CS 122 – Homework # 8 – “Warehouse” – due Tuesday, April 30, 2019

The purpose of this program is to review and practice several Java concepts you have seen throughout the course.

Suppose a company has just been formed and is preparing to start business. The company is in retail trade. The management would like to purchase or build several warehouses around the country to store their merchandise. A critical factor in determining the locations of these warehouses is to minimize the distances from these warehouses to the general public.

Every household in the United States is a potential customer. And for each customer, it would be possible to estimate the distance from that household to its nearest warehouse. The company wants to minimize the average distance that a package would have to travel from a warehouse to a customer. You will write a program to perform this calculation. The output of your program will be a list of (hopefully) optimal locations to select for the warehouses.

In order for this program to work, we need appropriate input data. We need input data that accurately reflects the geographical distribution of the population. On the class Web site, you will find a file called tract.txt that you should use as input for your program. Each line of tract.txt shows a census tract, which is, on average, a neighborhood of approximately 4000 people.

Here is an example of a line from tract.txt:

```
01001 AL Autauga County      020100  1912  1601   217    44 32477 086490 2424
```

There are basically two things you want to know about each tract: its population and its precise location. (The name of the tract is also nice for output.) If you treat this line as a string in Java, then the substring with indices (6, 29) gives you the name of the census tract. Inside the substring (29), if you tokenize on a space, the second token is the population of the tract. In the example above the population is 1912. The next 3 tokens can be ignored for this assignment. The last 3 tokens on the line give geographical information: the latitude, the longitude, and the land area of the tract. You need the latitude and longitude to give you the location of the tract. The latitude and longitude are expressed in thousandths of a degree. So the above example is located at 32.477 North, 86.490 West.

Each census tract represents a possible location for a warehouse. Suppose we want 10 warehouses around the country. How should we find the best 10 census tracts to choose for our warehouse locations? It turns out that a brute force exhaustive search would take too long! So, instead, your program will repeatedly select 10 tracts at random from around the country. For each random selection of 10 locations, the program will compute the average location from a person to the closest warehouse. Assume that everyone in America lives at the latitude and longitude given for each census tract. For example, in the census tract given above, we will assume that all 1912 people live at the location (32.477, 86.490). This is a useful approximation for the sake of the program.

It's unlikely that one random selection of 10 warehouses will be optimal! Therefore, your program will repeatedly make random selections of warehouse locations. Keep track of the best set-of-10 that you find. I recommend performing 25,000 random selections in order to keep the runtime of your program reasonably short.

Here is a recommended outline of your program. You should have a source file called Driver.java that contains the core algorithm:

```
Read tract.txt, and create an array of Tract objects.
Create an array A of 10 Tract objects representing the optimal
selection.
for trial = 1 to 25,000:
    Randomly generate a selection S of 10 Tracts.
    for each Tract t in the country:
        d = shortest distance from t to a Tract in S.
    Compute the weighted average of all values of d
      (i.e. weighted by the population of t)
    If this weighted average distance is less than the distance
      for the array A, then reset A to the selection S.
      Keep track of its weighted average distance,
      and also the trial number where this minimum was found.
    if trial is a multiple of 200, print onto the screen
      the current value of A and the corresponding average distance
      so that the user can see the progress of the algorithm.
```

The tract class should include attributes for the name, population, latitude and longitude. It should have an initial-value constructor, a get method for each attribute, and a toString(). The toString() method should display the latitude and longitude to 2 decimal places, as well as print the name of the tract.

When you print out your array of Tracts representing warehouse locations, you should print out the array in order from east to west. In other words, in ascending order of longitude. Therefore, I recommend that you write a comparator class that compares longitudes of a tract or warehouse.

Here is a hint about computing distances. To simplify the calculation, assume that the earth is flat, and that each degree of latitude or longitude is exactly 60 miles. Then, you can simply use the Cartesian distance formula between two points.

Your program will be run at the command line. By default (`java Driver`), you should assume that the user wants exactly 10 warehouse locations. But the user may supply a command-line argument specifying a different number, for example `java Driver 7`. You should notice that your program will take less time if there are fewer warehouses. So, in the early stages of development, you may want to just experiment with small input sizes so you are not waiting too long for your program to finish.

Here is an example final output for 5 warehouses and 25,000 trials. Your results may differ due to random luck!

```
trial   5110/ 25000 =  266.25 miles
   1 40.86  73.45  5070 NY Suffolk County       110102
   2 34.09  83.42  5494 GA Jackson County       010500
   3 41.84  88.04  3654 IL DuPage County        842709
   4 32.86  96.78  2469 TX Dallas County        007906
   5 34.26 118.52  7087 CA Los Angeles County   111301
```