# Heaps

Updated: 7th December, 2021

## Aims

- To implement a Heap and use it for HeapSort.

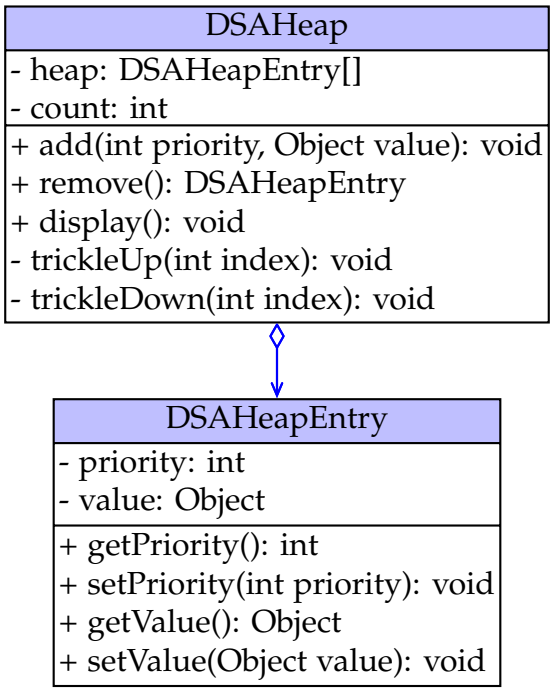
## Before the Practical

- Read this practical sheet fully before starting.

## Activities

## 1. Heap Implementation

Following the pseudocode in the lecture slides, implement a max heap so that larger values indicate higher priority using `DSAHeap` and `DSAHeapEntry` classes. At a minimum, your classes should include the following methods.




Write a test harness to test each method thoroughly, be sure to test all cases.

**Note:**

- Since we are using an array to represent the heap, remember that we use arithmetic to determine the array indexes of the children given the parent's index (and similarly to go from children to the parent's index). So, if we are at *currIdx*, the following arithmetic will get us the left child, right child and parent of the *currIdx* node.

```
leftChildIdx = (currIdx * 2) + 1
rightChildIdx = (currIdx *2) + 2
parentIdx = (currIdx - 1) / 2
```

- `add()` must add the priority-value pair at the correct place in the heap. To do this, remember that we start off at the end of the array, then trickle it up (by using arithmetic to get its parent) until the parent is of equal or higher priority.
- `remove()` must return the highest priority element and remove it from the heap array. This will involve removing the root element, placing the last element at the root and then trickling it down.

## 2. HeapSort Implementation

Following the pseudocode in the lecture slides, implement `heapSort()`, an O(N log N) sorting algorithm.

The idea is that an imported *array-to-sort* will have to be an array of `DSAHeapEntry` objects. `heapify()` will convert the array of `DSAHeapEntry` objects into a max heap. `heapSort()` will then sort this array.

Write a test harness using either a random list of varying sizes such as found in the `SortsTestHarness` or read in the numbers from `RandomNames7000.csv` from Practical 1.

## Submission Deliverable

- Your code is due 2 weeks from your current tutorial session.
  - You will demonstrate your work to your tutors during that session
  - If you have completed the practical earlier, you can demonstrate your work during the next session
- You must **submit** your code and any test data that you have been using **electronically via Blackboard** under the *Assessments* section before your demonstration.
  - Java students, please do not submit the *.class files

## Marking Guide

Your submission will be marked as follows:

- [5] Your DSAHeap is implemented properly.
- [3] Your Heap Sort algorithm is properly implemented.
- [2] Your test harness works and allows testing on either an array of arbitrary length of random numbers or loading in a large amount of numbers from a file (such as RandomNames7000.csv).

**End of Worksheet**