TD4- Heaps

This assignment will give you practice heaps.

Part 1 : write up questions

- Show the result of inserting 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, and 2, one at a time, into an initially empty binary heap
- Show the result of using the linear-time algorithm to build a binary heap using the same input (cf. Floyd's Method)
- Show the result of performing one *deleteMin* operation in the heap of the previous question
- Show the following regarding the inverse extremum item in the heap (that is the minimum for a max-heap or the maximum for a min-heap):
    - It must be at one of the leaves
    - Every leaf must be examined to find it

Part 2: binary heap implementation

In this part you must implement a binary heap class following what we learned during lecture but with the new following rules:

- your binary heap must be parametrized by the ordering on the elements
- you are to use the entire array, i.e., the first index 0 will be the index of the extreme element (minimum or maximum)

Complete the provided class template BinaryHeap by writing all the methods.

Supporting files

- BinaryHeap.java
- EmptyHeapException.java
- BinaryHeapTest.java

Part 3: deleting in a binary heap

Assuming values in a binary heap are all distinct, we want to implement the *delete* operation which deletes a given element in the heap (not necessary the extreme element).

- give a lower bound for the complexity of the *delete* operation.
- write the method delete in the same class BinaryHeap as in part 2
- give the detailed worst-case complexity of the delete method

We now assume that the elements in the heap are not unique, and it is likely that the heap contains many occurrences of the element to delete.

- give a different method to delete all the occurrences of a given element in the heap
- write the method *deleteAll* in the same class *BinaryHeap* of part 2

Part 4: *d*-heaps (Optional)

A simple generalization of the binary heap is a *d*-heap, which is exactly like a binary heap except that all nodes have *d* children (thus, a binary heap is a 2-heap).

- give the height of a *d*-heap containing *N* elements
- discuss the pros and cons of *d*-heaps versus 2-heaps
- write all the methods in a class named DHeap: as far as the implementation is concerned, a *d*-heap is just like a binary heap, except it has a new attribute *d* (the number of children of nodes).

For this problem, you don't have supporting file. Instead, you are to copy and paste the code from your BinaryHeap.java file and modify it accordingly. Your class should be named Dheap (and the file Dheap.java).