**A08**

**Due Date:** Tuesday, June 13   
**File to be submitted:** [PQ.java](http://cs.smu.ca/~myoung/csci2341/Assignments/A08/PQ.java)   
**Program file:** [RevisedPQTester.java](http://cs.smu.ca/~myoung/csci2341/Assignments/A08/RevisedPQTester.java)   
**Sample output:** [SampleOutput.html](http://cs.smu.ca/~myoung/csci2341/Assignments/A08/SampleOutput.html)

[SUBMIT](http://cs.smu.ca/~myoung/csci2341/protected/getSubmission.cgi?08)   /   [Check](http://cs.smu.ca/~myoung/csci2341/protected/viewasgn.cgi?A08)

**Revised PriorityQueue Class (Priority Queues)**

**Summary**

Revise the PQ class I created in lecture in two ways:

1. Allow it to grow from its original size up to a maximum size.
2. Allow its constructor to accept a **Comparator**.

Test your revised version using the program I have provided.

**Details**

I have provided a program for you to run, and the (slightly modified) code for **PQ** from today's lecture. Run the program to see what it does. Note that it ends with an error message.

The first revision you should make to **PQ** is to make it expandable. If the array ever gets full, then the items will be copied to a larger array -- typically twice the size of the current array. The maximum size of the array should be 10000 elements.

Note that this is just like the **ResizableArrayBag** I created for you. (See the code for that class.) You only need to copy the pertinent parts of that class to the **PQ** class and make the appropriate modifications.

After you've revised the code, the program I gave you should run to completion with no error messages.

The next revision is to allow the client to provide a suitable **Comparator** in the constructor. For example, the program above prints out the words in lexicographic order -- all the words that start with capital letter come before any of the words that start with small letters. We'd like to get a version that prints out in alphabetical order by specifying the **String.CASE\_INSENSITIVE\_ORDER** when we create the PQ object:

PQ pq3 = new PQ<>(String.CASE\_INSENSITIVE\_ORDER);

In order to do that, you'll need to:

1. Add a **Comparator<T>** instance variable.
2. Make a version of the constructor that accepts a **Comparator<T>** argument.
3. Revise the original constructor to call the revised constructor using the **Comparator**:
4. (t1, t2) -> {
5. return t1.compareTo(t2);

}

1. Revise the calls to compareTo in the methods to be calls to compare using the saved **Comparator**.

When you've made your revisions, un-comment the last four lines of main in the provided program. Run it to confirm that:

1. The original list is still printed in lexicographic order.
2. The second list is printed in alphabetical order.

**Notes**

The default size of the PQ must stay at 10. It's not a solution to the problem to just make the orginal array 10,000 elements long.

The **Comparator** to use when the client doesn't provide one just asks the objects to sort themselves. Thus the base type must still extend **Comparable<? super T>**.

Recall how one constructor calls another. Also recall how lambda expressions can be used in places where the program is expecting an object that implements a one-method interface -- like **Comparator**. We talked about this in the first class this term.

**Grading Outline**

* 50% -- Class performs as required
* 30% -- Class shows good design
* 20% -- Submitted material meets the standard requirements as described in the [rules for submissions](http://cs.smu.ca/~myoung/csci2341/Rules.html) and the [style rules](http://cs.smu.ca/~myoung/csci2341/Style.html).

[SUBMIT](http://cs.smu.ca/~myoung/csci2341/protected/getSubmission.cgi?08)   /   [Check](http://cs.smu.ca/~myoung/csci2341/protected/viewasgn.cgi?A08)