute. It was clear that we needed a Map since we needed to associate a ZIP Code to a violation average. We wanted specifically a TreeMap since one of the requirements, besides getting the violations per ZIP code, was printing them the system.out in ascending order. Since a TreeMap stores the information is ascending order, this data structure made the most sense to use.