PRG – ETSInf – THEORY – Academic year 2017-18
Second mid term exam – 4 June 2018 – Duration: 2 hours

NOTE: This exam is evaluated up to 10 points, but its weight in the final grade of PRG **3 points**.

1. 2.5 points **To be done:** implement an static method that, given an array of `int`, copies its elements, one per line, in a text file named `"ArrayElements.txt"`. Thus, if the array is {`5, 2, 8, 4`}, in the file the values will appear one per line in the following order: `5`, `2`, `8` and `4`. The method has to return, as a result, the of object of the class `File` created for opening the file.

   Exceptions of the class `FileNotFoundException` must be caught, and a message showing the error must be printed on screen in such case.

Solution:

```
public static File fromArrayToTextFile(int[] a) {
    File res = new File("ArrayElements.txt");
    PrintWriter pw = null;
    try {
        pw = new PrintWriter(res);
        for (int i = 0; i < a.length; i++) {
            pw.println(a[i]);
        }
    } catch (FileNotFoundException e) {
        System.err.println("Error opening " + res);
    } finally {
        if (pw != null) { pw.close(); }
    }
    return res;
}
```

2. 2.5 points **To be done:** implement an static method such that given an `String s` converts it into a linked sequence of characters. Characters must appear in the linked sequence in the same order they are stored in the `String s`. If `String s` is empty or `s` references to `null`, the resulting linked sequence should be `null` also, otherwise the method must return the reference to the first node in the linked sequence. For this problem, you can assume that it is available the class `NodeChar`, analogous to the class `NodeInt` you already know which is part of the `linear` package. The unique difference is that the internal attribute for storing data is a `char` instead of an `int`.

   For instance, given the `String s = "Examen"`, the sequence will be:

   → 'E' → 'x' → 'a' → 'm' → 'e' → 'n'

Solution:

```
    public static NodeChar fromStringToSeq(String s) {
        NodeChar res = null;
        if(s != null) {
           int i = s.length() - 1;
           while (i >= 0) {
               char c = s.charAt(i--);
               res = new NodeChar(c, res);
           }
        }
        return res;
    }
```

3. 2.5 points **To be done:** add a new method in the class `ListIntLinked` with the profile:

```
public void append(int x)
```

such that given an integer `x`, inserts it after the cursor and updating the cursor to reference the new inserted element. If the the cursor was after the last element, i.e. it is `null`, this method must throw an exception of the class `NoSuchElementException` with the message `"Cursor at end"`.

For instance, if the method is invoked this way `l.append(5)` being `l` a list (where the element between brackets is the cursor or interest point) `1 4 [7] 8 3 4`, then `l` must become `1 4 7 [5] 8 3 4`.

IMPORTANT: In the new method you can only use the attributes of the class and local variables used as temporary references. It is strictly forbidden to use other methods of the class `ListIntLinked`.

Solution:

```
public void append(int x) {
    if (cursor != null) {
        cursor.setNext( new NodeInt( cursor, x, cursor.getNext() ) );
        cursor = cursor.getNext();
        if (cursor.getNext() != null) cursor.getNext().setPrevious(cursor);
        size++;
    } else { throw new NoSuchElementException("Cursor at end"); }
}
```

4. 2.5 points **To be done:** implement an static method `merge` such that given two queues `QueueIntLinked q1` and `q2`, returns a new queue where both queues have been merged, the values appear in the same order they are in their respective queue, but alternated, one of one queue and another of the other queue. The elements remaining in the largest queue will be added at the end of the new queue. Queues `q1` and `q2` must remain with its initial state after the process is finished.

For instance, if `q1` is $\leftarrow \overline{3\ 6\ 20\ 1\ -3\ 4\ -5} \leftarrow$ and `q2` is $\leftarrow \overline{10\ 9\ 8} \leftarrow$ the method returns the queue $\leftarrow \overline{3\ 10\ 6\ 9\ 20\ 8\ 1\ -3\ 4\ -5} \leftarrow$.

IMPORTANT: It is supposed that the method is implemented outside the class `QueueIntLinked`, so, only the public methods of such class can be used.

Solution:

```
public static QueueIntLinked merge(QueueIntLinked q1, QueueIntLinked q2) {
    QueueIntLinked res = new QueueIntLinked();
    int i = Math.min(q1.size(), q2.size());
    for (int j = 0; j < i; j++) {
        res.add(q1.element()); q1.add(q1.remove());
        res.add(q2.element()); q2.add(q2.remove());
    }
    while (i < q1.size()) { res.add(q1.element()); q1.add(q1.remove()); i++; }
    while (i < q2.size()) { res.add(q2.element()); q2.add(q2.remove()); i++; }
    return res;
}
```

ANNEX

Methods of the class `QueueIntLinked`, attributes of the class `ListIntLinked` and class `NodeChar`.

```
public class QueueIntLinked {
    ...
    public QueueIntLinked() {...}
    public void add(int x) {...}
    public int remove() {...}
    public int element() {...}
    public int size() {...}
    public boolean empty() {...}
    public boolean equals(Object o) {...}
    public String toString() {...}
}
```

```
public class ListIntLinked {
    private int size;
    private NodeInt first, cursor, last;
}

class NodeChar {
    char data;
    NodeChar next;
    NodeChar(char c) { data = c;  next = null; }
    NodeChar(char c, NodeChar n) {
        data = c; next = n;
    }
}
```