

FOUNDATIONS OF COMPUTER SCIENCE
Programming Test - 17-02-2015

Extended Queue

=====

The Queue interface below defines the queue abstract data type that is a collection of items with FIFO (First In First Out) access policy, where the first item inserted (front) is the first extracted.

T is the parametric data type

```
public interface Queue <T> // ADT Queue
{
    /*
     * removes the front item from this queue
     * @return the front item removed from this queue
     * @throws EmptyQueueException if this queue is empty
     */
    T dequeue() throws EmptyQueueException;

    /**
     * inserts the specified item into this queue
     * @param x the specified item to be inserted
     * @throws java.lang.IllegalArgumentException if the specified item is null
     */
    void enqueue(T x) throws IllegalArgumentException;

    /**
     * provides the front item of this queue
     * @return the front item of this queue
     * @throws EmptyQueueException if this queue is empty
     */
    T front() throws EmptyQueueException;

    /**
     * checks if this queue is empty
     * @return true if this queue is empty, false otherwise
     */
    boolean isEmpty();

    /**
     * provides the number of items in this queue
     * @return number of items in this queue
     */
    int size();

    /**
     * returns an array containing the items of this queue in the FIFO
     * sequence with the front of this queue at index zero. The size of the array
     * is equal to the number of items present in this queue and the method
     * never throws exceptions
     * @return an array containing the items of this queue, if the queue is
     *         not empty, otherwise an empty array
     */
    Object[] toArray();
}
```

Write the code of the public class Q<T> implements Queue<T>, whose constructor constructs an empty queue.

Next, complete the following extended queue subclass whose constructor initializes an empty extended queue:

```
public class EQ<T extends Comparable> extends Q<T> // Extended queue
{
    public void enqueue(T x) { super.enqueue(x); } // code provided by teacher
    public Object[] toSortedArray() {...}
    public Object[] getOnlyUniqueItems(){ ... }
}
```

```
    public EQ[] split(int n){ ... }
}
```

where the class methods implement the operations below:

- toSortedArray() returns an array in which the items of the extended queue are sorted in ascending order, according to their natural order.
The length of the array must equals the number of items in this extended queue
 - getOnlyUniqueItems() returns an array that contains only the items that are unique in this extended queue
 - split divides this extended queue (called the source queue) into a specified number n of extended queues (called destination queues) according to the following rules:
 - . each item of the source queue is assigned to one and only one of the destination queues
 - . all destination queues have the same number of items, except for one unit
 - . if item A precedes item B in the source queue and A and B are assigned to the same destination queue, then A precedes B in the destination queue
 - . running the method does not change the source queue
- param n specified number of destination queues
return array of extended queues of specified size n
throws java.lang.IllegalArgumentException if n is less than 1

During test evaluation, the code will be tested with the following commands:

```
$ rm *.class
$ javac TestQ.java
$ java TestQ < fruit.txt
$ javac MainEQ.java
$ java MainEQ < fruit.txt
```

where the TestQ.java, fruit.txt and MainEQ.java files are provided.

The attached OKQ.txt e OKEQ.txt files contain examples of printouts on standard output when running the TestQ and MainEQ by reading the data from the fruit.txt file by redirection of the standard input.

Normally, code that causes errors at compile time is considered insufficient.

The test evaluation will consider the asymptotic time complexity of the methods enqueue, dequeue, toSortedArray and getOnlyUniqueItems.

The test evaluation will assign the following maximum score to each method: dequeue 5, enqueue 5, front 5, toArray 5, toSortedArray 5, getOnlyUniqueItems 2, split 2. Two points may be awarded for the time complexity and one point for the programming style. Total 32 scores.

In the implementation of the Q and EQ classes:

- apply the principle of encapsulation in the strict sense (private instance variables)
- do not add to the Q class methods with public or protected access specifiers, or without access specifiers (you may add private methods, if needed)
- do not use classes from the Java Platform API except those of the java.lang package

At the end of the programming test, the student will deliver in the working directory the files produced: Q.java, EQ.java e MainEQ.java and the files made available by the teacher

```
*****
* The Q.java and EQ.java files need to contain a first comment line with *
* first and family names, ID number, date, number of PC station in ADT *
*****
```

Name _____ IDno. _____ ADT PC _____

I deliver my programming test consisting of the following files : _____
Signature _____

I do not submit the programming test and I withdraw from the exam.

Signature
