Fecha	Versión	Descripción
08/02/2022	1.0.0	Versión inicial

Programación funcional y excepciones.

Ejercicio 1

Implementa la clase Libro que tiene los atributos de titulo, autor y precio, así como los respectivos getters y setters, constructor y redefinición de toString.

Implementa una lista que almacene en la variable libros un conjunto de libros.

Se pide implementar un programa que muestre por pantalla los libros ordenados de menor a mayor.

Ejercicio 2

Implementa una clase llamada Receta que almacene los datos de una receta de cocina: su nombre, su categoria (pasta, carnes, pescado, arroces, etc.) y sus calorias, incluyendo su constructor, getters y setters así como la función toString.

En el programa principal debes:

- Crear una lista con 6 o 7 recetas.
- Implementa las siguientes consultas:
    - Recetas con menos de 500 calorías.
    - Nombre de las recetas de "carnes", ordenadas alfabeticamente.
    - Media de calorías de las recetas de "verduras".
    - Cuántas recetas hay de más de 800 calorías.

Ejercicio 3

Queremos escribir un programa en Java que nos ayude a gestionar un hogar domótico. Para ello, se pide implementar los siguientes elementos:

- Una clase abstracta llamada **ElementoDomotico**. Tendrá como **atributo** el nombre del elemento (por ejemplo, "Ventana del salón"), junto con un constructor que le dé valor, y el correspondiente getter. Además, definirá dos métodos **abstractos** que todo elemento domótico deberá tener: uno llamado **interruptor()** que permitirá accionar el interruptor asociado al elemento para activarlo/desactivarlo alternativamente, y otro llamado **estado()** que devolverá el estado actual del elemento (si está encendido o apagado).
- Una clase llamada **Calefaccion** que será un subtipo de ElementoDomotico. Redefinirá el código de los dos métodos abstractos para encender o apagar la calefacción, y obtener su estado, y además tendrá como atributo propio la temperatura ambiente deseada, que se podrá obtener y modificar con los correspondientes getter y setter.
- Una clase llamada **PuertaGaraje**, que también será un subtipo de ElementoDomotico y deberá redefinir los métodos pendientes para subir o bajar la puerta, alternativamente.
- Una clase llamada **Ventana**, que también será un subtipo de ElementoDomotico y redefinirá los métodos pendientes para subir o bajar la persiana, alternativamente.
- Además, ***tanto la puerta del garaje como la ventana deben poderse bloquear/desbloquear***. Para ello, ambas clases deben implementar una interfaz llamada Bloqueable, que permitirá definir esos dos métodos, junto con otro para obtener si el elemento está bloqueado o no actualmente.

En los constructores de las diferentes clases, por defecto los elementos se iniciarán como activados (calefacciones encendidas, puertas de garaje subidas y ventanas con persianas subidas), con unos valores predeterminados para sus características particulares (por ejemplo, una temperatura por defecto para las calefacciones).

El programa principal deberá crear una lista de 5 elementos domóticos (en código, sin pedir nada al usuario), habiendo al menos uno de cada tipo (al menos una Calefaccion, una PuertaGaraje y una Ventana), y luego:

1. Mostrará el estado actual de todos los elementos: mostrará un listado de los elementos, ordenados alfabéticamente por su nombre, indicando el estado de cada uno y sus características particulares (por ejemplo, la temperatura en el caso de las calefacciones). Para ordenar el array de elementos **deberás emplear una expresión lambda**.
2. Desactivará los elementos cuyo estado sea activado (es decir, apagar las calefacciones, bajar las puertas de garaje y bajar las persianas de las ventanas). Para esto, **deberás emplear streams**.
3. Volverá a mostrar el estado de todos los elementos, igual que en el paso 1.