Gebze Technical University

Computer Engineering

CSE222-2020-SPRING

Homework-5_Part4 Report

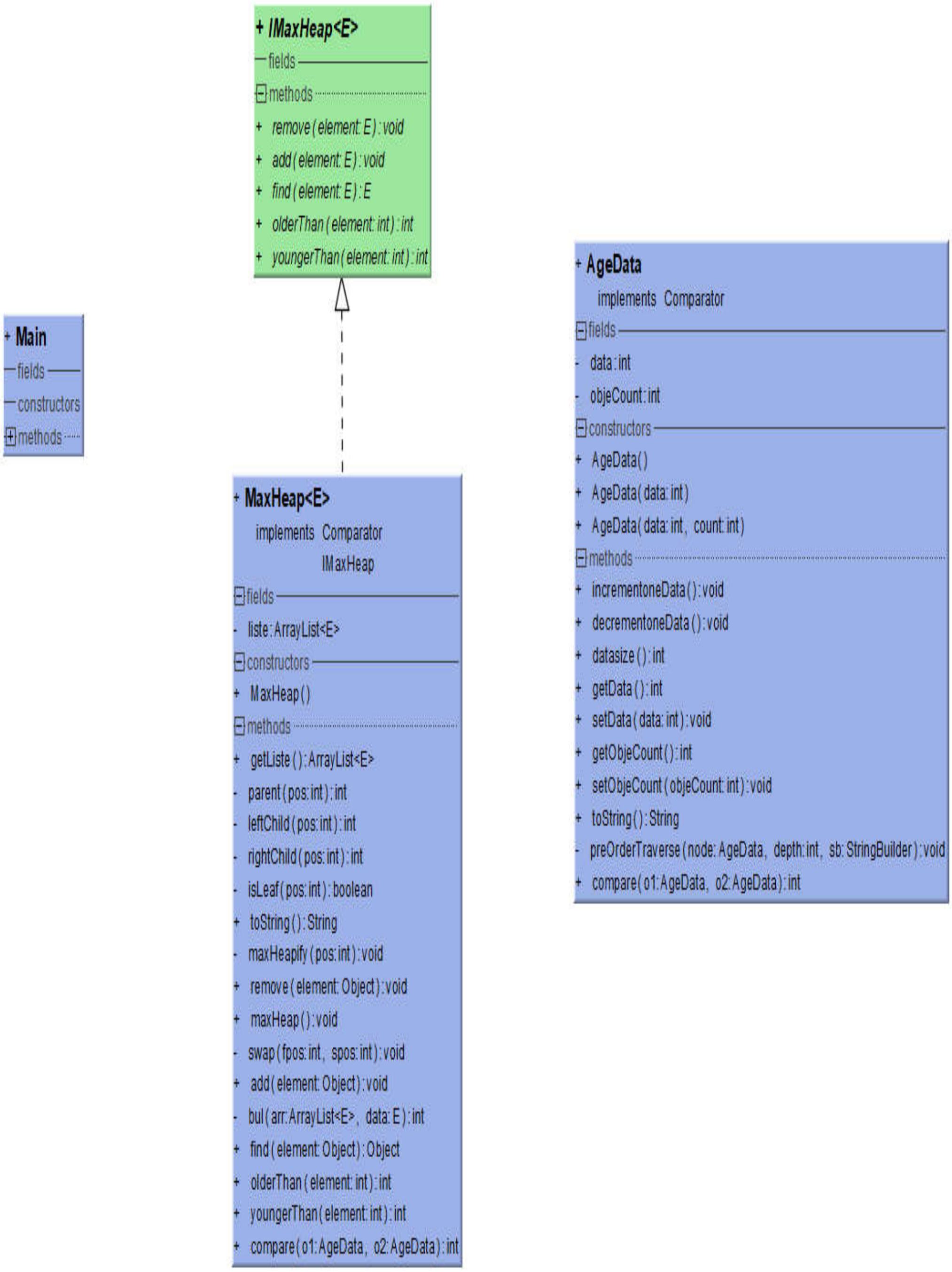
Ferdi Sönmez

161044046

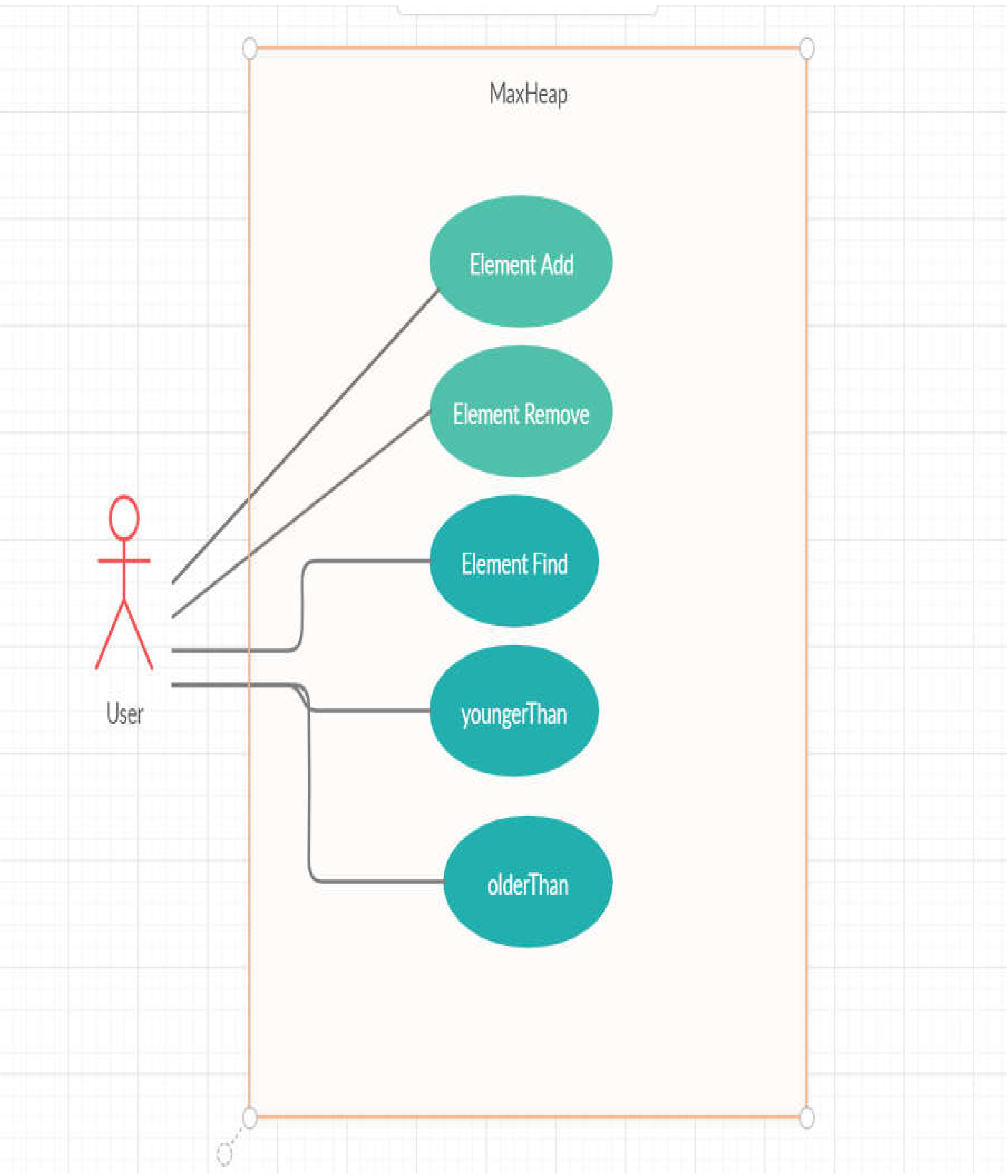
1)Problem Solutions Approach

I created the structure by keeping the arraylist in MaxHeap and adding the arraylist in the MaxHeap structure according to the number of people in the AgeData class. If the number of people in AgeData data is high, the location of this data is changed in MaxHeap structure. In deletions, the number of people in the item changes or is deleted, and the positions of other items change after deletion. Searches and returns the item in the Arraylist in discovery. Compares the item received as a parameter in the OlderThan function with the items in the Arraylist and adds the number of people of the larger ones. The YoungerThan function does the opposite.

2) Class Diagram


+ IMaxHeap<E>
fields
methods
+ remove(element:E):void
+ add(element:E):void
+ find(element:E):E
+ olderThan(element:int):int
+ youngerThan(element:int):int
+ Main
fields
constructors
methods
+ MaxHeap<E>
implements Comparator
IMaxHeap
fields
- liste:ArrayList<E>
constructors
+ MaxHeap()
methods
+ getListe():ArrayList<E>
- parent(pos:int):int
- leftChild(pos:int):int
- rightChild(pos:int):int
- isLeaf(pos:int):boolean
+ toString():String
- maxHeapify(pos:int):void
+ remove(element:Object):void
+ maxHeap():void
- swap(fpos:int, spos:int):void
+ add(element:Object):void
- bul(arr:ArrayList<E>, data:E):int
+ find(element:Object):Object
+ olderThan(element:int):int
+ youngerThan(element:int):int
+ compare(o1:AgeData, o2:AgeData):int
+ AgeData
implements Comparator
fields
- data:int
- objeCount:int
constructors
+ AgeData()
+ AgeData(data:int)
+ AgeData(data:int, count:int)
methods
+ incrementoneData():void
+ decrementoneData():void
+ datasize():int
+ getData():int
+ setData(data:int):void
+ getObjeCount():int
+ setObjeCount(objeCount:int):void
+ toString():String
- preOrderTraverse(node:AgeData, depth:int, sb:StringBuilder):void
+ compare(o1:AgeData, o2:AgeData):int

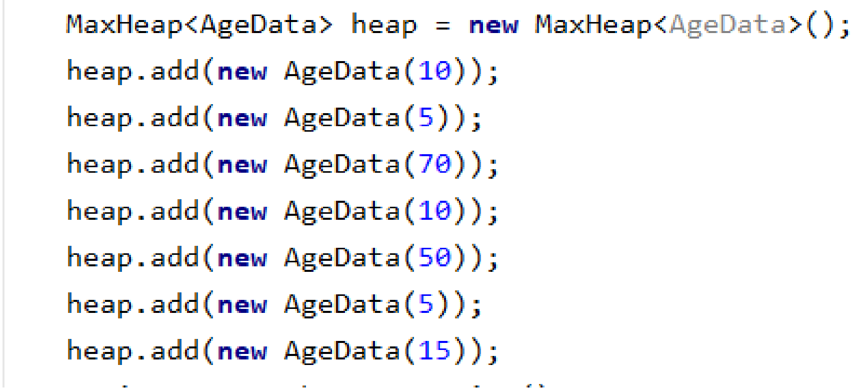
3)Use Case Diagram



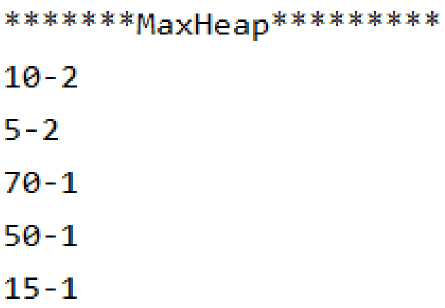
4)Test Case

a)Add

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T01	Given valid integer	Call the method with valid integer number	`heap.add(new AgeData(10));`	Adding new item	Add new element in the MaxHeap	Pass
T02	Given new element that is a space()	Call the method with space()	`heap.add(new AgeData());`	Adding new item	Add new element in the MaxHeap	Pass



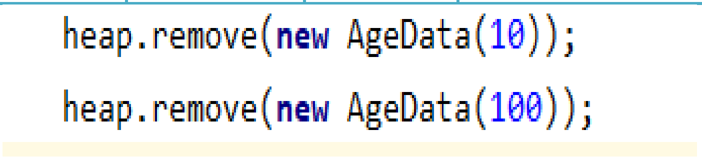
```
MaxHeap<AgeData> heap = new MaxHeap<AgeData>();
heap.add(new AgeData(10));
heap.add(new AgeData(5));
heap.add(new AgeData(70));
heap.add(new AgeData(10));
heap.add(new AgeData(50));
heap.add(new AgeData(5));
heap.add(new AgeData(15));
```



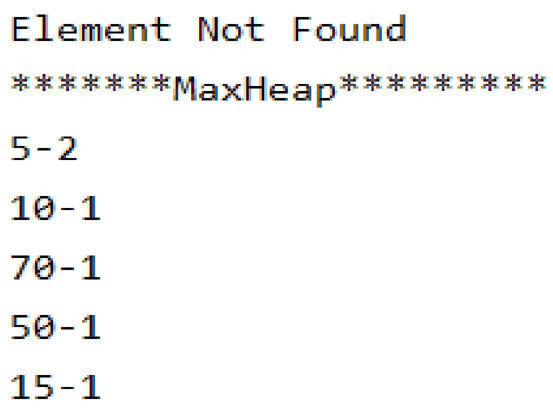
```
*******MaxHeap*********
10-2
5-2
70-1
50-1
15-1
```

b)Remove

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T01	Given valid value(one of the previously added values)	Call the method valid value	heap.remove(new AgeData(10));	The specified item has been deleted	The number of people in the AgeData has been reduced	Pass
T02	Given İnvalid value(An attempt to delete an element that is not in MaxHeap)	Call Method invalid value	heap.remove(new AgeData(100));	No element found	Element Not Found	Pass



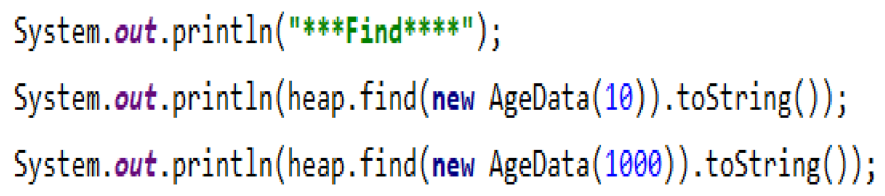
```
heap.remove(new AgeData(10));
heap.remove(new AgeData(100));
```



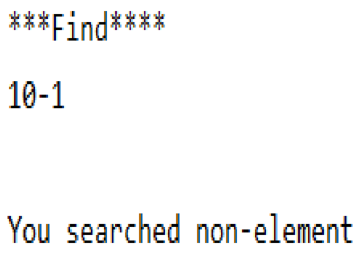
```
Element Not Found
*******MaxHeap*********
5-2
10-1
70-1
50-1
15-1
```

c)Find

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T01	One of the items in MaxHeap was searched	Call the method valid value	`System.out.println(heap.find(new AgeData(10)).toString());`	Item found and its value and prints the number of people in it	The value found and the number of people in it are printed on the screen	Pass
T02	If you want to search for an item that is not in MaxHeap	Call Method with invalid value	`System.out.println(heap.find(new AgeData(1000)).toString());`	must be a statement that the item is missing	throw the exception and catch in main if item is not found	Pass



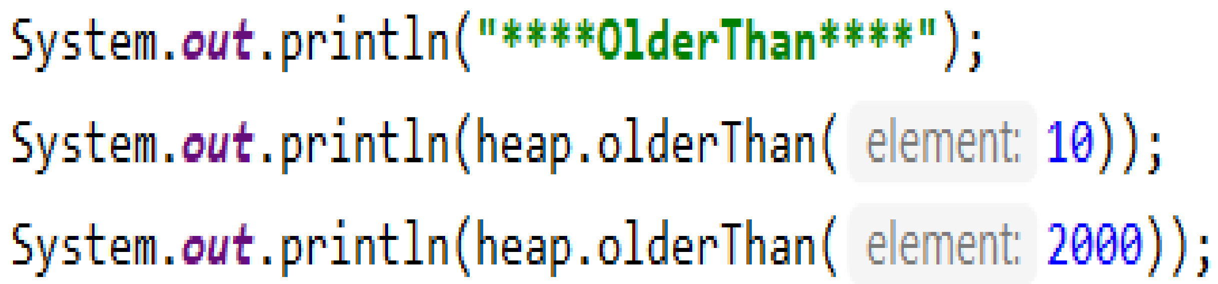
```
System.out.println("***Find****");
System.out.println(heap.find(new AgeData(10)).toString());
System.out.println(heap.find(new AgeData(1000)).toString());
```



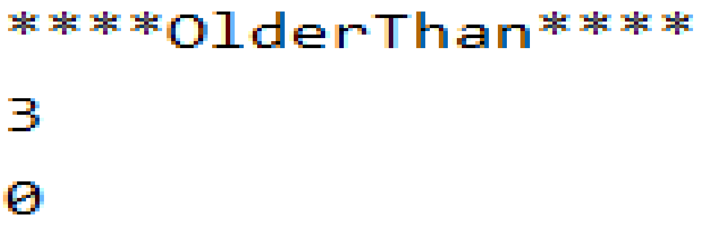
```
***Find****
10-1


You searched non-element
```

d)olderThan

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
T01	Given valid value	Call method valid input	`System.out.println(heap.olderThan(10));`	Shows how many people are larger than the given value	Total number of those greater than the given value	Pass
T02	Given invalid value	Call method invalid input	`System.out.println(heap.olderThan(2000));`	0	0	Pass



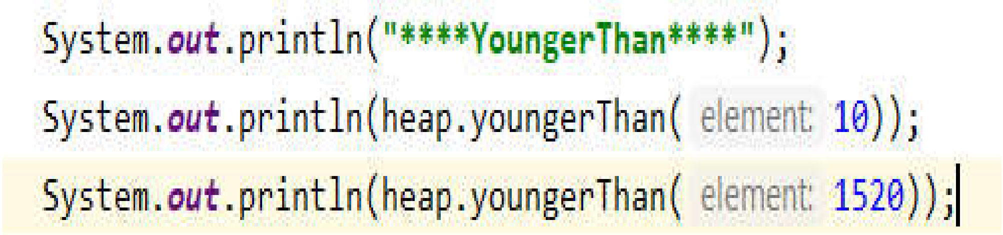
```
System.out.println("****OlderThan****");
System.out.println(heap.olderThan( element: 10));
System.out.println(heap.olderThan( element: 2000));
```



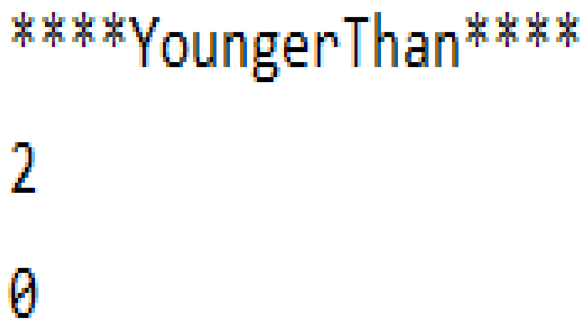
```
****OlderThan****
3
0
```

e)youngerThan

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/ Fail
T01	Given valid value	Call method valid input	`System.out.println(heap.youngerThan(10));`	Returns the sum of the number of people who are younger than the given value	Number of young people	Pass
T02	Given invalid value	Call method invalid input	`System.out.println(heap.youngerThan(1520));`	Returns 0 if the given value is not in MaxHeap	0	Pass



```
System.out.println("****YoungerThan****");
System.out.println(heap.youngerThan( element: 10));
System.out.println(heap.youngerThan( element: 1520));
```



```
****YoungerThan****

2

0
```