FOUNDATIONS OF COMPUTER SCIENCE - a.a. 2019-2020
Programming Test dated 29-01-2020

AIR FORCE CODE

The sorted set abstract data type is defined by the following interfaces Set, Iterable, and Iterator:

```
//-----------------------------------START OF INTERFACE DEFINITION
public interface Set<E extends Comparable> extends Iterable<E>
{
   /**
      Inserts the specified element into this set if it is not null and is not
      already present
      @param e the specified element
      @return false if this set already contained the specified element
   */
   boolean add(E e);
   /**
      Checks if the specified element is present in this set
      @param element the specified element
      @return true if this set contains the specified element
   */
   boolean contains(E element);
   /**
      Checks if this set is empty
      @return true if this set is empty
   */
   boolean isEmpty();
   /**
      Returns the number of elements of this set
      @return the number of items in this set
   */
   int size();
   /**
      Copies the elements of this set in ascending order according to their
      natural order to the specified array
      @param a the specified array
      @return number of elements of this set copied to the specified array
      @throws java.lang.ArrayIndexOutOfBoundsException if the specified array
              size is less than the set size
*/
   int toSortedArray(E[] a) throws ArrayIndexOutOfBoundsException;
} // *** END OF SET INTERFACE
public interface Iterable<T> // *** START OF ITERABLE INTERFACE
{
   /**
      Returns an iterator over the sequence
      @return an iterator over the sequence
   */
   Iterator<T> iterator();
} // *** END OF ITERABLE INTERFACE
public interface Iterator<T> // *** START OF ITERATOR INTERFACE
{
   /**
      Checks if there are more elements
      @return true if the iteration has subsequent elements
   */
   boolean hasNext();
   /**
      Inspects the element following the position of this iterator
      @return the next element in the iteration
      @throws java.util.NoSuchElementException if the iteration has no
              subsequent elements
   */
T    next() throws java.util.NoSuchElementException;
```

```
   /**
      Removes the last element inspected by this iterator
      @throws java.lang.IllegalStateException if the next method has not been
              called before or the remove method has been called after the last
              call to the next method
   */
   void remove() throws java.lang.IllegalStateException;
} // *** END OF ITERATOR INTERFACE
//-------------------------------------------------END OF INTERFACE DEFINITION
```

Complete the S class below where the min method is made available by the teacher.

```
public class S<E extends Comparable> implements Set<E> {
   public S() {…}        // constructs an empty sorted set
   public S(E[] a) {...} // constructs a sorted set containing the elements of
                         // the specified array
   public S(Set s) {...} // constructs a sorted set containing the elements of
                         // of the specified set
   // search for the minimum element
   public E min() throws java.util.NoSuchElementException {
      if (isEmpty()) // precondition – the set is empty
         throw new java.util.NoSuchElementException();
      // generate an iterator to inspect the set elements
      Iterator<E> iter = iterator();
      // first guess for the minimum
      E min = iter.next();
      // Inspect the elements and selecting the minimum element
      while (iter.hasNext()) {
         E tmp = iter.next();
         if (tmp.compareTo(min) < 0) min = tmp;
      }
   return min;
   }
}
```

with the specified constructors

In the evaluation, those solutions where the contains method has an asymptotic time complexity better than O(n) in the average and worst case will be considered excellent.

Complete, then, the code of the extended set class below that implements the java.lang.Comparable interface

```
public class SE<E extends Comparable<E>> extends S<E> implements
Comparable<SE<E>>{
...
   public E min() throws java.util.NoSuchElementException {...}
   public SE<E> subset(Iterator<E> iter){...}
}
```

where the constructors are the same as in the S class.

- the min method RECURSIVELY overwrites the superclass method of the same name.
- the subset method returns an extended set that contains the elements that can be inspected with the specified iterator or an empty set if the specified iterator is null

The natural order of the class is defined according to the size of the sets.

Let s1 and S2 be two sets:

- if s1.size() < s2.size(), then s1 precedes s2
- else if s1.size() > s2.size(), then s1 follows s2.

If the two sets have the same size, then compare the sorted elements of the two sets in pairs (one pair element for each set) starting with the largest elements.

Examples:

i) Let be s1 = {A, B} and s2 = {A, B, C}. Set s1 precedes s2 because set s1 has a lower size than set

ii) Let be s1 = {A, B, C} and s2 = {B, C, D}. Set s1 precedes s2 because the maximum element in s2 is greater than the maximum element in set s2

iii) Let be s1 s1 = {K, L, N} and s2 = {J, M, N}. Set s1 precedes s2 because the maximum elements in the two sets are the same and equal to N, but in the following comparison element L precedes

element M.
During correction, the code will be tested with the following commands:

```
$rm *.class
$javac SETester.java
$java SETester
```

where the SETester.java class is specified in the in-class programming 12.
The attached OK.txt file reports a possible result of the $ java SETester command.
Generally, the documents that give rise to compilation errors are considered insufficient.
During the evaluation, the following maximum scores will be assigned to the methods:
S - constructors 2, add 4, contains 4, iterator 2, hasNext 2, next 2, remove 3, toSortedArray 4 (Tot = 23)
SE - compareTo 2, min 2 if recursive, subset 2 (Tot = 6). One point can be assigned for the style and up to two points for the time complexity.
In the realization of classes S and SE it is not allowed to:
- use non-private instance variables and / or add non-private methods;
- use classes of the Java Platform API except those of the java.lang package and of the class java.util.NoSuchElementException.
At the end of the programming test the student will leave the files produced by her / him and the files provided by the teacher in the working directory. The files produced by the students must contain as the first line a comment with name and surname, serial number, date and number of the working station.

```
                FOUNDATIONS OF COMPUTER SCIENCE - a.a. 2019-20

Name & surname_____________________________Rollno. _______________ADT station_____

I submit my programming tests consisting of the following files:
[] S.java [] SE.java [] other_____________________________________

Signature _____________________________________________________

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

Signature _____________________________________________________
```