Segon Parcial d'IIP - ETSInf
Data: 16 de gener de 2013. Duració: 2:30 hores.

1. 2 punts Escriure un mètode estàtic que donat un enter $n >= 0$ mostre en la sortida una figura amb n línies. Cadascuna tindrà n-1 caràcters '*' i un caràcter 'a' en la posició de la diagonal principal de la figura. Per exemple, per a $n = 5$ escriuria:

```
a****
*a***
**a**
***a*
****a
```

Solució:

```
/** PRECONDICIÓ: n >= 0. */
public static void diagonal(int n) {
    for (int i=0; i<n; i++) {
       for (int j=0; j<i; j++) System.out.print('*');
       System.out.print('a');
       for (int j=i+1; j<n; j++) System.out.print('*');
       System.out.println();
    }
}
```

2. 2 punts Donat un array de nombres reals, es denomina *transició* a qualsevol parella d'elements de l'array en posicions consecutives d'ell. A més, per a qualsevol transició, es denomina *valor de la transició*, a la diferència en valor absolut entre els seus dos elements.

Es demana: escriure un mètode estàtic que donat un array de nombres reals `a`, amb almenys dues components (`a.length`>1), determine la transició amb *valor* màxim. El mètode haurà de retornar la posició del primer element d'aquesta transició; si hi haguera varies transicions amb el mateix valor màxim, es retornaria la de posició menor.

Per exemple, donat l'array `a = {-1.30, 2.22, 10.10, -1.03, 5.54, 7.32, 1.98}`, el mètode ha de retornar `2`, ja que la transició de major valor és la que es troba entre les posicions `2` i `3`, sent `11.13` la diferència en valor absolut entre `a[2]` i `a[3]`, que és la màxima d'entre les existents.

Solució:

```
/** PRECONDICIÓ: a.length > 1. */
public static int transicioMaxima(double[] a) {
    int pMax= 0;
    double dif = Math.abs(a[1]-a[0]);
    double max = dif;
    for (int i=1; i<a.length-1; i++) {
        dif = Math.abs(a[i+1]-a[i]);
        if (dif>max) { max=dif; pMax=i; }
    }
    return pMax;
}
```

3. 6 punts La Borsa de Valors gestiona el mercat d'accions. Cada acció es representa mitjançant un objecte de la classe `Accio`, ja coneguda; el seu codi es mostra a continuació:

```
public class Accio {
  private String empresa;
  private double apertura, minim, maxim, actual;
  /** Crea una Accio amb un nom d'empresa i valor d'apertura donats */
  public Accio(String nom, double ape) { this(nom, ape, ape, ape); }
  /** Crea una Accio amb un nom d'empresa i valors d'apertura, mínim i màxim donats */
  public Accio(String nom, double ape, double min, double max) {
    empresa = nom; apertura = ape;  minim = min; maxim = max; actual = ape;
  }
  /** Torna el nom de l'empresa */
  public String getEmpresa() { return empresa; }
  /** Torna el valor d'apertura */
  public double getApertura() { return apertura; }
  /** Torna el valor mínim */
  public double getMinim() { return minim; }
  /** Torna el valor màxim */
  public double getMaxim() { return maxim; }
  /** Torna el valor actual */
  public double getActual() { return actual; }
  /** Modifica el valor d'actual, actualitzant en el seu cas minim i maxim */
  public void setActual(double a) {
    if (a<minim) minim = a; else if (a>maxim) maxim = a;
    actual = a;
  }
  /** Comprova si l'acció està donant beneficis */
  public boolean capAmunt() { return (actual>apertura); }
  /** Comprova si l'acció en curs és igual a un altra; i.e. si són de la mateixa empresa */
  public boolean equals(Object o) {
    return o instanceof Accio && empresa.equals(((Accio) o).empresa);
  }
  /** Torna un String amb la informació de l'acció */
  public String toString() { return empresa + ": " + actual + " " + minim + " " + maxim; }
}
```

Es demana: implementar la classe `Borsa` per tal de gestionar un número variable d'accions (250 com a màxim), que ha d'incloure:

a) Atributs (0.5 punts):

- Un atribut `MAX_ACCIONS`, una constant que represente el número màxim d'accions a gestionar (250).
- Un atribut enter `accions`, que represente el número d'accions que estan sent gestionades en la borsa (valor entre 0 i `MAX_ACCIONS`).
- Un atribut `borsa`, array de tipus base `Accio`, amb capacitat màxima `MAX_ACCIONS`. Els objectes `Accio` s'emmagatzemaran a l'array `borsa` en posicions consecutives, des de 0 a `accions-1`.
- Un atribut enter `ambBeneficis`, que represente el número d'accions que estan donant beneficis (les que van cap a munt).

b) Un constructor (0.5 punts) que inicialice els atributs, creant l'array `borsa` sense cap acció inicial.

c) Un mètode (1 punt) amb perfil:

```
private int posicioDe(Accio a)
```

que, donada una `Accio a`, torne la seua posició a l'array si està o -1 si no es troba.

d) Un mètode (1.5 punts) amb perfil:

```
public boolean afegeixActualitza(Accio a)
```

que afegeix o actualitza la `Accio a` donada. S'ha d'usar el mètode privat `posicioDe(Accio)`. Si `a` ja està a l'array `borsa` l'actualitza i si no està l'afegeix si cap, tornant `true`. En ambdós casos, s'ha d'actualitzar l'atribut `ambBeneficis` si procedeix. Recorda que el mètode `capAmunt()` de la classe `Accio` permet comprovar si una acció aconsegueix beneficis o no.

Si `a` no està però no cap a l'array, el mètode torna `false`.

e) Un mètode (1 punt) amb perfil:

```
public Accio mesBeneficisQue(Accio a)
```

que torne la primera acció del array que està donant més beneficis que l'acció `a`, o torne `null` si no hi ha cap. Recorda que els beneficis que està donant una acció es calculen com la diferència entre el seu valor actual i el seu valor d'apertura.

f) Un mètode (1.5 punts) amb perfil:

```
public String[] empresesCapAMunt()
```

que torne un array de `String` que continga els noms de les empreses d'aquelles accions que estan donant beneficis. La longitud de l'array serà igual al número d'accions que estan donant beneficis, o 0 si no hi ha cap.

Solució:

```
public class Borsa {
    public static final int MAX_ACCIONS = 250;
    private Accio[] borsa;
    private int accions;
    private int ambBeneficis;

    public Borsa() {
       borsa = new Accio[MAX_ACCIONS];
       accions = ambBeneficis = 0;
    }

    private int posicioDe(Accio a) {
        int i = 0;
        while(i<accions && !borsa[i].equals(a)) i++;
        if (i<accions) return i;
        else return -1;
    }
```

```
    public boolean afegeixActualitza(Accio a) {
        boolean res = true;
        int pos = posicioDe(a);
        if (pos!=-1) {
            if (borsa[pos].capAmunt()) ambBeneficis--;
            borsa[pos] = a;
            if (a.capAmunt()) ambBeneficis++;
        }
        else if (accions!=MAX_ACCIONS) {
                borsa[accions++] = a;
                if (a.capAmunt()) ambBeneficis++;
             }
             else res = false;
        return res;
    }

    public Accio mesBeneficisQue(Accio a) {
        double beneA = a.getActual()-a.getApertura();
        int i = 0;
        while(i<accions && borsa[i].getActual()-borsa[i].getApertura()<=beneA)
            i++;
        if (i<accions) return borsa[i];
        else return null;
    }

    public String[] empresesCapAMunt() {
        String[] aux = new String[ambBeneficis];
        int k = 0;
        for (int i=0; i<accions; i++)
            if (borsa[i].capAmunt()) {
                aux[k] = borsa[i].getEmpresa();
                k++;
            }
        return aux;
    }
}
```