### Portfolio Function Title: MaxHeap

Version number: 1

### Function Description:

Allows for a user to build a MaxHeap of objects with a specified size. Once a MaxHeap is built a user can add elements until the MaxHeap reaches the specified size. Once the MaxHeap is populated, the user can look at and remove the maximum element (according to the element type’s compareTo() method) in the MaxHeap. Most of these methods rely on two private methods sift up and sift down that move an element from the bottom of the heap or the top of the heap to their correct position.

### Author:

Nathan Mankovich

### Date Written/Last Modified:

05-06-2015 / 05-07-2015

### Design overview:

The MaxHeap relies on the array data structure to store the elements in the heap.

When the MaxHeap is initially created it has a maximum size specified by the user, an initial size of 0 and has elements stored in the itms array of the user-specified maximum size.

Four private methods are key to the program’s functionality. The ChildIndex (2k+1) and ParentIndex ((k-1)/2) methods use very simple functions to calculate a certain element’s left child and parent respectively. Whenever an element is added or removed from the MaxHeap the SiftUp and SiftDown methods are called respectively.

The array holds all the elements, but the concept is that each element can have up to two children, a right child and a left child. The rule is that every parent in the MaxHeap is greater than its child.

The SiftUp method travels up the heap from the bottom comparing the current element in the heap with its parent and moves the current element up the heap if it is greater than its parent. If the current element is not greater its parent, it stays in its current slot.

The SiftDown method travels down the heap from the maximum element (the 0th element in the array structure) comparing it to the greatest of its two children. If the maximum child is greater than the element, the two child and element swap places. Otherwise, the element and child stay in their respective slots.

### Feature Specifications:

GetMax() gets the maximum element in the MaxHeap.

RemoveMax() removes the maximum element in the MaxHeap by replacing it with the last element in the heap then uses SortDown() to re-order the heap.

Add() adds a donor to the next available slot in the MaxHeap itms array as long as the MaxHeap is not full. Then, it uses SortUp() to move the element to its proper index in the MaxHeap array.

### Programmer User Interface:

User Interface: Java code and command line only.

### Input and Output Requirements and Restrictions:

Private Methods:

* ChildIndex
  + Input: an integer index in the array
  + Output: the left child index of the input index
* ParentIndex
  + Input: an integer index in the array
  + Output: the parent index of the input index
* SiftUp no Inputs or outputs
* SiftDown no inputs or outputs

Public Methods:

* GetMax
  + Input: none
  + Output: the maximum element in the MaxHeap
* RemoveMax
  + Input: none
  + Output: the old maximum element in the MaxHeap
* Add
  + Input: the element to be added to the MaxHeap
  + Output: none

### Assumptions and Dependencies:

The MaxHeap assumes that all elements input are the same type.

The size of the max heap depends on the user’s instantiation.

The MaxHeap assumes the user only needs to access the maximum element.

### Known problems and limitations:

This MaxHeap implementation does not allow the user to access the number of elements in the MaxHeap.

It would be nice to have a MaxHeap that is dynamically sizable instead of this statically sized array implementation. Thus, no elements can be added after the MaxHeap is full.

There is not a RemoveMin method because the minimum donor is difficult to find in a MaxHeap.

### Use Cases:

This is best used to store unsorted data and return it in sorted order of largest elements first to smallest elements last.

### Testing Methodology:

Be sure to put the tMaxHeap.java and the portfolio.jar file in the same folder.

To compile:

javac -cp .;portfolio.jar MaxHeap.java

To run:

java -cp .;portfolio.jar tMaxHeap > LogMaxHeapTest.txt

All command line outputs will be in the new LogMaxHeapTest.txt file in the folder that contains the tMaxHeap and the portfolio.jar files.

### Modification history:

Version 1: 05-07-2015

### Design detail and/or Diagrams:

When the MaxHeap is initially created it has a maximum size specified by the user, an initial size of 0 and has elements stored in the itms array of the user-specified maximum size.

Whenever an element is added to the MaxHeap it is added to the bottom of the heap and sorted up.

Whenever a maximum element is removed from the heap it is replaced by the last element in the heap’s array structure and that maximum element is sorted down.