PRG second quiz – ETSInf
Date: May, 9th 2011. Duration: 2 hours.

1. $\boxed{\text{4 points}}$ To solve the problem of representing the measurements of a weather station, we provide you, as additional material attached to this statement, the Java classes `Timestamp` and `Measure`.

   The class `Timestamp` represents a given date (month and day) and time, while the class `Measure` associates the value of a timestamp with a measure of temperature and rainfall.

   You should implement the Java class `MeteoManager` in order to manage several objects of the class `Measure` by means of an array.

   So, we ask you to do the following:

   a) Define the necessary attributes of the class `MeteoManager`, this definition must include: the array for storing several measures, an integer indicating the number of stored measures, and a constant value equal to 1000 to indicate the maximum number of measures.

   b) Write a constructor for the class `MeteoManager`, which reads the measurement data from a file (the file name is a parameter of the constructor). Each of the lines of the file have the format shown below:

   ```
   month  day  hour  maxTemp  rainfall
   ```

   being an example of the contents of the file the following lines:

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

   c) Write a method `getHighestTemp()` in the class `MeteoManager` for obtaining the date and time where the maximum temperature was measured. This method should return an object of the class `Timestamp`.

   d) Write a method `averageRainfall( int )` in the class `MeteoManager` for estimating the average rainfall in the month specified as parameter.

2. $\boxed{\text{4 points}}$ Considering all the aspects explained in the previous problem, we ask you to do the following:

   a) Write a method `compareTo( Timestamp )` in the class `Timestamp` that returns an integer value less than, equal to or greater than zero depending on if the variable of type `Timestamp` on which it is applied is prior, equal, or posterior with respect to the one received as parameter.

   b) Write a method `isPrevious( Measure )` in the class `Measure`, which returns a boolean and uses the method `compareTo( Timestamp )` of the `Timestamp` class. Remember that the method `compareTo()` returns `true` when the timestamp associated with the `Measure` object on which it is applied is prior to the timestamp of the `Measure` object that is received as a parameter, or `false` otherwise.

   c) Write a method `sortByTimestamp()` in the class `MeteoManager` for sorting in chronological order the elements in the array of measurements. This method must use the method `isPrevious( Measure )`.

   Notice: for helping you in the solution of this problem you will find, attached to this statement as additional material, a version of the selection-sort algorithm for integer arrays.

3. $\boxed{\text{2 points}}$ Considering the previous classes, we ask you to do the following: to implement a Java class `Main` containing the method `main( String [] )`, this method should ask the user for the name of the file where the measurements are described, and then create a `MeteoManager` object. Once the `MeteoManager` object is properly initialized, the main method must write the highest temperature on screen, and sort chronologically the measurements contained in the `MeteoManager` object.

Timestamp.java

```java
public class Timestamp
{
    private int hour, day, month;

    public Timestamp( int d, int m, int h ) {
        this.day   = d;
        this.month = m;
        this.hour  = h;
    }

    public int getDay()   { return day; }
    public int getMonth() { return month; }
    public int getHour()  { return hour; }

    public String toString () { return month + "  " + day + "  " + hour; }

    public boolean equals( Object o ) {
        if ( o instanceof Timestamp ) {
            Timestamp a = (Timestamp) o;
            return day==a.day && month==a.month && hour==a.hour;
        }
    }

    public int compareTo( Timestamp other ) {
        // TO BE DONE BY STUDENTS
    }
}
```

Timestamp.java

Measure.java

```java
public class Measure
{
    private Timestamp t;
    private double tempMax;
    private double pluvio;

    public Measure( Timestamp t, double tmax, double pl )
    {
        this.t = t;
        this.tempMax = tmax;
        this.pluvio = pl;
    }

    public Timestamp getTimestamp() { return t; }
    public double getTempMax() { return tempMax; }
    public double getPluvio() { return pluvio; }

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

    public boolean anterior( Measure other ) {
        // TO BE DONE BY STUDENTS, it is recommended
        // to use the compareTo() method of the Timestamp class.
    }
}
```

Measure.java

MeteoManager.java

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

public class MeteoManager
{
    // Declaration of attributes (instance variables)
    // TO BE DONE BY STUDENTS

    public MeteoManager( String nomFich ) throws IOException {
        // TO BE DONE BY STUDENTS
    }

    public Timestamp mayorTemp() {
        // TO BE DONE BY STUDENTS
    }

    public double averageRainfall( int month ) {
        // TO BE DONE BY STUDENTS
    }

    /**
     * Selection-Sort method for arrays of ints.
     * Use it as reference to implement the sortByTimestamp() method.
     */
    public void selectionSort( 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 sortByTimestamp() {
        // TO BE DONE BY STUDENTS, see the previous sorting method.
    }
}
```

MeteoManager.java

Main.java

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

public class Main
{
    public static void main( String args[] ) throws IOException {
        // TO BE DONE BY STUDENTS
    }
}
```

Main.java

Solution:

Timestamp.java

```java
public class Timestamp
{   ...
    public int compareTo( Timestamp other ) {
        int lesserEqualorGreater = 1;
        if (this.equals(other)) lesserEqualorGreater = 0;
        else if ((month<other.month) || (month==other.month && day<other.day) ||
                  (month==other.month && day==other.day && hour<other.hour))
                  lesserEqualorGreater = -1;
        return lesserEqualorGreater;
    }
}
```

Timestamp.java

Measure.java

```java
public class Measure
{   ...
    public boolean anterior( Measure other ) { return this.t.compareTo(other.t)<0; }
}
```

Measure.java

MeteoManager.java

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

public class MeteoManager
{
    // Declaration of attributes (instance variables)
    private Measure theArray[];
    private int numM;
    public static final int MAXM = 1000;

    public MeteoManager( String nomFich ) throws IOException {
        theArray = new Measure[MAXM];
        numM = 0;
        Scanner f = new Scanner(new File(nomFich)).useLocale(Locale.US);
        while(f.hasNext() && numM<MAXM){
            int month = f.nextInt();
            int day = f.nextInt();
            int hour = f.nextInt();
            double tempMax = f.nextDouble();
            double rainfall = f.nextDouble();
            theArray[numM++] = new Measure(new Timestamp(day,month,hour),tempMax,rainfall);
        }
        f.close();
    }

    public Timestamp mayorTemp() {
        int posMax = 0;
        for(int i=1; i<numM; i++)
            if (theArray[i].getTempMax() > theArray[posMax].getTempMax()) posMax = i;
        if (numM>0) return theArray[posMax].getTimestamp();
        else return null;
    }

    public double averageRainfall( int month ) {
        double rfAverage = 0;
        int count = 0;
        for(int i=0; i<numM; i++)
            if (theArray[i].getTimestamp().getMonth() == month) {
                rfAverage+=theArray[i].getPluvio();
                count++;
            }
        if (count>0) return rfAverage/count;
        else return -1;
    }

    public void sortByTimestamp() {
       for (int i=0; i<numM-1; i++) {
            int posMin = i;
            for (int j=i+1; j<numM; j++)
                if (theArray[j].anterior(theArray[posMin])) posMin = j;

            Measure aux = theArray[posMin];
            theArray[posMin] = theArray[i];
            theArray[i] = aux;
        }
    }
}
```

MeteoManager.java

Main.java

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

public class Main
{
    public static void main( String args[] ) throws IOException {
         Scanner in = new Scanner(System.in);

        System.out.print("File name: ");
        String fname = in.nextLine().trim();

        MeteoManager mM = new MeteoManager(fname);

        Timestamp t = mM.mayorTemp();
        if (t!=null) System.out.println("Month, day and hour: " + t);
        else System.out.println("There are no measures");

        mM.sortByTimestamp();
    }
}
```

Main.java