Segon Parcial d'IIP - ETSInf
Data: 19 de gener de 2012. Duració: 2:30 hores.

1. 2.0 punts Per a x, valor real qualsevol, es pot definir la recurrència següent:

$$t_0 = 0, \qquad t_{i+1} = t_i^2 + x$$

Si es calculen els valors successius t_i, per a diferents valors de x, es té que en algunes ocasions els termes t_i creixen fent-se cada vegada més grans, mentre que en altres ocasions els termes es mantenen dins d'un rang reduït de valors no superant mai el valor límit 2.0.

Per exemple,

- Si x inicialment val 1, els termes successius valen: 0, 1, 2, 5, 26 ...
- Si x inicialment val -1, els termes successius valen: 0, -1, 0, -1, ...

Com es pot vore, la primera successió de valors supera a partir del quart terme el valor 2.0, no fent-ho mai en el segon cas.

Escriure un mètode estàtic en Java que donat cert valor inicial real x determine, retornant cert o fals, si algun dels primers 100 termes de la successió definida, arriba a fer-se major o igual que el valor límit 2.0. El mètode que s'escriga haurà de realitzar el menor nombre possible d'iteracions.

Solució:

```
public static boolean exer1(double x) {
    final int NUM_IT = 100;
    final double VAL_LIM = 2.0;

    double term = 0;
    int it = 0;
    while (it<NUM_IT && term<VAL_LIM) {
        term = term*term + x;
        it++;
    }
    return term>=VAL_LIM;
}
```

2. 2.5 punts Escriure un mètode estàtic en Java que donat cert array d'enters a retorne, en una cerca ascendent, la posició de l'últim 0 anterior al primer 1, si n'hi ha, o retorne -1 en cas que no existeixca.

Per exemple,

- En l'array {3, 0, 2, 0, 3, 1, 6, 1, 1} (el primer 1 està en la posició 5), retornarà 3, ja que és la posició del 0 d'índex major existent abans de l'1 d'índex menor,
- En l'array {3, 2, 3, 4, 3, 1, 6, 0, 1} (el primer 1 està en la posició 5), retornarà -1 ja que no hi ha cap 0 abans del primer 1,
- En l'array {3, 2, 0, 3, 0, 4, 6, 7, 5} (no hi ha cap 1), tornarà -1 ja que no existeix cap 1.

Solució:

```
public static int zeroAbansUn (int[] a) {
    int pos0 = -1;
    int i = 0;
    boolean trobat = false;
    while (i<a.length && !trobat) {
        if (a[i]==1) trobat = true;
        else {
            if (a[i]==0) pos0=i;
            i++;
        }
    }
    if (trobat) return pos0;
    else return -1;
}
```

3. 5.5 punts Per a la gestió d'un forn industrial es prenen mesures de la temperatura i pressió d'aquest en determinats instants de temps al llarg d'una hora. Algunes d'aquestes mesures excedeixen el rang vàlid de les mateixes, per la qual cosa es fa necessari comptar amb algun tipus d'advertència que permeta conèixer el moment en que es va produir l'excés. El programa que es demana, del qual es donen algunes parts ja fetes, permet resoldre el seguiment proposat.

   Mitjançant la classe `Mesura`, ja realitzada i de la que es mostra el seu codi al complet a continuació, es representa una mesura individual:

```
/**
 * La classe Mesura representa els valors de la temperatura i pressio d'un forn
 * industrial en un instant determinat.
 * @author (IIP - ETSINF - DSIC - UPV)
 * @version (curs 2011-12)
 */
public class Mesura {
    public static final double MAX_PRES = 100.0;
    public static final int MAX_TEMP = 1200;

    public static final int CORRECTA = 0;
    public static final int PROBLEMA_1 = 1;
    public static final int PROBLEMA_2 = 2;

    private double pres;        // la pressio
    private int temp;           // la temperatura

    /** Inicialitza una Mesura amb certa pressio p i temperatura t. */
    public Mesura(double p, int t) {
        pres = p;
        temp = t;
    }

    /** Retorna la pressio. */
    public double getPres() { return pres; }
```

```
    /** Retorna la temperatura. */
    public int getTemp() { return temp; }

    /** Retorna un String amb la informacio de la Mesura. */
    public String toString() {
        return "Pressio: " + pres + "MHp\n" +
               "Temperatura: " + temp + "Grds\n";
    }

   /**
    * Retorna PROBLEMA_2 si temperatura i pressio son els dos excessius,
    * torna PROBLEMA_1 si be la temperatura, be la pressio, son excessius,
    * torna CORRECTA en cas contrari (ningun es excessiu).
    */
   public int estat() {
        if (pres>MAX_PRES && temp>MAX_TEMP) return PROBLEMA_2;
        else if (pres>MAX_PRES || temp>MAX_TEMP) return PROBLEMA_1;
        else return CORRECTA;
    }
}
```

Al llarg d'una hora es poden fer diferents mesures. Cadascuna d'elles va associada temporalment al segon determinat de l'hora en què aquesta es fa, de manera que el nombre màxim de mesures que poden fer-se és de 3600.

Per exemple, es pot conèixer la mesura feta en el segon 35, la del segon 283 i la presa en el segon 1125.

Per a representar aquest conjunt de mesures s'ha d'escriure una classe `MesuresForn` que ha d'incloure:

a) Atributs (0.5 punts):

- Un atribut `mesures`, array de `Mesura`, amb 3600 elements. Cadascun d'aquests elements pot contenir una `Mesura` individual. La posició de l'element a l'array indica el segon en què s'ha realitzat la mesura.
  Nota: Al llarg de l'execució pot passar que hi haja elements de l'array no inicialitzats per que en el segon que representen no s'ha efectuat cap mesura,
- un atribut enter `numProblem`, que permetrà portar un comptador de les mesures amb problemes que es produeixen. Aquest atribut s'ha d'actualitzar quan s'incloga una nova mesura tal que l'estat de la qual no siga `Mesura.CORRECTA`,

b) un constructor (0.5 punts) que inicialitza els atributs, creant l'array de 3600 posicions sense cap `Mesura` inicial,

c) un mètode (1.0 punts) amb capçalera:

```
public void altaMesura(int segon, Mesura m)
```

que donada certa `Mesura` i el segon en que aquesta es realitza, actualitze l'element corresponent de l'array així com l'atribut `numProblem` en el cas de que l'estat de la mesura no siga `Mesura.CORRECTA`,

d) un mètode (1.5 punts) amb capçalera:

```
public int minPressio()
```

que torne la posició en que es troba la mesura de menor pressió. Si hi ha diverses mesures amb la mateixa pressió mínima es retornarà la posició de l'última de totes elles. Si no hi ha cap mesura es retornarà -1 per indicar-ho.

Nota: Cal recordar que la constant `Double.MAX_VALUE` representa el màxim valor de tipus `double` representable en Java,

e) un mètode (2.0 punts) amb capçalera:

```
public int[] mesuresAmbProblemes()
```

que retorne un array que continga la posició de totes les mesures que han presentat algun problema (és a dir, aquelles mesures amb qualsevol estat diferent de `Mesura.CORRECTA`). L'array que s'ha de retornar tindrà tants elements com mesures problemàtiques s'hagen pogut donar. L'array que es torne haurà d'estar ordenat temporalment en forma ascendent, és a dir, les mesures posteriors apareixeran darrere a l'array de les mesures anteriors.

Solució:

```
/**
 * @author (IIP - ETSINF - DSIC - UPV)
 * @version (curs 2011-12)
 */
public class MesuresForn {
    public int NUM_MESURES = 3600;
    private Mesura[] mesures;
    private int numProblem;

    public MesuresForn() {
       mesures = new Mesura[NUM_MESURES];
       numProblem = 0;
    }

    public void altaMesura(int segon, Mesura m) {
        if (mesures[segon]!=null &&
            mesures[segon].estat()!=Mesura.CORRECTA) numProblem--;
        mesures[segon] = m;
        if (m.estat()!=Mesura.CORRECTA) numProblem++;
    }

    public int minPressio() {
        double min = Double.MAX_VALUE;
        int posMin = -1;
        for (int i=0; i<mesures.length; i++)
            if (mesures[i]!=null && mesures[i].getPres()<=min) {
                min = mesures[i].getPres();
                posMin = i;
            }
        return posMin;
    }

    public int[] mesuresAmbProblemes() {
        int[] aux = new int[numProblem]; int k = 0;
        for (int i=0; i<mesures.length; i++)
            if (mesures[i]!=null &&
                mesures[i].estat()!=Mesura.CORRECTA) {
                aux[k] = i; k++;
            }
        return aux;
    }
}
```