PRG - ETSInf. TEORIA. Curs 2017-18. Parcial 2.
4 de juny de 2018. Duració: 2 hores.

Nota: L'examen s'avalua sobre 10 punts, però el seu pes específic en la nota final de PRG és de **3 punts**.

1. 2.5 punts **Es demana:** implementar un mètode estàtic tal que, donat un array d'`int`, copie els seus elements, un per línia, en un fitxer de text de nom `"ArrayElements.txt"`. Així, si l'array és {`5, 2, 8, 4`}, al fitxer s'emmagatzemaran, un per línia, els valors `5`, `2`, `8` i `4`. El mètode ha de retornar com a resultat l'objecte `File` creat.

   Haurà de tractar la possible excepció `FileNotFoundException`, mostrant un missatge d'error en cas que aquesta es produisca.

   Solució:

```
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 en obrir " + res);
    } finally {
        if (pw != null) { pw.close(); }
    }
    return res;
}
```

2. 2.5 punts **Es demana:** implementar un mètode estàtic tal que, donat un `String s`, el convertisca en una seqüència enllaçada de caràcters on el primer element de la seqüència ha de ser el primer caràcter de l'`String s`. Si l'`String s` està buit o és `null`, la seqüència resultant serà també `null`; si no, retornarà el primer node de la seqüència. Per a això, es suposa accessible una classe `NodeChar` idèntica a la clase `NodeInt` utilitzada en el paquet `linear`, excepte en el tipus de la dada.

   Per exemple, donat l'`String s = "Examen"`, la seqüència serà: → `'E'` → `'x'` → `'a'` → `'m'` → `'e'` → `'n'`

   Solució:

```
    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 punts] **Es demana:** Afegir un mètode a la classe `ListPIIntLinked` amb el perfil:

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

tal que, donat un enter `x`, l'inserisca en la posició següent al punt d'interès, avançant aquest al nou element introduït. Si el cursor ja està al final en el moment d'invocar al mètode, aquest ha de llançar l'excepció `NoSuchElementException` amb el missatge `"Cursor al final"`.

Per exemple, si s'invoca al mètode `l.append(5)` sent `l` la llista (on l'element distingit és el marcat entre claudàtors) `1 4 [7] 8 3 4` , aleshores `l` queda de la forma `1 4 7 [5] 8 3 4` .

IMPORTANT: En la solució només es permet accedir als atributs de la classe, quedant prohibit l'accés als seus mètodes.

Solució:

```
public void append(int x) {
    if (pI != null) {
        prevPI = pI;
        pI.next = new NodeInt(x, pI.next);
        pI = pI.next;
        size++;
    } else { throw new NoSuchElementException("Cursor al final"); }
}
```

4. [2.5 punts] **Es demana:** implementar un mètode estàtic `fusion` que, donades dues cues `QueueIntLinked` `q1` i `q2`, retorne una nova cua en la que s'hagen fusionat els elements de `q1` i `q2` de manera que apareguen per ordre de cua, però alternant-se un element d'una i altra cua. Els elements sobrants d'alguna de les cues apareixeran al final de la cua resultat. Les cues `q1` i `q2` han de quedar en el seu estat original.

Per exemple, si `q1` és $\leftarrow \overline{\texttt{3 6 20 1 -3 4 -5}} \leftarrow$ i `q2` és $\leftarrow \overline{\texttt{10 9 8}} \leftarrow$ el mètode retorna la cua $\leftarrow \overline{\texttt{3 10 6 9 20 8 1 -3 4 -5}} \leftarrow$.

IMPORTANT: Es suposarà que el mètode s'implementa en una classe distinta de `QueueIntLinked`, per tant, només es podran usar els mètodes públics de la classe.

Solució:

```
public static QueueIntLinked fusion(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

Mètodes de la classe `QueueIntLinked`, atributs de la classe `ListPIIntLinked` i classe `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 ListPIIntLinked {
    private int size;
    private NodeInt first, pI, prevPI;
}

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