Segundo Parcial de PRG - ETSInf
Fecha: 9 de mayo de 2011. Duración: 2 horas.

1. [4 puntos] Para resolver el problema de representar las medidas de una estación meteorológica se proporcionan como material adicional, adjuntas a este enunciado, las clases `Momento` y `Medida`.

   Mediante la clase `Momento` se representa una fecha (día y mes) y hora determinada; mientras que la clase `Medida` asocia un valor de `Momento` dado a una medida de temperatura y pluviosidad determinada.

   Se desea implementar en Java una clase `GestorMeteo` que gestione (mediante un array) datos de tipo `Medida` para poder procesarlos posteriormente.

   Para ello, **se pide**:

   a) Definir los atributos necesarios de la clase `GestorMeteo`, debiendo incluir dicha definición: el array soporte de las distintas medidas, un entero que indique el número de medidas existentes, y un valor constante igual a 1000 que indique el número máximo de medidas.

   b) Escribir un constructor para la clase `GestorMeteo` que lea los datos de las medidas desde un fichero (cuyo nombre será un parámetro de dicho método). Cada una de las líneas del fichero tendrá el formato que se muestra a continuación:

   ```
   mes  dia  hora  tempMax  pluvio
   ```

   siendo un ejemplo de las primeras líneas de dicho fichero el siguiente:

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

   c) Escribir un método `mayorTemp()` en la clase `GestorMeteo` para devolver el momento (`Momento`) con la mayor temperatura de entre todas las medidas.

   d) Escribir un método `pluvioMedia(int)` en la clase `GestorMeteo` para determinar la pluviosidad media de un mes dado (siendo el número de mes el parámetro del método).

2. [4 puntos] Teniendo en cuenta todos los aspectos ya mencionados en el problema anterior, **se pide**:

   a) Escribir un método `compareTo(Momento)` en la clase `Momento` que devuelva un valor entero menor, igual o mayor que cero según que la variable de tipo `Momento` sobre la que se aplique sea, respectivamente, anterior, igual o posterior al `Momento` que se recibe como parámetro.

   b) Escribir un método `anterior(Medida)` en la clase `Medida`, utilizando el método `compareTo(Momento)`, que devuelva `true` cuando el `Momento` asociado a la `Medida` sobre la que se aplica es anterior al de la `Medida` que se recibe como parámetro, o `false` en caso contrario.

   c) Escribir un método `ordenarPorMomento()` en la clase `GestorMeteo` que ordene de anterior a posterior, utilizando el método `anterior(Medida)`, el array de medidas asociado a la clase `GestorMeteo`.

   Nota: Para facilitar la resolución se acompaña como material adicional una versión del método de ordenación de un array de enteros por Selección directa; utilícese como guía para resolver el problema pedido.

3. [2 puntos] Considerando las clases anteriores, **se pide**: implementar en Java una clase `Principal` con un método `main` que pida al usuario el nombre de un fichero que contiene medidas como las descritas, inicialice adecuadamente un objeto de tipo `GestorMeteo`, escriba en la pantalla mes, día y hora en que se produjo la mayor temperatura y, finalmente, ordene temporalmente las medidas de `GestorMeteo`.

Momento.java

```java
public class Momento {

    private int hora, dia, mes;

    public Momento (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) {
        Momento a = (Momento) o;
        return dia==a.dia && mes==a.mes && hora==a.hora;
    }

    public int compareTo (Momento otro) {
        // A COMPLETAR POR EL ALUMNO ...

    }
}
```

Momento.java

Medida.java

```java
public class Medida {

    private Momento mom;
    private double tempMax;
    private double pluvio;

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

    public Momento getMomento () { return mom; }
    public double getTempMax () { return tempMax; }
    public double getPluvio () { return pluvio; }

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

    public boolean anterior (Medida otra) {
        // A COMPLETAR POR EL ALUMNO, utilizando el método compareTo de la clase Momento.

    }
}
```

Medida.java

GestorMeteo.java

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

public class GestorMeteo {

    // Definición de atributos: A COMPLETAR POR EL ALUMNO ...

    public GestorMeteo (String nomFich) throws IOException {
        // A COMPLETAR POR EL ALUMNO ...

    }

    public Momento mayorTemp () {
        // A COMPLETAR POR EL ALUMNO ...

    }

    public double pluvioMedia (int mes) {
        // A COMPLETAR POR EL ALUMNO ...

    }

    /**
     * Método de ordenación de un array de enteros por Selección Directa.
     * Utilícese como guía para resolver el método ordenarPorMomento().
     */
    public void SeleccionDirecta (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 ordenarPorMomento () {
        // A COMPLETAR POR EL ALUMNO, utilizando el método anterior de la clase Medida.

    }
}
```

GestorMeteo.java

Principal.java

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

public class Principal {
    public static void main (String args[]) throws IOException {
        // A COMPLETAR POR EL ALUMNO ...

    }
}
```

Principal.java

Solución:

Momento.java

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

Momento.java

Medida.java

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

Medida.java

GestorMeteo.java

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

public class GestorMeteo {

    // Definición de atributos
    private Medida elArray[];
    private int numM;
    public static final int MAXM = 1000;

    public GestorMeteo (String nomFich) throws IOException {
        elArray = new Medida[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();
            elArray[numM++] = new Medida(new Momento(dia,mes,hora),tempMax,pluvio);
        }
        f.close();
    }

    public Momento mayorTemp () {
        int posMax = 0;
        for(int i=1; i<numM; i++)
            if (elArray[i].getTempMax() > elArray[posMax].getTempMax()) posMax = i;
        if (numM>0) return elArray[posMax].getMomento();
        else return null;
    }
```

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

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("Nombre de fichero: ");
9           String nF = tcl.nextLine().trim();
10
11          GestorMeteo gM = new GestorMeteo(nF);
12
13          Momento m = gM.mayorTemp();
14          if (m!=null) System.out.println("Mes, dia y hora: " + m);
15          else System.out.println("No existen medidas");
16
17          gM.ordenarPorMomento();
18      }
19  }
```

Principal.java