Segon Parcial de PRG - ETSInf
Data: 7 de juny de 2012. Duració: 2 hores i 30 minuts

1. 1.5 punts El següent mètode demana a l'usuari un valor enter, que és llegit com una String, `val`, que després és transformada, mitjançant l'operació `Integer.parseInt(val)` en el valor enter que aquesta conté.

```
public static int llegirInt() {
    Scanner tcl = new Scanner(System.in);
    System.out.print("Dona'm valor: ");
    String val = tcl.nextLine().trim();
    int ret = Integer.parseInt(val);
    return ret;
}
```

El problema és que si lo que conté la `String` no és un enter correctament escrit (per exemple perque continga alguna lletra) es produirà una excepció `NumberFormatException`, detenint-se a continuació el procés de lectura.

Es demana: Modificar el mètode `llegirInt()` per a que, cas de produir-se una excepció `NumberFormatException` durant la transformació de la `String` a `int`, torne a demanar les vegades que calga el valor fins que aquest siga un nombre correcte.

Solució:

```
public static int llegirInt() {
    Scanner tcl = new Scanner(System.in);
    int ret=0;
    boolean eixir = false;
    while (!eixir)
        try {
            System.out.print("Dona'm valor: ");
            String val = tcl.nextLine().trim();
            ret = Integer.parseInt(val);
            eixir = true;
        } catch (NumberFormatException e) {System.out.println("ERROR");}
    return ret;
}
```

2. 1.5 punts Donat cert fitxer de text anomenat en el sistema "origen.txt", així com cert valor enter `llindar`, **es demana** dissenyar un mètode que escriga en el sistema un nou fitxer "desti.txt"que continga exclusivament i amb el mateix orde d'aparició, totes les línies de text del fitxer original que tinguen una llargària, en nombre de caràcters, major o igual que `llindar`.

Si el fitxer origen estiguera buit s'haurà de generar un de nou també buit.

Solució:

```
public static void nouFitxer(int llindar) throws IOException {
    String sin = "origen.txt"; String sout = "desti.txt";

    Scanner fin = new Scanner(new File(sin));
    PrintWriter fout = new PrintWriter(new File(sout));
    while (fin.hasNextLine()) {
        String aux = fin.nextLine();
        if (aux.length() >= llindar) fout.println(aux);
    }
    fin.close(); fout.close();
}
```

3. 3.0 punts Es desitja afegir a la classe `LlistaPIIntEnla` un parell de nous mètodes, **interns**, que permitisquen obtindre una llista a partir dels elements d'un array i viceversa. **Aquestes noves operacions hauran de fer-se sense fer servir les operacions públiques ja existents en la classe**.

   Mitjançant aquests mètodes es podrà ordenar els elements d'una Llista amb punt d'interès transformant-la en un array, ordenant-lo i, fet això, obtenint una nova Llista amb punt d'interès usant l'array. Per a això, **es demana** resoldre els següents apartats:

   1. Mètode `toArray()`, que haurà de tornar un array que continga exactament, i en el mateix orde original, els elements de la `LlistaPIIntEnla`.
   2. Constructor `LlistaPIIntEnla(int[] a)`, que donat un array d'enters construïsca una `LlistaPIIntEnla` amb els elements que es troben en l'array, respectant el seu orde.
   3. Suposeu la classe `LlistaPIIntEnla` completada amb els dos mètodes anteriors. Suposeu, a més, que en una classe `ProvaLlista` es disposa ja fet d'un mètode d'ordenació ràpida d'arrays amb la capçalera següent:

      `public static void ordFussio(int[] a, int ini, int fi)`

      Escriu les instruccions necessàries per a ordenar, en aquesta classe `ProvaLlista`, certa `LlistaPIIntEnla` emmagatzemada en una variable anomenada `laLlista`.

Solució:

```
public int[] toArray() {
    int[] aux = new int[talla];
    int k; NodeInt p;
    for (p=primer, k=0; p!=null; p=p.seguent, k++) aux[k] = p.dada;
    return aux;
}

public LlistaPIIntEnla(int[] a) {
    this.primer = null;
    for (int k=a.length-1; k>=0; k--) primer = new NodeInt(a[k],primer);
    this.talla = a.length;
```

```
    this.PI = primer; this.antPI = null;     // PI al començament
}

// seqüència d'instruccions per a ordenar laLlista:
  int[] copia = laLlista.toArray();
  ordFussio(copia,0,copia.length-1);
  laLlista = new LlistaPIIntEnla(copia);
```

4. 1.5 punts Tenint en compte la definició de `PilaIntEnla` vista en classe i accedint a la seva representació interna, **es demana** escriure un mètode:

   `public void cimBase()`

   que intercanviï el valor de l'element al cim de la pila amb el de la base de la mateixa (el més antic). Precondició de la operació serà que la pila no estiga buida.

Solució:

```
/** PRECONDICIÓ: !esBuida() */
public void cimBase() {
    NodoInt aux = cim;
    int e = aux.dada;
    while(aux.seguent!=null)
        aux = aux.seguent;
    cim.dada = aux.dada;
    aux.dada = e;
}
```

5. 2.5 punts Donades dues Llistes amb punt d'interès `LlistaPIIntEnla la` i `lb`, que es troben ordenades ascendentment i que no contenen elements duplicats, **es demana** fer un mètode anomenat `intersecció` que torne una nova `LlistaPIIntEnla` que continga tan sols els elements que es troben tant en `la` com en `lb`. Aquesta nova llista tampoc haurà de contindre elements duplicats.

   Si no existeixen elements en comú la nova `LlistaPIIntEnla` estarà buida.

   Exemple: Si inicialment es té que:

   ```
   la = 2 4 6 7 9 11 33 45 67 112 129 310 516 555 610
   lb = 8 11 22 33 44 45 46 112 113
   ```

   Com a efecte de l'execució del mètode que es demana, la llista resultant serà:

   ```
   11 33 45 112
   ```

Solució:

```
/** PRECONDICIÓ: la i lb ordenades ascendentment, sense repetits */
public static LlistaPIIntEnla interseccio (LlistaPIIntEnla la,
                                           LlistaPIIntEnla lb) {
    LlistaPIIntEnla lc = new LlistaPIIntEnla();
    la.inici(); lb.inici();
    while (!la.fi() && !lb.fi()) {
        int a = la.recuperar(); int b = lb.recuperar();
        if (a<b) la.seguent();
        else if (b<a) lb.seguent();
        else {
            lc.inserir(a);
            la.seguent(); lb.seguent();
        }
    }
    return lc;
}
```