Name : Ramzy Izza Wardhana
NIM : 21/472698/PA/20322
Class : IUP CS1

Assignment 5 - Lab Algorithm and Data Structure

Full Source Code on Zip file :
https://drive.google.com/file/d/1s19vDElfywDpICeCYgXPxRpXM4ge7hHZ/view?usp=sharing

1. Create heap tree for array with size of 15 and data key
   Key = {78, 3, 9, 10, 23, 77, 34, 86, 90, 100, 20, 66, 94, 63, 97}

```
Declare the size of data you want to input: 15
Input data seperated by spaces: 78 3 9 10 23 77 34 86 90 100 20 66 94 63 97

Max Heap Array [Unsorted]:
100 90 97 78 86 77 94 3 23 10 20 9 66 34 63
```

2. Create a heap sort function to use heap from this chapter.

```
Heap Array in Ascending [Sorted with HeapSort Method]:
3 9 10 20 23 34 63 66 77 78 86 90 94 97 100

Heap Array in Descending [Sorted with HeapSort Method]:
100 97 94 90 86 78 77 66 63 34 23 20 10 9
```

Screenshots of Output:

```
PS C:\Users\themi\Downloads\Kuliah\Semester 2\Algorithm & Data Structure\Assignment
 2\Algorithm & Data Structure\Assignment or Quiz\Assignment 5 Lab ASD Heap\Main.java
nExceptionMessages' '-cp' 'C:\Users\themi\Downloads\Kuliah\Semester 2\Algorithm & Da

Declare the size of data you want to input: 15
Input data seperated by spaces: 78 3 9 10 23 77 34 86 90 100 20 66 94 63 97

Max Heap Array [Unsorted]:
100 90 97 78 86 77 94 3 23 10 20 9 66 34 63

Heap Array in Ascending [Sorted with HeapSort Method]:
3 9 10 20 23 34 63 66 77 78 86 90 94 97 100

Heap Array in Descending [Sorted with HeapSort Method]:
100 97 94 90 86 78 77 66 63 34 23 20 10 9

PS C:\Users\themi\Downloads\Kuliah\Semester 2\Algorithm & Data Structure\Assignment
```

Screenshot of Code:

Node.java

```java
package HeapAssignment;

public class Node {
    private int num;
    public Node(int value){
        num = value;
    }
    public int getKey(){
        return num;
    }
}
```

Main.java

```java
package HeapAssignment;
/*
Ramzy Izza Wardhana
21/472698/PA/20322
Heap Assignment
*/
import java.util.*;
public class Main {
    Run | Debug
    public static void main(String[]args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Declare the size of data you want to input: ");
        int arrSize = sc.nextInt();
        Heap heap = new Heap(arrSize);
        int[] heapSort = new int[arrSize+1];

        System.out.print("Input data seperated by spaces: ");
        for(int i = 0; i < arrSize; i++){
            int input = sc.nextInt();
            heap.insert(input); //insert value to heap array
            heapSort[i] = input; //insert value to heapSort array
        }

        System.out.println("\nMax Heap Array [Unsorted]:  ");
        for (int i = 1;i <= arrSize; i++){
            System.out.print(heap.heap[i].getKey() + " "); //output the value of max heap
        }

        System.out.println("\n\nHeap Array in Ascending [Sorted with HeapSort Method]: ");
        heap.sort(arrSize, heapSort); //invoke the heap sort function to sort the data
        for (int i = 0; i < arrSize; i++){
            System.out.print(heapSort[i] + " "); //output the value of min heap to max heap
        }

        System.out.println("\n\nHeap Array in Descending [Sorted with HeapSort Method]: ");
        for (int i = arrSize-1; i > 0; i--){
            System.out.print(heapSort[i] + " "); //output the value of max heap to min heap
        }
        System.out.println("\n");
    }
}
```

Heap.java

```java
package HeapAssignment;

public class Heap {
    Node[] heap; //heap implemented on array
    int sizeMax; //bound for array
    int size; //current size of heap array

    //set constructor for heap
    public Heap(int max){
        this.sizeMax = max;
        this.size = 0;
        this.heap = new Node[sizeMax +1]; //declare size of heap which index start at 1
    }
    public boolean isFull(){
        return size == sizeMax; //if size == maxInput, return true
    }

    //Insertion method
    public boolean insert(int num){ //int as parameter
        if(isFull())
            return false; //base case
        Node newNode = new Node(num); //create new node
        size++; //increase the size
        heap[size] = newNode; //insert the value to the array
        trickleUp(size); //recursive trickling up to max

        return true;
    }
    //trickle up method for inserting the max to the root
    public void trickleUp(int index){
        int parent = index/2;
        Node last = heap[index]; //last data on bottom == last index
        while(index > 1 && last.getKey() > heap[parent].getKey()){ //trickle up the biggest value to the root
            heap[index] = heap[parent];
            index = parent;
            parent = index/2;
        }
        heap[index] = last;
    }
```

```java
    //heapify function to trickle the value to the root recursively
    public void heapcursive(int index, int heap[], int size ){
        int max = index; //root
        //if value on left > value of known max, replace the max
        if((2*index)+1 < size && heap[(2*index)+1] > heap[max])
            max = (2*index)+1;
        //if value on right > value of known max, replace the max
        if((2*index)+2 < size && heap[(2*index)+2] > heap[max])
            max = (2*index)+2;;
        //if max != with index, substitute it
        if(max != index){
            int hold = heap[index];
            heap[index] = heap[max];
            heap[max] = hold;

            heapcursive(max, heap, size); //recursive
        }
    }
    //heapSort algorithm
    public void sort(int size, int heap[]){
        //invoke the heapify from index/2 -1 to 0
        for(int i = (size/2)-1; i >= 0; i--){
            heapcursive(i, heap, size);
        }
        for(int i = size-1; i>0; i--){
            int hold = heap[0];
            heap[0] = heap[i];
            heap[i] = hold;

            heapcursive(0, heap, i);
        }
    }
}
```