Problem Solution Approach

I used array list for every node in the heap. Because this is usual approach for heap implementation.

Detailed system Requirements

FUNCTİONAL REQIREMENTS		
FR01	`insertNode(E i)`	You can send any type for the adding node to the heap.
FR02	`Search(E element)`	You can search any element.
FR03	`mergeHeap(Heap<E> New)`	You don't have to send null heap.
FR04	`printList()`	It will print the heap like list.

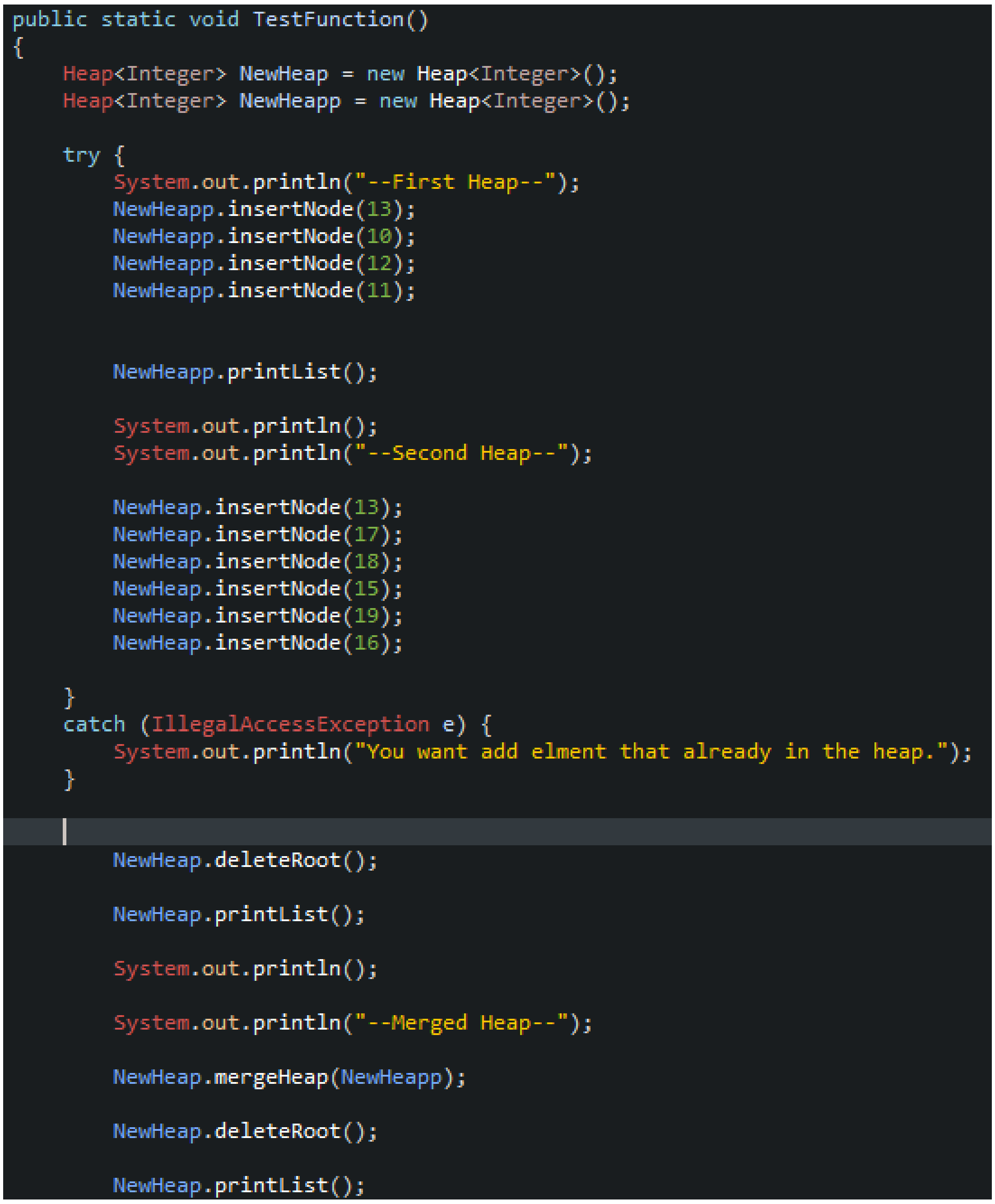
Class Diagrams

Class diagram is in the file.

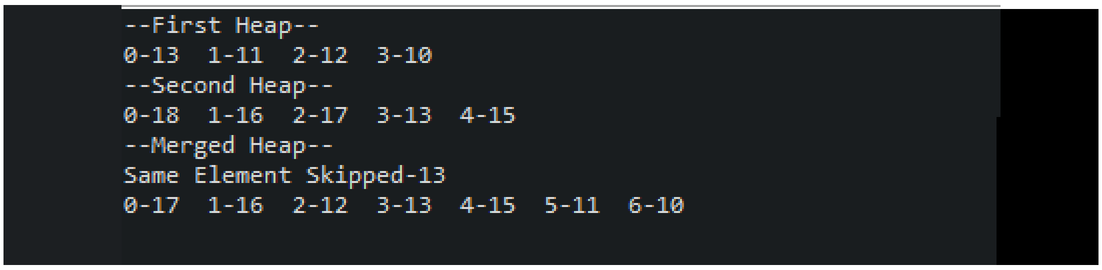
Test Cases

FUNCTİONAL REQIREMENTS	
TC01	Add element to empty heap.
TC02	Print heap.
TC03	Merge two heap.
TC04	Delete last element in the heap.
TC05	Add element that already in the heap.
TC06	Merge two heap but one heap has a element that already in the other heap.

Running Command and Results



```
public static void TestFunction()
{
    Heap<Integer> NewHeap = new Heap<Integer>();
    Heap<Integer> NewHeapp = new Heap<Integer>();

    try {
        System.out.println("--First Heap--");
        NewHeapp.insertNode(13);
        NewHeapp.insertNode(10);
        NewHeapp.insertNode(12);
        NewHeapp.insertNode(11);


        NewHeapp.printList();

        System.out.println();
        System.out.println("--Second Heap--");

        NewHeap.insertNode(13);
        NewHeap.insertNode(17);
        NewHeap.insertNode(18);
        NewHeap.insertNode(15);
        NewHeap.insertNode(19);
        NewHeap.insertNode(16);

    }
    catch (IllegalAccessException e) {
        System.out.println("You want add elment that already in the heap.");
    }

        NewHeap.deleteRoot();

        NewHeap.printList();

        System.out.println();

        System.out.println("--Merged Heap--");

        NewHeap.mergeHeap(NewHeapp);

        NewHeap.deleteRoot();

        NewHeap.printList();
```



```
--First Heap--
0-13  1-11  2-12  3-10
--Second Heap--
0-18  1-16  2-17  3-13  4-15
--Merged Heap--
Same Element Skipped-13
0-17  1-16  2-12  3-13  4-15  5-11  6-10
```