Algorithms Tick: Implementing a Binary Max-Heap

Please make sure to attend the Algorithms practicals on Thu 2014-02-13 from 2pm-4pm or 4pm-6pm in the Intel Teaching Lab, since this is the only opportunity to get assistance from the Algorithms demonstrators. This tick will be marked in the same way as those in the Programming in Java course: after passing the tests in the unit tester you should sign up for a ticking slot using the Programming in Java sign up sheets.

Introduction

In this workbook you will implement a binary heap (refer to pages 75-76 in the Algorithms handout). Despite its simplicity, a binary heap is a very fundamental data structure which is implicitly used by the heapsort algorithm. Further, binary heaps form the basis of more sophisticated data structures such as binomial heaps and Fibonacci heaps which will be covered at a later point in the Algorithms course.

Outline

Specifically, your task is to implement a binary max-heap. Your program `MaxHeap.java` should have one method for inserting an item, one method for extracting the maximum key (if the heap is non-empty) and two constructors: one for creating an empty heap, and another for creating a heap containing a given set of n items. Implementing an $O(n \lg n)$ version of this second constructor is trivial; but, if you are good at this stuff, as an optional exercise you might choose to implement an $O(n)$ constructor instead. It's not required, but proudly show your ticker if you did.

Please follow the structure and adhere to the precise identifiers in the skeleton below. The name of a heap should be any lower-case character between 'a' and 'z', and a key value should be any upper-case character between 'A' and 'Z', where 'Z' is considered to be the largest value. *Note:* You may have to add additional variable(s) (which ones?) to make it work.

```
package uk.ac.cam.your-crsid.alg1;

public class MaxHeap
{
  private char heapName;

  MaxHeap(char name)
  {
    ...
  }

  MaxHeap(char name, String str)
  {
    ...
  }

  ...
}
```

In addition to the two constructors, the class `MaxHeap` should also contain the two basic heap operations, one for inserting an item and another one for extracting the maximum element. First, the method for inserting an element should look as follows:

```
  void insert(char x)
  {
    ...
  }
```

The method for extracting the Maximum should return the special character '_' if the heap is empty, and otherwise remove and return the largest key:

```
  char getMax()
  {
    ...
  }
```

Obviously, you are allowed to add additional methods for implementing these operations or just for debugging. Finally, you may use the `main` method to exercise your code, as in the following example (your own tests will hopefully be more thorough, but here is a starting point):

```
public static void main(String[] args)
{
    char c;
    MaxHeap h = new MaxHeap('h', "CAMBRIDGEALGORITHMS");
    c = h.getMax();
    System.out.println(c); // expect T
    h.insert('Z');
    h.insert('A');
    c = h.getMax();
    System.out.println(c); // expect Z
    c = h.getMax();
    System.out.println(c); // expect S
}
```

Submitting your program

To submit your program `MaxHeap.java`, produce a jar file called `crsid-alg1.jar` with the following content (refer to page 6 of Workbook 1 from the Programming in Java course for details):

```
META-INF/
META-INF/MANIFEST.MF
uk/ac/cam/crsid/alg1/MaxHeap.class
uk/ac/cam/crsid/alg1/MaxHeap.java
```

The jar file should have its entry point set to `uk.ac.cam.crsid.alg1.MaxHeap`. You should submit your jar file by emailing it as an attachment to `ticks1a-java@cl.cam.ac.uk`. Once your code successfully passes the tests you should sign up and attend a ticking slot from the Programming in Java course.