M3: Pre Lab

Introduction

This pre-lab is designed to help prepare you to complete the programming component of M3:Lab. Future labs will also include a pre-lab component.

Because the pre-lab section is designed to help you plan your implementation, it is strongly recommended for you to complete this section **before** writing any code. However, it is also natural for your understanding of the implementation requirements and your own strategy to evolve as you complete the code, so feel free to update your answers to the pre-lab section during/after the coding phase of the lab, up to the pre-lab deadline.

Questions

The programming component of this lab consists of implementing several important min-heap operations by filling in code for method stubs in the file `MinHeap.java`. For each of the methods to be filled in (listed below), answer the following questions. You may assume that the questions pertain to the main logic of the method only; you do not need to consider instrumentation variables such as those related to the `ticker` when answering.

- If the method has any parameters (i.e. inputs passed to the method), what are they and what does each one represent, in your own words? How will each parameter be used by the method? (An example of a parameter is that the method `insert` has one parameter `thing` of type `T`.)
- If the method has a non-`void` return value, what does it return, in your own words?
- If the method has no parameters and/or its return value is void, please state this fact in your answer.

Here are the methods you will be filling in:

1. `public Decreaser<T> insert(T thing){}`
2. `void decrease(int loc){}`
3. `public T extractMin(){}`
4. `private void heapify(int where){}`

Here is a sample of an expected response for a method called `selectionSort` (not related to this assignment) that has the header

```
public int[] selectionSort(int[] arr){}
```

and whose goal is to perform a selection sort on an array of `int`s.

Method: `public int[] selectionSort(int[] arr){}`

- *Parameters*:
  `arr`: a 1-dimensional array of ints that stores the inputs to be sorted. The method will read this input array (multiple times) and write a sorted version of the inputs to a newly allocated output array.
- *Return value*: a new array containing the inputs in sorted order.

Once you finish your responses for the methods listed, answer the following additional questions:

5. Which instance variable in the `MinHeap` class holds the actual contents of the heap?
6. Which method(s) will call `heapify`?
7. Why is the `decrease` method not private to the `MinHeap` class?