Segon Parcial de PRG - ETSInf
Data: 9 de maig de 2011. Duració: 2 hores.

1. [4 punts] Per a resoldre el problema de representar les mesures d'una estació meteorològica es proporcionen com a material addicional, adjuntes a aquest enunciat, les classes `Moment` i `Mesura`.

   Mitjançant la classe `Moment` es representa una data (dia i mes) i hora determinada; mentres que la classe `Mesura` associa un valor de `Moment` determinat a certa mesura de temperatura i pluviositat.

   Es desitja implementar en Java una classe `GestorMeteo` que gestione (mitjançant un array) dades de tipus `Mesura` per a poder processar-los posteriorment.

   Per a això, **es demana**:

   a) Definir els atributs necessaris de la classe `GestorMeteo`, havent d'incloure la definició que es faça: l'array suport de les distintes mesures, un enter que indique el nombre de mesures existents, i un valor constant igual a 1000 que indique el nombre màxim de mesures.

   b) Escriure un constructor per a la classe `GestorMeteo` que llija les dades de les mesures des d'un fitxer (el nom del qual serà un paràmetre d'aquest mètode). Cadascuna de les línies del fitxer tindrà el format que es mostra a continuació:

   ```
   mes  dia  hora  tempMax  pluvio
   ```

   sent un exemple de les primeres línies d'aquest fitxer el següent:

   ```
    6  15  12  32.15    0.0
   12   1   3  18.10   25.5
    3  19  20  21.3   335.1
   ..  ..  ..  .....  .....
   ..  ..  ..  .....  .....
   ```

   c) Escriure un mètode `majorTemp()` en la classe `GestorMeteo` per a tornar el moment (`Moment`) amb la major temperatura d'entre totes les mesures.

   d) Escriure un mètode `pluvioMitjana(int)` en la classe `GestorMeteo` per a determinar la pluviositat mitjana de cert mes (sent el nombre de mes el paràmetre del mètode).

2. [4 punts] Tenint en compte tots els aspectes ja mencionats en el problema anterior, **es demana**:

   a) Escriure un mètode `compareTo(Moment)` en la classe `Moment` que torne un valor enter menor, igual o major que zero segons que la variable de tipus `Moment` sobre la qual s'aplique siga, respectivament, anterior, igual o posterior al `Moment` que es rep com a paràmetre.

   b) Escriure un mètode `anterior(Mesura)` en la classe `Mesura`, utilitzant el mètode `compareTo(Moment)`, que torne `true` quan el `Moment` associat a la `Mesura` sobre la qual s'aplica és anterior al de la `Mesura` que es rep com a paràmetre, o `false` en cas contrari.

   c) Escriure un mètode `ordenarPerMoment()` en la classe `GestorMeteo` que ordene d'anterior a posterior, utilitzant el mètode `anterior(Mesura)`, l'array de mesures associat a la classe `GestorMeteo`.

   Nota: Per a facilitar la resolució, s'acompanya com a material addicional una versió del mètode d'ordenació d'un array d'enters per Selecció directa; utilitze's com guia per a resoldre el problema demanat.

3. [2 punts] Considerant les classes anteriors, **es demana**: implementar en Java una classe `Principal` amb un mètode `main` que demane a l'usuari el nom d'un fitxer que conté mesures com les descrites, inicialitze adequadament un objecte de tipus `GestorMeteo`, escriga en la pantalla mes, dia i hora en què es va produir la major temperatura i, finalment, ordene temporalment les mesures de `GestorMeteo`.

Moment.java

```java
public class Moment {

    private int hora, dia, mes;

    public Moment (int d, int m, int h) {
        this.dia  = d;
        this.mes  = m;
        this.hora = h;
    }

    public int getDia ()  { return dia;  }
    public int getMes ()  { return mes;  }
    public int getHora () { return hora; }

    public String toString () { return mes + "  " + dia + "  " + hora; }

    public boolean equals (Object o) {
        Moment a = (Moment) o;
        return dia==a.dia && mes==a.mes && hora==a.hora;
    }

    public int compareTo (Moment altre) {
        // A COMPLETAR PER L'ALUMNE ...

    }
}
```

Moment.java

Mesura.java

```java
public class Mesura {

    private Moment mom;
    private double tempMax;
    private double pluvio;

    public Mesura (Moment m, double tmax, double pl) {
        this.mom = m;
        this.tempMax = tmax;
        this.pluvio = pl;
    }

    public Moment getMoment ()  { return mom; }
    public double getTempMax () { return tempMax; }
    public double getPluvio ()  { return pluvio; }

    public String toString () { return mom.toString() + "  " + tempMax + "  " + pluvio; }

    public boolean anterior (Mesura altra) {
        // A COMPLETAR PER L'ALUMNE, emprant el mètode compareTo de la classe Moment

    }
}
```

Mesura.java

GestorMeteo.java

```
import java.util.*;
import java.io.*;

public class GestorMeteo {

    // Definició dels atributs: A COMPLETAR PER L'ALUMNE ...

    public GestorMeteo (String nomFich) throws IOException {
        // A COMPLETAR PER L'ALUMNE ...

    }

    public Moment majorTemp () {
        // A COMPLETAR PER L'ALUMNE ...

    }

    public double pluvioMitjana (int mes) {
        // A COMPLETAR PER L'ALUMNE ...

    }

    /**
     * Metode d'ordenació d'un array d'enters per Selecció Directa.
     * Utilitze's com guia per a resoldre el metode ordenarPerMoment().
     */
    public void SeleccioDirecta (int v[]) {
        for (int i= 0; i<v.length-1; i++) {
            int posMin = i;
            for (int j=i+1; j<v.length; j++)
                if (v[j]<v[posMin]) posMin = j;

            int aux = v[posMin];
            v[posMin]= v[i];
            v[i] = aux;
        }
    }

    public void ordenarPerMoment () {
        // A COMPLETAR PER L'ALUMNE, emprant el metode anterior de la classe Mesura

    }
}
```

GestorMeteo.java

Principal.java

```
import java.util.*;
import java.io.*;

public class Principal {
    public static void main (String args[]) throws IOException {
        // A COMPLETAR PER L'ALUMNE ...

    }
}
```

Principal.java

Solució:

Moment.java

```
public class Moment {
    ...
    public int compareTo (Moment altre) {
       int menorIgualoMajor = 1;
        if (this.equals(altre)) menorIgualoMajor = 0;
        else if ((mes<altre.mes) || (mes==altre.mes && dia<altre.dia) ||
                 (mes==altre.mes && dia==altre.dia && hora<altre.hora))
                menorIgualoMajor = -1;
        return menorIgualoMajor;
    }
}
```

Moment.java

Mesura.java

```
public class Mesura {
    ...
    public boolean anterior (Mesura altra) { return this.mom.compareTo(altra.mom)<0; }
}
```

Mesura.java

GestorMeteo.java

```
import java.util.*;
import java.io.*;

public class GestorMeteo {

    // Definició dels atributs
    private Mesura lArray[];
    private int numM;
    public static final int MAXM = 1000;

    public GestorMeteo (String nomFich) throws IOException {
        lArray = new Mesura[MAXM];
        numM = 0;
        Scanner f = new Scanner(new File(nomFich)).useLocale(Locale.US);
        while(f.hasNext() && numM<MAXM){
            int mes = f.nextInt();
            int dia = f.nextInt();
            int hora = f.nextInt();
            double tempMax = f.nextDouble();
            double pluvio = f.nextDouble();
            lArray[numM++] = new Mesura(new Moment(dia,mes,hora),tempMax,pluvio);
        }
        f.close();
    }

    public Moment majorTemp () {
        int posMax = 0;
        for(int i=1; i<numM; i++)
            if (lArray[i].getTempMax() > elArray[posMax].getTempMax()) posMax = i;
        if (numM>0) return lArray[posMax].getMoment();
        else return null;
    }
```

```
33
34      public double pluvioMitjana (int mes) {
35          double plMitjana = 0;
36          int cont = 0;
37          for(int i=0; i<numM; i++)
38              if (lArray[i].getMoment().getMes() == mes) {
39                  plMitjana+=lArray[i].getPluvio();
40                  cont++;
41              }
42          if (cont>0) return plMitjana/cont;
43          else return -1;
44      }
45
46      public void ordenarPerMoment () {
47         for (int i=0; i<numM-1; i++) {
48              int posMin = i;
49              for (int j=i+1; j<numM; j++)
50                  if (lArray[j].anterior(lArray[posMin])) posMin = j;
51
52              Mesura aux = lArray[posMin];
53              lArray[posMin] = lArray[i];
54              lArray[i] = aux;
55          }
56
57      }
58  }
```

GestorMeteo.java

Principal.java

```
1   import java.util.*;
2   import java.io.*;
3
4   public class Principal {
5       public static void main (String args[]) throws IOException {
6           Scanner tcl = new Scanner(System.in);
7
8           System.out.print("Nom de fitxer: ");
9           String nF = tcl.nextLine().trim();
10
11          GestorMeteo gM = new GestorMeteo(nF);
12
13          Moment m = gM.majorTemp();
14          if (m!=null) System.out.println("Mes, dia i hora: " + m);
15          else System.out.println("No hi han mesures");
16
17          gM.ordenarPerMoment();
18      }
19  }
```

Principal.java