CMU 17-630 Data Structures and Algorithms for Engineers Fall 2018

ASSIGNMENT 4

Priority Queues

Learning Objectives:

The purpose of this assignment is to practice the concepts associated with:

- trees, specifically heaps
- dynamically allocated linked objects
- designing, implementing, and analyzing ADTs
- coupling and cohesion in ADT design
- utilizing an object oriented language

Assignment Description:

Note that all your work must be your own work. You are not permitted to use code you find on the web. You are not permitted to share code with other students.

You will design and implement a priority queue ADT using the Java language. The queue ADT you design will use dynamically allocated objects of your own design. YOU ARE NOT PERMITTED TO UTILIZE JAVA LANGUAGE TREE OBJECTS OR TREE LIBRARIES. YOU ARE REQUIRED TO BUILD YOUR OWN LINKED OBJECTS INFRASTRUCTURE TO IMPLEMENT YOUR TREE/HEAP ADT. You will create a test harness to test and demonstrate your ADT. Each of these parts is described in detail below.

Part 1: Priority Queue ADT

A priority queue is a special purpose queue, because rather than being a "first-in-first-out" data structure, data comes out of the queue by priority. A priority queue might be used to handle the jobs processed by a server, operating system, traffic control systems, factory assembly lines, and so on. Your priority queue will contain nodes that contain the following information:

- **Job Name:** A string that describes the name of the job. The job name is a combination of alphanumeric characters assigned by the user. You can assume that job names will be no longer than 15 characters in length. Assume that each job name will be different.
- **Job Priority:** An integer value of 1 to 20 that indicates the priority of the job. Priority 1 is considered the highest priority, and 20 is the lowest priority. Note that there might be jobs of the same priority. If jobs of the same priority are added to the queue, a first-come-first serve policy for jobs of the same priority will be utilized.

For your priority queue you will utilize a heap ADT of your own design and construction **(no built-in or third party tree libraries)**. Recall that a heap is a specialized tree ADT in which all the nodes of the tree are in a specific order (please review the lecture notes and readings on heaps). To build your ADT and test harness (see part 2) you will use the Java language and linked objects. The example code shown in class illustrates a list of linked objects. This should be a good starting point.

Part 2: Test Harness

Create a textual (not a GUI) test harness to test your priority queue ADT and enables a user to do the following:

Options:
1. Create a priority queue from a file (user provides a string path name)
2. Add an item to the queue (user provides job name and priority – if the list doesn't exist create it)
3. Remove the next priority item to process from the queue (removes item, prints name and priority).
4. Delete a job from the queue by job name (can be anywhere in the list – each job name is unique).
5. Print the contents of the queue from highest to least to highest priority (job name and priority #).
6. Print the number of elements in the queue.

X. Exit

The user must be able to select one of the above options. After selecting the option, the application will prompt the user for any further information required and/or execute the operation. After executing the operation, the application should provide proper feedback to the users and then will present the menu of options again and wait for the user to make another selection. If the user enters 'X' or 'x' the program will end.

For option #2 above you will read the data into the priority queue from a text file using the following format:

```
JobName1:6
JobName2:3
JobName3:1
JobName4:7
JobName5:8
JobName6:20
JobName7:19
JobName8:7
:
:
```

Do not deviate from this text file specification. We will test your code with our input file.

If any of these requirements of this assignment are unclear, please communicate with the course instructor and/or TA.

Design Guidance:

As always, your test harness and priority queue ADT should be as decoupled as possible. Again as you design your priority queue you should consider the programmer that might use your ADT. Design your ADT to be as easy as possible to use and reuse. Your ADT should hide the details of all the queue operations. To the greatest extent possible, the test harness should have as little visibility into the queue ADT's variables, datatypes, methods, etc. Try to expose only those services that a programmer might need to use to utilize and manipulate their data. Be very careful about introducing I/O (and similar) dependencies in your ADT that might make it difficult to use in a general way. You may create other separate and distinct functions as necessary to meet the requirements outlined in above especially to maintain separation of concerns.

As always, design and create bullet proof code… we will try to break it.

Given that you will be using a strongly typed, object oriented language to implement this application, you should take this opportunity to reflect on how much harder/easier it is to practice information hiding than it was with a structured language like C.

Assignment Preparation and Setup

You must use the console java compiler, *javac* to compile your code. We will test your code by putting all of your source code (*.java files) into a test directory and compiling it as follows:

javac *.java

We will then execute your program and test your application against our test plan and data. You must follow these instructions precisely – if we can't figure out how to build your app and run it, we won't. Note that we will not install any additional tools. You may use any tools you like to write/edit your source code, build, and test your application. However, what you submit to us must adhere the guidelines outlined here.

If you are not familiar with Java, there is a nice tutorial here: https://www.tutorialspoint.com/java/

You will also provide a BRIEF write-up to accompany the source code that includes the following:

- Describe the design of your application. Show the static structure and relationship of the code modules you design. Provide a prose explanation of the design. Include rationale for why you designed the program the way you did.

- Provide an algorithmic analysis of the ADT describing the big-oh of your operations and space requirements of the services in your ADT. Show any correctness analysis you performed on your operations.

Packaging and Submitting Your Work

Post on blackboard *a single archive file* of your report and code. Please use Windows or Mac OS compressed folder to create your archive – other archive formats (rar, pkzip, 7zip, etc.) are not acceptable. Please include your name on all of your work including your source code. Name your archive as follows:

A4+<your family name>

For example, my archive name for assignment 4 would be A4+Lattanze. You will lose points for not following these directions. Please check with the instructor/TA if you are unclear.

Grading:

This assignment is worth 100 points as follows:

- **Application:** 75 points**:** Correct functionality, robustness, and the quality of design with respect to how well the list ADT you create hides complexity and is decoupled from the test harness; The quality of comments in the source code.

- **Quality of Report:** 25 Points: Grammar, format, content, and thoroughness of the write-up.