Project 5 (The Last One):
New York City Restaurant Inspection Results
Using Priority Queus, Trees and Maps to Complete Some of the Tasks
Due date: May 3, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating. If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

In this project you will continue working on the code that you developed for project 1. The objective in this project is to improve the performance of the sorting algorithms.

Objectives

The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- working with code developed previously,
- using data structures provide by Java API,
- understanding data structures and their performance in order to select an appropriate data structures for given tasks.

Your implementation of the code for projects 1, 2 and 3 used linear structure of an array and a binary search tree to store all of the inspection records. In this project you should use the implementation of other data structures that we discussed (provided by Java API classes) to store the inspection records when performing some of the tasks. Note that objects of these classes should not be complete replacements for the `MyArrayList` class or `BST` class that you developed previously. As in project 4, they should be used “in parallel” with the current data structures to facilitate faster/better processing of some of the tasks.

Program Input and Output

The inputs and outputs for this project are exactly the same as in project 4.

Data Storage and Organization

In this project, you will be modifying the implementation of your `findByNameAddress`, `findByScore` and `findByDate` methods in the `ListOfRecords` class.
Computational Task

In previous projects, your implementation of the find... methods iterated through the entire list of inspection records and in some cases, the list needed to be sorted as well. At this point in the course, you probably realized that these were not the most efficient implementations.

If the data was organized in a priority queue (implemented using a min-heap) based on the inspection score, then findByScore method could extract the needed records much faster. The added complication (or possible simplification) is that the records we are looking for need to come from a particular zip code. You may need a hash table of priority queues.

For the findByNameAddress command we need to access data easily when given name and address. These two fields identify the restaurant uniquely, but there are many possible inspection records associated with each restaurant. Hash tables provide fast access to data but only if keys are unique for each <key, value> pair. Using an auxiliary class similar to the DBAList we can make such mapping unique.

There are also ways to improve the design of findByDate method. This is left for you to discover on your own.

Classes and Data Structures

You do not need to implement any additional data structures for this project. Use the implementation of data structures that is provided by Java API. You may need to implement your own auxiliary classes (similar to DBAList from the last assignment)

The details of data organization and implementation are up to you.

As with project 4, the data structures should be created at the same time as the ListOfRecords object is populated.

For the three methods that you revise in this project, include a paragraph describing why your change in the code is beneficial. Write those comments in the Javadoc comments for that method (3-4 sentences suffice).

Programming Rules

You should follow the rules outlined in the document Code conventions posted on the course website at http://cs.nyu.edu/~joannakl/cs102.03_s16/notes/CodeConventions.pdf.

You must document all your code using Javadoc. Your class documentation needs to provide a description of what it is used for and the name of its author. Your methods need to have description, specification of parameters, return values, exceptions thrown and any assumptions that they are making.

Classes’ data fields and methods should not be declared static unless they are to be shared by all instances of the class or provide access to such data.

Working on This Assignment

You should start right away! There is not much code to be written for this assignment, but you should allow sufficient time for debugging, testing, and writing of the report.

You should backup your code whenever you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, do anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even a few hours before the due dates - make sure that you have working code if that happens.

Grading

Make sure that you are following all the rules in the Programming Rules section above.

You can use the tests that were posted for the project 1 to test correctness of your program.

If your program does not compile or if it crashes (almost) every time it is run, you will get a zero on the assignment.
If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

25 points  implementation of the findByNameAddress
25 points  implementation of the findByScore
30 points  implementation of the findByDate
20 points  proper documentation and program style (note: you may need to update some of your documentation from project 1 to reflect the changes)

How and What to Submit

You should submit all of your source code files (the ones with .java extensions only) in a single zip file to NYU Classes.

If you wish to use your (one and only) freebie for this project (one week extension, no questions asked), then complete the form at http://goo.gl/forms/rCKfB6iocX before the due date for the assignment. All freebies are due seven days after the original due date and should be submitted to NYU Classes.