GTU Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework #4 Report

Yunus Emre Öztürk
1901042619

1. SYSTEM REQUIREMENTS

Non-Functional System Requirements:

- 50 KB memory (For tests)
- Java Runtime Environment

Functional System Requirements (Heap):

```
public HeapPartOne(int cap)
public HeapPartOne(Comparator<? super E> comparator)
public HeapPartOne(HeapPartOne<E> other)
```

There is three constructor other than the default one (Object should be created with one of these or default constructor):

- Constructor that takes capacity, defines a starting capacity for Object.
- Constructor that takes Comparator, takes a comparator for compare method inside of the Object.
- Copy constructor, to copies the elements from other Heap and creates a new Heap.

```
public boolean search(Object o)
```

User of this class can search for an element inside of heap by giving object as parameter.

```
public boolean merge(HeapPartOne<E> other)
```

- User can merge another heap object with current heap object by giving other one as parameter.

```
public boolean removeBiggest(int i)
```

- Removes the biggest ith element from heap, needs to take index as input.

```
public E set(E element)
```

- In heap iterator there is a set method which changes the last returned element from heap to given element.

```
public int compareTo(HeapPartOne<E> o)
```

- User can compare another heap with current heap, comparison happens by comparing peek values of the heaps.

Other than these methods, users can use methods from the priority queue stated in the Java documentation.

Functional System Requirements (BSTHeapTree):

User can create objects of this class by using default constructor. It uses heap and binary search tree to store elements inside of it.

```
public int add(E item)
```

- User must give the item as parameter and use this method to add item to the structure, after insertion can get number of occurrence of item.

```
public int remove(E item)
```

- User must give the item that will be removed as parameter, after deletion can get number of occurrence item, if structure doesn't have the item method will throw NoSuchElementException, this exception should be handled by user.

```
public int find(E item)
```

- User must give the item as parameter to find the number of occurrences of that item inside of structure, if structure doesn't have the item method will throw NoSuchElementException, this exception should be handled by user.

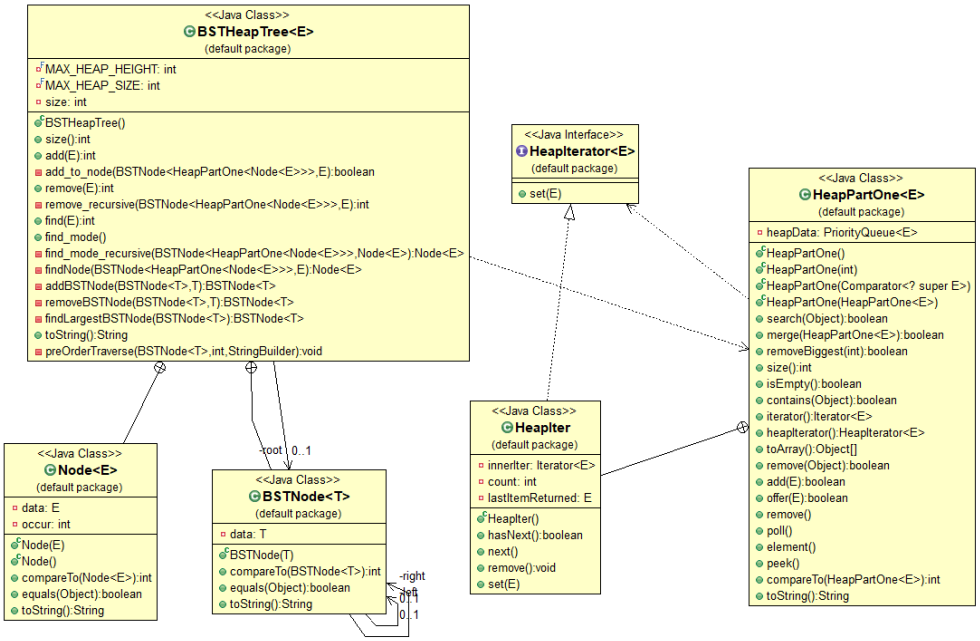
```
public E find_mode()
```

- It returns the mode (item that has most occurrence) as return value, if there is no item in the structure method will throw NullPointerException, this exception should be handled by user.

User can use toString method to print to take the string representation of tree.

2.

USE CASE AND CLASS DIAGRAMS



---

3. PROBLEM SOLUTION APPROACH

For the heap part need to find a structure system to define the methods. I used PriorityQueue to implement the methods, when I think about implementation it doesn't make difference between using PriorityQueue and my implementation of heap, so I used PriorityQueue.

For the BSTHeapTree part the biggest problem is addition after remove, because after a heap node is deleted and a new element added to that heap peek value can change and tree structure can break. So I remove the heap from the tree after a node inside of the heap deleted, and then my program adds the remaining elements from the heap to the structure again.

TEST CASES / RUNNING AND RESULTS

Heap

```
Search Test:
20 integer added to the heap
0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-
Search for 20 integer:
0 is found
1 is found
2 is found
3 is found
4 is found
5 is found
6 is found
7 is found
8 is found
9 is found
10 is found
11 is found
12 is found
13 is found
14 is found
15 is found
16 is found
17 is found
18 is found
19 is found

Searching for an element that heap doesn't have:
999 is not found
************************************
Merge:
Created new heap with values from 21 to 35
First heap after merging with second heap:
0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-28-29-30-31-32-33-34-
************************************
Removing ith largest element:
Removing 6th largest element (which is 28 in this case)
First heap after remove:
0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-34-29-30-31-32-33-
Trying to remove negative index -1:
Method returned false
Trying to remove an index bigger than size (60):
Method returned false
Last situation in first heap:
0-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-21-22-23-24-25-26-27-34-29-30-31-32-33-
************************************
Set method in iterator:
Seth 5th element with value of 4 to 999:
0-1-2-3-9-5-6-7-8-19-10-11-12-13-14-15-16-17-18-33-21-22-23-24-25-26-27-34-29-30-31-32-999-
Trying to use set again before using next (Return value should be null):
Return value: null
--------------------------------------------------------------------------
```

BSTHeapTree

```
Number: 156
Occurence count in array: 2
Occurence count in BSTHeapTree: 2

Number: 159
Occurence count in array: 2
Occurence count in BSTHeapTree: 2

Number: 160
Occurence count in array: 2
Occurence count in BSTHeapTree: 2

Number: 164
Occurence count in array: 1
Occurence count in BSTHeapTree: 1

Number: 165
Occurence count in array: 1
Occurence count in BSTHeapTree: 1

Number: 167
Occurence count in array: 1
Occurence count in BSTHeapTree: 1

Number: 168
Occurence count in array: 2
Occurence count in BSTHeapTree: 2

Number: 169
Occurence count in array: 1
Occurence count in BSTHeapTree: 1
```

```
TRYING TO FIND ELEMENTS THAT TREE DOESN'T HAVE
Number: 5001
NoSuchElementException has been throwed, test is successful
Number: 5002
NoSuchElementException has been throwed, test is successful
Number: 5003
NoSuchElementException has been throwed, test is successful
Number: 5004
NoSuchElementException has been throwed, test is successful
Number: 5005
NoSuchElementException has been throwed, test is successful
Number: 5006
NoSuchElementException has been throwed, test is successful
Number: 5007
NoSuchElementException has been throwed, test is successful
Number: 5008
NoSuchElementException has been throwed, test is successful
Number: 5009
NoSuchElementException has been throwed, test is successful
Number: 5010
NoSuchElementException has been throwed, test is successful
```

```
Number will be removed: 23
Occurence in array before remove: 1
Occurence in tree before remove: 1
Occurence in array after remove: 0
Occurence in tree after remove: 0

Number will be removed: 24
Occurence in array before remove: 2
Occurence in tree before remove: 2
Occurence in array after remove: 1
Occurence in tree after remove: 1

Number will be removed: 24
Occurence in array before remove: 1
Occurence in tree before remove: 1
Occurence in array after remove: 0
Occurence in tree after remove: 0

Number will be removed: 25
Occurence in array before remove: 1
Occurence in tree before remove: 1
Occurence in array after remove: 0
Occurence in tree after remove: 0
```

```
MODE TEST
Mode in array: 1613
Count of mode in array: 5
Mode in tree: 1613
Count of mode in tree: 5
```

```
TRYING TO REMOVE ITEMS THAT NOT IN THE ARRAY
Number: 5001
NoSuchElementException has been throwed, test is successful

Number: 5002
NoSuchElementException has been throwed, test is successful

Number: 5003
NoSuchElementException has been throwed, test is successful

Number: 5004
NoSuchElementException has been throwed, test is successful

Number: 5005
NoSuchElementException has been throwed, test is successful

Number: 5006
NoSuchElementException has been throwed, test is successful

Number: 5007
NoSuchElementException has been throwed, test is successful

Number: 5008
NoSuchElementException has been throwed, test is successful

Number: 5009
NoSuchElementException has been throwed, test is successful

Number: 5010
NoSuchElementException has been throwed, test is successful
```