LAB EXERCISE

Heapsort

Background:

A priority queue, implemented as a heap, can be used for sorting. To do that we must add all the items to a priority queue in any order, and then remove them one by one. The items will be returned in ascending order. This efficient algorithm is called *Heapsort*.

To realize the heapsort algorithm, it is necessary to develop a class that implements the `PriorityQueue` interface using a binary heap.

This lab will use a text file, "*file20.txt*", which is similar to the one used in the binary search lab in Lesson 28. The file has been saved in random order by `Id` number. Your program must build a binary heap based on the `Id` field. The priority queue should be implemented as a `HeapPriorityQueue` of type `Item`.

Assignment:

1. Here are some of the specifications for the methods to be added to the `HeapPriorityQueue` class:

   a. You are to write a method `isEmpty` that returns true if the number of element in the priority queue is 0; otherwise it returns false.

   b. An `add` method will add a new item to the heap, rearranging the heap as necessary to preserve the heap structure.

   c. The `removeMin` method will return and remove the highest priority item from the priority queue. If the queue is empty, a `NoSuchElementException` should be thrown.

   c. The `peekMin` method will return the highest priority item from the priority queue. If the queue is empty, a `NoSuchElementException` should be thrown.

   e. It is recommended that a `heapify` helper method be created as described in the lesson to reorganize the heap to preserve the heap structure after the removal of the root item.

   f. The heap structure should be contained in an `ArrayList`. To aid in coding, the root of the binary heap should start at index 1.

   g. Reading the data file is a similar process to that used in *Store.java* in Lesson 27.

   h. Printing the list involves the same code as used in the previous lessons.

2. If your instructor chooses, you will be provided with a program shell consisting of a `main` menu method, testing methods, and stubbed methods for routines you must develop. Here are some of the specifications of this program shell.

a. A `HeapSort` test method is provided. A method to read the data file is provided. However, the `sort` method is stubbed out as a print statement.

b. The `Item` class is provided.

c. The `remove` method returns a **`null`** value.

d. A shell for the `HeapPriorityQueue` class is provided. The `add`, `removeMin`, `peekMin`, and `isEmpty`, methods are stubbed out.

g. Methods to read the data file and print the list are provided.

Instructions:

1. Modify and write code as necessary to satisfy the above specifications.
2. Print out the entire source code.
3. Include a printed run output of the file in original and sorted order.