CS 3345 Project 2: Binary Min Heap

Develop a class called BinaryMinHeap that stores references to Name objects. Here is a Name object:

```
 class Name { // keep to this specification
      Name(int id, int ky); // constructor
      public int getKey();
      public void setKey(int newKey);
      public int getID();

      private int ID;
      private int key;
}
```

The BinaryMinHeap class is as follows:

```
   class BinaryMinHeap {  // keep to this specification
       BinaryHeap(int initialSize);  // constructor
       public void insert(Name n);   // use the heap insert method
       public void addAtEnd(Name n); // insert a reference to the name object in the first
                                     // free space in the array. Don't percolate it.
       public void buildHeap();      // build a heap from the unordered entries in array A.

       public Name deleteMin();      // returns null if the heap is empty.
       public boolean isEmpty();
       public int size();            // returns the number of Names stored in the heap.
       public void increaseKey(int id, int delta); // increase the key of the Name object
                                                   // with ID = id by amount delta and
                                                   // percolate the element down.
                                                   // Do nothing if ID doesn't exist
       public void decreaseKey(int id, int delta); // decrease the key of the Name object
                                                   // with ID = id by amount delta and
                                                   // percolate the element up.
                                                   // Do nothing if ID doesn't exist
       private int arraySize;
       private int nextFreeSlot;      // the index of the next free array slot.
       private Name [] A;             // holds the references to Names stored in the heap.
   }
```

Names in the heap will be arranged according to the following Heap Property:
All the Names in the left and right subtrees of a Name with key `x` will have keys larger than or equal to `x`

There will be duplicate keys but Name IDs within the heap will be unique.

If the array capacity is exceeded, the class must create a new array of double the size of A, copy all the references from A to the new one, and set A to refer to the new array.

Use a Java Hashtable within the BinaryMinHeap class to keep track of the positions of Names in the array A:

```
private Hashtable<Integer,Integer> ht;
```

The table will store positions of Name objects in the BinaryMinHeap array and will be accessed by the corresponding Name ID values.

Commands
Your program will read commands from `System.in`. Each command will be given on a separate line.

Here are the commands that your program must interpret: The capital letters indicate the commands. Variable i is a Name ID, and variable k is an integer key value. Variable d is an integer increment or decrement.

```
N           \\ The first line of every data file will begin with this command.
            \\ Print your name on a line by itself.
C t         \\ Create a new heap capable of storing t name references.
I i k       \\ Do nothing if ID i already exists in the heap, otherwise
            \\ create a new Name object n with ID=i and key=k and call insertName(n)
J i k       \\ Do nothing if ID i already exists in the heap, otherwise
            \\ create a new Name object n with ID=i and key=k and call addAtEnd(n)
B           \\ Call buildHeap
D           \\ Call deleteMin and print the key and ID on a line on its own in format "k i"
            \\ where k is the key and i is the ID,
            \\ Or print "Heap empty" if that is the case.
P i d       \\ Do nothing if ID is not found, otherwise
            \\ call increaseKey(i,d) where i is the ID and d is the amount to add.
M i d       \\ Do nothing if ID is not found or d is greater than the current key value.
            \\ otherwise call decreaseKey(i,d) where i is the ID and d is the amount to subtract.
E           \\ End of Data - just exit
```

Boundary Conditions

- Keys are in the range [0,9999]. IDs are in the range [0,9999].
- In any data file there could be between 1 and 200,000 commands.
- All data files begin with an N command and end with an E command. Each of these commands occur only once.
- In all command lines the tokens will be separated by exactly one space and there will not be extra spaces before the newline character.
- A single newline character will follow the E character at the end of the file

Here is a sample data file and the corresponding program output:

Line #	Sample Input	Sample Output	From Command on Line
1	N	Ivor Page	1
2	C 20	1 101	16
3	J 101 1	2 106	17
4	J 102 8	3 103	18
5	J 103 3	4 107	19
6	J 104 5	5 104	20
7	J 105 7	6 110	21
8	J 106 2	7 105	22
9	J 107 4	8 102	23
10	J 108 9	9 108	24
11	J 109 10	10 109	25
12	B	11 112	26
13	I 110 6	12 111	27
14	I 111 12	Heap empty	28
15	I 112 11	0 108	46
16	D	1 101	47
17	D	2 107	48
18	D	4 111	49
19	D	4 103	50
20	D	5 110	51
21	D	6 105	52
22	D	7 106	53
23	D	8 102	54
24	D	10 109	55
25	D	13 104	56
26	D	17 112	57
27	D	Heap empty	58
28	D		
29	C 30		
30	I 110 5		
31	I 112 11		
32	I 101 1		
33	I 102 8		
34	I 103 4		
35	I 108 9		
36	I 104 3		
37	I 105 6		
38	I 111 12		
39	I 106 7		
40	I 107 2		
41	I 109 10		
42	P 104 10		
43	M 111 8		
44	P 112 6		
45	M 108 9		
46	D		
47	D		
48	D		
49	D		
50	D		
51	D		
52	D		
53	D		
54	D		
55	D		
56	D		
57	D		
58	D		
59	E		

RULES FOR PROGRAMMING AND SUBMISSION:

1. Your program must be your own work. Do not use code from a book, the web or anyone else. Use the course notes as a guide.
2. Write your program as one source file and do not use the "package" construct in your Java source.
3. Name your source file by the name given to you in the Program Names file on the eLearning main page with the suffix "p2" in place of "p1".
4. Your program must not output any prompts. Its output must exactly match the format of the **Sample Output** above.
5. Do not use any Java Collection Classes, other than the array in the BinaryMinHeap class. You may use a String to read command lines and an array of Strings to facilitate the Java String.split() method.
6. **You program must read from System.in and output to System.out. IT MUST NOT READ FROM A NAMED FILE**
7. Use good style and layout and comment your code well.
8. Use file redirection to test your program. That's how I will test it.
9. Submit your ONE source code file to the eLearning drop box for this project. Don't submit a compressed file.
10. **There will be a 1% penalty for each minute of lateness. After 60 late minutes, a grade of zero will be recorded.**

Study the sample data files on eLearning and send questions and report errors or suspicions of errors to ivor@utdallas.edu