IIP second partial - Retake exam - ETSInf
Date: January 28th, 2014 – Time: 2h 30'

1. 6.5 points The class `Sensor`, that represents a temperature sensor attached to a device, is available. The physic sensor, apart from measuring temperature, can change (up or down) the device temperature (in Celsius degrees). Moreover, the user can define a ideal temperature for the area where the sensor is placed. The following figure presents a summary of the documentation for that class:

Constructor Summary

Constructors

Constructor and Description
`Sensor(java.lang.String name, int idealTemp)` Creates a Sensor with a name and an ideal temperature.

Method Summary

Methods

Modifier and Type	Method and Description
void	`decreaseCurrentTemp()` Decreases the current temperature with a random amount of degrees.
boolean	`equals(java.lang.Object o)` Checks if the current Sensor is equals to another one; i.e. if their names match.
double	`getCurrentTemp()`
double	`getIdealTemp()`
java.lang.String	`getName()`
void	`increaseCurrentTemp()` Increases the current temperature with a random amount of degrees.
java.lang.String	`toString()` Returns a String with the information of the current Sensor.

You must: implement the `DomoticHouse` class that represents a domotic house where there could be many sensors in different places. This class is defined by the following components (attributes and methods):

a) (0.5 points) Attributes:
- `MAX_SENSORS`, a constant with the maximum number of sensors that can be placed in a domotic house: 100
- `numSensors`, integer in the interval `[0..MAX_SENSORS]` which represents the current number of sensors placed in the domotic house
- `sensors`, an array of the `Sensor` datatype and size `MAX_SENSORS`, where the different sensors are stored consecutively in the initial positions (from 0 to `numSensors-1`)
- `maxDegreeDev`, a positive real number that represents the maximum deviation in degrees that the user admits for the domotic house sensors
- `numSensorsWithDev`, integer that represents how many sensors in the domotic house have currently a deviation, in absolute value degrees, higher than `maxDegreeDev`

b) (0.5 points) A general constructor, with the needed parameters, that creates a domotic house with 0 sensors and a given maximum deviation in degrees

c) (1 point) A method with header: `private boolean existsSensor(Sensor s)` that given `Sensor s` returns if the corresponding sensor is in the house or not

d) (0.5 points) A method with header: `private boolean sensorWithDeviation(Sensor s)` that, given a `Sensor`, returns `true` when that `Sensor` presents a deviation between its ideal temperature and its real temperature higher that the maximum deviation in degrees defined for the domotic house

e) (1 point) A method with header: `public int registerSensor(Sensor s)` that inserts (registers) `Sensor s` in the array of the domotic house and returns the position in which `Sensor s` has been registered; if there is no room to register `s` or it was registered previously, the method must return `-1` in order to indicate that it could not be registered; employ the private method `existsSensor(Sensor)` to verify if the sensor was present and the private method `sensorWithDeviation(Sensor)` to verify if `Sensor s` was deviated

f) (1 point) A method with header: `public Sensor[] sensorsWithDeviation()`
that returns an array of `Sensor` objects that present a deviation between its current temperature and its ideal temperature higher that the maximum deviation in degrees defined for the domotic house; the returned array length must be equal to the number of sensors that present that deviation, or 0 when no sensor with deviation is present; employ the method `sensorWithDeviation(Sensor)`

g) (1 point) A method with header: `public void fitTemperature()`
that regulates the temperature of all `Sensor` objects in the domotic house that present a deviation of its current temperature with respect to its ideal temperature higher than the maximum temperature defined for the domotic house; the regulation must be done for each `Sensor` object that presents deviation, by using its corresponding methods in order to fit the current temperature, until the `Sensor` object does not present that deviation

h) (1 point) A method with header: `public Sensor sensorWithHighestTemperature()`
that returns the `Sensor` that measures the maximum temperature when the method is called, or `null` when no `Sensor` is registered in the house

Solution:

DomoticHouse.java

```
/**
 * Represents a DomoticHouse, that includes an array of Sensor
 * that obtain the temperature in some areas of the house.
 */
public class DomoticHouse {
    private Sensor[] sensors;
    private int numSensors;
    private int numSensorsWithDev;
    private double maxDegreeDev;
    public static final int MAX_SENSORS = 100;

    public DomoticHouse(double maxDegreeDev) {
      sensors = new Sensor[MAX_SENSORS];
      numSensors = numSensorsWithDev = 0;
      this.maxDegreeDev = maxDegreeDev;
    }

    private boolean existsSensor(Sensor s) {
      int i;
      for (i=0; i<numSensors && !sensors[i].equals(s); i++);
      return i<numSensors;
    }

    private boolean sensorWithDeviation(Sensor s) {
      return Math.abs(s.getCurrentTemp() - s.getIdealTemp()) > maxDegreeDev;
    }

    public int registerSensor(Sensor s) {
      int res = -1;
      if (!existsSensor(s) && numSensors < MAX_SENSORS) {
        sensors[numSensors] = s; numSensors++; res = numSensors;
        if (sensorWithDeviation(s)) numSensorsWithDev++;
      }
      return res;
    }
```

```
    public Sensor[] sensorsWithDeviation() {
        Sensor[] aux = new Sensor[numSensorsWithDev];
        int k = 0;
        for (int i=0; i<numSensors; i++)
            if (sensorWithDeviation(sensors[i])) {
                aux[k] = sensors[i];
                k++;
            }
        return aux;
    }

    public void fitTemperature() {
      for (int i=0; i<numSensors; i++) {
         Sensor s = sensors[i];
         while (sensorWithDeviation(s))
           if (s.getCurrentTemp() > s.getIdealTemp()) s.decreaseCurrentTemp();
           else s.increaseCurrentTemp();
      }
         numSensorsWithDev = 0;
    }

    public Sensor sensorWithHighestTemperature() {
      Sensor max = sensors[0];
      for (int i=0; i<numSensors; i++)
         if (sensors[i].getCurrentTemp() > max.getCurrentTemp()) max = sensors[i];
      return max;
    }
}
```

DomoticHouse.java

2. 1.5 points Given an integer `h` in the range [0..23], implement a public static method that writes on the standard output, hour by hour and minute by minute, all the hours in the five character format `hh:mm` from `00:00` until the last minute of the hour `h`. Each hour must be written in a different line. E.g., for `h=13`, it must show:

```
00:00
00:01
00:02
...
00:59
01:00
01:01
01:02
...
01:59
...
13:00
13:01
13:02
...
13:59
```

N.b.: suspension points (. . .) represent intervals of hours that have not been written in the statement because of space reasons

Solution:

```
/** h is in the range [0..23] */
public static void displayHour(int h) {
  int h1, m;
  for (h1=0; h1<=h; h1++)
    for (m=0; m<60; m++) {
      if (h1<10) System.out.print("0"+h1); else System.out.print(h1);
      System.out.print(":");
      if (m<10) System.out.println("0"+m); else System.out.println(m);
    }
}
```

3. 2 points **You must:** a) Write what shows on the screen the following code and describe (in two or three lines at most) what it does

   b) Indicate what will happen if the guard of the inner loop in `function(int[][])` was `j<m.length`

```
public class Trace {
    public static void main(String[] args) {
        int[][] a = { { 1, 2, 2 }, { 0, 1, 2 }, { 0, 0, 1 } };
        function( a );
        for (int i=0; i<a.length; i++) {
            for (int j=0; j<a.length; j++)
                System.out.printf( " %3d ", a[i][j] );
            System.out.println();
        }
    }
    public static void function(int[][] m ) {
        for (int i=0; i<m.length; i++)
            for (int j=0; j<i; j++)
                exchange( m, i, j );
    }
    public static void exchange(int[][] m, int a, int b) {
        int aux = m[a][b]; m[a][b] = m[b][a]; m[b][a] = aux;
    }
}
```

Solution:

a) It shows on the screen:

```
  1    0    0
  2    1    0
  2    2    1
```

It performs matrix transpose

b) The matrix will remain the same, since transpose is done twice