Formulario Práctica JUnit

1. Con la información proporcionada para la realización de la práctica, ¿qué tipo de pruebas se llevan a cabo en la práctica? Justifique su respuesta:

En el caso del ejercicio de la clase Buscador, se trata de pruebas de caja blanca debido a que al momento de realizarlas se tenía visibilidad del código interno, es decir, se sabía el código fuente y la estructura de control del programa, permitiendo la corrección de errores en este tras la realización de las pruebas.

Para las pruebas realizadas sobre las clases de Gestor y Producto, se definen como pruebas de caja negra y, al contrario de las explicadas anteriormente, estas se caracterizan por la independencia del código en las pruebas, no teniendo en cuenta cómo se estructura el código ni el programa en sí.

1. En el ejercicio 2 de la práctica ejecutó la batería de pruebas pruebasINSW3.jar, ¿cuántos tests se realizaron con la batería? ¿qué resultados obtuvo? ¿qué deduce a partir de los resultados obtenidos?

Se realizaron un total de 16 pruebas distintas sobre las clases de la librería de insw3.jar, 11 de ellas fueron sobre la clase Gestor y 5 sobre la clase Producto. En estas pruebas de caja negra se probaron las funcionalidades básicas de ambas clases y, a la vez, se utilizaron datos de entrada que deliberadamente tratasen de romper el programa para poder comprobar si se había realizado correctamente el encapsulamiento y tratamiento de excepciones dentro de las funciones disponibles.

Las únicas pruebas que hicieron saltar errores en las pruebas fueron: en la clase Gestor, una de la función añadir producto (permitió añadir una cantidad negativa de productos), y las 3 que se hicieron sobre la función de quitar productos (por algún motivo todas daban el error de IndexOutOfBoundsException); y en la clase Producto, una de añadir cantidad (permitió añadir una cantidad negativa a un producto existente). A partir de estas pruebas se puede deducir que se ha de revisar el código de esas 3 funciones y arreglar los fallos encontrados a través del testing.

1. En el ejercicio 3, ha creado dos tests y uno de ellos produce error. ¿Cuál de ellos? ¿Qué quiere decir el error que produce el test ejecutado?

La prueba que daba error era la que se realizaba sobre la función devolverUltimoElemento() puesto que esta intentaba buscar el elemento en la posición igual al tamaño de la lista (size).

Esto daba el error IndexOutOfBoundsException, el cual significa que la posición introducida supera al tamaño de la lista, puesto que en programación al buscar una posición dentro de un array hay que tener en cuenta que se empieza por la posición 0. En términos prácticos, si se tiene una lista con 3 elementos, la función .size() nos devolverá el número de elementos en la lista (3), pero si queremos utilizar la función .get(), tendremos que restar 1 al número del elemento que queremos obtener (en el caso del último elemento, size – 1).

1. ¿En qué consiste el criterio de clases de equivalencia y cómo lo ha aplicado para realizar su batería de pruebas?

El criterio de clases de equivalencia se refiere a dividir el conjunto de entradas válidas en clases de datos equivalentes. Si un caso de prueba funciona para un dato dentro de una clase, debería funcionar para todos los datos de esa clase. Como ejemplo, en el caso de la batería de pruebas que se ha realizado, si se está probando la función de añadir artículos en la clase Gestor, se pueden hacer 3 pruebas en las que se intente:

* Añadir un número menor que 1 de productos.
* Añadir un número de productos que tenga sentido (de 1 a 100, por ejemplo).
* Añadir un número desorbitado de productos.

Al hacer esas 3 pruebas, nos aseguraremos de que esa función del programa acepta respuestas válidas dentro de unos límites definidos que consigan probar todas las alternativas que pueda haber en el código.

1. Complete la siguiente tabla con los tests que ha realizado en su batería de pruebas, justifique la utilidad del test realizado y los resultados obtenidos en cada uno de los tests. Puede añadir todas las filas que considere necesarias en la tabla:

|  |  |  |
| --- | --- | --- |
| CLASE GESTOR | | |
| Test | **Justificación del test** | **Resultado** |
| Añadir Producto | | |
| Se añade un producto correctamente | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: true |
| Se lanza excepción al introducir producto = null | Comprobar que al introducir un elemento null en el almacén se lanza excepción | Expected: false  Obtained: false |
| Se añade una cantidad negativa de productos | Comprobar si al haber menos de 0, el producto no debería ser añadido. | Expected: false  Obtained: true |
| Se añade una cantidad desorbitada de productos | Comprobar que, al añadir una cantidad tan grande, el producto no debería ser añadido | Expected: false  Obtained: false |
| Se añaden dos productos iguales | Comprobar que el producto se añade a la cantidad existente del que ya está en el gestor | Expected: true  Obtained: true |
| Quitar Producto | | |
| Se quita un producto | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: IndexOutOfBoundsException |
| Se quitan más productos de los que hay | Comprobar que no se pueda quitar una mayor cantidad de productos que las que hay | Expected: false  Obtained: IndexOutOfBoundsException |
| Se quita una cantidad negativa de productos | Comprobar que no se pueda quitar una cantidad negativa de productos | Expected: false  Obtained: IndexOutOfBoundsException |
| Get Almacen | | |
| Se intenta conseguir los productos del almacen | Comprobar funcionalidad básica del módulo | Expected: not null  Obtained: not null |
| Get Cantidad Total | | |
| Se intenta conseguir la cantidad total del almacén | Comprobar funcionalidad básica del módulo | Expected: not null  Obtained: not null |
| Se intenta conseguir la cantidad total del almacén | Comprobar que la cantidad obtenida sea la que hay en el almacén | Expected: true  Obtained: true |
| CLASE PRODUCTO | | |
| Test | **Justificación del test** | **Resultado** |
| Get Nombre | | |
| Se trata de conseguir el nombre de un producto | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: true |
| Get Precio | | |
| Se trata de conseguir el precio de un producto | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: true |
| Get Cantidad | | |
| Se trata de conseguir la cantidad de un producto | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: true |
| Añadir Cantidad | | |
| Se intenta añadir una cantidad del producto | Comprobar funcionalidad básica del módulo | Expected: true  Obtained: true |
| Se intenta añadir una cantidad negativa del producto | Comprobar que la función no reste la cantidad negativa introducida a la cantidad existente del producto | Expected: false  Obtained: true |