PRG - ETSInf. TEORIA. Curs 2018-19. Recuperació Parcial 2.
11 de juny de 2019. Duració: 2 hores.

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

1. 3.0 punts Tenim un fitxer de text facilitat per la ATP amb informació dels tenistes professionals, on en cada línia figura el cognom, l'edat, els punts aconseguits i el nombre de campionats als que ha participat el tenista. El fitxer pot contindre errors perquè:
   – La informació no està completa (pot faltar alguna dada) o pot haver més dades de les esperades.
   – En lloc de l'edat, o dels punts, o del nombre de campionats poden aparèixer seqüències de caràcters no vàlides o valors negatius.

   Es demana: implementar un mètode que rep com a paràmetres un `String fileIn`, amb el nom del fitxer de text a analitzar, i un altre `String fileOut`, amb el nom del fitxer de text resultat del procés. El mètode ha de llegir les dades del fitxer `fileIn` i, per a cada línia en la qual hi haja un error, escriure un missatge d'error en el fitxer d'eixida indicant l'error detectat. El mètode ha de propagar les excepcions que es puguen produir de la classe `FileNotFoundException` si ha hagut cap problema en obrir els fitxers. Recordeu que aquest tipus d'excepció és comprovada.

   Per exemple, si el fitxer d'entrada fos:

```
Djokovic    31  12115   17
Nadal       -32 7945    16
Federer     37  5770    17
Thiem       25  4845    24  4
Zverev      22  4745
Nishikori   29  3860    23.5
Tsitsipas   20  3790    28
Anderson    32  3755.3  17
```

   El de sortida hauria de contindre:

```
Error línia 2: Valor negatiu.
Error línia 4: Incorrecte el nombre de dades.
Error línia 5: Incorrecte el nombre de dades.
Error línia 6: Format d'enter no vàlid.
Error línia 8: Format d'enter no vàlid.
```

   Si les columnes en cada línia poden estar separades per blancs o tabuladors, es pot fer ús de la següent instrucció per separar les dades que apareixen en una línia:
   `String[] tokens = linia.split("([ \t])+");`
   Per exemple, si la variable `linia` (de tipus `String`) conté `"Djokovic 31 12115 17"`, l'array de `String` resultant ha de ser `["Djokovic", "31", "12115", "17"]`.
   Es pot utilitzar el mètode `parseInt` de la classe `Integer` per convertir una `String` en l'enter que representa. Fixeu-vos que aquest mètode pot llançar una excepció de la classe `NumberFormatException` si el format no és l'apropiat.

Solució:

```
public static void filtraErrors(String fileIn, String fileOut) throws FileNotFoundException  {
    File fE = new File(fileIn); File fS = new File(fileOut);
    Scanner entrada = new Scanner(fE); PrintWriter eixida = new PrintWriter(fS);
    int cont = 0;
    while (entrada.hasNext()) {
        try {
            String linia = entrada.nextLine();  cont++;
            String[] tokens = linia.split("([ \t])+");
            if (tokens.length != 4) {
                eixida.println("Error línia " + cont + ": " + "Incorrecte el nombre de dades.");
            } else {
```

```
                int edat = Integer.parseInt(tokens[1]);
                int punts  = Integer.parseInt(tokens[2]);
                int campionats  = Integer.parseInt(tokens[3]);
                if (edat < 0 || punts < 0 || campionats < 0) {
                    eixida.println("Error línia " + cont + ": " + "Valor negatiu.");
                }
            }
        } catch (NumberFormatException e) {
            eixida.println("Error línia " + cont + ": " + "Format d'enter no vàlid.");
        }
    }
    entrada.close(); eixida.close();
}
```

2. 3.5 punts **Es demana:** implementar un mètode d'instància en la classe `ListPIIntLinked` amb el perfil i la precondició següents:

```
/** Precondició: la llista this conté dades emmagatzemades en ordre creixent */
public void removeGreaterThan(int e)
```

Donat un enter `e`, el mètode modifica la llista esborrant de la mateixa tots els elements majors que `e`. Al final de l'execució, el `PI` ha d'estar a l'inici de la llista.

Per exemple, si la llista `l` inicialment és `[10] 12 14 15`, i l'enter `e` és `12`, aleshores, la llista, després d'executar el mètode, quedarà com `[10] 12`. Considerant la mateixa llista `l`, i l'enter `e` = `9`, aleshores, la llista es quedarà buida després d'executar el mètode.

REQUISIT: Utilitzar solament els atributs de la classe i referències auxiliars a `NodeInt`, no als mètodes de la classe.

Solució:

```
public void removeGreaterThan(int e) {
    if (first != null && first.data > e) {
        first = null;
        size = 0;
    }
    else {
        int cont = 0;
        NodeInt aux = first, prev = null;
        while (aux != null && aux.data <= e) {
            prev = aux;
            aux = aux.next;
            cont++;
        }
        if (aux != null) {
            prev.next = null;
            size = cont;
        }
    }
    prevPI = null;
    pI = first;
}
```

3. 3.5 punts **Es demana:** implementar un mètode estàtic amb perfil

```
public static void moureAlFinal(QueueIntLinked q, int x)
```

que cerque la primera ocurrència de l'element `x` dins de la cua `q`, i,

- en cas de trobar-lo, el lleve d'on està i el pose com el darrer element de la cua.
- En cas contrari, deixa la cua intacta.

REQUISIT: Es suposarà que el mètode s'implementa en una classe diferent a `QueueIntLinked`, aleshores, sols es podrà fer ús dels mètodes públics de la classe `QueueIntLinked`.

Solució:

```
public static void moureAlFinal(QueueIntLinked q, int x) {
     int n = q.size(), i = 0;
     while (i < n && q.element() != x) {
          q.add(q.remove());
          i++;
     }
     if (i < n) {
          q.remove();
          for(int j = i + 1; j < n; j++) {
               q.add(q.remove());
          }
          q.add(x);
     }
}
```

ANNEX

Atributs de la classe `ListPIIntLinked` i mètodes de la classe `QueueIntLinked`.

```
public class ListPIIntLinked {
    private NodeInt first;
    private NodeInt pI;
    private NodeInt prevPI;
    private int size;
    ...
}

public class QueueIntLinked {
    ...
    public QueueIntLinked() { ... }
    public void add(int x) { ... }
    public int remove() { ... }
    public int element() { ... }
    public boolean empty() { ... }
    public int size() { ... }
}
```