Problem Solution Approach

I used node in HeapTree class and this node has one heap list.

And I used node for heap list and this node has two value.

I used generic types for you, so you can use this heap tree for any type.

Detailed system Requirements

FUNCTİONAL REQIREMENTS		
FR01	`find_mod()`	It returns mod number.
FR02	`remove(E element)`	You can remove any specific element.
FR03	`add(E newValue)`	You can add element.
FR04	`find(E element)`	It finds specific element.
FR05	`travelPrint()`	You can print every node of HeapTree. Every line has heap list.

Class Diagrams

Class diagram is in the file.

Test Cases

	FUNCTİONAL REQIREMENTS
TC01	Add element to empty heap.
TC02	Find element in heap.
TC03	Delete specific element in heap.
TC04	Add element that already in the heap.
TC05	Find element that is not in the heap.

Running Command and Results

```
HeapTree<Integer> NewSecondTree = new HeapTree<Integer>();
HeapTree<Integer> NewTree = new HeapTree<Integer>();


for(int i=0; i<3000 ; i++)
{
    int randN = rand.nextInt(100);
    NewSecondTree.add(randN);
}

try {
    System.out.println("Mod Number"+NewSecondTree.find_mod()+"  "+"Mod NumberOfOcurrence-"+NewSecondTree.getNumberOfOcurrence
}
catch(IllegalAccessException e) {
    System.out.println("There are not any element in this heap.");
}

 NewTree.add(16);
 NewTree.add(17);
 NewTree.add(125);
 NewTree.add(122);
 NewTree.add(1000);
 NewTree.add(102);
 NewTree.add(109);
 NewTree.add(11);
 NewTree.add(1);
 NewTree.add(5);
 NewTree.add(102);
 NewTree.add(109);
 NewTree.add(11);
 NewTree.add(1000);

 NewTree.add(1100);
 NewTree.add(1200);
 NewTree.add(500);
 NewTree.add(105);
 NewTree.add(107);
 NewTree.add(1100);
 NewTree.add(1040);

 NewTree.add(122);
 NewTree.add(122);
 NewTree.add(122);

 NewTree.add(10);

 NewTree.add(1);
 NewTree.add(5);

 try {
     NewTree.remove(500);
     NewTree.remove(125);
     NewTree.remove(1000);
     NewTree.remove(1000);

 }
 catch(IllegalAccessException e) {
     System.out.println("There are not any element in this heap.");
 }

 NewTree.add(5000);

 System.out.println("--HeapTree--");
 NewTree.travelPrint();
 System.out.println();

 try {
     System.out.println("Mod Number-"+NewTree.find_mod());
 }
 catch(IllegalAccessException e) {
     System.out.println("There are not any element in this heap.");
 }

 System.out.println("If element is founded it will print numberofocurrence otherwise it will print -1 : "+NewTree.find(1100));
 System.out.println("If element is founded it will print numberofocurrence otherwise it will print -1 : "+NewTree.find(11));
 System.out.println("If element is founded it will print numberofocurrence otherwise it will print -1 : "+NewTree.find(1040));
 System.out.println("If element is founded it will print numberofocurrence otherwise it will print -1 : "+NewTree.find(122));
```

```
Mod Number97  Mod NumberOfOcurrence-38


--HeapTree--
5000-1 || 102-2 || 122-4 || 16-1 || 17-1 || 109-2 ||
107-1 || 105-1 || 10-1 || 1-2 || 11-2 || 5-2 ||
1200-1 || 1100-2 || 1040-1 ||

Mod Number-122
If element is founded it will print numberofocurrence otherwise it will print -1 : 2
If element is founded it will print numberofocurrence otherwise it will print -1 : 2
If element is founded it will print numberofocurrence otherwise it will print -1 : 1
If element is founded it will print numberofocurrence otherwise it will print -1 : 4
```