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

Part-1

Ömer Faruk Bitikçioğlu
161044010

1. SYSTEM REQUIREMENTS

1- Introduction

1.1- Purpose

To give extra capabilities to heap (priority queue) structure.

1.2- Intended Audience

Teaching assistances of Computer Engineering, Gebze Technical University

1.3- Intended Use

This version of heap structure have some additional features like searching for an item in the heap, merging with another heap, removing ith largest element, and has a different iterator with the capability of setting value of last item returned.

2- Overall Description

2.1- User Needs

User needs to know how priority queues work and use this new version as it is plus additional features.

3- System Features and Requirements

3.1- Functional Requirements

The user of the program should use the priority queue in a way it is, plus with additional features. The new priority queue called MyPriorityQueue has the capability of searching an element, merging with another MyPriorityQueue, and removing ith largest element of the heap structure.

3.2- System Features

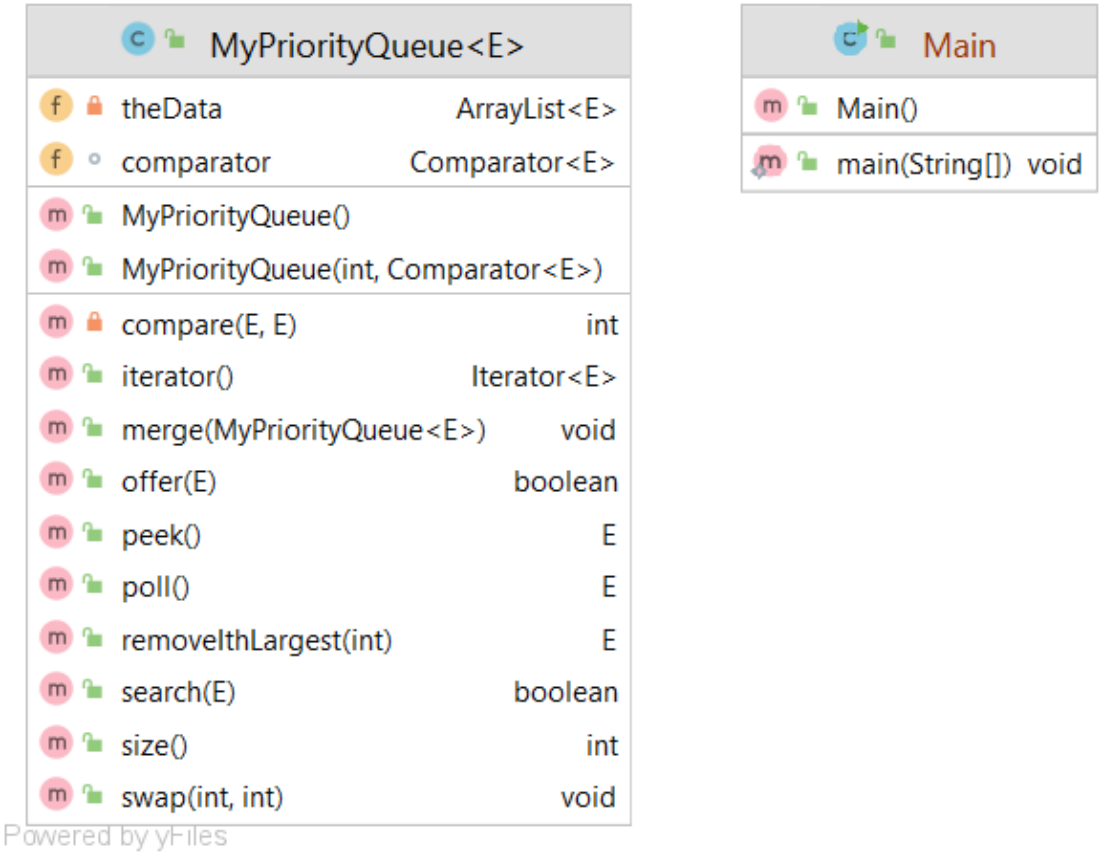
The new heap (priority queue) structure has several new features. First it can search for an element. Second it can merge with another heap structure while keeping heap structure. Third it can find and remove ith largest element at the

heap. Last but not least, its iterator can set the value of last item returned from next methods.

3.3- Nonfunctional Requirements

A computer with a modern java development environment and compiler.

2. CLASS DIAGRAM



3. PROBLEM SOLUTION APPROACH

To add new features to the heap structure, I used the priority queue implementation of the book. I copied the code and added necessary methods for new features by designing their algorithms and delegating some existing methods of the priority queue.

Searching for an item uses linear search on the array list. It uses the compare method of the priority queue to compare the given item with each element of the priority queue.

Merging with another heap depends on a simple logic. The method takes the priority queue to be merged and polls items one by one and offers these items to the existing priority queue. Since poll and offer does not change the heap structure, they are safe to use. At the end these two priority queues will be merges successfully by keeping the correct structure and order.

The ith largest element in the heap is the (size+1-i)th smallest item in the heap. So, I updated the value of i with size+1-i. The reason I do this is because the heap always keeps the smallest item at root. If we can find the equivalent smallest item by polling one by one to the updated i value, we can remove this item. Removed items are stored in a temporary array list. When the process of polling to the i is done the last item of temporary array list is removed. After that unnecessarily deleted items are added to the heap one by one. By doing so ith largest element of heap is removed and heap structure and order is not interfered.

To add a new method to iterator I simply returned from iterator method a new kind of iterator with the new method included.

4. TEST CASES

- Search for existing element
- Search for non-existing element
- Merge with another heap
- Remove 1st largest element which is 96
- Remove 3rd largest element which is 55
- Remove smallest element which is 6th largest element is 15
- Check if the rest of the items are removed or not

5. RUNNING AND RESULTS

```
Value 20 is in the heap. Passed!
Value 230 is not in the heap. Passed!
Two heaps are merged successfully. Passed!
Largest element removed successfully. Passed!
3th largest element removed successfully. Passed!
6th largest element removed successfully. Passed!
Rest of the elements stays safe. Passed!
```