Heap Sort

Data Structures Made Easy

DUBLIN CITY UNIVERSITY

1. ***Heap Sort***

```
class heap_Sort{

	private static int left_Sort(int element){

		return 2 * element + 1;
	}

	private static int right_Sort(int element){

		return 2 * element + 2;
	}

	private static void swap(int [] array, int a, int b){

		int temp = array[a];
		array[a] = array[b];
		array[b] = temp;
	}

	public static void max_Heap(int [] heap_Array, int heap_Integer, int heap_Size){

		int left;
		int right;
		int maximum;
		int temporary;

		left = left_Sort(heap_Integer);
		right = right_Sort(heap_Integer);

		if(left < heap_Size && heap_Array[left] > heap_Array[heap_Integer])
			maximum = left;
		else
			maximum = heap_Integer;

		if(right < heap_Size && heap_Array[right] > heap_Array[maximum])
			maximum = right;

		if(maximum != heap_Integer){

			swap(heap_Array, heap_Integer, maximum);
			max_Heap(heap_Array, maximum, heap_Size);
		}
	}

	public static void build_Heap(int [] heap_Array, int heap_Size){

		for(int index = heap_Size / 2; index >= 0; index--)
			max_Heap(heap_Array, index, heap_Size);
	}
```

```
	public static void heap_Sort(int [] heap_Array, int heap_Size){

		build_Heap(heap_Array, heap_Size);

		for(int index = heap_Size - 1; index > 0; index--){

			swap(heap_Array, index, 0);
			heap_Size--;
			max_Heap(heap_Array, 0, heap_Size);
		}
	}

	public static void main(String [] args){

		System.out.print("Enter the number of elements: ");
		int size = Console.readInt();

		int [] heap = new int[size];

		System.out.print('\n' + "Enter the elements: ");

		for(int index_1 = 0; index_1 < size; index_1++){

			int element = Console.readInt();
			heap[index_1] = element;
		}

		heap_Sort(heap, size);

		System.out.print('\n' + "The sorted list is: ");

		for(int index_2 = 0; index_2 < size; index_2++)
			System.out.print(heap[index_2] + " ");
	}
}
```