Formulario Práctica Junit - Diego Rodríguez Camarmol

Debe rellenar este formulario y subirlo al Blackboard de la asignatura para poder evaluar la práctica.

Ejercicio 1 – Calculadora

Ejercicio 2 – Buscador

Ejercicio 3 - Almacén

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 la práctica realizada se utilizan dos tipos de pruebas, las de caja blanca y las de caja negra. Las de caja blanca, permiten que, una vez realizados los test, se pueda cambiar la clase original para seguir testeando y hacer que el código no tenga brechas. Con esto, se consigue una estructura bien sólida y aligera el tiempo de desarrollo y el empleado en fallos en caso del despliegue del software. Las pruebas de caja negra, son aquellas que no permiten la modificación de la clase original, teniendo que detectar solamente los fallos que hay en el código, sin poder editar este. Esto dificulta la eficiencia, ya que para este caso, una persona es quién hace el código y otra quien lo testea, teniendo que establecer comunicación para algo que podría hacer uno solo.

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?

En el ejercicio 2 (BuscadorTestCase), he realizado un total de 13 tests, en los que compruebo en cada caso los posibles parámetros a añadir en los métodos. Dependiendo del parámetro que sea, así será la respuesta. Es decir, si le metemos un null a un método que espera un int, el resultado de la prueba deber ser erróneo. Por ello, en el ejercicio 2 al tratarse de pruebas de caja blanca, podremos modificar la clase Buscador para que estas pruebas funcionen correctamente y conseguir un programa completo y seguro ante cualquier despliegue del código.

Los resultados obtenidos de las pruebas del ejercicio 2 son fructíferos, puesto que al ser de caja blanca podemos editar la clase original para que falle lo menos posible. Por ello, se ha conseguido que ninguna prueba falle y que el código original esté preparado para cualquier ambigüedad que se pueda introducir. Se ha conseguido por tanto que al introducirse parámetros como nulos o como incompatibilidades del propio programa no salte ningún error.

Se puede deducir que gracias a las pruebas unitarias de caja blanca es posible conseguir un código completo, compacto y sin errores. Por el contrario, en las de caja negra, el cambio de la clase original para manejar las pruebas lo hace aquella persona que creó la clase, por lo que no es eficiente tener que detectar las pruebas y que, si dan fallidas, no poder cambiarlas.

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?

En las pruebas de gestión, los resultados que he obtenido son los siguientes. Todos los métodos funcionan correctamente, salvo el quitarXProducto, ya que en este siempre retorna true, cuando si por ejemplo se le introduce más cantidad de la que existe en el propio producto debería retornar false, pero no es así. Por otra parte, también daría error al introducir los datos permitidos, puesto que la cantidad que se le mete por parámetro y por tanto la que se quiere quitar, nunca se utiliza.

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

El criterio llevado a cabo es sencillo. Lo primero de todo, he comprobado que los métodos funcionasen correctamente de base, por si hubiese que modificar algo en la clase original. Tras comprobar eso y haberlo solucionado, he empezado a meter pequeños problemas que podrían darse al añadir parámetros al método, como nulos o listas vacías. Una vez comprobado esto, se procede al cambio del método en cuestión para que maneje estas excepciones. Por último, he seguido comprobando dentro de la lógica del propio método, añadiéndole, por ejemplo, números negativos o demasiado altos (indexo ut of bounds).

En cuanto al criterio usado para las pruebas de caja negra ha sido distinto, pues al no poder modificar la clase original, simplemente me he dedicado a hacer pruebas aún sin poder mejorarlas.

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:

|  |  |  |
| --- | --- | --- |
| Test | Justificación del test | Resultado |
| Se añade un producto correctamente | Comprobar funcionalidad básica del módulo | Éxito |
| Se lanza excepción al introducir producto = null | Comprobar que al introducir un elemento null en el almacén se lanza excepción | Éxito |
| testQuitarXProducto() | Comprobar la funcionalidad básica del método | Fallido |
| testQuitarXProducto() | Comprobar que al introducir un elemento null se lance una excepción. | Fallido |
| testDevolverPalabraNegativa() | Comprobar que al meterle una posición negativa a la lista salte la excepción especificada para ello. | Éxito |
| testDevolverPalabraExcedida() | Comprobar que al meterle una posición de la lista que no existe salte la excepción de index out of bounds. | Éxito |
| testDevolverPrimerElemento() | Comprobar la funcionalidad Básica del método | Éxito |
| testDevolverPrimerElementoNull() | Comprobar que al meterle un null salte la excepción adecuada para ello | Éxito |
| testDevolverUltimoElemento() | Comprobar la funcionalidad básica del método | Éxito |
| testBuscarPalabra() | Comprobar la funcionalidad básica del método | Éxito |
| testBuscarPalabraNull() | Comprobar que al meterle una palabra nula salte la excepción adecuada. | Éxito |