**Introduction**

This assignment lets you read data from a file again but store it in a tree-based data structure, which allows for faster lookups.

This assignment builds a class to create, populate and search for items in different data structures then compare the execution time of each implementation.

**Directions**

Rewrite your class WeatherReport from the previous unit so that it calculates the statistic you proposed in the discussion. If you didn’t get that piece of the discussion, you may implement someone else’s proposed statistic, but at a loss of points. Replace the Temperature class with a new class that contains data you need, but nothing else.

1. Constructor with no parameters. It should create a list of 8-10 objects which you can use for testing.
2. Constructor with one String parameter – the name of a text file with weather information. This is the same as for the previous assignment, except storing the data in a different object, since your statistic will need different pieces of the input data. If you were to implement my example of lowest temperature at each weather station, your object would contain the weather station name and the low temperature.
3. computeByList() – this method has no parameters, accesses the data created by the constructor, and computes your statistic by creating, searching and updating a new List (either LinkedList or ArrayList. It should return the List that was created. Using my example, the returned list would contain [‘BHM’ 27, ‘HSV’ 25, ‘MOB’ 35 …]
4. computeByTree() – this method has no parameters, accesses the data created by the constructor, and computes your statistic by creating, searching and updating a new TreeMap. It should return the TreeMap that was created. As in the previous step, the TreeMap should have the same elements and values, though when you print the TreeMap you will probably get the items in a different order. If your items are not the same, then something is amiss.
5. main – test your class by writing a main method that creates at least two WeatherReport objects and computes the run times of calling computeByList() and computeByTree()for each one. Then print each of the returned objects. The statistics computed should be the same for each method.
6. Answer the following questions in your Word document:
   * Describe the state of your project, what works and what doesn’t.
   * Describe how you tested your program, since the data file is large
   * Show your timing data and write a paragraph on the benefits/drawbacks of using a List vs. a TreeMap
   * In a sentence or two, what did you learn?
   * In a sentence or two, what did you like about this project?
   * In a sentence or two, what did you find confusing or would like to see done differently regarding this project?
   * In a sentence or two, if you had another hour or two, what would you like to add to the project or how would you do things differently?

**Notes**

After your program is working for each step of the process, you should commit your changes to the repository. This allows you to go back to a previous working version in case you decide to throw out new code and try a different approach.

**Submit**

Be sure to put your Word document in your repository and do one final commit. Generate a .zip file from the repository and submit that file to Canvas. Double check that what you submit to Canvas has all your .java files.

**Due Dates**

* by 11:59 p.m., Sunday, CT