Homework 4: Heaps and Priority Queues

Due: See T-Square

Important

There are general homework guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 8**. This includes unused code, so don't submit extra files that don't compile. (Java is backwards compatibile so if it compiles under JDK 7 it *should* compile under JDK 8.)
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. Code exactly what is asked for. No more and no less.
8. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.

Heaps

A heap has one simple rule:
In a heap, for every node X with parent P, the key value in P is smaller than or equal to X which forces the minimum value to be at the tree root node

For this homework, you will implement a heap backed by an array. The array should have an initial size of 10, and double its length whenever it reaches capacity. You must write your own code for this; do not use the Arrays class or the System.arrayCopy method.
The smallest element in the head should always be contained at index 1 of the array. Index 0 must always be equal to null.
To implement the heap, you will need to implement private methods like heapify, buildHeap, and resize to maintain heap property. Feel free to add private helper methods in your code.

Priority Queue

Priority Queue is a data structure with the ability to add items at any time and remove the smallest item at any time.
For this homework, you will implement a Priority Queue backed by a heap you write.

Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing. It is located in resources along with instructions on how to use it. We will take off a point for every style error that occurs. If you feel like what you wrote is in accordance with good style but still sets off the style checker, please email Jonathan Jemson (jonathanjemson@gatech.edu) and Matt Gruchacz (mattgruchacz@gatech.edu) with the subject header of "Style XML".

Javadoc

Javadoc any helper methods you create in a style similar to the Javadoc for the methods in the interface.

JUnits

For this homework, no JUnit testing is required, but you may find it beneficial to unit test your Heap and Priority Queue as you write them to help avoid problems.

Provided

The following file(s) have been provided to you.

1. `HeapInterface.java` This is the interface you will implement for your heap. All instructions for what the methods should do are in the Javadoc. **Do not alter this file.**
2. `Heap.java` This is the class in which you will actually implement the interface. Feel free to add private helpers but **do not add any new public methods.**
3. `PriorityQueueInterface.java` This is the interface for the PriorityQueue you will implement. All instructions for what the methods should do are in the Javadoc. **Do not alter this file.**
4. `PriorityQueue.java` This is the class that will implement your PriorityQueue using the heap you built as backend. Look at the Javadoc for the given methods for information on how to implement them. Feel free to add private helpers but **do not add any new public methods.**
5. `Gradable.java` This is the interface for grading purposes. The Heap class must implement this interface. All instructions for what the methods should do are in the Javadoc. **Note that what you return for the toArray method should be the entire backing array, including the empty spaces. Do not alter this file.**
6. `StudentTests.java` These are basic JUnit tests for this assignment. You are encouraged to write additional tests to ensure you have accounted for all edge cases.

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc.

1. `PriorityQueue.java`
2. `Heap.java`

You may attach each file individually, or submit them in a zip archive.