Joanna Klukowska
joannakl@cs.nyu.edu

Project 4: Implementations of a Priority Queue

Due date: Nov. 14, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own . Any sharing or copying of assignments will be considered cheating (this includes posting of partial or complete solutions on Piazza or any public forum). If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

In this project you will be reading the source code for two different implementations of a priority queue data structure. The objectives are to understand that code and be able to answer some questions about it.

The two implementations are:

- Open JDK version of the `PriorityQueue` class - the source code for that class is linked to on the course website (you can also download it with the zip file that provides the source code for all Java classes), the Javadoc documentation page is at https://docs.oracle.com/javase/8/docs/api/java/util/PriorityQueue.html
- the `HeapPriorityQueue` class from the textbook - the source code for that class is linked to on the course website, the discussion about the class and its implementation is in chapter 9 of the book.

You need to answer the following questions regarding the two implementations. Each answer should be placed on a separate page.

1. Both classes are generic classes. The `PriorityQueue` class uses a single generic specifier called `E`. The `HeapPriorityQueue` class uses two generic specifiers called `K` and `V`. Discuss what each of the generic specifiers represent. Explain, in your own words, why the Open JDK implementation uses only one generic type and the book's class uses two.
2. How are the elements stored in each of the implementations. Describe the type of the container used, type of the elements stored and how the size is handled.
3. Which methods of the `PriorityQueue` class may be called when adding an element to an object of type `PriorityQueue` (list all the methods that are called)? Briefly describe what each of these method calls accomplishes (use your own words, do not copy verbatim from the documentation).
4. Which methods of the `HeapPriorityQueue` and the `ArrayList` classes may be called when adding an element to an object of type `HeapPriorityQueue` (list all the methods that are called)? Briefly describe what each of these method calls accomplishes (use your own words, do not copy verbatim from the documentation).
5. Which methods of the `PriorityQueue` class may be called when removing an element from an object of type `PriorityQueue` (list all the methods that are called)? Briefly describe what each of these method calls accomplishes (use your own words, do not copy verbatim from the documentation).
6. Which methods of the `HeapPriorityQueue` and the `ArrayList` classes may be called when removing an element to an object of type `HeapPriorityQueue` (list all the methods that are called)? Briefly describe what each of these method calls accomplishes (use your own words, do not copy verbatim from the documentation). For each method specify which class it is implemented in.
7. The `HeapPriorityQueue` class has a two parameter constructor as follows:
   ```
   public HeapPriorityQueue(K[] keys, V[] values)
   ```
   Consider the code fragment below that uses the `HeapPriorityQueue` class:

Joanna Klukowska
joannakl@cs.nyu.edu

```java
public static void main (String [] args ) {
  String [] words = {"hello", "goodbye", "bye", "good morning",
                        "good evening", "good afternoon" };
  Integer [] sizes = new Integer  [words.length];
  for (int i = 0; i < sizes.length; i++ ) {
    sizes[i] = words[i].length();
  }

  HeapPriorityQueue<Integer, String> hpq =
                  new HeapPriorityQueue<Integer, String> (sizes,words);
  Entry<Integer,String> e = null;
  while (!hpq.isEmpty()) {
    e = hpq.removeMin();
    System.out.println(e.getKey() +", " + e.getValue());
  }
}
```

Rewrite the `main()` method so that it performs the same task as the one above, but uses the `PriorityQueue` class from OpenJDK. Note: you may need to implement additional classes.

8. Both classes provide the `void heapify()` methods. Analyze both methods and determine if they perform the same task or not. If they do, establish correspondence of each step. If they do not, the explain to what extent the tasks overlap and how they differ.

9. Both classes use a class called `Comparator`. Discuss, in your own words, what the purpose of this class is, why it is used and its similarities and differences to the `Comparable` interface.

10. Analyze and discuss the performance of the remove operation in both classes. Use the big-O notation to describe what the running time is. Explain why it is what you think it is. Make sure your discussion covers both implementations.

Grading

The answer to each question is worth 10 points.

How and What to Submit

For the purpose of grading, your answers have to be provided on separate pages. The answers have to be typed - do not submit scanned/photographed hand written documents.

Your should submit your solutions as a single document to Gradescpe. When you submit, you need to assign page numbers to particular questions (if the graders cannot find your answers easily, they will assign zero credit for the question.)