FUNDAMENTOS DE INFORMÁTICA

PRÁCTICA

CURSO 2019/2020

Contenido

1 INTRODUCCIÓN

Esta práctica se realizará en Java, un lenguaje orientado a objetos. El primer concepto que se debe conocer para programar dentro del paradigma de la orientación a objetos es el de 'objeto'. En este paradigma todo está basado en los objetos, que toman vida cuando nuestro programa está en ejecución; por tanto, un programa Java se organizará en objetos que además podrán relacionarse de diferentes maneras entre sí.

Las relaciones entre objetos proporcionan al programador las funcionalidades necesarias para resolver una amplia gama de problemas. La definición de un objeto viene dada por lo que se conoce como 'clase', que es lo que modelaremos y programaremos, aquello que escribiremos definiendo campos y métodos. Estos campos y métodos nos permitirán añadir funcionalidad a la clase y, una vez definida, ya podremos crear objetos (o instancias de clase) que son los que existirán cuando nuestro programa se ejecute.

Por tanto, podemos ver los objetos como los componentes que nos servirán para diseñar nuestro programa, y las clases como las definiciones de dichos componentes. Una misma clase puede, por tanto, dar lugar a muchos objetos diferentes, cada uno de ellos con características diferentes que vendrán dadas por los valores particulares de sus campos.

2 DESCRIPCIÓN GENERAL DE LA PRÁCTICA

El objetivo de esta práctica es que el alumno sea capaz de crear un sistema informático simplificado para el registro y control de mediciones de variables atmosféricas y concentración de gases de efecto invernadero en diferentes estaciones meteorológicas distribuidas por el continente europeo.

La aplicación que se pide desarrollar deberá permitir el acceso a los registros de mediciones de temperatura (T), precipitaciones (P) y valores de concentración de óxido nitroso (N2O) tomados en diferentes instantes y en cada una de las diferentes estaciones meteorológicas.

Las estaciones meteorológicas disponen de un código de identificación y se caracterizan por tener una localización concreta, expresada por medio de dos coordenadas angulares (latitud y longitud), junto con otro valor *Altitud* que representa la altura al nivel del mar a la que se encuentra la estación. Cada estación meteorológica dispondrá (o podrá disponer) de un instrumento para medir la temperatura (termómetro), otro para medir las precipitaciones (pluviómetro) y otro de medición de concentración de N2O, y deberá ser capaz de almacenar junto al valor medido de las variables atmosféricas (temperatura, precipitaciones) y concentración de gases (concentración de N2O), el instante en el que se realizó la medición.

Cada uno de los instrumentos de medición tendrá un código de identificación y guardará las mediciones realizadas, así como el último valor que haya registrado. Además, cada uno de ellos tendrá establecido un valor máximo a partir del cual saltaría una alarma en el instrumento; es decir, que el propio instrumento de medición deberá lanzar un aviso en forma de alarma si, por ejemplo, el termómetro contenido en una determinada estación meteorológica registra valores cercanos o superiores al valor máximo de temperatura establecido según el caso. De un modo análogo se tratarían las precipitaciones y concentraciones de N2O.

> **Nota**: En esta práctica vamos a simular el uso de instrumentos de medición reales, ya sea de temperatura, precipitaciones o de concentración de gases. Por ello, cuando hablemos de "tomar una medición" en realidad lo que vamos a hacer es "crear" una medición; es decir, que vamos a establecer nosotros mismos el valor (de temperatura, precipitaciones o de concentración de N2O) de la medición.
>
> El enunciado de la práctica se acompaña de código (un método *main()* de una clase Lanzador, que se verá en detalle más adelante) que servirá para validar en cada parte del enunciado si la solución a la práctica que se está dado es correcta o no. En este *main()* es donde se crearán las mediciones simulando que se las pedimos a un instrumento de medición.

El sistema informático que desarrollaremos deberá ser capaz también de ofrecer las siguientes funcionalidades:

- Consultar la secuencia de los últimos *n* valores registrados de T, P o concentración de N2O en una estación meteorológica dada, indicando junto con el valor el momento exacto en que se registró la medida.
- Calcular el valor medio de las últimas *n* mediciones de T, P o concentración de N2O en una estación meteorológica dada.
- Consultar el valor más alto de T, P o concentración de N2O en una estación meteorológica dada.
- Lanzar el aviso cuando el último valor medido supere el valor máximo determinado (o se acerque a él, según el caso) para T, P o concentración de N2O.

La forma de desarrollar la práctica será partiendo de un enfoque básico, para ir avanzando progresivamente y de forma iterativa incluyendo nuevas características y funcionalidades. En cada etapa se partirá inicialmente del diseño y desarrollo de la etapa anterior, pero habrá que incluir nuevas funcionalidades, poniendo en juego los conceptos del temario que se van estudiando progresivamente. Es posible que en determinadas etapas haya que modificar el diseño previo si éste no cubre la nueva especificación, o si en ese momento ya hemos adquirido determinados conceptos que nos permitan mejorar nuestro código.

RECOMENDACIÓN: la práctica se realiza a lo largo del cuatrimestre y forma parte del estudio de la asignatura. Para cada etapa el tutor indicará las correspondientes fechas de entrega.

Es conveniente hacer una primera lectura completa de este documento para tener una visión global de lo que se pide, y organizar apropiadamente el estudio de la asignatura.

3 ETAPA 1

3.1 ETAPA 1.1: ORIENTACIÓN A OBJETOS

3.1.1 OBJETIVOS

En la primera etapa de la práctica se van a afianzar los conceptos básicos de la programación en Java. Trabajaremos con las características fundamentales de **clase, objeto, campos y métodos** (constructores, métodos que ayudan a la encapsulación de la clase y métodos de visualización).

Veámoslo con un ejemplo que nos permita comenzar esta primera etapa de la práctica. En primer lugar, vamos a necesitar considerar información acerca de localizaciones y mediciones de diferentes tipos; en otras palabras, cada localización y medición será un objeto de la clase correspondiente. Las localizaciones y las mediciones de diferentes tipos tendrán unas características comunes (campos de clase), pero cada una podrá ser diferente si los valores de dichas características son diferentes (los valores que toman los campos en un objeto). Lo veremos más en detalle un poco más adelante.

3.1.2 REQUISITOS PREVIOS

Para la realización de esta primera etapa se requiere haber estudiado los temas 4, 5 y 6 del temario detallado de la asignatura, correspondientes a los capítulos 1, 2 y 3, así como los apéndices B, C, D y F del libro de referencia de la Unidad Didáctica II.

3.1.3 ENUNCIADO: CREACIÓN DE LAS CLASES *LOCALIZACIÓN* Y *MEDICIÓN*

Se desea diseñar un sistema informático cuyo fin sea el registro y control de mediciones de variables atmosféricas (T y P), así como de emisiones de un determinado tipo de gas de efecto invernadero (N2O) en diferentes estaciones meteorológicas distribuidas por el continente europeo.

Dentro de esta primera parte de la práctica se realizará una aproximación muy básica al modelado inicial de algunos de los elementos del sistema, que posiblemente tengamos que ir modificando según vayamos avanzando en los contenidos del curso, y según vayamos avanzando en el desarrollo de las diferentes partes de esta práctica. Los elementos básicos de nuestro sistema en este punto del desarrollo serán las localizaciones y los diferentes tipos de mediciones que estamos considerando.

Una localización indica dónde se encuentra situada una estación meteorológica; por tanto, en primer lugar, tendremos que definir el concepto localización, y para ello vamos a crear una clase con ese nombre: *Localizacion*. Toda localización tendrá las siguientes características (campos o atributos):

- *Nombre* que identifique el punto de medición. Considerar un tipo de dato String.
- *Latitud*; expresada en forma decimal, para lo que usaremos un tipo de dato double.
- *Longitud*. Considerar también un tipo de dato double.
- *Altitud*. Considerar también un tipo de dato double.

Debemos tener en cuenta que todo objeto de esta clase recién creada deberá permitir consultar y modificar estos campos a través de una serie de métodos creados a tal efecto. Por tanto, por cada campo deberemos contar con dos métodos: uno para consultar su valor y otro para modificarlo. Por ejemplo, para el campo *longitud* tendríamos los métodos *getLongitud() y setLongitud()*.

Ahora, de un modo análogo al caso de las localizaciones, tendremos que modelar los conceptos correspondientes a los diferentes tipos de mediciones que estamos considerando: el concepto de medición de temperatura, de cantidad de precipitaciones, así como de concentración de N2O. Para ello tendremos que definir el siguiente conjunto de clases: *MedicionTemperatura, MedicionPrecipitaciones* y *ConcentracionN2O.*

Cada una de esto tipos de mediciones tendrá las siguientes características; es decir, deberá tener los siguientes campos o atributos:

- Identificador de la medición. Puede usarse un tipo de dato *int*.
- *Unidad de medida*, que será una constante de tipo *String* con valor:
  - *" ºC "* en el caso de la clase *MedicionTemperatura*.
  - *" Litros_m2 "* en el caso de *MedicionPrecipitaciones,* que representa la unidad *Litros/m^2*.
  - *" mg_m3 "* en el caso de *ConcentracionN2O,* que representa la unidad *mg/m^3*.

  Nota: Si queremos usar el concepto de constante en Java lo modelaremos como un atributo de clase, de modo que su valor será compartido por todos los objetos de la clase. Las constantes en Java se definen como un atributo definido como *static* y con el modificador *final* del siguiente modo:

  ```
  static final String nombreConstante = valor;
  ```

- *Momento en el que se ha realizado la medición*. En este caso debemos definir un atributo de la clase *Date*.

  Nota: La clase *Date* viene incluida en Java, pero debemos importarla en nuestro código. *Date* se localiza en el paquete java.util, por lo que al inicio del programa se deberá escribir lo siguiente para importarlo:

  ```
  import java.util.Date;
  ```

  Para establecer el momento en el que realizamos una medición habrá que crear un objeto de la clase *Date* de Java del siguiente modo (ponemos un ejemplo de cómo habría que recoger el instante actual –al que llamaremos en el ejemplo "*ahora"*– e imprimirlo por pantalla):

  ```
  Date ahora = new Date();

  System.out.println("Hora y fecha actual: " + ahora.toString());
  ```

- *Valor medido*. Considerar un tipo de dato *double*.

Análogamente a lo que se hizo en el caso de la clase *Localizacion*, habrá que definir también métodos de acceso y modificación para los campos de cada una de las clases que acabamos de definir: *MedicionTemperatura, MedicionPrecipitaciones* y *ConcentracionN2O.*

Por otro lado, cada una de estas clases deberá tener también un método llamado *print()* que devuelva la información de la localización (nombre, latitud, longitud y altitud) y de la medición realizada (id, unidades de medida, valor medido, así como fecha y hora) con un formato legible como, por ejemplo, los mostrados en las Figura 1 y 2. Para ello el método deberá devolver un String donde se respete el formato mostrado en las Figura 1 y 2 (considerar "\n" para producir los saltos de línea).

```
Longitud: <valor_de_longitud>
Latitud: <valor_de_latitud>
Altitud: <valor_de_altitud>

Nombre del punto de localización: <nombre_pto_localizacion>
```

Figura 1: Formato de salida por pantalla con la información básica de una localización.

```
Valor medido de T: <valor_medido>
Unidad de medida: ºC

Instante de la medición: <fecha_y_hora_de_la_medicion>
```

Figura 2: Formato de salida por pantalla con la información básica de una medición (en este caso de una medición de temperatura). Nótese que los diferentes tipos de mediciones encajan todos ellos con este esquema.

Los elementos entre '<' y '>' son los valores correspondientes a los campos referidos arriba con dicho nombre.

Nota: Más adelante, y según vayamos adquiriendo los conceptos y mecanismos de la Programación Orientada a Objetos, veremos que es posible agrupar los diferentes tipos de mediciones en una clase que represente una medición general.

RECOMENDACIÓN: Experimentar con BlueJ creando diferentes localizaciones y diferentes tipos de mediciones, modificando sus datos, escribiéndolos por pantalla, etc.

¿Cuántas localizaciones puedes crear? ¿Y mediciones de temperatura? ¿Y de concentración de CO2?

¿Cómo puedes acceder a ellas en BlueJ?

3.2 ETAPA 1.2: EJECUCIÓN DE APLICACIONES

3.2.1 OBJETIVOS

Es importante comprender que Java y BlueJ son cosas diferentes. Java es un lenguaje de programación, mientras que BlueJ es un entorno de desarrollo que nos permite programar en Java. BlueJ está pensado para el aprendizaje del lenguaje Java y proporciona distintas herramientas que permiten inspeccionar las partes de un programa, tanto en la parte de diseño (cuando estamos escribiendo el código), como en la de ejecución (cuando el código se ejecuta, se crean los objetos en memoria, etc.)

Una de las herramientas de BlueJ que pueden resultar más útiles, pero a la vez pueden despistar más, es el *banco de objetos* u *object bench* (capítulo 1 del libro; se muestra un ejemplo en la Figura 3). En conjunto con otras herramientas como el *inspector de objetos*, resulta muy interesante para experimentar con los objetos en tiempo de ejecución de forma interactiva. BlueJ nos permite ejecutar nuestras aplicaciones mediante la invocación de los métodos de sus objetos, pero no es esta la única manera de hacerlo.

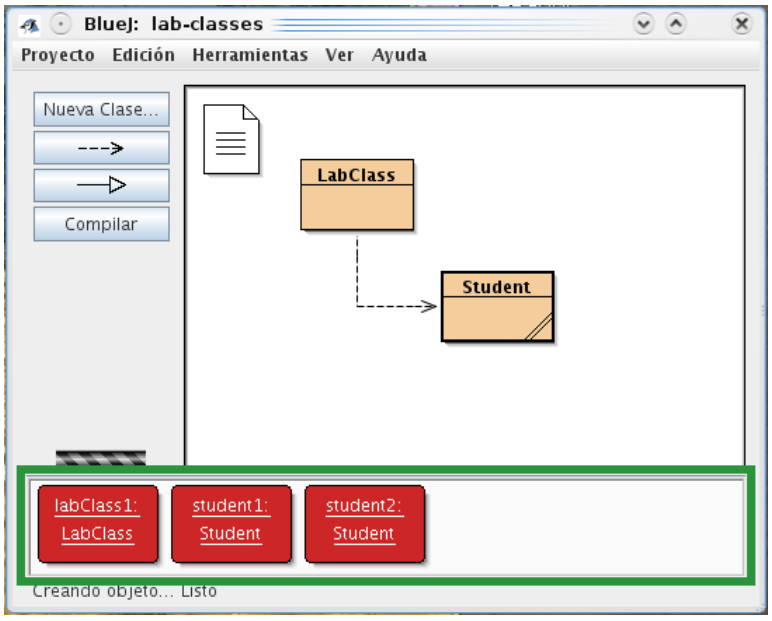

Figura 3: Banco de objetos (recuadrado en verde) en BlueJ

A estas alturas sabemos que cada clase dispone de una serie de métodos y campos que son propios de los objetos que creamos a partir de dicha clase; es decir, que para poderlos utilizar necesitamos crear primero un objeto de esa clase. Por ejemplo, la clase *Localizacion* que hemos creado tiene un método *print()* para mostrar por pantalla su información básica, que sólo podrá ser invocado una vez tengamos algún objeto de esta clase *Localizacion* (y que imprimirá por pantalla, en este caso, el valor concreto de los campos del objeto de esa clase que hayamos creado, tal como se muestra en la Figura 1). Pues bien, en Java existe otro tipo de métodos, llamados métodos de clase o métodos estáticos, que no son propios de los objetos sino de las clases. De esta forma, no necesitamos una instancia particular de la clase (objeto) para invocarlos.

Java nos permite ejecutar los programas que creamos de forma independiente a BlueJ. La ejecución fuera de BlueJ comienza sin que exista ningún objeto (recordemos que en BlueJ primero se crean objetos que se almacenan en el banco de objetos para posteriormente invocar sus métodos). Entonces, antes de tener objetos, lo único que se tiene son las clases que los definen, por lo que deberemos usar un método de clase (de los que hablábamos en el párrafo anterior) para poder iniciar la ejecución.

Un caso particular de estos métodos de clase es el método *main()*, que es un método especial ya que es el que se utiliza como punto de partida para iniciar la ejecución de todo programa en Java (véase el apartado 3.2.2).

RECOMENDACIÓN: Al final de este apartado se debería entender la diferencia entre ejecutar un programa en BlueJ y hacerlo de forma independiente, esto es, por ejemplo, ejecutarlo desde la línea de comandos (símbolo de sistema en Windows/Linux/MacOS) por medio de la llamada a un método de clase *main()*.

3.2.2 REQUISITOS PREVIOS

Además de los expuestos en la sección 3.1.2, debe consultarse el libro de texto, capítulos 6.15 y 6.16, sobre métodos de clase. En el apéndice E del libro de texto puede encontrar más información sobre el método *main()*; en concreto, en el apartado E1. Puede consultar también el capítulo 3.10 (Objetos que crean objetos) del libro de texto si necesita más información acerca de la creación de objetos desde el código.

3.2.3 ENUNCIADO: JAVA, BLUEJ Y EL MÉTODO MAIN

Para ejecutar nuestro programa de manera independiente de BlueJ y crear el flujo principal del sistema, crearemos una clase llamada *Lanzador*, donde implementaremos el método *main*(). Esta clase contendrá el código con el que se crean las instancias de objetos necesarias y las llamadas para poder comprobar el correcto funcionamiento del sistema.

El ejercicio a realizar en este punto consiste en crear dicha clase Lanzador con un método *main*() e implementar en este método el siguiente código[1]:

```
        public static void main(String[] args) {

        // Creamos una localización en Madrid y otra en Barcelona
        Localizacion madrid = new Localizacion("Madrid",40.4165000,-3.7025600, 667.0);
        Localizacion barcelona = new Localizacion("Barcelona",41.3818,2.1685, 13.0);

        // Modificamos los campos de las localizaciones e imprimimos los nuevos valores
        madrid.setNombre("Madrid-Barajas");
        madrid.setLatitud(40.4918100);
        madrid.setLongitud(-3.5694800);
        System.out.println("La nueva localización de la estación meteorológica en Madrid se llama "
+ madrid.getNombre() + ", con latitud: " + madrid.getLatitud() + " , longitud: " +
madrid.getLongitud() + " y altitud: " + madrid.getAltitud());

        barcelona.setNombre("Barcelona-Sans");
        barcelona.setLatitud(41.3726311);
        barcelona.setLongitud(2.1545999);
        System.out.println("La nueva localización de Barcelona es: " + "\n"+ barcelona.print());


        // Creamos mediciones en un momento actual e imprimimos los valores medidos en cada caso
        Date momentoActual = new Date();
        MedicionTemperatura temp1 = new MedicionTemperatura(1,18.34,momentoActual);
        System.out.println("La medición de temperatura tomada en este momento es " + temp1.print());

        MedicionPrecipitaciones prec1 = new MedicionPrecipitaciones (2,0.55,momentoActual);
        System.out.println("La medición de precipitaciones tomada en este momento es " +
prec1.print());

        ConcentracionN2O concent1 = new ConcentracionN2O (3,0.67,momentoActual);
        System.out.println("La medición de concentración de N2O tomada en este momento es " +
concent1.print());

        }
```

[1] En el curso virtual, junto con el enunciado de la práctica se pueden descargar este método *main()* de la clase *Lanzador*.

Este código debería de hacer lo siguiente: en primer lugar, crear dos instancias (objetos) de la clase Localización; a continuación, modificar los datos correspondientes a ambas localizaciones y mostrar los nuevos datos por pantalla; finalmente, crear tres mediciones (una de temperatura, otra de precipitaciones y la última de concentración de N2O) y mostrar su información por pantalla.

Dentro de este *main()* pruebe ahora a escribir el código necesario para crear una nueva localización con diferentes valores y pruebe a acceder a éstos mediante los métodos pertinentes. Muestre a continuación su información básica por pantalla.

Ejecute el programa desde BlueJ y desde la línea de comandos (consola o símbolo de sistema, según el sistema operativo que utilice). En el caso de los diferentes tipos de mediciones podemos hacer lo mismo.

IMPORTANTE: La presente práctica debe poderse ejecutar independientemente de BlueJ de cara a su entrega y corrección.

4 ETAPA 2

4.1 ETAPA 2.1: USO DE LA COMPOSICIÓN

4.1.1 OBJETIVOS

Los objetos pueden contener otros objetos y utilizarlos para realizar distintas tareas. De esta forma, cuando un objeto contiene otros, hablamos del mecanismo de ***Composición***. En Java existen además estructuras de datos que nos permiten agrupar objetos, como es el caso de las listas, los conjuntos o los mapas. De esta forma si quisiéramos representar una Universidad con personas, podríamos, por ejemplo, crear una clase *Universidad* que tuviera un campo de tipo de lista de objetos de la clase *Persona*, como se muestra en la Figura 4. Tendríamos así que una Universidad se compone de personas.

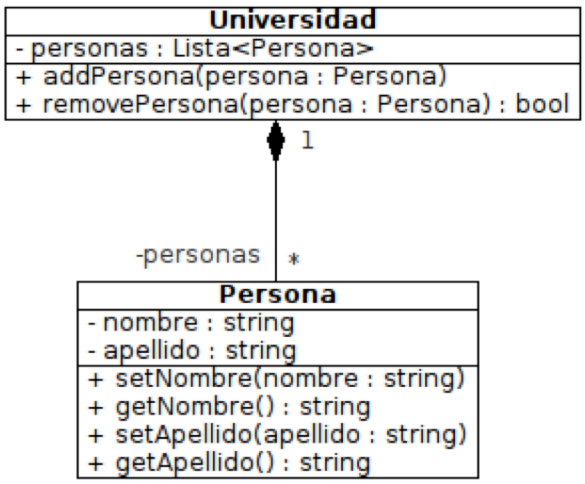

Figura 4: Diagrama de clases correspondiente al ejemplo de la Universidad (se han omitido los constructores)

Por otro lado, no hay que olvidar que, a medida que la aplicación que estamos construyendo se hace más grande y compleja, conviene ir documentando las clases que hemos creado, no sólo para que terceras personas puedan comprender su funcionalidad sin tener que leer minuciosamente el código, sino para que nosotros mismos podamos en un futuro reutilizarlas de una forma sencilla. Para la generación de documentación en Java disponemos de la herramienta *Javadoc*, que por medio de determinado tipo de comentarios que podemos incluir en nuestro código, permite la generación posterior de documentación a partir del mismo.

4.1.2 REQUISITOS PREVIOS

Esta etapa requiere haber estudiado los temas 7, 8 y 9 del temario detallado incluido en la guía de estudio de la asignatura. Encontrará información más detallada sobre el mecanismo de composición y la generación de documentación en el libro de texto de la asignatura en los capítulos 4, 5 y 6. La lectura de las secciones comprendidas entre la 8.3 y la 8.6 del libro, ambas incluidas, sobre buenas prácticas en el diseño de aplicaciones es también recomendable.

4.1.3 ENUNCIADO: CREACIÓN DE LAS CLASES CORRESPONDIENTES A LOS DIFERENTES *INSTRUMENTOS DE MEDICIÓN Y DE LA CLASE ESTACIÓN METEOROLÓGICA*

RECOMENDACIÓN: leer primero los requisitos completos antes de tomar decisiones acerca de cómo desarrollar la solución.

En este punto de la práctica se deberán crear diferentes clases para representar cada uno de los diferentes instrumentos de medición que vamos a considerar dentro de nuestro sistema, y que se añadirán a una estación meteorológica.

Los instrumentos que vamos a modelar son los siguientes: para la medición de temperatura, Termómetro; para la medición de precipitaciones, Pluviómetro; y para la medición de la concentración de N2O, un detector específico que podemos llamar DetectorN2O.

Por tanto, cada uno de estos instrumentos deberá modelarse con una clase diferente (*Termómetro*, *Pluviómetro* y *DetectorN2O*), y todas ellas deberán cumplir las siguientes condiciones:

1. Cada clase deberá tener un código de identificación del instrumento. Considerar un tipo de dato *int*.

2. Cada instrumento deberá ser capaz de guardar el conjunto de mediciones registradas, correspondiente a la variable atmosférica o concentración de gas correspondiente, así como el valor correspondiente a la última medición realizada.

   Nota: por simplicidad, y como ya se ha avanzado, la obtención de las mediciones en los diferentes instrumentos se hará directamente en el método *main*() de la clase *Lanzador* con el que se va probando el desarrollo de la práctica en cada apartado.

Además, cada uno de estos instrumentos de medición tiene características propias que deberán tenerse en cuenta en el modelado de las clases:

- *Termometro* tendrá un *valor máximo de temperatura* (se expresará en grados centígrados) que establecerá a partir de qué temperatura se lanzará un aviso; es decir, que cuando se actualice la última medición de temperatura en el termómetro, si el valor de temperatura de ésta supera el 90% del *valor máximo de temperatura* establecido*,* el termómetro emitirá un aviso en forma de mensaje por pantalla indicando que se ha superado la temperatura máxima.

- *Pluviometro* tendrá un *valor máximo de precipitaciones* (se expresará en litros/m2) que establecerá a partir de qué valor de precipitaciones se lanzará un aviso; es decir, que cuando se actualice la última medición de precipitaciones en el pluviómetro, si el valor de éstas en la última medición supera el 80% de este *valor máximo de precipitaciones,* el pluviómetro emitirá un aviso en forma de mensaje por pantalla indicando que se ha superado el valor máxima de precipitaciones establecido.

- *DetectorN2O,* con un *valor máximo de concentración de N2O* (se expresará en miligramos/m3) a partir del cual el instrumento lanzará un aviso; es decir, que cuando se actualice la última medición de concentración de N2O, si el valor de esta concentración es superior al *valor máximo de concentración de N2O* establecido, el detector emitirá un aviso en forma de mensaje por pantalla indicando que se ha superado el valor máximo.

Análogamente a lo que se hizo en la definición de las clases anteriores, habrá que definir los métodos de acceso y modificación para los campos de cada una de las clases que acabamos de definir. En este punto es importante pensar en qué método habrá que comprobar la condición de superación (o de que el valor se acerca a un determinado porcentaje) del valor máximo de temperatura, precipitaciones o concentración de N2O que lanza el aviso correspondiente en cada instrumento, una vez establecido como requisito que el aviso se deberá lanzar si, al actualiza la última medición dentro del instrumento, se cumple una determinada condición (diferente en cada caso) en el valor de la medida.

A continuación debe crearse una nueva clase, que llamaremos *EstacionMeteorologica,* y que representará a cada una de las estaciones meteorológicas que se vayan a considerar dentro del sistema que queremos desarrollar. Las características de cada una de estas estaciones meteorológicas son las siguientes:

1. Deberá tener un código *id* que la identifique. Considerar un tipo de dato *int*.

2. Deberá contener una localización que indique el punto en el que se encuentra situada la estación.

3. Deberá contener un único instrumento de medición de cada tipo: un termómetro, un pluviómetro y un detector de N2O; y deberían poder modificarse, es decir, que si tenemos una estación con un termómetro deberíamos poder cambiarlo por otro termómetro en cualquier momento, manteniéndose la restricción de que solo podemos tener un instrumento de cada tipo dentro de una estación.

4. Deberá poder almacenar el conjunto de mediciones de cada uno de los tipos que estamos considerando. Consecuentemente deberán poderse añadir y eliminar mediciones de cada tipo. El motivo de querer guardar las mediciones en la propia estación, aparte de que cada instrumento guarde las mediciones que realice, es permitir el acceso al histórico de todas las mediciones realizadas en la estación meteorológica, independientemente de que éstas se hayan tomado con un instrumento u otro.

5. Deberá poder presentar por pantalla la información de todas las mediciones de cada tipo; es decir, listar el conjunto de valores medidos con sus unidades (de temperatura, precipitaciones o concentración de N2O), junto con el instante en el que se realizó la medición.

6. Deberá poder presentar por pantalla el valor más alto registrado en cada una de las variables atmosféricas (T y P) y de concentración de N2O.

De nuevo, deberán definirse los métodos de acceso y modificación para los campos de esta clase.

Documente las clases que ha creado hasta ahora. En el apartado 6.11.2 del libro de texto se detalla la mínima documentación que se debería incluir.

A continuación, vamos a modificar el método *main*() de la clase *Lanzador*. Al ejecutar este código, el sistema debería:

1. Crear una localización correspondiente con las coordenadas geográficas de Madrid y crear a partir de ella una estación meteorológica. A continuación, crear un termómetro, un pluviómetro y un detector de concentración de N2O y añadirlo a la estación. Tomar mediciones de temperatura, precipitaciones y concentración de N2O en tres momentos diferentes y añadirlas también a cada instrumento de medición y a la propia estación meteorológica.

2. Actualizar la última medición en cada uno de los tres instrumentos.

3. Lanzar el aviso de que se ha superado el nivel máximo de concentración de N2O, ya que la última medición de concentración de N2O era de 5.82 mg/m3, un valor que es superior al *valor máximo de concentración de N2O* que se había fijado en 5 mg/m3 al crear el detector de N2O.

4. Presentar por pantalla los valores más altos registrados para la temperatura, precipitaciones y concentración de N2O.

Código a implementar en el main() de la clase Lanzador:

```
        // Creamos una localización en Madrid
        Localizacion madrid = new Localizacion("Madrid",40.4165000,-3.7025600, 667);

        // Creamos una estación meteorológica en la localización creada anteriormente
        EstacionMeteorologica estacionMadrid = new EstacionMeteorologica(1,madrid);

        // Creamos tres instrumentos de medición, uno de cada tipo
        Termometro termometro1 = new Termometro(1,40);
        Pluviometro pluviometro1 = new Pluviometro(2,70);
        DetectorN2O detectorN2O1 = new DetectorN2O (3,5);

        // Añadimos a la estación los tres instrumentos de medición creados anteriormente
        estacionMadrid.setTermometro(termometro1);
        estacionMadrid.setPluviometro(pluviometro1);
        estacionMadrid.setDetectorN2O(detectorN2O1);

        // Tomamos mediciones en un momento t0
        Date momentoActual = new Date();
        MedicionTemperatura temp1 = new MedicionTemperatura(4,19.14,momentoActual);
        MedicionPrecipitaciones prec1 = new MedicionPrecipitaciones (5,56,momentoActual);
        ConcentracionN2O concent1 = new ConcentracionN2O (6,4.79,momentoActual);

        // Tomamos mediciones en un momento t1, añadiendo 1 segundo (1000ms) al momentoActual
        long momentoActualEnMilisegundos = momentoActual.getTime();
        Date momentoSig = new Date(momentoActualEnMilisegundos+1000);

        MedicionTemperatura temp2 = new MedicionTemperatura(7,19.44,momentoSig);
        MedicionPrecipitaciones prec2 = new MedicionPrecipitaciones (8,60,momentoSig);
        ConcentracionN2O concent2 = new ConcentracionN2O (9,4.90,momentoSig);

        // Creamos mediciones en un momento t2, añadiendo 2 segundos (2000ms) al momentoActual
        Date momentoFinal = new Date(momentoActualEnMilisegundos+2000);
        MedicionTemperatura temp3 = new MedicionTemperatura(10,38.54,momentoFinal);
        MedicionPrecipitaciones prec3 = new MedicionPrecipitaciones (11,72,momentoFinal);
        ConcentracionN2O concent3 = new ConcentracionN2O (12,5.82,momentoFinal);


        // Añadimos a la estación el conjunto de mediciones que han sido tomadas, actualizando las
últimas mediciones en cada instrumento
        estacionMadrid.getTermometro().addMedicion(temp1);
        estacionMadrid.getTermometro().setValorUltimo(temp1.getValorMedido());
        estacionMadrid.addTemp(temp1);
        estacionMadrid.getTermometro().addMedicion(temp2);
        estacionMadrid.getTermometro().setValorUltimo(temp2.getValorMedido());
        estacionMadrid.addTemp(temp2);
        estacionMadrid.getTermometro().addMedicion(temp3);
        estacionMadrid.getTermometro().setValorUltimo(temp3.getValorMedido());
        estacionMadrid.addTemp(temp3);

        estacionMadrid.getPluviometro().addMedicion(prec1);
        estacionMadrid.getPluviometro().setValorUltimo(prec1.getValorMedido());
        estacionMadrid.addPrecip(prec1);
        estacionMadrid.getPluviometro().addMedicion(prec2);
        estacionMadrid.getPluviometro().setValorUltimo(prec2.getValorMedido());
        estacionMadrid.addPrecip(prec2);
        estacionMadrid.getPluviometro().addMedicion(prec3);
        estacionMadrid.getPluviometro().setValorUltimo(prec3.getValorMedido());
        estacionMadrid.addPrecip(prec3);

        estacionMadrid.getDetectorN2O().addMedicion(concent1);
        estacionMadrid.getDetectorN2O().setValorUltimo(concent1.getValorMedido());
        estacionMadrid.addConcentrN2O(concent1);
        estacionMadrid.getDetectorN2O().addMedicion(concent2);
        estacionMadrid.getDetectorN2O().setValorUltimo(concent2.getValorMedido());
        estacionMadrid.addConcentrN2O(concent2);
```

```
estacionMadrid.getDetectorN2O().addMedicion(concent3);
estacionMadrid.getDetectorN2O().setValorUltimo(concent3.getValorMedido());
estacionMadrid.addConcentrN2O(concent3);



// Presentamos por pantalla los valores más altos registrados para T, P y N2O
System.out.println("Valor máx de T en la estación: " + estacionMadrid.valorMasAltoT());
System.out.println("Valor máx de Precipitaciones: " + estacionMadrid.valorMasAltoP());
System.out.println("Valor máx de N2O: " + estacionMadrid.valorMasAltoN2O());
```

Es importante probar el sistema ante otras situaciones, con diferentes localizaciones, mediciones de T, P y N2O, etc. Cree mediciones que sobrepasen los valores de temperatura y precipitación a partir de los cuales se lanzan los avisos pertinentes y compruebe qué sucede cuando se superan los porcentajes sobre los valores máximos de T y P.

4.2 ETAPA 2.2: USO BÁSICO DE LA HERENCIA

4.2.1 OBJETIVOS

Hasta ahora hemos modelado los diferentes tipos de mediciones que estamos considerando, los diferentes tipos de instrumentos de medición que necesitamos, una localización y una estación meteorológica. Pero vemos que tenemos muchas clases y que, además, muchas de ellas comparten gran parte de sus métodos. Por tanto, el siguiente paso dentro de nuestro desarrollo será considerar clases más generales que engloben aquellas que comparten campos y métodos; en nuestro caso podremos hacerlo tanto con los diferentes tipos de mediciones como con los diferentes instrumentos de medición.

Tal como hemos definido los diferentes tipos de mediciones podemos observar que todas ellas comparten un conjunto de campos (el identificador, el valor medido y el momento en el que se realiza la medición), y que se diferencian únicamente en la unidad de medida, que además es un valor constante en cada caso.

Por otro lado, en el caso de los instrumentos de medición vemos también que todos ellos tienen una serie de campos comunes (el código de identificación y la última medición realizada*)*, pero además tienen campos y métodos diferentes. Un ejemplo es el campo correspondiente al *valor máximo de temperatura* en el caso del termómetro, el *valor máximo de precipitaciones* en el caso del pluviómetro, y el *valor máximo de concentración de N2O* en el caso del detector de N2O. También hemos visto que cada uno de los instrumentos de medición que hemos definido tiene unas condiciones diferentes a la hora de emitir avisos de alarma, lo que nos indica que tendrán métodos diferentes.

Para este tipo de situaciones, los lenguajes orientados a objetos como Java disponen de un mecanismo llamado **Herencia** que permite construir clases que "hereden" las características de una clase "padre" más general. Por tanto, la clase "heredera" o "hija" tendrá toda la funcionalidad de la clase de la que hereda, permitiendo además la inclusión de nuevos métodos y campos para extender dicha funcionalidad. De este modo la clase resultante será una extensión de la clase "padre", que aportará las nuevas funcionalidades que la clase extendida no contemplaba.

4.2.2 REQUISITOS PREVIOS

Esta etapa requiere haber estudiado los capítulos del libro base de las etapas anteriores, así como los temas 10, 11 y 12 del temario detallado de la asignatura, correspondientes a las Secciones de la 8.3 a la 8.6, y los Capítulos 9, 10 y 11 del libro base para la Unidad Didáctica II. En concreto la sección 10.7 y relacionadas, así como el capítulo 11 resultarán útiles en la resolución de esta etapa.

4.2.3 ENUNCIADO: GENERALIZANDO CLASES

En este punto de la práctica queremos tratar de generalizar las clases que hemos creado, tanto para el caso de los tipos de mediciones como de los diferentes tipos de instrumentos de medición.

En el primer caso se debe crear una clase que llamaremos *Medicion* a partir de las clases definidas previamente: *MedicionTemperatura, MedicionPrecipitaciones* y *ConcentracionN2O* que habíamos creado en el apartado 3.1.3. Esta clase *Medicion* deberá tener un campo correspondiente a la unidad de medida, que solo podrá tomar los siguiente valores dependiendo del tipo de medición: " ºC ", " l_m2 " y "mg_m3". El resto de campos son comunes en todas las clases que teníamos, por lo que podemos modelar todos los conceptos de medición de temperatura, de precipitaciones y de concentración de N2O como una sola clase *Medicion*, con la ayuda de una clase de tipo enumerado que llamaremos *UnidadMedida*.

Por otro lado, en el caso de los instrumentos de medición podemos definir una clase "padre", utilizando el mecanismo de la herencia, a la que llamaremos *Sensor*. Tal como hemos indicado en el punto 4.2.1., esta clase *Sensor* contendrá la parte común que compartan todas sus clases hijas, mientras que éstas mantendrán campos y métodos específicos de cada instrumento. Al crear esta clase *Sensor* evitamos duplicidades y favorecemos la reutilización de código. Sin embargo, no instanciaremos objetos de la clase Sensor, sino que lo seguiremos haciendo a partir de sus clase hijas (*Termómetro*, *Pluviómetro* y *DetectorN2O*). La razón es que un sensor solo puede ser de uno de los tipos definidos como clases hijas de *Sensor*.

Más adelante veremos las ventajas que nos ofrece el hecho de haber generalizado los instrumentos de medición.

Modifique el diseño, para introducir la clase Medición y Sensor, redefiniendo los campos y métodos correspondientes, (lea primero hasta el final de la sección para saber qué tienen que incluir las clases y los métodos)

A continuación, modificamos de nuevo el contenido del método *main()* de la clase *Lanzador* con el siguiente código con el que probamos el funcionamiento del sistema:

```
        // Creamos una localización en Madrid
        Localizacion madrid = new Localizacion("Madrid",40.4165000,-3.7025600, 667);

        // Creamos una estación meteorológica en la localización creada anteriormente
        EstacionMeteorologica estacionMadrid = new EstacionMeteorologica(1,madrid);

        // Creamos tres instrumentos de medición, uno de cada tipo
        Termometro termometro1 = new Termometro(4,25);
        Pluviometro pluviometro1 = new Pluviometro(5,65);
        DetectorN2O detectorN2O1 = new DetectorN2O (6,4.9);

        // Añadimos a la estación los tres instrumentos de medición creados anteriormente
        estacionMadrid.setTermometro(termometro1);
        estacionMadrid.setPluviometro(pluviometro1);
        estacionMadrid.setDetectorN2O(detectorN2O1);

        // Tomamos mediciones en un momento t0
        Date momentoActual = new Date();
        Medicion temp1 = new Medicion(13,20.24,momentoActual,UnidadMedida.C);
        Medicion prec1 = new Medicion(14,50,momentoActual,UnidadMedida.LITROS_M2);
        Medicion concent1 = new Medicion(15,4.79,momentoActual,UnidadMedida.MG_M3);

        // Tomamos mediciones en un momento t1, añadiendo 1 segundo (1000ms) al momentoActual
        long momentoActualEnMilisegundos = momentoActual.getTime();
        Date momentoSig = new Date(momentoActualEnMilisegundos+1000);

        Medicion temp2 = new Medicion(16,22.74,momentoSig,UnidadMedida.C);
        Medicion prec2 = new Medicion(17,60,momentoSig,UnidadMedida.LITROS_M2);
        Medicion concent2 = new Medicion(18,4.90,momentoSig,UnidadMedida.MG_M3);

        // Creamos mediciones en un momento t2, añadiendo 2 segundos (2000ms) al momentoActual
        Date momentoFinal = new Date(momentoActualEnMilisegundos+2000);
        Medicion temp3 = new Medicion(19,21.14,momentoFinal,UnidadMedida.C);
        Medicion prec3 = new Medicion(20,70,momentoFinal,UnidadMedida.LITROS_M2);
        Medicion concent3 = new Medicion(21,4.99,momentoFinal,UnidadMedida.MG_M3);


        // Añadimos a la estación el conjunto de mediciones que han sido tomadas, actualizando las
últimas mediciones en cada instrumento
        estacionMadrid.getTermometro().addMedicion(temp1);
        estacionMadrid.getTermometro().setValorUltimo(temp1.getValorMedido());
        estacionMadrid.addTemp(temp1);
        estacionMadrid.getTermometro().addMedicion(temp2);
        estacionMadrid.getTermometro().setValorUltimo(temp2.getValorMedido());
        estacionMadrid.addTemp(temp2);
        estacionMadrid.getTermometro().addMedicion(temp3);
        estacionMadrid.getTermometro().setValorUltimo(temp3.getValorMedido());
        estacionMadrid.addTemp(temp3);

        estacionMadrid.getPluviometro().addMedicion(prec1);
        estacionMadrid.getPluviometro().setValorUltimo(prec1.getValorMedido());
        estacionMadrid.addPrecip(prec1);
        estacionMadrid.getPluviometro().addMedicion(prec2);
        estacionMadrid.getPluviometro().setValorUltimo(prec2.getValorMedido());
```

```
estacionMadrid.addPrecip(prec2);
estacionMadrid.getPluviometro().addMedicion(prec3);
estacionMadrid.getPluviometro().setValorUltimo(prec3.getValorMedido());
estacionMadrid.addPrecip(prec3);

estacionMadrid.getDetectorN2O().addMedicion(concent1);
estacionMadrid.getDetectorN2O().setValorUltimo(concent1.getValorMedido());
estacionMadrid.addConcentrN2O(concent1);
estacionMadrid.getDetectorN2O().addMedicion(concent2);
estacionMadrid.getDetectorN2O().setValorUltimo(concent2.getValorMedido());
estacionMadrid.addConcentrN2O(concent2);
estacionMadrid.getDetectorN2O().addMedicion(concent3);
estacionMadrid.getDetectorN2O().setValorUltimo(concent3.getValorMedido());
estacionMadrid.addConcentrN2O(concent3);


// Presentamos por pantalla los valores más altos registrados para T, P y N2O
System.out.println("Valor máx de T en la estación: " + estacionMadrid.valorMasAltoT());
System.out.println("Valor máx de Precipitaciones: " + estacionMadrid.valorMasAltoP());
System.out.println("Valor máx de N2O: " + estacionMadrid.valorMasAltoN2O());
```

Al ejecutar el *main*() con este código, el sistema debería:

1. Crear una localización correspondiente con las coordenadas geográficas de Madrid y crear una estación meteorológica en dicha localización. Crear un nuevo termómetro, pluviómetro y detector de concentración de N2O y añadirlos a la estación. Tomar nuevas mediciones de temperatura, precipitaciones y concentración de N2O en tres momentos diferentes y añadirlas a cada instrumento de medición, así como a la estación meteorológica. Debe actualizarse el valor de la última medición en cada uno de los tres instrumentos.

2. Lanzar el aviso de que se ha superado el nivel máximo de temperatura, ya que la segunda medición era de 22.74 ºC, un valor que es superior al 90% del *valor máximo de temperatura* que se había fijado en 25 ºC al crear el termómetro.

3. Lanzar el aviso de que se ha superado el nivel máximo de concentración de N2O, ya que la última medición de concentración de N2O era de 4.99 mg/m3, un valor que es superior al *valor máximo de concentración de N2O* que se había fijado en 4.9 mg/m3 al crear el detector de N2O.

4. Presentar por pantalla los valores más altos registrados para la temperatura, precipitaciones y concentración de N2O.

De nuevo, como en el apartado anterior, es importante probar el sistema ante otras situaciones, con diferentes localizaciones, mediciones de T, P y N2O, etc. Cree mediciones que sobrepasen los valores máximos en los tres tipos de sensores y compruebe cómo responde el sistema.

5 ETAPA 3

5.1 ETAPA 3.1: USO AVANZADO DE LA HERENCIA Y POLIMORFISMO

5.1.1 OBJETIVOS

Ahora que tenemos creada una clase general que representa todos los instrumentos de medición (la clase *Sensor*), podemos hacer uso de otro mecanismo fundamental de la Programación Orientada a Objetos: el **Polimorfismo**. En Programación Orientada a Objetos el polimorfismo aparece en distintos contextos. Como la propia palabra indica, polimorfismo se refiere a que un elemento concreto puede tener varias formas. En el caso de las variables, el polimorfismo permite que una variable pueda contener objetos de diferentes tipos o clases, ya sea el tipo declarado o cualquier subtipo de éste.

Por ejemplo, una variable declarada de la forma:

```
Sensor s;
```

puede contener objetos no sólo de la clase *Sensor*, sino también objetos de cualquiera de las clases que hereden de *Sensor*, que en nuestra jerarquía serán clases hijas (subclases o subtipos) de Sensor; es decir: DetectorN2O, Termometro o Pluviometro. Si en un futuro se quisiera considerar un tipo nuevo de sensor podría hacerse creando simplemente una nueva clase hija de *Sensor*. De este modo, un objeto de la clase *Sensor* podría contener entonces objetos de esta nueva clase.

5.1.2 REQUISITOS PREVIOS

Esta etapa se basa en los mismos principios que la etapa 2.2, por lo que los temas que deben estudiarse son los mismos. Requiere, eso sí, un dominio mayor de los conceptos que se presentan, con el objetivo de profundizar en el uso del mecanismo de composición, herencia y polimorfismo para ampliar la aplicación que estamos desarrollando.

5.1.3 ENUNCIADO: AÑADIENDO SENSORES A UNA ESTACIÓN METEOROLÓGICA (SIN RESTRICCIONES) Y GESTIONANDO MÁS DE UNA ESTACIÓN METEORÓGICA.

Hasta ahora en nuestra aplicación se han usado tres tipos diferentes de instrumentos de medidas (sensores), y habíamos fijado que todos ellos debían ser incluidos en una estación meteorológica. En este momento se desea poder incluir los sensores pero eliminando todas las restricciones que habíamos considerado hasta este momento.

Se quiere diseñar el sistema pensando en la posibilidad de incluir en un futuro otros tipos de sensores más allá de los tres que hemos considerado hasta ahora y **que esto implique no modificar las clases existentes**.

Por último, se quiere poder calcular el valor más alto y el valor medio para una unidad de medida concreta, de todas las mediciones realizadas por todos los sensores de una *Estación Meteorológica*, para la unidad de medida concreta.

Para cubrir estas restricciones dentro de nuestro sistema, gracias al mecanismo del polimorfismo podemos modificar ahora la clase *EstacionMeteorologica*, de modo que se cubra completamente la nueva especificación.

Además, otro objetivo de este apartado es la gestión de un conjunto de puntos de medición, para lo que será necesario definir una clase *GestorMedioambiental* que podrá contener un conjunto de estaciones meteorológicas. Deberá contar con métodos que permitan calcular los valores medios y máximos registrados por todas las estaciones que gestiona, para una unidad de medida concreta, por todos los sensores.

Como prueba, debe implementar en la clase *Lanzador* las siguientes llamadas y comprobar el correcto funcionamiento del sistema:

```
// Tomamos mediciones en un momento t0
Date momentoActual = new Date();
Medicion temp1 = new Medicion(22,20.14,momentoActual,UnidadMedida.C);
Medicion prec1 = new Medicion(23,1.61,momentoActual,UnidadMedida.LITROS_M2);
Medicion concent1 = new Medicion(24,4.34,momentoActual,UnidadMedida.MG_M3);

// Tomamos mediciones en un momento t1, añadiendo 1 segundo (1000ms) al momentoActual
long momentoActualEnMilisegundos = momentoActual.getTime();
Date momentoSig = new Date(momentoActualEnMilisegundos+1000);

Medicion temp2 = new Medicion(25,22.74,momentoSig,UnidadMedida.C);
Medicion prec2 = new Medicion(26,1.43,momentoSig,UnidadMedida.LITROS_M2);
Medicion concent2 = new Medicion(27,4.50,momentoSig,UnidadMedida.MG_M3);

// Creamos mediciones en un momento t2, añadiendo 2 segundos (2000ms) al momentoActual
Date momentoFinal = new Date(momentoActualEnMilisegundos+2000);
Medicion temp3 = new Medicion(28,21.41,momentoFinal,UnidadMedida.C);
Medicion prec3 = new Medicion(29,1.13,momentoFinal,UnidadMedida.LITROS_M2);
Medicion concent3 = new Medicion(30,4.99,momentoFinal,UnidadMedida.MG_M3);

// Creamos tres instrumentos de medición, uno de cada tipo
Sensor termometro1 = new Termometro(4,25);
Sensor pluviometro1 = new Pluviometro(5,65);
Sensor detectorN2O1 = new DetectorN2O (6,4.9);

// Simulamos las mediciones en los sensores, añadiéndolas a los instrumentos
termometro1.addMedicion(temp1);
termometro1.addMedicion(temp2);
termometro1.addMedicion(temp3);

pluviometro1.addMedicion(prec1);
pluviometro1.addMedicion(prec2);
pluviometro1.addMedicion(prec3);

detectorN2O1.addMedicion(concent1);
detectorN2O1.addMedicion(concent2);
detectorN2O1.addMedicion(concent3);

// Creamos una localización en Madrid
Localizacion madrid = new Localizacion("Madrid",40.4165000,-3.7025600, 667);

// Creamos una estación meteorológica en la localización creada anteriormente
EstacionMeteorologica estacionMadrid = new EstacionMeteorologica(1,madrid);

// Añadimos a la estación los tres instrumentos de medición creados anteriormente
estacionMadrid.addSensor(termometro1);
estacionMadrid.addSensor(pluviometro1);
estacionMadrid.addSensor(detectorN2O1);

// Añadimos a la estación el conjunto de mediciones que han sido tomadas
estacionMadrid.addMedicion(temp1);
estacionMadrid.addMedicion(temp2);
estacionMadrid.addMedicion(temp3);

estacionMadrid.addMedicion(prec1);
estacionMadrid.addMedicion(prec2);
estacionMadrid.addMedicion(prec3);

estacionMadrid.addMedicion(concent1);
estacionMadrid.addMedicion(concent2);
estacionMadrid.addMedicion(concent3);

// Creamos un nuevo termómetro y lo añadimos a la estación
Termometro termometro2 = new Termometro(4,24);
estacionMadrid.addSensor(termometro2);
```

```
        // Tomamos mediciones en dos nuevos momentos
        Date momentoUltimo = new Date(momentoActualEnMilisegundos+5000);
        Medicion temp4 = new Medicion(31,20.51,momentoUltimo,UnidadMedida.C);
        Date momentoPostUltimo = new Date(momentoActualEnMilisegundos+5500);
        Medicion temp5 = new Medicion(32,20.57,momentoPostUltimo,UnidadMedida.C);

        // Simulamos las mediciones en el termometro, añadiéndolas
        termometro2.addMedicion(temp4);
        termometro2.addMedicion(temp5);

        // Añadimos a la estación las nuevas mediciones realizadas con el nuevo termómetro
        estacionMadrid.addMedicion(temp4);
        estacionMadrid.addMedicion(temp5);

        // Presentamos por pantalla los valores más altos y los valores medios registrados para
todos los sensores de las 3 unidades de medida
        System.out.println("Valor máx de T en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMasAlto(UnidadMedida.C));

        System.out.println("Valor medio de T en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMedio(UnidadMedida.C));

        System.out.println("Valor máx de Precipitaciones en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMasAlto(UnidadMedida.LITROS_M2));

        System.out.println("Valor medio de Precipitaciones en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMedio(UnidadMedida.LITROS_M2));

        System.out.println("Valor máx de N2O en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMasAlto(UnidadMedida.MG_M3));

        System.out.println("Valor medio de N2O en la estación de " +
        estacionMadrid.getLocalizacion().getNombre() + ": " +
        estacionMadrid.valorMedio(UnidadMedida.MG_M3));

        // Creamos una nueva localización correspondiente a la ciudad de Barcelona
        Localizacion barcelona = new Localizacion("Barcelona",41.3818,2.1685, 13);

        // Creamos una estación meteorológica en la localización correspondiente a Barcelona
        EstacionMeteorologica estacionBarcelona = new EstacionMeteorologica(2,barcelona);

        // Creamos un instrumento de medición de tipo termómetro
        Termometro termometro3 = new Termometro(7,21.1);

        // Añadimos a la estación de Barcelona el termómetro creado anteriormente
        estacionBarcelona.addSensor(termometro3);

        // Tomamos una medición en este nuevo momento actual 21 segundos después del momento actual
        Date nuevoMomentoActual = new Date(momentoActualEnMilisegundos+21000);
        Medicion temp6 = new Medicion(33,20.71,nuevoMomentoActual,UnidadMedida.C);

        // Simulamos las mediciones en el termometro, añadiéndolas
        termometro3.addMedicion(temp6);

        // Añadimos a la estación la medición que ha sido tomada
        estacionBarcelona.addMedicion(temp6);
```

```
        // Presentamos por pantalla el valor más alto registrado para T en Barcelona
        System.out.println("Valor máx de T en la estación de " +
        estacionBarcelona.getLocalizacion().getNombre() + ": " +
        estacionBarcelona.valorMasAlto(UnidadMedida.C));

        // Presentamos por pantalla el valor medio de T registrado por todos los sensores en
Barcelona
        System.out.println("Valor medio de T en la estación de " +
        estacionBarcelona.getLocalizacion().getNombre() + ": " +
        estacionBarcelona.valorMedio(UnidadMedida.C));

        // Creamos un objeto de la clase GestorMediambiental
        GestorMedioambiental gestor = new GestorMedioambiental();

        // Añadimos las dos estaciones meteorológicas (Madrid y Barcelona) al gestor
        gestor.addEstacion(estacionMadrid);
        gestor.addEstacion(estacionBarcelona);

        // Presentamos por pantalla el valor medio y el valor más alto registrado por todas las
estaciones que gestiona el gestor medioambiental para las distintas medidas
        System.out.println("Valor medio de T registrado en las estaciones gestionadas por el gestor
        medioambiental: " + gestor.valorMedio(UnidadMedida.C));
        System.out.println("Valor medio de Precipitaciones registrado en las estaciones gestionadas
        por el gestor medioambiental: " + gestor.valorMedio(UnidadMedida.LITROS_M2));
        System.out.println("Valor medio de N2O registrado en las estaciones gestionadas por el
        gestor medioambiental: " + gestor.valorMedio(UnidadMedida.MG_M3));

        System.out.println("Valor máximo de T registrado en las estaciones gestionadas por el gestor
        medioambiental: " + gestor.valorMasAlto(UnidadMedida.C));
        System.out.println("Valor medio de Precipitaciones registrado en las estaciones gestionadas
        por el gestor medioambiental: " + gestor.valorMasAlto(UnidadMedida.LITROS_M2));
        System.out.println("Valor medio de N2O registrado en las estaciones gestionadas por el
        gestor medioambiental: " + gestor.valorMasAlto(UnidadMedida.MG_M3));
```

Al ejecutar el *main*() con este código, el sistema debería:

1. Crear una localización correspondiente con las coordenadas geográficas de Madrid. A continuación, crear una estación meteorológica en dicha localización y añadirle un termómetro, un pluviómetro y un detector de concentración de N2O. Realizar mediciones de temperatura, precipitaciones y concentración de N2O en tres momentos diferentes. Añadir las mediciones a la estación.

2. Lanzar el aviso de que se ha superado el nivel máximo de temperatura, ya que la segunda medición era de 22.74 ºC, un valor que es superior al 90% del *valor máximo de temperatura* que se había fijado en 25 ºC al crear el termómetro. Lanzar el aviso también de que se ha superado el nivel máximo de concentración de N2O, ya que la última medición de concentración de N2O era de 4.99 mg/m3, un valor que es superior al *valor máximo de concentración de N2O* que se había fijado en 4.9 mg/m3 al crear el detector de N2O.

3. Crear un nuevo termómetro y añadirlo al sistema. A continuación, presentar por pantalla los valores más altos y medios registrados para la temperatura, precipitaciones y concentración de N2O.

4. Crear una estación medioambiental correspondiente con las coordenadas geográficas de Barcelona, añadir un termómetro, tomar una medición de temperatura y añadirla a la estación meteorológica creada en Barcelona y mostrar por pantalla el valor medio y máximo de Tª registrados en Barcelona.

5. Crear un gestor mediambiental y añadir las estaciones meteorológicas correspondientes a Madrid y Barcelona. Mostrar el valor medio y máximo de las tres medidas registradas en el gestor en todas sus estaciones por todos sus sensores.

De nuevo, es importante probar el sistema ante otras situaciones, con diferentes estaciones meteorológicas, localizaciones, mediciones, etc. Cree mediciones que sobrepasen los valores máximos en los tres tipos de sensores y compruebe cómo responde el sistema. Cree más gestores medioambientales, añada nuevas estaciones, simule nuevas mediciones y compruebe el correcto funcionamiento de todos los métodos implementados en todas las clases.

5.2 ETAPA 3.2: (opcional para Mecánica y Tecnología Industrial): EXTENDIENDO EL SISTEMA

5.2.1 OBJETIVOS

Hasta llegar a esta última etapa hemos hecho un recorrido por los aspectos más importantes de la programación orientada a objetos en Java. Para finalizar el diseño e implementación de la aplicación que tenemos entre manos, vamos a introducir un nuevo tipo de sensor en el sistema. Hasta ahora nuestra aplicación solo permitía la gestión de termómetros, pluviómetros y detectores de concentración de N2O. Ahora queremos poder introducir en una estación medioambiental otro tipo de instrumento de medición: un detector de concentración de CO2.

5.2.2 REQUISITOS PREVIOS

Para la realización de esta parte es necesario haber estudiado y tener claro todo lo que se ha ido proponiendo a lo largo de la presente práctica.

5.2.3 ENUNCIADO: EXTENDIENDO EL SISTEMA CON MÁS SENSORES

En este último apartado queremos ampliar la funcionalidad del sistema permitiendo añadir un nuevo tipo de sensor para la detección de niveles de concentración de CO2 y que esto no implique modificar las clases creadas en la última etapa, salvo añadir una nueva medida al sistema (MG_CO2_M3).

Un *DetectorCO2,* además de tener las mismas características comunes a todos los sensores, contará con un *valor máximo de concentración de CO2* (se expresará en miligramos/m3), de modo que cuando se actualice la última medición de concentración de CO2, si el valor de esta concentración es superior al 75% del *valor máximo de concentración de CO2* establecido, el detector deberá emitir un aviso en forma de mensaje por pantalla indicando que se está alcanzando el valor máximo.

Para comprobar el nuevo funcionamiento del sistema se debe añadir al último Lanzador las siguientes instrucciones:

```
        // Simulamos nuevas mediciones de CO2
        Date momentoNuevo = new Date(momentoActualEnMilisegundos+100000);
        Medicion co21 = new Medicion(31,3.1,momentoNuevo,UnidadMedida.MG_CO2_M3);
        Date momentoNuevo2 = new Date(momentoActualEnMilisegundos+200000);
        Medicion co22 = new Medicion(32,4.8,momentoNuevo2,UnidadMedida.MG_CO2_M3);

        // Creamos un nuevo sensor detector de co2
        Sensor detectorCO21 = new DetectorCO2(8,5);

        // Simulamos las mediciones en el sensor
        detectorCO21.addMedicion(co21);
        detectorCO21.addMedicion(co22);

        // Añadimos a la estación de Barcelona el nuevo detector creado anteriormente
        estacionBarcelona.addSensor(detectorCO21);

        // Añadimos a la estación las mediciones que han sido tomadas
        estacionBarcelona.addMedicion(co21);
        estacionBarcelona.addMedicion(co22);

        // Presentamos por pantalla el valor más alto registrado para CO2 en Barcelona
        System.out.println("Valor máx de CO2 en la estación de " +
        estacionBarcelona.getLocalizacion().getNombre() + ": " +
        estacionBarcelona.valorMasAlto(UnidadMedida.MG_CO2_M3));

        // Presentamos por pantalla el valor medio de CO2 registrado por todos los sensores en
Barcelona
        System.out.println("Valor medio de CO2 en la estación de " +
        estacionBarcelona.getLocalizacion().getNombre() + ": " +
        estacionBarcelona.valorMedio(UnidadMedida.MG_CO2_M3));

        // Presentamos por pantalla el valor medio y el valor más alto registrado por todas las
```

```
estaciones que gestiona el gestor medioambiental para la nueva medida
        System.out.println("Valor medio de CO2 registrado en las estaciones gestionadas por el
        gestor medioambiental: " + gestor.valorMedio(UnidadMedida.MG_CO2_M3));
        System.out.println("Valor máximo de CO2 registrado en las estaciones gestionadas por el
        gestor medioambiental: " + gestor.valorMasAlto(UnidadMedida.MG_CO2_M3));
```

De nuevo, es importante probar el sistema ante otras situaciones, con diferentes estaciones meteorológicas, localizaciones, mediciones, etc. Cree mediciones que sobrepasen los valores máximos de todos los nuevos sensores y compruebe cómo responde el sistema. Cree más gestores medioambientales, añada nuevas estaciones, simule nuevas mediciones y compruebe el correcto funcionamiento de todos los métodos implementados en todas las clases.

6 Fechas y Normas de Entrega

6.1 Fechas

La realización de la práctica se llevará a cabo en los centros asociados, siendo las sesiones organizadas y supervisadas por el tutor de la asignatura. Habrá como mínimo tres sesiones presenciales de obligatoria asistencia. En cada sesión se abordará cada una de las partes de las que consta la práctica. **Los alumnos deberán ponerse en contacto con su centro asociado para informarse acerca de cuándo tendrán que asistir a las sesiones**.

Las fechas orientativas para la realización de cada una de las etapas serán:

- Finales de Marzo. Realización de la primera parte de la práctica.
- Finales de Abril. Realización de la segunda parte de la práctica.
- Mediados de Mayo. Realización de la tercera parte de la práctica.

7 Normas de Entrega

La práctica se entregará a través de la tarea definida en el entorno virtual de la asignatura. Cada alumno creará y comprimirá una carpeta nombrada con el DNI y primer apellido separados por un guión (”8345385X-gonzalez”). Esta carpeta contendrá:

- Una memoria de **no más de 6 hojas** donde se explique la especificación y el diseño realizados en cada parte de la práctica.
- Los ficheros .JAVA, sin caracteres especiales; por ejemplo *ñ* o *tildes*.

NOTA IMPORTANTE: Los nombres de los ficheros y carpetas/paquetes que compongan la práctica entregada, deben contener SÓLO caracteres correspondientes a las letras de la A a la Z, tanto mayúsculas como minúsculas, números del 0 al 9 y los caracteres especiales ’-’ y ’ ’. **No deben utilizarse otros, tales como tildes o símbolos**.

Los tutores de la asignatura deberán evaluar en el entorno virtual las prácticas antes del 18 de mayo.

NOTA IMPORTANTE: Los tutores tienen que cumplir una serie de requisitos ante los alumnos debido a que la práctica cuenta para la calificación de la asignatura. Por tanto, antes de entregar las calificaciones al equipo docente deberán:

1. Publicar la nota de las prácticas en un lugar accesible para los alumnos (ya sea vía web o mandar un fax al centro asociado)
2. Establecer un día de revisión de prácticas (previo al período de entrega de las calificaciones al equipo docente), dado que éstas forman parte de la evaluación del alumno.

Es importante que se mantengan todos los identificadores definidos en el enunciado, es decir, el nombre de las clases, campos y métodos deben ser tal y como se definen en este enunciado.

7.1 Evaluación de la práctica

Las prácticas tienen carácter **INDIVIDUAL**, para evitar posibles copias todas las prácticas pasarán por un software detector de copias. La detección de prácticas copiadas implicará un **SUSPENSO** en **TODO** el curso, es decir, convocatorias de Junio y Septiembre, para todos los implicados.

Es requisito indispensable para aprobar el examen, la asistencia a las 3 sesiones obligatorias de la práctica, así como superar la propia práctica. El informe del tutor se considera a efectos de subir nota.

Las prácticas NO se guardan de un curso para otro.

7.2 Preguntas al Equipo Docente

El equipo docente atenderá preguntas de carácter metodológico y de diseño, preferentemente a través de los foros del curso virtual de la asignatura. Las preguntas relativas a la instalación del entorno de desarrollo, puesta en funcionamiento y errores de compilación deben ser remitidas a los tutores de los centros asociados.

Felisa Verdejo Maillo, Catedrática de Universidad
Tutorías: Martes y miércoles de 11:00 a 13:00h

Teléfono: 91 398 64 84
Email: felisa@lsi.uned.es

Víctor Fresno Fernández, Profesor Titular de Universidad
Tutorías: Martes y Miércoles de 11:30 a 13:30

Teléfono: 91 398 82 17
Email: vfresno@lsi.uned.es

Enrique Amigó Cabrera, Profesor Contratado Doctor
Tutorías: Jueves de 15:00 a 19:00

Teléfono: 91 398 86 51
Email: enrique@lsi.uned.es

Roberto Centeno Sánchez, Profesor Contratado Doctor
Tutorías: Jueves de 11:00 a 13:00h y de 15:00 a 17:00 h.

Teléfono: 91 398 96 96
Email: rcenteno@lsi.uned.es