Lab 13 -- Min-Max Priority Queue (CANCELLED)

The goal of this design and implement an Adapter design pattern.

Min-Max Priority Queue

A priority queue is a popular data structure. It is useful in many applications (e.g. simulating customers at an airline counter) and algorithms (e.g. Dijkstra's shortest path algorithm). A priority queue stores items, each of which is associated with a priority. The priority is usually denoted as a single number. The main operations offered by a priority queue are addition, and removal of the item of highest (or lowest) priority. A priority queue can be implemented in several ways: one of the most popular is a heap data structure. The binary heap allows addition and removal (the minimum or maximum priority, depending on implementation) in *O(log n)*, where *n* is the number of items in the queue.

A drawback of a typical priority queue implementation is that it allows efficient removal of *either* the item with minimum priority or the maximum priority, not both. In this lab you will implement a priority queue data structure that allows addition, and  *O(log n)* (or better) removal of both maximum and minimum priority item.

What to do

**Package:** priority

Write a generic interface MinMaxPriorityQueue that can hold items of any type. It offers the following operations:

* void add(T item, int priority) -- add an item with the given priority into the queue.
* T minPriorityItem() -- remove and return the item with the minimum priority (as defined by the lowest numeric priority above). If such an item does not exist, this method should return null.
* T maxPriorityItem() -- remove and return the item with the maximum priority (as defined by the highest numeric priority above). If such an item does not exist, this method should return null.

Now write an implementation MinMaxPriorityQueueImpl of the above interface. This interface must obey the following constraints and have the following characteristics:

1. There may be multiple items with the same priority. Your implementation must handle this. In case there are several items of the lowest (or highest) priority, the queue must remove them in the order in which they were added.
2. It must be implemented using existing data structures (i.e. you must not implement a heap data structure yourself, but *use* existing interfaces and classes in Java. In other words, use the idea of adapters and facades.
3. You are not allowed to use any existing implementations of a priority queue (e.g. the PriorityQueue class) in the JDK in any way.
4. You must attempt to make your implementation efficient, at least as efficient as the PriorityQueue class. Read the documentation to learn more about how efficient it is.