Programming Assignment #1

75 points

Due Date/Time:

M/W: Your project is due on Monday February 19th at 4 P.M.
T/TH: Your project is due on Tuesday February 20th at 8 A.M.
Hardcopy of your Java source code files is also due at that same time.

The Program:

For this assignment, you will write two implementations of a **`Priority Queue`**. For this ADT, removal operations always return the object in the queue of highest priority that has been in the queue the longest. That is, no object of a given priority is ever removed as long as the queue contains one or more object of a higher priority. Within a given priority FIFO order must be preserved.

Your implementations will be:

1. Ordered Array
2. Unordered Array

Both implementations must have identical behavior, and must implement the PriorityQueue interface (provided). The implementations must have two constructors, a default constructor with no arguments that uses the DEFAULT_MAX_CAPACITY constant from the `PriorityQueue` interface, and a constructor that takes a single integer parameter that represents the maximum capacity of the priority queue. The PriorityQueue interface follows:

```
/*  The PriorityQueue ADT may store objects in any order.  However,
    removal of objects from the PQ must follow specific criteria.
    The object of highest priority that has been in the PQ longest
    must be the object returned by the remove() method.  FIFO return
    order must be preserved for objects of identical priority.

    Ranking of objects by priority is determined by the Comparable<E>
    interface.  All objects inserted into the PQ must implement this
    interface.
*/

package data_structures;

import java.util.Iterator;
```

```
public interface PriorityQueue<E extends Comparable <E>> extends Iterable<E> {
    public static final int DEFAULT_MAX_CAPACITY = 1000;

    //  Inserts a new object into the priority queue.  Returns true if
    //  the insertion is successful.  If the PQ is full, the insertion
    //  is aborted, and the method returns false.
    public boolean insert(E object);

    //  Removes the object of highest priority that has been in the
    //  PQ the longest, and returns it.  Returns null if the PQ is empty.
    public E remove();

    //  Deletes all instances of the parameter obj from the PQ if found, and
    //  returns true.  Returns false if no match to the parameter obj is found.
    public boolean delete(E obj);

    //  Returns the object of highest priority that has been in the
    //  PQ the longest, but does NOT remove it.
    //  Returns null if the PQ is empty.
    public E peek();

    //  Returns true if the priority queue contains the specified element
    //  false otherwise.
    public boolean contains(E obj);

    //  Returns the number of objects currently in the PQ.
    public int size();

    //  Returns the PQ to an empty state.
    public void clear();

    //  Returns true if the PQ is empty, otherwise false
    public boolean isEmpty();

    //  Returns true if the PQ is full, otherwise false.  List based
    //  implementations should always return false.
    public boolean isFull();

    //  Returns an iterator of the objects in the PQ, in no particular
    //  order.
    public Iterator<E> iterator();

}
```

Thus, your project will consist of the following files. You must use exactly these filenames.

- `PriorityQueue.java` The ADT interface (provided above)
- `OrderedArrayPriorityQueue.java` The ordered array implementation.
- `UnorderedArrayPriorityQueue.java` The unordered array implementation.

Additional Details:

- Each method must be as efficient as possible. That is, a O(n) is unacceptable if the method could be written with O(log n) complexity. Accordingly, the ordered array implementation **must** use binary search where possible, such as the `contains()`, the `delete(E obj)` method and also to identify the correct insertion point for new additions.
- By convention, a **lower number=higher priority**. If there are five priorities for a given object, 1 .. 5, then 1 is the highest priority, and 5 the lowest priority.
- Your project must consists of only the three files specified (including the provided interface), no additional source code files are permitted. (Do not hand in a copy of PriorityQueue.java, as it is provided to you).
- You may not make any modifications to the PriorityQueue interface provided. I will grade your project with my copy of this file.
- All source code files must have your name and class account number at the beginning of the file.
- All of the above classes must be in a package named 'data_structures'.
- You may import `java.util.Iterator`, and `java.util.NoSuchElementException` only. If you feel that you need to import anything else, let me know. You are expected to write all of the code yourself, and you may not use the Java API for any containers.
- Your code must not print anything.
- Your code should never crash, but must handle any/all error conditions gracefully. i.e. if the user attempts to call the clear() method on an empty PQ, or remove an item from an empty PQ, the program should not crash. Be sure to follow the specifications for all methods.
- You must write generic code according to the interface provided. You may not add any public methods to the implementations, but you may add private ones, if needed.
- Your code may generate unchecked cast warnings when compiled, but it must compile and run correctly on edoras to receive any credit.
- Tester/driver programs will be provided going forward to help you test your code.

Turning in your project:

To submit your project, you must copy both Java source code files into your `handin/prog1` subdirectory. You will submit a printout of these files in class on the due date.***[IMPORTANT NOTE:*** *Do not recreate the data_structures subdirectory in the handin subdirectory--just copy your two files into the* `handin/prog1/` *directory itself.]* Be sure to check the Program Submission Guidelines page.

Cheating Policy

There is a zero tolerance policy on cheating in this course. You are expected to complete all programming assignments on your own. Collaboration with other students in the course is not permitted. You may discuss ideas or solutions in general terms with other students, but you must not exchange code. During the grading process I will examine your code carefully. Anyone caught cheating on a programming assignment (or on an exam) will receive an "F" in the course, and a referral to Judicial Procedures.