COMILLAS
UNIVERSIDAD PONTIFICIA

Mayo 2019

Programación Orientada a Objetos

- Convocatoria Ordinaria -

3h

Apellidos:_____ Nombre:_____

JIoT

Se desea desarrollar un programa Java que gestione una plataforma IoT de sensores de una casa.

En la plataforma existen varios tipos de sensores (temperatura, humedad y movimiento) los cuales poseen un identificador único alfanumérico ("TEMP01", "HUM04", …), una ubicación de donde se encuentran ("Salón", "Habitación 1", …), un valor (entero en el caso de la humedad, float en el caso de la temperatura y un valor sí/no en el caso de movimiento) y la unidad en la que se representa su valor ("ºC", "%" y "").

La información de los sensores se leerá cada minuto (ver clases que se suministran) y se mostrará en una ventana de monitorización similar a la que aparece en la Figura 1. Las ventanas de monitorización pueden suscribirse a los sensores mediante la caja de texto y el botón al efecto (escribiendo el identificador del sensor) de tal forma que cada ventana mostrará solo la información de los sensores que desee. Una ventana puede estar suscrita a más de un sensor. Además, la información de un sensor puede ser leída por más de una ventana de monitorización de forma simultánea. A medida que lleguen nuevos datos de los sensores, se añadirán al final de la lista (JTextArea y se recuerda que se deberán utilizar los métodos getText() y setText() de esa clase) de la ventana que corresponda.

La información de los sensores se encuentra en el fichero ("sensores.obj"), sin embargo, sus valores (22.1 en el caso de TEMP01) serán almacenados por un programa externo de forma temporal en un fichero de texto llamado identificador.txt ("TEMP01.txt"). Este fichero solo almacenará el valor (pe, 22.1). A medida que los sensores generen un nuevo dato se guardará en el fichero escribiendo una línea al final del mismo. La frecuencia de actualización de este fichero será también de un minuto. Como se ha destacado, la acción de escritura de los valores de los sensores en el fichero identificador.txt será realizado por un programa externo, por lo que no deberá ser codificado.

Es importante saber, para no tener problemas de bloqueos y sincronismo, que el fichero de almacenamiento de los valores del sensor **solo podrá ser abierto una vez** por uno de los objetos/procesos del programa. Es decir, no podremos tener el mismo fichero abierto por varios objetos/procesos simultáneamente.

Antes de mostrar la información en la ventana, como los sensores son susceptibles de tener mucho ruido, se deberá validar y normalizar la información que van a mostrar. La validación consiste en verificar que los valores de los sensores están dentro de un rango (temperatura entre -30 y 60, humedad entre 0 y 100 y movimiento los valores discretos sí/no). La normalización consiste en que cada valor de un sensor no puede tener una variación superior del ±10% respecto al anterior. En ese caso, ese valor ni se almacenará ni se visualizará. En el caso de los sensores de movimiento siempre se dará el valor por válido siempre que coincida con los valores sí/no.

Al finalizar el programa (cierre de la ventana, cualquiera) se deberá guardar toda la información de los sensores en un fichero llamado ("sensoresValores.obj") junto con los valores y la hora (ver clase Time) en la que se leyeron para cada sensor. Este fichero deberá estar ordenado por la hora y el identificador del sensor.

La lectura y escritura de objetos se realizará mediante la clase que se suministra IOSensores.

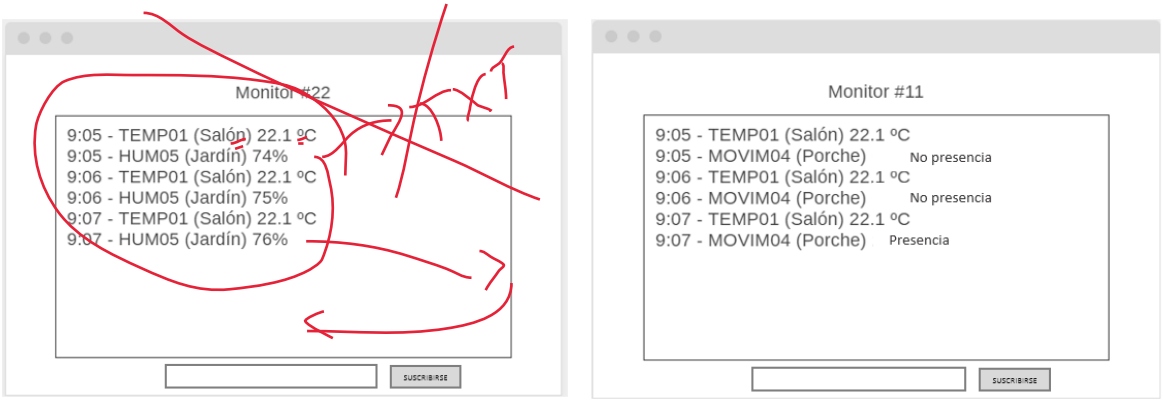

Figura 1. Ejemplo de ejecución del programa

```
public class Time implements Comparable

public class Util
     public void wait(byte seconds)
                          throws InterruptedException

public class IOSensores
     public static void setFile(String file)
     public static void writeObjects(Collection)
```

```
public static Collection readObjects()
```

Nota:

- Se deberán seguir las normas de diseño y programación vistas en clase.
- El examen se realizará dentro del paquete “examen”.
- No se podrán utilizar la estructura *foreach* de la versión 1.5 ni colecciones tipificadas, por lo que habrá que recorrer las colecciones con Iterators.
- Se podrán utilizar imports con comodines (*).
- Sólo se implementarán los métodos que se vayan a utilizar.