PRG - ETSInf. TEORIA. Curs 2016-17. Recuperació Parcial 2.
26 de juny de 2017. 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 disposa d'un fitxer de text tal que les seues línies contenen, separats per espais en blanc, tokens que són representacions vàlides de nombres enters i tokens que no ho són. **Es demana:** implementar un mètode estàtic que, donat un `String` amb la ruta i el nom del fitxer, retorne la suma de tots els enters que continga el fitxer. El mètode ha de:

   - Capturar l'excepció `FileNotFoundException` que pot ocórrer en intentar obrir el fitxer, mostrant un missatge per pantalla.
   - Llegir (mentre en queden) les línies del mateix, usant el mètode `nextLine()` de la classe `Scanner`.
   - Dividir cada línia en els tokens corresponents, usant el mètode `split(expReg)` de la classe `String`. Mitjançant aquest mètode, s'obté un array d'`String`, tal que cada component del mateix és un dels tokens que apareixen a l'`String` sobre el qual s'aplica, utilitzant com a separadors vàlids els descrits en `expReg`. En aquest exercici es farà servir com a separador l'espai en blanc, és a dir, `split(" ")`.
   - Convertir cada token o `String` de l'array anterior al valor enter corresponent, usant el mètode `Integer.parseInt(String)`.
   - Si el token és una representació vàlida d'un nombre enter, aquest nombre es sumarà al resultat a retornar.
   - Si el token no és una representació vàlida d'un nombre enter, aleshores s'ha de capturar l'excepció `NumberFormatException` que es genera en tal cas, mostrant per pantalla el valor del token que l'ha generada. Aquesta circumstància no ha d'impedir que continue la conversió de tokens a enters ni la lectura del fitxer.

Solució:

```
public static int lligISuma(String nomFitxer) {
    int suma = 0;
    Scanner s = null;
    try {
        s = new Scanner(new File(nomFitxer));
        while (s.hasNextLine()) {
            String[] linia = s.nextLine().trim().split(" ");
            for (int i = 0; i < linia.length; i++) {
                try {
                    suma += Integer.parseInt(linia[i]);
                } catch (NumberFormatException e) {
                    System.out.println(linia[i]);
                }
            }
        }
    } catch (FileNotFoundException e) {
        System.out.println("No s'ha trobat el fitxer");
    } finally {
        if (s != null) { s.close(); }
    }
    return suma;
}
```

2. 2.5 punts Es disposa de la següent classe que implementa una seqüència enllaçada amb nodes de tipus `NodeInt`:

```
public class SecEnla {
    private NodeInt sec;  // referència al primer element de la seqüència
    . . .
}
```

Es demana: implementar un mètode d'instància a la classe `SecEnla` que, donat un valor enter `x`, trobe la primera ocurrència d'aquest valor en la seqüència i l'avance una posició. Si `x` es troba en el primer node de la seqüència, és a dir, no es pot avançar una posició, passarà a l'últim i viceversa. En cas que `x` no es trobe, no farà res. Se suposa, per precondició, que la seqüència té, almenys, 2 nodes.

Per exemple, si la seqüència conté `[3, 7, 2, 8, 5]` i executem la crida al mètode `avançar(8)`, la seqüència quedarà `[3, 7, 8, 2, 5]`. Si, a continuació, executem `avançar(3)`, la seqüència quedarà `[5, 7, 8, 2, 3]`.

Solució:

```
/** Precondició: la seqüència té, almenys, 2 nodes */
public void avançar(int x) {
    NodeInt ant = null, aux = sec;
    while (aux != null && aux.dada != x) {
        ant = aux;
        aux = aux.seguent;
    }
    if (aux != null) {
        if (ant == null) {
            ant = aux;
            while (ant.seguent != null) {
                ant = ant.seguent;
            }
        }
        aux.dada = ant.dada;
        ant.dada = x;
    }
}
```

3. 2.5 punts **Es demana:** implementar un mètode d'instància a la classe `LlistaPIIntEnla` amb perfil

```
public void inserirAl(int x, boolean inici)
```

que, donat un enter `x`, l'inserisca a l'inici de la llista si el paràmetre `inici` és `true` i, en cas contrari, l'inserisca al final. El punt d'interès ha de quedar sobre l'element inserit.

Nota: Només es permet accedir als atributs de la classe, quedant prohibit l'accés als seus mètodes.

Solució:

```
public void inserirAl(int x, boolean inici) {
    NodeInt nou = new NodeInt(x);
    if (primer == null) { primer = nou; }
    else if (inici) {
        nou.seguent = primer;
        primer = nou;
        antPI = null;
    }
    else {
        NodeInt aux = primer;
        while (aux.seguent != null) { aux = aux.seguent; }
        aux.seguent = nou;
        antPI = aux;
    }
    pI = nou;
    talla++;
}
```

4. [2.5 punts] **Es demana:** implementar un mètode estàtic fora de la classe `CuaIntEnla` tal que, donada una `CuaIntEnla` amb valors en [0..9], retorne el nombre enter format pels dígits de la mateixa. La cua ha de quedar en el seu estat original. Per exemple, si la cua és ← `5 1 4 7` ←, l'enter resultant serà `5147`. Fixa't que aquest enter pot obtenir-se com segueix: `(((((5 * 10) + 1) * 10) + 4) * 10) + 7`.

Solució:

```
/** Versió 1: sense estructures auxiliars */
public static int fromCuaToInt(CuaIntEnla q) {
    int res = 0;
    for (int i = 0; i < q.talla(); i++) {
        int x = q.desencuar();
        res = res * 10 + x;
        q.encuar(x);
    }
    return res;
}

/** Versió 2: usant una cua auxiliar */
public static int fromCuaToInt(CuaIntEnla q) {
    int res = 0;
    CuaIntEnla qAux = new CuaIntEnla();
    while (!q.esBuida()) {
        int x = q.desencuar();
        res = res * 10 + x;
        qAux.encuar(x);
    }
    while (!qAux.esBuida()) {
        int x = qAux.desencuar();
        q.encuar(x);
    }
    return res;
}
```