Data Structures
Homework 3

Subjects:
Binary Heap

Honor code:

1. Do not copy the answers from any source
2. You may work in small groups but write your own solution (mention this in the homework)
3. Cheating students will face Committee on Discipline (COD)

Submittion date - 14/4 - through moodle

In this homework you'll implement a Max Binary Heap.
Why would you do it?
A bank manager has hired you to help him manage the queues of the people who come to the bank.
The policy is very simple - the one with the more money will enter first (no matter when he joined the line).
The bank manager would also wants to know (whenever he wants) who is the person with the least money in the queue (he wants to have a good laugh) and what is the average of the money in the queue.
One small thing… the bank sometimes… just sometimes... steals money from the person with the biggest amount of money (the one that should get into the bank first). Pay attention that maybe after the steal, this person (who unexplainably wants to stay in the queue), would drop in priority and drop down in the queue.
The bank queue could be initialized in two ways, in both of them the maximal number of expected people is given.

1. There are no people yet in the bank.
2. There are already a set of people, and their data is stored in an array.

You would implement two classes - next to the method you can see the expected complexity (it's obligatory):

<u>ExtendedMaxHeap</u>

constructors:

- public ExtendedMaxHeap(int capacity); O(1)
  Should throw HeapException in case the capacity is bad

- public ExtendedMaxHeap(HeapElement [] elements, int capacity); O(n)
  Should throw HeapException if the elements array is null / of size 0 and if the capacity is smaller than the given array size

methods:

- public HeapElement deleteMax(); O(logn)
  Should throw HeapException if the queue is empty

- public void insert(HeapElement e); O(logn)
  Should throw HeapException on overload

- public long getKeysAverage(); O(1)

- public HeapElement getElementWithMinKey(); O(1)
  Should throw HeapException if the queue is empty

<u>BankQueue</u>

constructors:

- public BankQueue(int maxQueueSize); O(1)
  Should build a BankQueue with specified max size

- public BankQueue(PersonWithMoney [] people, int maxQueueSize); O(n)
  Should build a Bank from a provided array of PersonWithMoney with a max capacity of maxQueueSize

methods:

- public void enqueue(PersonWithMoney p); O(logn)
  Enqueue p to the BankQueue

- public PersonWithMoney dequeue(); O(logn)
  Dequeue the person with the most money

- public void stealFromFirstInQueue(int amount); O(logn)

  Steal amount (of money) from the first person in queue and return him to the queue

- public int getAverageMoneyInQueue(); O(1)

  Return the average money amount of people in the queue

- public PersonWithMoney getPoorestPerson(); O(1)

  Return the person with the least money in the queue (don't remove him from the queue)

Implementation instructions:

1. The heap should be implemented with an array as learned in the class
2. You may add classes and methods that were not defined in the given API, as long as they conform to proper Object Oriented Design.
3. Do not change the given method signatures.
4. Your code will be extracted to a folder with a Main class that will run and test your code.
5. **You shouldn't use any java ready DS (LinkedList, Set, …). Do not import unnecessary packages. If you have question about this one - approach the TA**
6. You can assume that the constructors of BankQueue will be called with correct input.
7. BankQueue shouldn't hold any information about the number of people that are in the queue. Only the ExtendedMaxHeap (that the BankQueue holds) should hold this information
8. You are responsible for testing your code.
9. You should provide full documentation of all classes and methods.
10. All methods should be readable and well documented.
11. All your methods should be efficient.

Here are the files that are given in the project and their explanation

1. BankQueue.java (you should fill it)
2. ExtendedMaxHeap.java (you should fill it)
3. HeapElement.java (don't modify it)
4. PersonWithMoney.java (don't modify it)
5. BankQueueTest.java (don't modify it) - it's a set basic test cases
6. TestOutput.txt (don't modify it) - it's the output of the BankQueueTest.java

Submission instructions:

1. The submission is in pairs (not obligatory but highly recommended).
2. If you submit in pari the submission file will be ID_ID.zip (e.g. 123456789_987654321.zip). If you submit by yourself the file will be named ID.zip (e.g. 123456789.zip)
3. Don't create any directories in the zip - it should include just the java files of the solution
4. If your code won't compile - it won't be checked
5. Test your code with the provided BankQueueTest check class and the provided output file
6. Some functions should throw HeapException when called on a heap with wrong status (deleteMax on an empty heap). The error message doesn't matter as long as it's logical (e.g. "Performing delete on empty stack")
7. In the check process we will unzip your file and replace the BankQueueTest.java with one of our own

Grading

- Correctness: 50%
- Efficiency: 10%
- Robustness: 10%
- Architecture & Design: 10%
- Documentation & Readability: 10%
- Submission instructions 10%