Homework Turnin

Account:	6G_06 (rgalanos@fcps.edu)
Section:	6G
Course:	TJHSST APCS 2016-17
Assignment:	09-02
Receipt ID:	2aefb325aab8511c84a404c2bce7bf26

Turnin Successful!

The following file(s) were received:

HeapPriorityQueue.java (2011 bytes)

```
1.  //Name:   Date:
2.
3.  //implement the API for java.util.PriorityQueue
4.  //test this class by using it in McRonaldPQ_working.java.
5.  //add(E) and remove()  must work in O(log n) time
6.
7.  import java.util.*;
8.  public class HeapPriorityQueue<E extends Comparable<E>>
9.  {
10.    private ArrayList<E> myHeap;
11.
12.    public HeapPriorityQueue()
13.    {
14.       myHeap = new ArrayList<E>();
15.       myHeap.add(null);
16.    }
17.
18.    public void heapDown(ArrayList<E> array, int k, int size)
19.    {
20.       int left = 2 * k;
21.       int right = 2 * k + 1;
22.       if(k > size || left > size)
23.          return;
24.       if(right > size)
25.       {
26.          if(array.get(k).compareTo(array.get(left))>0)
27.             swap(array, k, left);
28.       }
29.       else
30.       {
31.          int minChild = (array.get(left).compareTo(array.get(right))<0)? left:right;
32.          if(array.get(k).compareTo(array.get(minChild))>0)
33.          {
34.             swap(array, k, minChild);
35.             heapDown(array, minChild, size);
36.          }
37.       }
38.    }
39.    public void heapUp(ArrayList<E> array, int k, int size)
40.    {
41.       int parent = k / 2;
42.       if(k > size || parent < 1)
43.          return;
44.       if(array.get(k).compareTo(array.get(parent))>0)
45.       {
46.          swap(array, k, parent);
47.          heapUp(array, parent, size);
48.       }
49.    }
50.
```

```
51.     public void swap(ArrayList<E> array, int a, int b)
52.     {
53.        E temp = array.get(a);
54.        array.set(a, array.get(b));
55.        array.set(b, temp);
56.     }
57.     public String toString()
58.     {
59.        return myHeap+"";
60.     }
61.     public E peek()
62.     {
63.        if(myHeap.size()>1)
64.           return myHeap.get(1);
65.        else
66.           return null;
67.     }
68.
69.     public E remove()
70.     {
71.        E temp = myHeap.get(1);
72.        swap(myHeap, 1, myHeap.size()-1);
73.        myHeap.remove(myHeap.size()-1);
74.        heapDown(myHeap, 1, myHeap.size()-1);
75.        return temp;
76.     }
77.
78.     public boolean add(E o)
79.     {
80.        myHeap.add(o);
81.        heapUp(myHeap, myHeap.size()-1, myHeap.size());
82.        return true;
83.     }
84.     public boolean isEmpty()
85.     {
86.        return myHeap.size()<2;
87.     }
88. }
89.
```