Laboratorio S9
Singleton y JUnit

Objetivos
- Entender la necesidad y el uso del patrón Singleton
- Seguir practicando con la documentación en JavaDoc y con la verificación con JUnit.

Herramientas a utilizar
- Eclipse, JavaDoc, JUnit

Entregable
Se debe entregar un fichero **ZIP** con la exportación del **Proyecto** del laboratorio S9. Este proyecto debe contener el **código** desarrollado (incluyendo las clases de Test) **y** la **documentación** generada.
- Nombre del fichero: *S9_apellido_nombre*.zip.
- La **entrega** es **individual** y se subirá a eGela
- **Fecha límite** de entrega: **viernes 26** de **marzo** a las **23:55**

Tareas a realizar
En todos los cambios que se realices, debe quedar la documentación totalmente actualizada. Asegúrate que las clases están totalmente documentadas al finalizar el laboratorio.

1. Importa el proyecto del laboratorio anterior y cámbialo de nombre a *S9Supermarket*
2. Modifica la clase Stock para que siga el patrón Singleton
3. Añade a la clase Stock los siguientes métodos
    a. containsProduct: que devuelve cierto si contiene el Product dado y falso en caso contrario
    b. stockSize: Devuelve el número de Productos diferentes que tenemos en Stock
4. Actualiza la clase SuperOnline para que no de errores de compilación al cambiar la clase Stock
5. Verifica el método *setAmount* de la clase **Product** usando JUnit considerando el tratamiento de excepciones. Realiza estas pruebas
    a. Pasándole un valor positivo
    b. Pasándole un valor negativo
    c. Pasándole el 0
6. Verifica el método *getNextCode* de la clase **Stock** usando JUnit
    a. Comprobando que la primera vez da el valor correcto
    b. Comprobando que da el valor correcto la segunda vez
7. Prueba los test de JUnit definidos hasta ahora
8. Verifica el método *obtainProduct* de la clase **Stock** usando JUnit considerando también el tratamiento de excepciones.
    a. Pasándole un código de producto que exista
    b. Pasándole un código de producto que no exista
9. Verifica el método *removeProduct* que recibe un código como parámetro usando JUnit
    a. Pasándole un código de producto que exista
    b. Pasándole un código de producto que no exista

Tareas complementarias

10. Añade los siguientes métodos en la clase **Stock**:

    a. *obtainProductListToOrder*: devuelve una lista con los códigos de los productos para los que en el stock queden menos unidades que el valor dado por parámetro

    b. *removeAllProductsWith0Units*: Elimina del **Stock** aquellos productos para los que queden 0 unidades.