Explanation of the method makeHeap()

```java
public static int[] makeHeap3(int[] a){

    for(int i =a.length/2-1;i>=0;i--){
        heapify(a,a.length,i);

    }
    return a;
}
public static void heapify(int[] a,int size,int root){
    int minimum = root; Track the min data index (Assume @root)
    int left = 2*root+1;
    int right = 2*root+2;
    if(left<size && a[left]<a[minimum]){
        minimum =left;
    }
    if(right<size && a[right]<a[minimum]){
        minimum =right;
    }
    if(minimum!=root){
        int temp = a[root];
        a[root] = a[minimum];
        a[minimum] = temp;

        heapify(a,size,minimum);
    }
```

This for loop iterates through half the array starting at the half point, because we know that the last half of the array are leaf nodes

This heapify uses the binary tree form of heap to help sorting.

If any child node has less value data change the minimum's index

If the minimum is not at root make it so.

A reccursive call since this index no longer holds the minimum data a recursive call is made to ensure heap property is preserved

An Example

Array = [7, 4, 13, 3, 10, 5, 12]

The binary tree representation

First for loop starts at index 2

Array = [7, 4, 13, 3, 10, 5, 12]

heapify (Array, 7, 2) is called

since 13 > 12 and 12 is not at the root, a swap occurs

Array = [7, 4, 13, 3, 10, 5, 12]

minimum

Swap

Array = [7, 4, 5, 3, 10, 13, 12]

minimum

heapify (Array, 7, 6) is called
No swaping occurs ∵ leaf

i is decremented
an heapify (Array, 7, 1) is called

Array = [7, 4, 5, 3, 10, 13, 12]

since 4 > 3 a swap occurs

Array = [7, 4, 5, 3, 10, 13, 12]

Minimum

Array = [7, 3, 5, 4, 10, 13, 12]

Minimum

heapify (Array, 7, 3) is called
No swapping occurs
i is decremented
an heapify (Array, 7, 0) is called

Array = [7, 3, 5, 4, 10, 13, 12]

Since 7 > 3 swapping occurs

Array = [7, 3, 5, 4, 10, 13, 12]

Minimum

Array = [3, 7, 5, 4, 10, 13, 12]

minimum

heapify (Array, 7, 1) is called and swapping occur

and heapify (Array, 7, 3) is called

and thus the final array

Array = [3, 4, 5, 7, 10, 13, 12]