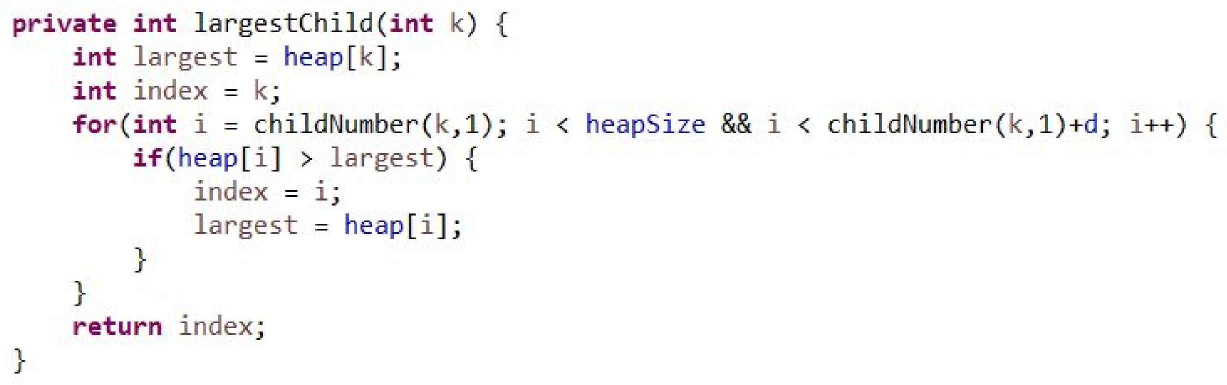
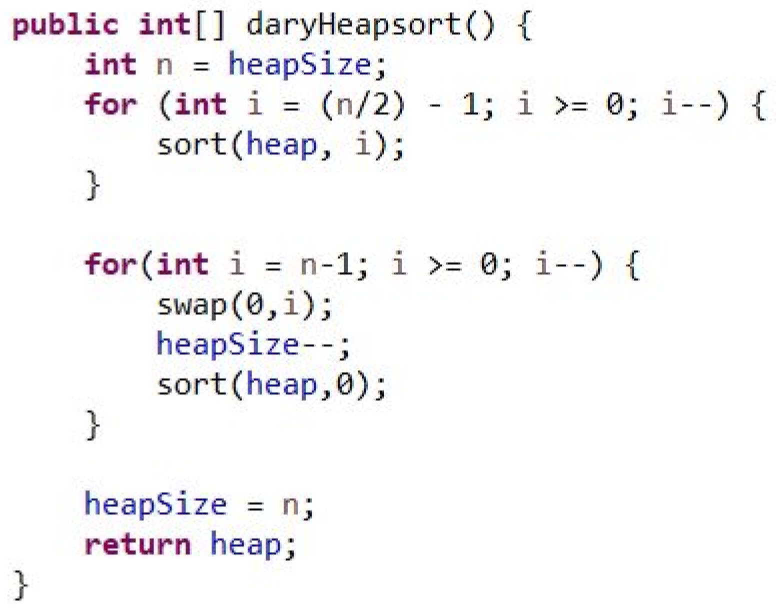
The Big O Notation of delMax() is O(logn). First it will swap and delete the old largest array. However after this, it needs to take account of all the child class that the heap has. It will run through the for loop as many times until it is certain that the value is the largest child. And this part is the core of the delMax() code therefore depending on the number of child class affects the time complexity.



```java
private int largestChild(int k) {
    int largest = heap[k];
    int index = k;
    for(int i = childNumber(k,1); i < heapSize && i < childNumber(k,1)+d; i++) {
        if(heap[i] > largest) {
            index = i;
            largest = heap[i];
        }
    }
    return index;
}
```

The Big O notation of the heapsort is O(nlogn). This is because of recursion. The heapsort will call the sort function until all the values are sorted properly. This heap sort function acts similar to finding the max. Which was determined to be O(logn). Therefore n in front of the log depends on the number of times that the sort function will be called.



```java
public int[] daryHeapsort() {
    int n = heapSize;
    for (int i = (n/2) - 1; i >= 0; i--) {
        sort(heap, i);
    }

    for(int i = n-1; i >= 0; i--) {
        swap(0,i);
        heapSize--;
        sort(heap,0);
    }

    heapSize = n;
    return heap;
}
```