PRUEBA AUTODIAGNÓSTICA UD 3

Ejercicio 1 – 6 puntos

Codifica una herencia de clases para modelar **calentadores domésticos**, sabiendo que:

- Todos los **calentadores** tendrán una potencia máxima que será un número real (por ejemplo: 1000.0 W), ofreciendo métodos para establecer y consultar dicha propiedad.
- Además, todos los calentadores sabrán si están encendidos o apagados, ofreciendo métodos para encenderlos, apagarlos y consultar su estado.
- Todos los calentadores tendrán un método *calculaPotenciaActual()* que devuelva su potencia actual del siguiente modo: si el calentador está apagado el consumo será 0. Si está encendido se devolverá la potencia máxima.
- Un **radiador** es un tipo de calentador que permite graduar porcentualmente la potencia del calentador. Es decir, podremos, por ejemplo, encender el radiador y ponerlo al 60% de la potencia máxima.
- El radiador tendrá un método que permita establecer este porcentaje y otro método que permita consultar este valor.
- El radiador, además, tendrá que sobrescribir el método *calculaPotenciaActual()* para que contemple la aplicación del porcentaje sobre la potencia máxima, siempre que el radiador esté encendido.
- Un **secador** es un tipo de calentador que tiene 2 resistencias para calentar el aire. En todo momento, el secador debe saber si tiene cero, una o dos resistencias activas, ofreciendo métodos para activarlas y apagarlas por separado, así como para consultar su estado independientemente.
- El secador, además, tendrá que sobrescribir el método *calculaPotenciaActual ()* para que refleje que, si está encendido, y tiene solo una resistencia activa entonces devolverá la mitad de su potencia máxima. En el caso de que tuviera ambas resistencias activas entonces devolverá su potencia máxima. Si el secador estuviese encendido, pero ninguna de las resistencias estuviese encendida entonces se devolvería 0.

Por último, debes crear una clase *PruebaCalentadores* con un método *main* que cree un radiador y un secador, ambos de 1000W de potencia máxima realizando las siguientes pruebas:

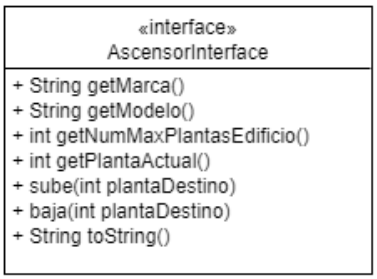
Para el radiador:

- Se debe establecer el porcentaje de potencia al 80%.
- Se debe mostrar la potencia actual con el radiador apagado.
- Se debe mostrar la potencia actual con el radiador encendido.

Para el secador:

- Se debe mostrar la potencia actual con el secador apagado.
- Se debe mostrar la potencia actual con el radiador encendido pero ninguna resistencia activa.
- Se debe mostrar la potencia actual con una resistencia encendida.
- Se deben mostrar la potencia actual con las dos resistencias encendidas.

Ejercicio 2 – 4 puntos



Codifica el siguiente diagrama UML creando un archivo AscensorInterface.java en tu proyecto.

A continuación, crea una clase Ascensor que implemente la interfaz anterior y dótala de código sabiendo que:

- La clase debe tener un constructor que reciba como parámetros la marca, el modelo y el número de plantas máximo que tiene el edificio en el que está instalado el ascensor.
- Las plantas del edificio empiezan en 0 y terminan en el valor del número máximo de plantas que se haya especificado en el constructor.
- El ascensor debe conocer en todo momento la planta actual en la que está parado y poder devolverla con el método *getPlantaActual.* También sabemos que cuando creamos un ascensor, la planta actual se establecerá a 0.
- El ascensor también debe saber devolver en todo momento el número máximo de plantas que tiene el edificio con el método *getNumMaxPlantasEdificio*
- El método *sube(int plantaDestino)* comprobará que la planta de destino existe en el edificio, devolviendo un error en caso contrario. Si la planta de destino está por debajo de la planta actual entonces el ascensor no se moverá y se imprimirá el mensaje *"Mejor usa el botón de bajar"*. En caso contrario, haremos que la planta actual coincida con la planta de destino y mostraremos el mensaje *"Has llegado a la planta XX"*.
- El método *baja(int plantaDestino)* comprobará que la planta de destino existe en el edificio, devolviendo un error en caso contrario. Si la planta de destino está por encima de la planta actual entonces el ascensor no se moverá y se imprimirá el mensaje *"Mejor usa el botón de subir"*. En caso contrario, haremos que la planta actual coincida con la planta de destino y mostraremos el mensaje *"Has llegado a la planta XX"*.
- El método *toString()* sobrescribe el método de la clase Object y debe mostrar todas las propiedades de la clase.

Por último, debes crear una clase *PruebaAscensor* con un método *main* que cree un ascensor con los valores de marca y modelo que consideres pero que esté instalado en un edificio de 10 plantas. A continuación, realiza la siguiente secuencia de acciones: sube(20), sube(5), sube(1), baja(1), baja(5), baja(-1) imprimiendo una línea con el estado del ascensor (usando toString) entre cada dos acciones. Por último, imprime el mensaje "El ascensor está en la planta actual XX".

NOTA: NO hay que usar arrays.