### Portfolio Function Title: MinHeap

Version number: 1

### Function Description:

Allows for a user to build a MinHeap of objects with a specified size. Once a MinHeap is built a user can add elements until the MinHeap reaches the specified size. Once the MinHeap is populated, the user can look at and remove the minimum element (according to the element type’s compareTo() method) in the MinHeap. Author:

Nathan Mankovich

### Date Written/Last Modified:

05-06-2015 / 05-07-2015

### How to use the Function:

To use the stack program just instantiate a MinHeap<obj> with an object of your choice and then you can use the different methods in your java code.

The program is stored in the portfolio.jar file, put that file in the same folder as YourProgram.java.

Open a Windows command window and enter:

To compile the program with YourProgram.java enter:

javac -cp .;portfolio.jar YourProgram.java

pause

Then, to run the program with the provided sample-mailing-list.csv enter:

java -cp .;portfolio.jar YourProgram

pause

User Interface: Java code and command line only.

### How the Function works:

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

When the MinHeap 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 MinHeap 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 MinHeap is less than its child.

### Supported Methods (including Inputs, Outputs, Features and Results by method):

* GetMin gets the minimum element in the MinHeap
  + Input: none
  + Output: the minimum element in the MinHeap
* RemoveMin removes the minimum element in the MinHeap
  + Input: none
  + Output: the old minimum element in the MinHeap
* Add adds an element to the MinHeap
  + Input: the element to be added to the MinHeap
  + Output: none

### Known problems and limitations:

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

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

There is not a RemoveMax method because the maximum donor is difficult to find in a MinHeap.