PRG - ETSInf. TEORIA. Curs 2013-14. Parcial 2.
10 de juny de 2014. Duració: 2 hores.

1. [2 punts] Donat un fitxer de text, es vol escriure el seu contingut línia a línia en l'eixida estàndard, però transformant els seus caràcters alfabètics a majúscules.

   Es demana escriure un mètode, la capçalera del qual comença així:

   ```
   public static void escriuAmbMajuscules(String nomF1) ...
   ```

   que haurà d'escriure en l'eixida estàndard, línia a línia, el contingut del fitxer el nom del qual està emmagatzemat en `nomF1`, canviant els caràcters alfabètics en minúscula per les corresponents majúscules.

   En cas que `nomF1` no existeixca, s'haurà de propagar una excepció del tipus `FileNotFoundException`.

   NOTA: Recordeu que l'operació `toUpperCase()` aplicada a una `String` torna una còpia d'ella en la que els caràcters alfabètics venen transformats a la majúscula corresponent.

   Solució:

   ```
   public static void escriuAmbMajuscules(String nomF1) throws FileNotFoundException {
       Scanner sIn = new Scanner(new File(nomF1));
       while(sIn.hasNextLine())
           System.out.println(sIn.nextLine().toUpperCase());
       sIn.close();
   }
   ```

2. [1 punt] Donada una `PilaIntEnla p`, escriu un mètode que mostre per l'eixida estàndard el valor del cim de la pila. Si la pila està buida ha de mostrar un missatge d'error. No es poden utilitzar els mètodes `talla()` o `esBuida()` de `PilaIntEnla`. S'ha de capturar l'excepció `NoSuchElementException` que es produeix en els mètodes `cim()` o `desempilar()` quan la pila està buida.

   Solució:

   ```
   public static void mostraCim(PilaIntEnla p) {
        try{
            System.out.println(p.cim());
        }catch(NoSuchElementException ex){
            System.err.println("Error: Pila Buida");
        }
   }
   ```

3. [2 punts] A una classe `ExamenEstructures` es desitja afegir un mètode `incrementaParells` estàtic que donada una cua d'enters `c` implementada mitjançant una seqüència de nodes enllaçats, torne una nova cua d'enters on s'hauran canviat els números parells sumant-los 1, deixant els números imparells sense tocar. De manera que si la cua original és:

   ```
   <- 5 10 14 13 22 31 <-
   ```

   la cua a tornar ha de contindre els següents números (en qualsevol ordre):

```
<- 5 11 15 13 23 31 <-
```

NOTA: Per resoldre aquest problema no es podrà gastar cap estructura de dades addicional.

a) (1.75 punts) Implementa el mètode `incrementaParells`.

b) (0.25 punts) Modifica el mètode anterior en cas que la cua s'implemente mitjançant un array.

Solució:

a) Una solució possible (iterativa) entre altres pot ser:

```
public static CuaIntEnla incrementaParells(CuaIntEnla q) {
    CuaIntEnla q1 = new CuaIntEnla();
    int n = q.talla(), cont = 0;
    for(int i=0; i<n; i++) {
        if (q.primer()%2==0) q1.encuar(q.desencuar()+1);
        else                 q1.encuar(q.desencuar());
    }
    return q1;
}
```

b) Bastaria amb canviar en la capçalera i declaracions de variables el nom de la classe CuaIntEnla per CuaIntArray i usar també la constructora de CuaIntArray.

4. 3 punts Es desitja modificar el comportament de l'operació `encuar(int)` en la classe `CuaIntEnla` de forma que els elements es mantinguen de forma ordenada ascendentment; això és, l'element **primer** de la cua serà el de menor valor d'entre tots, mentre que l'**últim** serà el de valor major.

   Es demana construir el mètode `encuar(int)` tenint en compte la definició anterior.

Solució:

```
/** Encuar ordenadament: versió primera. */
public void encuar(int val) {
    NodeInt q = null, p = primer;
    while (p!=null && val>p.dada) { q = p; p = p.seguent; }

    NodeInt nou = new NodeInt(val);
    if (q==null) {
        nou.seguent = primer;
        primer = nou;
    } else {
        nou.seguent = p;
        q.seguent = nou;
    }

    if (p==null) ultim = nou;
    talla++;
}
```

```
/** Encuar ordenadament: versió segona. */
public void encuar2(int val) {
    NodeInt q = null, p = primer;
    while (p!=null && val>p.dada) { q = p; p = p.seguent; }

    NodeInt nou;
    if (q==null) primer = nou = new NodeInt(val, primer);
    else q.seguent = nou = new NodeInt(val,p);

    if (p==null) ultim = nou;
    talla++;
}
```

5. 2 punts Es vol implementar un mètode anomenat `comptar` tal que:

   - Ha de rebre com arguments una llista `l` pertanyent a la classe `LlistaPIIntEnla`, i dos enters `i`, `j`, que es considera que delimiten un rang de valors `[i,j]`.
   - Ha de retornar el nombre d'elements en `l` que estiguen dins del rang `[i,j]`.

   S'ha de suposar que el mètode s'està escrivint dins d'una classe diferent a `LlistaPIIntEnla`.

Solució:

```
public static int comptar(LlistaPIIntEnla l, int i, int j) {
    int compt = 0;
    l.inici();
    while ( !l.esFi() ) {
        int x = l.recuperar();
        if ( x>=i && x<=j ) compt++;
        l.seguent();
    }
    return compt;
}
```