Reflexión y Revisión DPOO Taller 2

José Cristóbal Arroyo 202011404 - Juan Alegría 202011282

Febrero 2021

Punto 5 diagramas UML con extensión

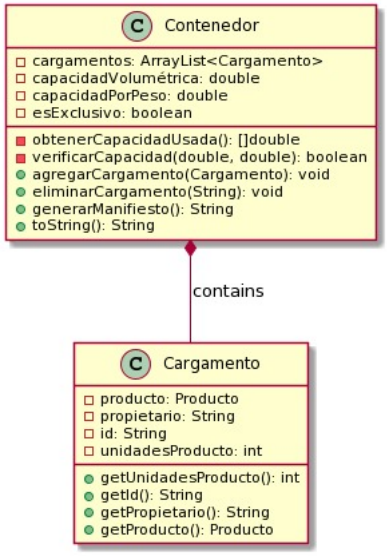

Figure 1: Diagrama UML del contenedor previo a la extensión

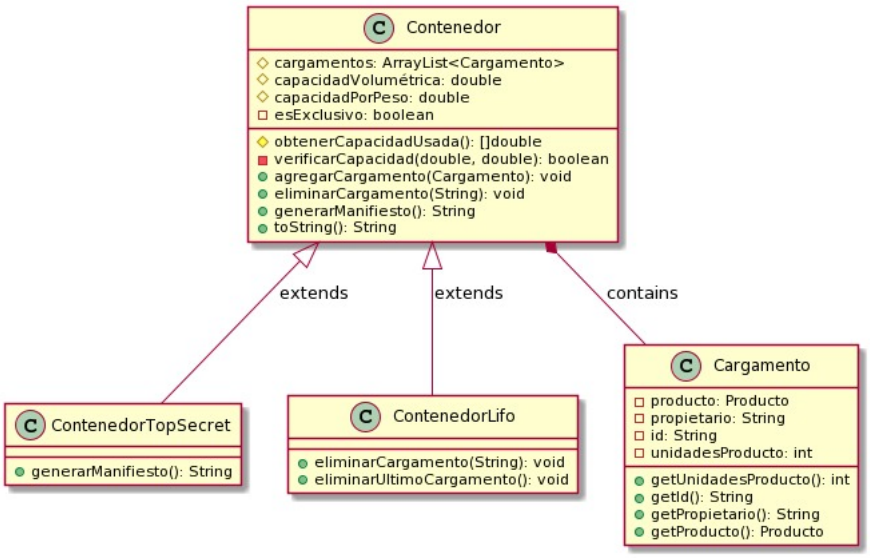

Figure 2: Diagrama UML del contenedor posterior a la extensión

En este caso se puede apreciar cómo los cambios radican en la creación de dos nuevas clases las cuales heredan la características de la clase contenedor, así como la protección de un método y de los atributos que antes era privado. Esto con el fin de que los hijos puedan acceder a dichos comportamientos.

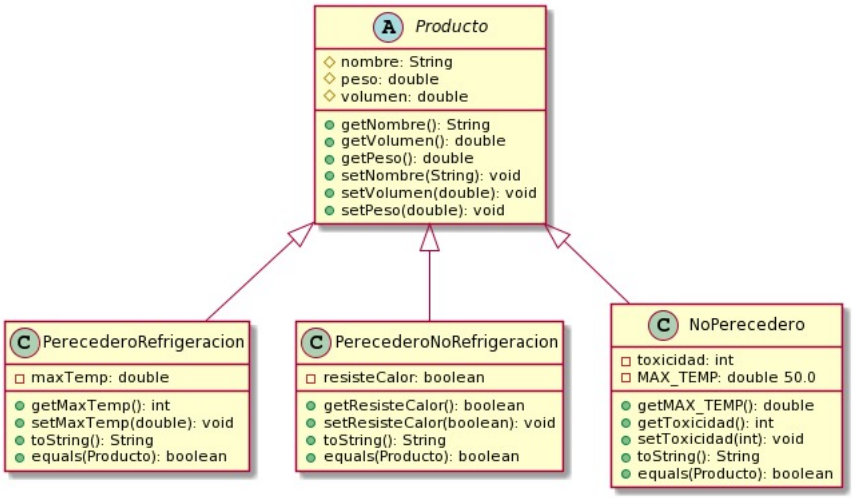

Figure 3: Diagrama UML del producto previo a la extensión

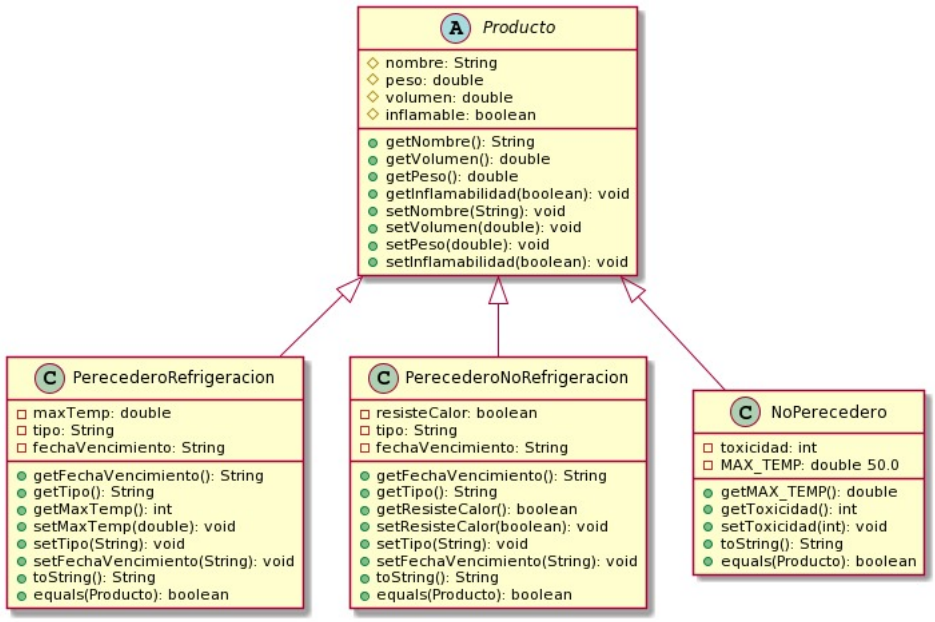

Figure 4: Diagrama UML del producto posterior a la extensión

Como se puede apreciar, los cambios principales radican en la creación de nuevos métodos para los productos. No obstante, los atributos y métodos getters y setters de la fecha de vencimiento y el tipo de producto perecedero se repitió 2 veces. Esto puede ser un indicador de que hubiera sido válido crear una nueva clase de alimento perecedero de la cual extendieran clases distintas.

Herencia

1. Con base en lo que aprendió durante el desarrollo del taller, ¿cuándo tiene sentido usar interfaces y cuándo clases abstractas?

Las interfaces son útiles a la hora de implementar funcionalidades y no características de los objetos. Es decir, esto se debe a que las interfaces solo proporcionan métodos abstractos, mientras que una clase abstracta proporciona también atributos que pueden ser heredados a sus hijos.

2. Java soporta cuatro tipos de visibilidad para los elementos de una clase: public, private, protected y default (en blanco). ¿Cuándo tiene sentido utilizar la visibilidad protected para un atributo en lugar de private?

Cuando los hijos necesitan tener visibilidad de los métodos o atributos. Por ejemplo, en el modelo de productos. Las subclases heredan los atributos protegidos de Producto (nombre, peso volumen), por lo que sus subclases podrán editarlo y se podrán acceder a ellos por medio de getters y setters.

Polimorfismo

3. ¿En qué lugares del taller utilizó el polimorfismo y qué ventajas o desventajas le trajo?

Se utilizó para la creación de productos. En este caso era muy complicado escribir los tipos de cada producto, así que el polimorfismo permitió crear todos los objetos sin necesidad de especificar su tipo, así como tratarlos en los algoritmos simplemente como "Productos". Esta es una ventaja con respecto a la aplicación específica de tipos, sin embargo una desventaja podría ser que se puede caer en un error si se tratan a los hijos de forma independiente en un algoritmo.

Genericidad

4. ¿Dónde utilizó la genericidad en el taller?

La genericidad se utlizó principalmente para crear una estructura de los tipos tratados en el taller. Esto es específicamente en la clase de contenedores y cargamentos.

5. ¿Qué ventajas y qué desventajas trajo el uso de la genericidad?

La genericidad tiene como ventajas el uso de estructuras complejas de clases. Sin embargo, una desventaja puede ser que es más complicado crear una estructura que tenga distintos tipos de clases, esto es porque se especifica una clase T para almacenar un solo tipo de clase.

Java Collections

6. ¿En qué casos tiene sentido usar cada una de las interfaces Set, List, Map y Collection?

Los sets deberían ser utilizados cuando simplemente se quiere almacenar una estructura y caracterizarla sin importar el orden. Por ejemplo contar el total de productos en un cargamento.

La list debería ser utilizada cuando se necesitan organizar los elementos, utilizando algún sorting para conocer la posición exacta del objeto que se quiere buscar.

Las collections vienen muy bien cuando se tiene una pseudo estructura, en donde una clase necesita insertar o eliminar elemetos dependiendo del requerimiento utilizado.

Los mapas son muy útiles a la hora de encontrar elementos específicos, podría ser utilizado para encontrar códigos específicos de productos o contenedores.

Iteradores

7. En una frase, ¿cuál es la principal ventaja de usar iteradores para recorrer estructuras de datos?

Simplicidad y orden.