Foundations of Computer Science - a.a. 2018-2019
Programming tests dated 3.7.2019

ITERABLE PARAMETRIC LIST

The iterable parametric list abstract data type defines a sequence of elements each of which is associated with a rank such that the first element of the list has rank 0 and the next of the element of rank n has rank n + 1. The following interfaces define the iterable parametric, where E is the parametric type:

```
public interface List <E> extends Container, Iterable
{
   /**
     Inserts the specified element at the last position in this list
     @param element the specified element
     @throws IllegalArgumentException if the specified element is null
   */
   void add(E element);

   /**
     Inserts the specified element at the specified position i this list
     Shifts the element in that position and the following elements to the right
     (adds 1 to the indeces)
     @param index the specified position
     @param element the specified element
     @throws java.lang.IndexOutOfBoundsException if the the specified position
             is outside the valid range [0, number of elements in the list]
     @throws IllegalArgumentException if the specified element is null
   */
   void add(int index, E element);

   /**
      Returns the element at the specified position in this list
      @param index the specified position
      @return the element at the specified position
     @throws java.lang.IndexOutOfBoundsException if the the specified position
             is outside the valid range [0, number of elements in the list - 1]
   */
   E get(int index);

   /**
      Returns the position of the specified element if it exists in this list
      @param element the specified element
      @return the position of the first occurrence of the specified element in
              this list or -1 if the element is not present in this list
   */
   int rankOf(E element);

   /**
      returns an array containing the elements of this list in the sequence in
      which they are in this list
      @return an array containing the elements of this list in the sequence in
              which they are in this list
   */
   Object[] toArray();
} //*** END OF LIST INTERFACE
public interface Container
{
   /**
      checks if this container is empty
      @return true if this container is empty, otherwise false
   */
   boolean isEmpty();

   /**
```

```
      makes this container empty
   */
   void makeEmpty();

   /**
      provides the number of data items in this container
      @return the number of data items in this container
   */
   int size();
} // *** END OF CONTAINER INTERFACE
public interface Iterable<T>
{
   /**
      Returns an iterator over the elements in a list in proper sequence
      and positioned at the beginning of the list
      @return an iterator over the elements in a list in proper sequence
   */
   Iterator<T> iterator();
} // *** END OF ITERABLE INTERFACE
public interface Iterator<T>
{
   /**
      Checks if there are more elements in this iterable sequence
      @return true if the  iterator has more elements when traversing the list
   */
   boolean hasNext();

   /**
      Returns the next element in the list and advances the cursor position
      @return the next element in the list
      @throws java.util.NoSuchElementException if the iteration has no more
              elements
   */
   T next();

   /**
      Removes from the list the last element that was returned by next(). This
      call can only be made once per call to next
      @throws IllegalStateException if next has not been called, or remove or
              add have been called after the last call to next
   */
   void remove();
} // *** END OF ITERATOR INTERFACE
```

Implement the **`public class L<E> implements List<E>`** class, in which the constructor constructs an empty list.

Then, write the private section and complete the code of the following extended list class

```
public class LE<E extends Comparable> extends L<E>
{
  public int rankOfSortedList(E element) {...}
  public List<E> toSortedList()    {...}
}
```

where the specified methods perform the following processing:

- rankOfSortedList(E element) returns the rank of the first occurrence of the specified element in the sorted list or -1 if the element is not present in this listre
- toSortedList() returns a list containing the same elements as this list arranged in ascending order according to their natural order

During the correction the code will be tested with the commands:

```
$rm *.class
$javac LETester.java
$java LETester numbers.txt
```

where numbers.txt is the attached file and the LETester.java must be coded according to the instructions reported in the classroom programming 11.

The attached OK.txt file reports the result to standard output when the LETester class performs by acquiring data from the numbers.txt file. Generally, the cod that gives rise to compilation errors are considered insufficient. For the purposes of evaluating the programming test, the following maximum scores will be assigned:
L: add 4, get 2, iterator 4, rankOf 4, hasNext 2, next 4, remove 4, TOT 24 points
LE: rankOfSortedList 2, toSortedList 2, TOT 4 points
Two points can be assigned for the style and two points for the time complexity of the toSortedList and rankOfSortedList methods (TOT 24 + 4 + 2 + 2 = 32).
In the realization of classes L and LE it is not allowed:
- to add non-private variables and / or methods;
- to use Java API classes with the exception of java.util.NoSuchElementException and the classes of the java.lang package.
At the end of the test the students will leave the files produced by them and the files provided by the teacher in the working directory. The files submitted by the students must contain as the first line a comment with student's name and surname, serial number, date and number of the work station.

FOUNDATIONS OF COMPUTER SCIENCE
Name & Surname___________________________Rollno. _________Post. ADT_______
I submit my programming tests consisting of the files:_____________________
Signature ____________________________
I do not submit my programming test and I withdraw from the exam.
Signature ____________________________