Laboratorio S07

Proyecto SuperOnline (PARTE 1)- Excepciones

Objetivos

- Conocer el tratamiento de excepciones en Java: definición, captura y elevación.
- Seguir practicando con la documentación en JavaDoc
- Trabajar con Salida de datos a ficheros.
- Como complementario se trabajará con JUnit.

Herramientas a utilizar

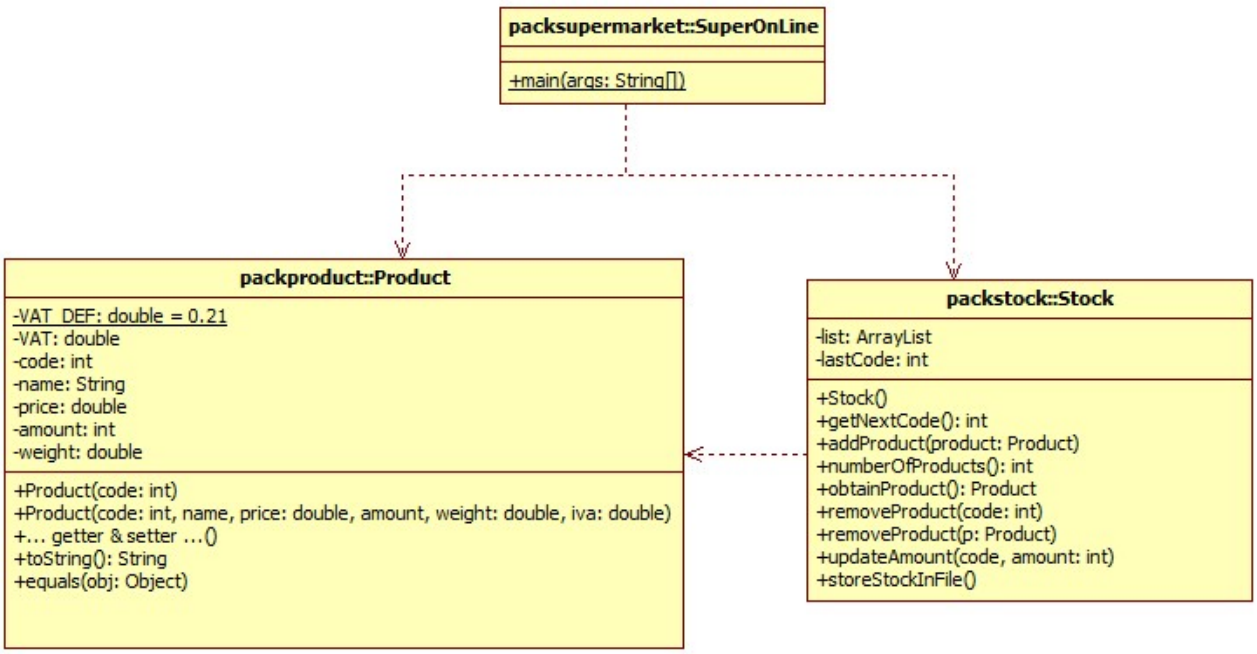
- Eclipse, Javadoc, JUnit

Entregable

Al finalizar el laboratorio se deberá exportar el proyecto realizado y se subirá a eGela. Este proyecto debe contener el **código** desarrollado y la **documentación** generada. El fichero *.zip* deberá nombrarse con apellidos y nombre según el formato: Apellido1_Apellido2_Nombre*.zip*

Enunciado general

El objetivo de este laboratorio es el desarrollo de una aplicación que gestiona el stock de productos de un supermercado: *SuperOnline*. Para ello, implementaremos las clases **`Product`** y **`Stock`**. El siguiente diagrama de clases UML muestra las clases correspondientes a este laboratorio incluyendo el paquete en el que se definen. El **`Stock`** representa el almacén de productos y tiene una lista de los productos disponibles en stock. **`Product`** representa los productos que hay en stock y tiene toda la información relevante de cada producto. El código (code) de **`Product`** es único para cada producto y será añadido al producto por la clase **`Stock`** cuando sea añadido a su lista. Para ello la clase Stock tiene un atributo `lastCode` que guarda el último código puesto a un producto. Se va dando valores correlativos a los nuevos productos. Pero si se elimina un producto ese código no vuelve a asignarse a otro producto.



Tareas a realizar:

1 Descarga el proyecto *SuperOnline.zip* de eGela e impórtalo en Eclipse. No te preocupes que te de errores ya que falta por implementar código que de coherencia a toda la aplicación.

2 Vamos a realizar algunos cambios en las clases, verás que en Eclipse te aparecerán como tareas a realizar porque están marcados con comentarios especiales en el código: "//TODO …". Para verlos selecciona en el menú superior *Window → ShowView → Task* verás que te aparecerá una lista con las tareas pendientes de realizar. Cada vez que termines una tarea, borra el término TODO para que desaparezca de la lista de tareas pendientes.

3 Completa la clase **`Product`** en el paquete `packProduct`.

3.1 Los atributos y constantes se dan definidos. Revísalos y comprueba que se corresponden con los del UML.

3.2 Implementa de manera automática las siguientes operaciones. Para ello utiliza los menús *Source → Generate constructor using Fields ...* y *Source → Generate getters and setters ...*

- Una segunda constructora con un valor por cada atributo de la clase.
- Métodos *getter* y *setter* de todos los atributos.

3.3 Sobreescribe[1] los siguientes métodos (están implementados parcialmente):

- **`toString`**: Crea un String sólo con los valores de los atributos de la clase separados por un espacio y en este orden: code, name, price, amount, weight y VAT.
- **`equals`**: Completa el método indicando que dos productos son iguales si tienen el mismo código.

4 **Definición de excepciones de usuario.** Define las siguientes excepciones como clases independientes en un paquete `packexceptions`:

- **`NegativeAmountException`**, para expresar que es un error actualizar la cantidad de un producto con un valor negativo. Implementa dos constructoras: una sin parámetros y la otra con un parámetro `String` para el mensaje
- **`UnknowCodeException,`** para expresar que es un producto desconocido. No implementes ninguna constructora (quedará definida implícitamente una sin parámetros).

5 Actualiza los siguientes métodos para usar excepciones:

- Método **`setAmount`** de **`Product`**. Este método elevará la excepción **`NegativeAmountException`** si la cantidad dada es negativa. Para elevarla se debe usar la constructora con el parámetro `String`, pasándole un mensaje apropiado (indicando el valor negativo concreto del parámetro, por ejemplo). Al hacerlo te dará varios errores que iremos corrigiendo en los siguientes apartados.
- Método **`obtainProduct`** de **`Stock.`** En aquellos casos que el código de producto no está en el inventario, en lugar de los mensajes que se imprime y la devolución de un producto ficticio (suprime esas sentencias) eleva la excepción **`UnknowCodeException`**. Recuerda que es necesario indicar en la cabecera del método que esa excepción puede elevarse.
- Métodos **`removeProduct`** (hay dos métodos) de **`Stock.`** Estos métodos deben elevar la excepción **`UnknowCodeException`** si no existe ese código de producto en el stock. Elimina el mensaje que se escribe (en el método con parámetro **`Product`**) asegurándote que se eleva correctamente la excepción.

[1] Se dice sobreescribe, porque esos métodos ya están definidos por defecto para toda clase que se defina. Se solicita implementarlo con la semántica indicada.

- Método **`updateAmount`** de **`Stock`**. Revisa bien el código de este método. Se debe considerar lo siguiente:
  - Si no existe un producto con el código dado, debe seguir elevando la excepción **`UnknowCodeException.`**
  - Tratará la excepción **`NegativeAmountException`** y escribirá simplemente el mensaje obtenido de la excepción elevada por **`setAmount`**. Se eliminará el mensaje que se escribía por pantalla cuando el producto devuelto era el ficticio de código -11111.

6 **Ejecútalo y comprueba el correcto funcionamiento de todo lo que has implementado**:

- Corrige los errores de compilación del método `main` de la clase **`SuperOnline.`** Para ello haz un tratamiento adecuado de las excepciones.
- **¡OJO!, el método `main` no debe elevar ninguna excepción,** por lo que todas las excepciones en el `main` se deben capturar y mostrar un mensaje por pantalla.
- La captura de una excepción no debe impedir la ejecución de las siguientes sentencias del `main`.
- Asegúrate que se elevan las dos excepciones. Si no sucede eso, completa el `main` con las acciones pertinentes para que se eleven y se escriban los mensajes correspondientes.

7 Agrega en la clase **`Stock`** el método **`storeStockInFile`**, que guarda el stock en un archivo (stock.txt). En su implementación:

- Utiliza el método `toString` del producto.
- Se debe escribir la información de cada producto en una línea. Recuerda que el String "\n" hace el salto de línea).
- Captura adecuadamente la excepción que se puede elevar al utilizar la clase FileWriter[2] y trátala mediante un mensaje de error apropiado
- Actualiza el `main` de la clase **`SuperOnline`** y guarda el stock en el fichero y comprueba que el fichero se genere correctamente. Debe contener algo así como:

```
1001 banana 2.3 50 1.0 0.04
1002 gel_ducha 1.58 100 1.2 0.21
1003 alitas_de_pollo 2.13 40 0.5 0.1
```

[2] Tenéis un fichero "Tema0- Java Entada Salida" en la pestaña del Tema0 en eGela. FileWriter pertenece a la librería java.io