GTU DEPARTMENT of COMPUTER ENGINEERING

CSE222/505 – Spring 2022

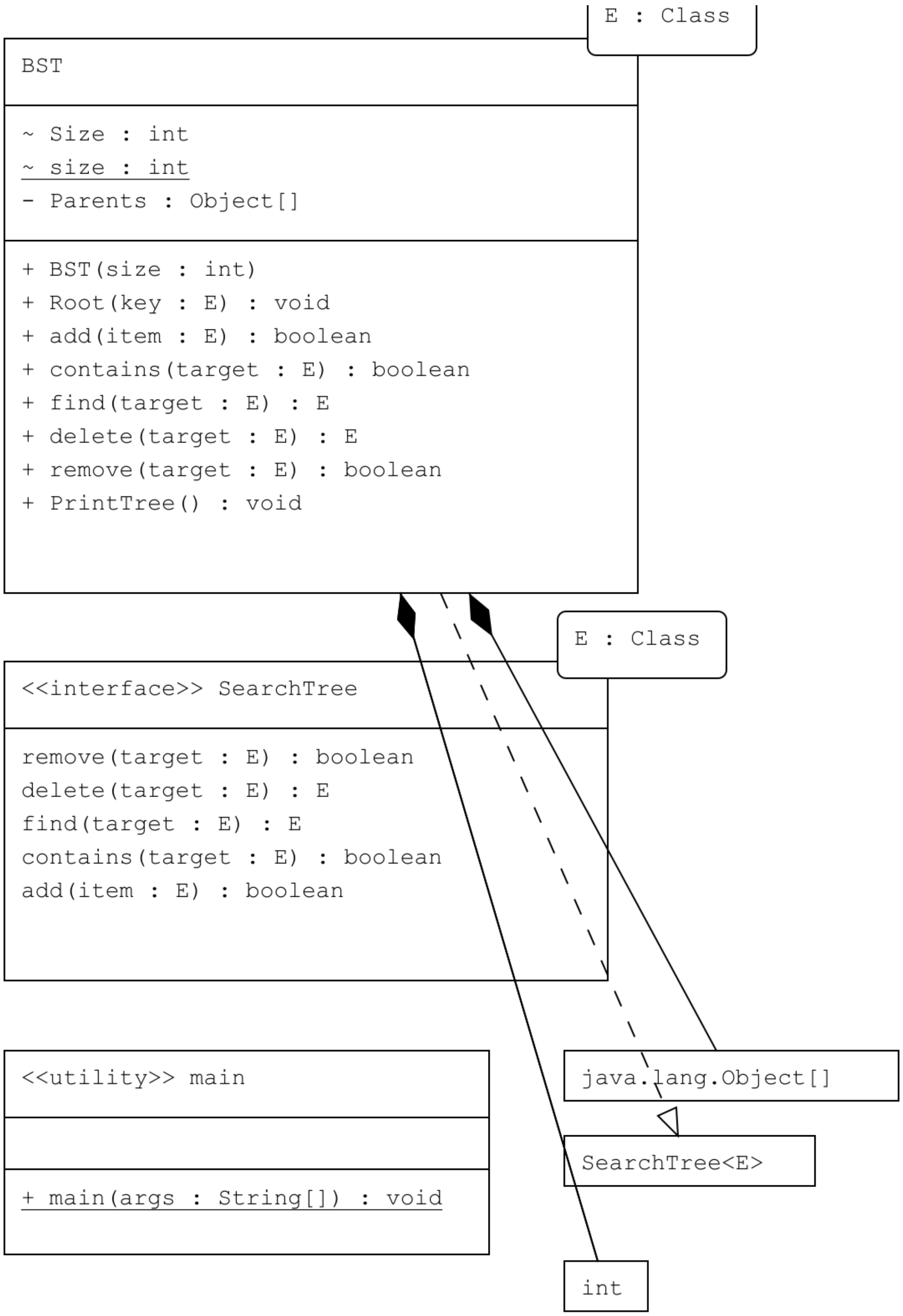
HOMEWORK #01 Report

Muhammet Çağrı Yılmaz

1901042694

QUESTION 3

1. CLASS DIAGRAM



2. PROBLEM SOLUTION APPROACH

My Problem solution steps are that;

– Specify the problem requirements

– Analyze the problem

– Design an algorithm and Program

– Implement the algorithm

– Test and verify the program

– Maintain and update the program

Specify the problem: I understand the problem.

Analyze the problem:

– Input data

– Output data

– Additional requirements and constraints

Design an algorithm and Program : I wrote the algorithm in Java by converting each step into statements of Java (classes ,methods etc.) Firstly,according to the assignment, I created an Interface whose name is SearchTree . There are 5 methods. After that I created a BST(BinarySearchTree) and I implement my Interface. I wrote these Interface and Class from the book which extends from BinaryTree and there are a problem. In book we use Node. In this assignment however we need to use array.

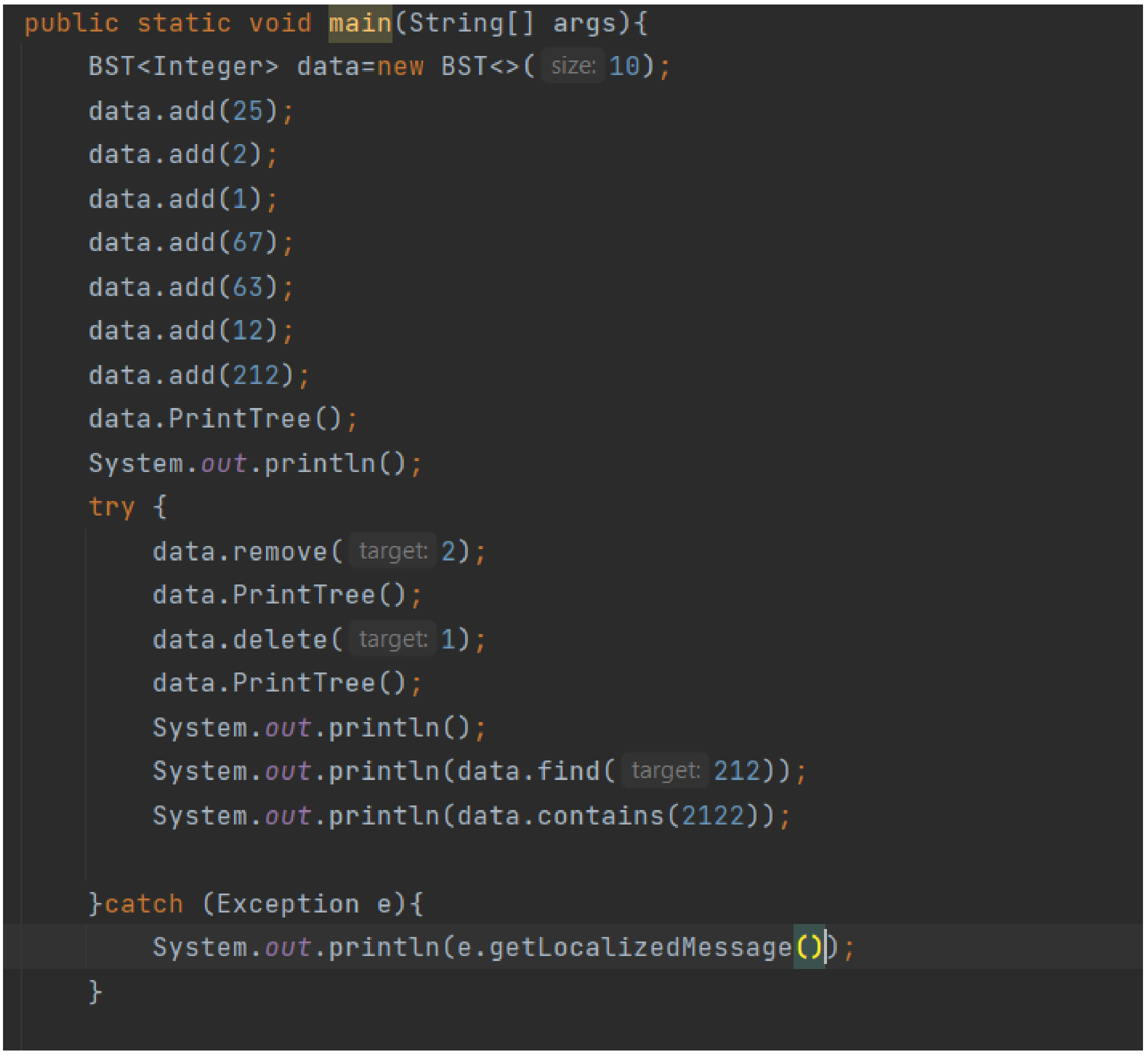
Test and verify the program: To test program I wrote the Main class in this class in main method I test each method of BST class by calling the methods and I printed the test results.

3.TEST CASES

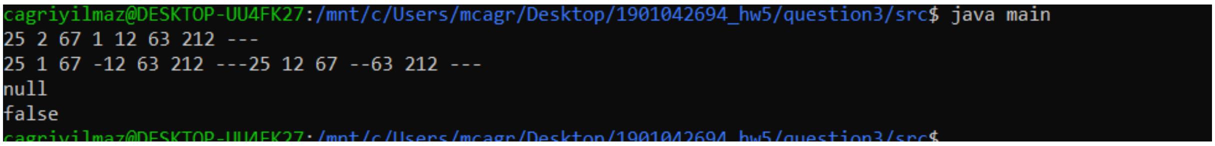
I have 5 methods and these are that add remove find contains delete.

These are working well. You can look 4 th section test results.

4. RUNNING AND RESULTS



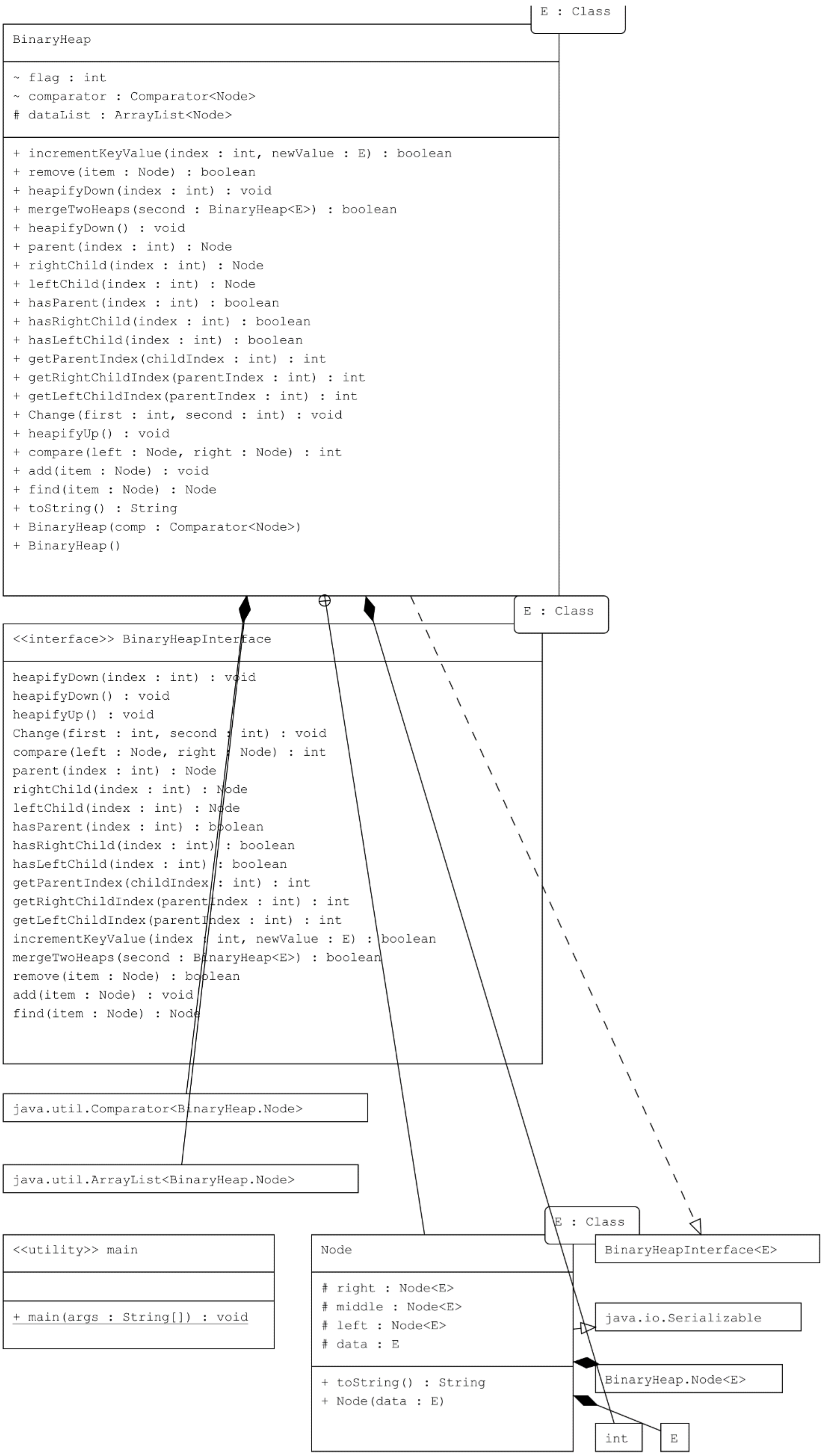
```java
public static void main(String[] args){
    BST<Integer> data=new BST<>( size: 10);
    data.add(25);
    data.add(2);
    data.add(1);
    data.add(67);
    data.add(63);
    data.add(12);
    data.add(212);
    data.PrintTree();
    System.out.println();
    try {
        data.remove( target: 2);
        data.PrintTree();
        data.delete( target: 1);
        data.PrintTree();
        System.out.println();
        System.out.println(data.find( target: 212));
        System.out.println(data.contains(2122));

    }catch (Exception e){
        System.out.println(e.getLocalizedMessage());
    }
```



```
cagriyilmaz@DESKTOP-UU4FK27:/mnt/c/Users/mcagr/Desktop/1901042694_hw5/question3/src$ java main
25 2 67 1 12 63 212 ---
25 1 67 -12 63 212 ---25 12 67 --63 212 ---
null
false
cagriyilmaz@DESKTOP-UU4FK27:/mnt/c/Users/mcagr/Desktop/1901042694_hw5/question3/src$
```

QUESTION 3

1. CLASS DIAGRAM


E : Class
BinaryHeap
~ flag : int
~ comparator : Comparator<Node>
# dataList : ArrayList<Node>
+ incrementKeyValue(index : int, newValue : E) : boolean
+ remove(item : Node) : boolean
+ heapifyDown(index : int) : void
+ mergeTwoHeaps(second : BinaryHeap<E>) : boolean
+ heapifyDown() : void
+ parent(index : int) : Node
+ rightChild(index : int) : Node
+ leftChild(index : int) : Node
+ hasParent(index : int) : boolean
+ hasRightChild(index : int) : boolean
+ hasLeftChild(index : int) : boolean
+ getParentIndex(childIndex : int) : int
+ getRightChildIndex(parentIndex : int) : int
+ getLeftChildIndex(parentIndex : int) : int
+ Change(first : int, second : int) : void
+ heapifyUp() : void
+ compare(left : Node, right : Node) : int
+ add(item : Node) : void
+ find(item : Node) : Node
+ toString() : String
+ BinaryHeap(comp : Comparator<Node>)
+ BinaryHeap()
E : Class
<<interface>> BinaryHeapInterface
heapifyDown(index : int) : void
heapifyDown() : void
heapifyUp() : void
Change(first : int, second : int) : void
compare(left : Node, right : Node) : int
parent(index : int) : Node
rightChild(index : int) : Node
leftChild(index : int) : Node
hasParent(index : int) : boolean
hasRightChild(index : int) : boolean
hasLeftChild(index : int) : boolean
getParentIndex(childIndex : int) : int
getRightChildIndex(parentIndex : int) : int
getLeftChildIndex(parentIndex : int) : int
incrementKeyValue(index : int, newValue : E) : boolean
mergeTwoHeaps(second : BinaryHeap<E>) : boolean
remove(item : Node) : boolean
add(item : Node) : void
find(item : Node) : Node
java.util.Comparator<BinaryHeap.Node>
java.util.ArrayList<BinaryHeap.Node>
<<utility>> main
+ main(args : String[]) : void
E : Class
Node
# right : Node<E>
# middle : Node<E>
# left : Node<E>
# data : E
+ toString() : String
+ Node(data : E)
BinaryHeapInterface<E>
java.io.Serializable
BinaryHeap.Node<E>
int
E

2. PROBLEM SOLUTION APPROACH

My Problem solution steps are that;

– Specify the problem requirements

– Analyze the problem

– Design an algorithm and Program

– Implement the algorithm

– Test and verify the program

– Maintain and update the program

Specify the problem: I understand the problem.

Analyze the problem:

– Input data

– Output data

– Additional requirements and constraints

Design an algorithm and Program : I wrote the algorithm in Java by converting each step into statements of Java (classes ,methods etc.) Firstly,according to the assignment, I created an Interface whose name is BinaryHeapInterface . There are some methods. After that I created a Binary Heap Class and I have a nested class Node to hold items and I implement my Interface. I wrote these Interface and Class from the book which extends from BinaryTree and Binary Heap there are a problem. In book we use Node. In this assignment however we need to use array.

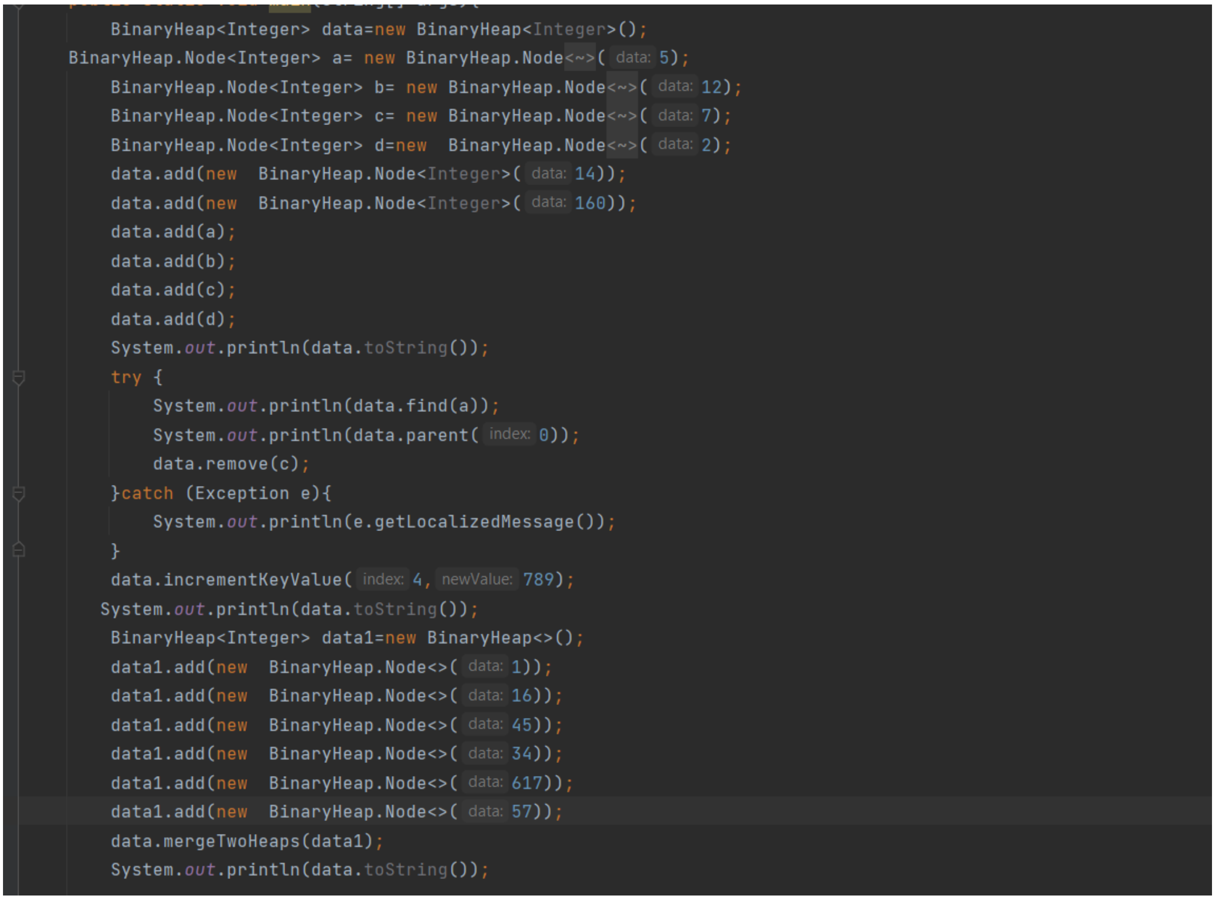
Test and verify the program: To test program I wrote the Main class in this class in main method I test each method of BinaryHeap class by calling the methods and I printed the test results.

3.TEST CASES

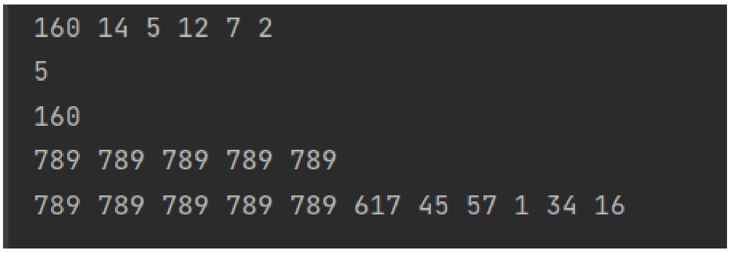
I have some methods and some of them are that add, remove, merge increment.

These are working well. You can look 4 th section test results.

4. RUNNING AND RESULTS



```
public static void main(String[] args){
    BinaryHeap<Integer> data=new BinaryHeap<Integer>();
BinaryHeap.Node<Integer> a= new BinaryHeap.Node<~>( data: 5);
    BinaryHeap.Node<Integer> b= new BinaryHeap.Node<~>( data: 12);
    BinaryHeap.Node<Integer> c= new BinaryHeap.Node<~>( data: 7);
    BinaryHeap.Node<Integer> d=new  BinaryHeap.Node<~>( data: 2);
    data.add(new  BinaryHeap.Node<Integer>( data: 14));
    data.add(new  BinaryHeap.Node<Integer>( data: 160));
    data.add(a);
    data.add(b);
    data.add(c);
    data.add(d);
    System.out.println(data.toString());
    try {
        System.out.println(data.find(a));
        System.out.println(data.parent( index: 0));
        data.remove(c);
    }catch (Exception e){
        System.out.println(e.getLocalizedMessage());
    }
    data.incrementKeyValue( index: 4, newValue: 789);
   System.out.println(data.toString());
    BinaryHeap<Integer> data1=new BinaryHeap<>();
    data1.add(new  BinaryHeap.Node<>( data: 1));
    data1.add(new  BinaryHeap.Node<>( data: 16));
    data1.add(new  BinaryHeap.Node<>( data: 45));
    data1.add(new  BinaryHeap.Node<>( data: 34));
    data1.add(new  BinaryHeap.Node<>( data: 617));
    data1.add(new  BinaryHeap.Node<>( data: 57));
    data.mergeTwoHeaps(data1);
    System.out.println(data.toString());
```



```
160 14 5 12 7 2
5
160
789 789 789 789 789
789 789 789 789 789 617 45 57 1 34 16
```