Implementing min-heaps

Preamble

This assignment is the first part of assignment A6. It will be needed in the rest of A6 when you perhaps implement Dijkstra's shortest path algorithm and other algorithms.

Collaboration policy and academic integrity

You may do this assignment with one other person. Both members of the group should get on the CMS and do what is required to form a group well before the assignment due date. Both must do something to form the group: one proposes, the other accepts.

People in a group must *work together*. It is against the rules for one person to do some programming on this assignment without the other person sitting nearby and helping. Take turns "driving" —using the keyboard and mouse.

With the exception of your CMS-registered group partner, you may not look at anyone else's code, in any form, or show your code to anyone else (except the course staff), in any form. You may not show or give your code to another student in the class.

This assignment is worth 20% of your total grade on A6.

Getting help

If you don't know where to start, if you don't understand testing, if you are lost, etc., please SEE SOMEONE IMMEDIATELY —an instructor, a TA, a consultant. Do not wait. A little in-person help can do wonders. See the course homepage for contact information.

The release code

We give you three files in zip file a6a.zip:

1. Interface MinHeap, which defines the methods for implementing a min-heap.
2. Class MyHeap, which implements MinHeap, with the overriding methods stubbed in so that the class compiles.
3. Class HeapTester, which has a method main to test MyHeap. It doesn't do complete testing, but it does enough to help you get a great start. Method main prints its output on the console. It has a String[] parameter, which you can use to give some arguments to control the testing that is done. The comment at the head of the class explains what the arguments are.

Place the three files in the default package in a new project called a6aHeaps.

What to do for this assignment

Your job is to implement the methods in class MyHeap so that the class actually implements a heap. Submit (only) class MyHeap on the CMS by the due date.

Hints and suggestions

1. Your class must implement the methods within the specified time bounds. You may, of course, write additional private methods in the class, but be sure to specify them well. Things like bubbleUp, buubbleDown, perhaps a procedure to swap to elements of the ArrayList, keeping their indices updated, can come in handy.
2. Put good specs on all methods of your implementation, including the overriding methods and ones you add.

3. Make sure function toString is correct! It is used in the testing class. Note that it says to put a comma and a space between adjacent items.
4. HeapTester does a fairly good job of testing, but it is not foolproof.
5. It is best to implement the heap in an ArrayList.
6. Special problem. Class MyHeap is fairly easy to write except for one issue. Method updatePriority(t, p) allows the priority of item t to be changed to p. This requires finding item t in the ArrayList —and without any other data structure to help, this could cost time linear in the size of the ArrayList!

   To overcome this problem, we suggest two steps:

   - 1. Have a static inner class C (say, you give it a better name) whose fields are (1) a priority and (2) an index into the ArrayList. Then, each item put into the ArrayList can be represented by an object of this inner class.
   - Have a field of your min-heap class that is a HashMap<T, C>, which maps an item in the heap to an object of class C that contains its priority and index in the ArrayList.

   Of course, when an item in the heap is moved to a different index, the field of the associated element of class C that contains its index has to be changed appropriately.

   Using this technique, the expected time for updating a priority should be O(1) for finding the corresponding element in the HashMap and then O(log n) for updating the heap.