

Java Standard Web Programming

Módulo 3 - Desafío

Ejercicio 1: Gestión de un almacén

Primera parte

Crear un programa que permita **manejar los Productos y Clientes de un almacén** con las siguientes condiciones:

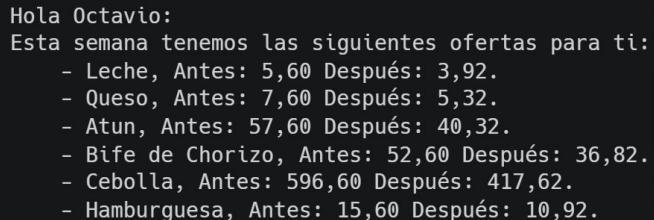
1. Existen dos tipos de productos (Perecederos y No Perecederos).
2. Ambos tipos de productos poseen descripción, precio y cantidad de inventario. Estos atributos pueden ser nulos.
3. Los productos perecederos poseen un atributo que indica la cantidad de días que le quedan para vencer.
4. Crear un método que calcule el precio del inventario y que tenga en cuenta que los productos perecederos tienen un 30% menos de valor “oferta” si poseen menos de 10 días por vencer.
5. Los clientes poseen nombre, apellido, tipo y número de documento.

6. Crear un método utilitario que **imprima la lista** de productos en oferta como texto:

```
Hola {nombre}:  
Esta semana tenemos las siguientes ofertas  
para ti:  
- {descripcion} Antes: {precio}  
Después:{oferta}.
```

7. Para **mostrar**, en consola, los datos básicos de los objetos: usar el método **toString**.

Nota: se debe cumplir con todos los puntos dictados en el módulo 3 del curso.



```
Hola Octavio:  
Esta semana tenemos las siguientes ofertas para ti:  
- Leche, Antes: 5,60 Después: 3,92.  
- Queso, Antes: 7,60 Después: 5,32.  
- Atun, Antes: 57,60 Después: 40,32.  
- Bife de Chorizo, Antes: 52,60 Después: 36,82.  
- Cebolla, Antes: 596,60 Después: 417,62.  
- Hamburguesa, Antes: 15,60 Después: 10,92.
```

Segunda parte

Vamos a mejorar el código que creaste en la primera parte del ejercicio:

1. La clase que posee el método utilitario que imprime el mensaje no se debe poder instanciar.
2. El ciclo del método utilitario que imprime el mensaje debe ser un **forEach**.
3. No se puede instanciar la clase padre **Producto**.
4. El método **getPrecioInventario()** debe ser un método que tenga cuerpo solo en las clases hijas.
5. En la base de datos se ha definido que todas las entidades (Clientes y Productos, por ahora) tendrán un **atributo obligatorio id** que es un entero de gran longitud. En el programa, se deberá garantizar que se implemente un método que retorne y dé valor a dicho atributo en las entidades.
6. El programa va a tener un **auto incrementable id** de tipo para cada entidad o clase (Clientes y Productos) por lo que se debe implementar un método que lo encapsule.

Ejercicio 2: Productos bancarios

Primera parte

Crear un programa que pueda **almacenar los productos de una entidad bancaria** (CA, CC y TC), con las siguientes condiciones:

1. Las **CA y CC** poseen Banco, Sucursal, Número de Producto.
2. Las **Tarjetas de Crédito** poseen Banco, Sucursal, Número de Producto y Clave de Seguridad.
3. El **número de producto** se genera automáticamente e incrementa de uno en uno por producto.
4. El número de producto no se puede pedir por el constructor.



5. Hacer un método que genere un **formato** para cada uno de los números de productos y reciba la estructura que debe imprimir:

CC y CA (###-#-#####/#)
y TC (#### #### #### ##).

5. El **toString** debe mostrar el formato correcto.
6. Para este desafío, el Banco y la Sucursal son **números enteros**.

Segunda parte

Ahora, vamos a hacer las siguientes mejoras:

1. El método que genera el **número de producto** debe ser único en todo el programa.
2. La **clase Producto** no se debe instanciar.



A continuación encontrarás su resolución para que verifiques cómo te fue.



¡Terminaste el módulo!
Todo listo para rendir el examen