Laboratorios S13 y S14

Objetivos

- Trabajar los siguientes conceptos de POO en Java: herencia, polimorfismo, clases abstractas e interfaces.

Herramientas que utilizaremos

- El entorno de desarrollo *Eclipse* y Javadoc

Entregable

Se debe entregar un fichero **ZIP** con el **diagrama de clases UML del laboratorio** y el **proyecto de *eclipse* exportado (código + documentación)**. Os recuerdo la nomenclatura de los diferentes elementos:

- Nombre del proyecto Java: *apellido_nombre_S13*
- Nombre del fichero a entregar: *apellido_nombre*.zip
- La entrega será individual a través de eGela
- Último día para realizar la entrega: 6 de mayo, jueves, antes de las 23:55

Semana 13:

Descripción de las clases de la aplicación

Se desea realizar una aplicación para monitorizar el conjunto de bombillas de un edificio inteligente. El objetivo principal de la aplicación, es ser capaces de determinar la energía consumida (unidad:Watt), la intensidad luminosa (unidad:lumen) y la eficacia luminosa (unidad: lm/W) del edificio en un momento dado.

GC1 Estas son las características generales de las bombillas que se podrán utilizar en el edificio: identificador (String), potencia nominal o energía que consume (cantidad de Watt, double), luminosidad o intensidad luminosa nominal (lumen en potencia máxima, double) y si está encendida o no (boolean). Además, tendrá las siguientes operaciones:

a) **Constructora**, dados un identificador, potencia nominal e intensidad luminosa nominal creará un objeto de tipo Bombilla. Esta operación siempre creará la bombilla apagada.

b) Operaciones getter y setter para los atributos potencia y luminosidad nominal.

c) Operación getter para el id y el atributo encendido de la Bombilla.

d) **`Método turnOn`:** enciende la bombilla. Pero si la bombilla ya está encendida eleva la excepción `AlreadyOnException`.

e) **`Método turnOff`:** apaga la bombilla. Este método lanzará la excepción `AlreadyOffException` si la bombilla ya estaba apagada.

f) **`Método efficacy:`** Calcula y devuelve la eficacia luminosa de la bombilla (lm/W, número real). De manera general, la eficacia luminosa de la bombilla se calcula dividiendo la luminosidad nominal por la potencia nominal. Es decir, el flujo luminoso emitido por la bombilla respecto a su consumo (si la bombilla está apagada se devuelve el valor de eficacia correspondiente a los valores nominales).

GC2 Se considerarán diferentes tipos de bombillas, para las que se debe definir su jerarquía. De manera general, las bombillas pueden ser fluorescentes o regulables. Para las fluorescentes es necesario conocer el tipo de cebador que precisan para funcionar (String).

-Para las bombillas regulables, se debe considerar que pueden estar encendidas en la potencia nominal o en una potencia inferior. Por lo tanto, tendrán como característica el porcentaje de potencia a la que están encendidas (número entero, representa el porcentaje que están consumiendo respecto a la potencia nominal). Al encender una bombilla regulable su potencia actual será el 50% de su potencia nominal; este porcentaje se puede regular a gusto del usuario (`setPercentage`) mientras siga encendida. Al apagar la bombilla su porcentaje toma el valor 0.

La luminosidad actual de las bombillas regulables (`currentBrightness`) se calculará aplicando el porcentaje de potencia actual a la luminosidad nominal. Si la bombilla está apagada la luminosidad será 0.

-Cuando están encendidas, la eficacia (`efficacy`) de las bombillas regulables se calcula dividiendo la luminosidad actual por la potencia actual. Cuando están apagadas, se debe devolver la eficacia en base a la luminosidad y potencia nominales.

Hay dos tipos de bombillas regulables: las incandescentes y las no incandescentes. Las bombillas no incandescentes tienen como característica principal su necesidad o no de transformador (boolean). En este grupo podemos distinguir las halógenas y las LED. Las características más destacadas de las bombillas LED son su color (rojo, naranja, cyan o azul) y porcentaje de estanqueidad (double). La luminosidad de las LED depende no solo de la potencia actual y de su luminosidad nominal sino que además se debe multiplicar por un factor en base a su color: naranja 1.29, cyan 1.5, azul 1.31 y rojo 1.15. En todo caso, la luminosidad actual nunca puede superar la luminosidad nominal[1].

GC3 La clase **`Building`** representará el conjunto de bombillas de un edificio y tendrá la siguiente funcionalidad:

a) **Constructora:** Crea el edificio considerando que no hay aún ninguna bombilla en el edificio.

b) `turnOnLightBulb`: dado un identificador, enciende la bombilla caracterizada con dicho identificador. Este método puede elevar dos excepciones:

- `UnknownIdentifierException`, si no existe una bombilla con el identificador dado.
- `AlreadyOnException`, cuando la bombilla con el identificador dado ya está encendida.

[1] Podéis utilizar el método Math.min para realizarlo más fácil

c) `turnOffLightBulb`: dado un identificador, apaga la bombilla caracterizada con ese identificador. Este método elevará la excepción `UnknownIdentifierException` cuando no exista una bombilla con ese identificador. Si la bombilla ya estaba apagada no hará nada.

d) `printBulbs`: Imprime la información de todas las bombillas del edificio utilizando el método `toString` de las bombillas.

e) `obtainLightBulb`: Devuelve la bombilla cuyo identificador se pase como parámetro. Este método lanzará la excepción `UnknowIdentifierException` si no existe una bombilla con dicho identificador.

f) `currentConsumption`: Devuelve la suma de la potencia nominal de todas las bombillas que están encendidas en el edificio.

g) `maximumConsumption`: Calcula y devuelve el consumo de energía máximo posible del edificio con todas las bombillas encendidas en su potencia nominal.

h) `addLightBulb`: Añade una bombilla al edificio. Si ya existía la bombilla la reemplaza (suponiendo que es un cambio de bombilla).

Semana 14:

En los sitios húmedos, tales como los cuartos de baño, solo se pueden instalar bombillas fluorescentes o de tipo LED. Para estos casos, se necesita alguna manera de poder determinar cuál es el nivel de seguridad de la bombilla (double entre 0 y 3). Estos valores dependen del tipo de cebador y de la estanqueidad de la bombilla, respectivamente. Para las fluorescentes, si el cebador es del tipo “DEOS ST171”[2] el nivel de seguridad será 3, si es de tipo “S2 4-22W SER” será de nivel 2 y en cualquier otro caso será de nivel 1. En cambio, para las LED, su nivel de seguridad se obtiene multiplicando por 3 el porcentaje de estanqueidad de la misma.

Además, se tiene que poder ordenar las bombillas en base a su identificador utilizando la interfaz `Comparable`. Esto conllevará añadir un nuevo método `sort` en la clase `Building`.

[2] Para comparar los String sin tener en cuenta su caja (mayúscula o minúscula) podéis utilizar el método equalsIgnoreCase(String s) que proporciona la clase String

Tareas a realizar

Tarea 1. Hacer el diagrama de clases de la aplicación, considerando las especificaciones del enunciado.

Tarea 2. Implementa y documenta, utilizando el estilo Javadoc, todas las clases.

Tarea 3. Crea una clase `DemoBuildingBulb` con un método `main`. El método `main` deberá hacer lo siguiente:

3.1. Crear un edificio y asignarle diversas bombillas de cada uno de los tipos.

3.2. Encender algunas bombillas y calcular el consumo de las bombillas que están encendidas, así como el posible consumo máximo del edificio.

3.3. Haz que se lancen las excepciones definidas, y trátalas de la siguiente manera en cada caso:

- `UnknowIdentifierException:` Mostrando un mensaje en la pantalla
- `AlreadyOnException:` No hay que hacer nada especial, pero el método main se tiene que seguir ejecutando.

Aclaración: El método `main` no debe lanzar excepciones, por lo tanto, cuando se eleva una excepción se debe tratar y se debe seguir con las siguientes instrucciones del `main`.

Tarea 4. Una vez realizada la parte de la semana 14

4.1. Ordena las bombillas del edificio en base a su identificador y escribe la información en la pantalla utilizando el método `toString()` de las bombillas.

4.2. Muestra por pantalla qué bombillas, entre las que están en el edificio, pueden estar en lugares húmedos y cuál es su nivel de seguridad. Para ello, en la clase `Building` se debe añadir el método `printWetPlaceBulbs`.