

Documento de diseño de pruebas

Clase Categoria

Patrón de cuatro fases

1) Definición de la estructura

Given (Diseño de escenario para las pruebas)

El escenario inicial que se crea para las pruebas es el siguiente: Calzado – Nike

When (Comportamiento esperado de las pruebas)

testAgregarNodo: al haberse agregado ya un nodo al momento de crear el escenario, se espera que haya una categoría con el nombre Nike para así comprobar que se haya agregado el nodo.

testDarNodos: como solo se agregó un nodo al momento de crear el escenario, se espera que a estructura que devuelve este método sea de tamaño 1.

testTieneHijo: se espera como resultado true ya que efectivamente Nike es hijo de Calzado.

testBuscarPadre: se espera como resultado la categoría Calzado ya que Nike tiene como padre a Calzado.

testBuscarNodo: se espera que efectivamente se retorne el nodo buscado.

testEliminarNodo: se espera que despues de eliminar el a Nike, la longitud de los nodos hijos de Calzado sea 0.

testBuscarProducto: se espera un null al no haber ningun producto con el id ingresado.

testDarProductos: se espera un 0 al no haber producto en Calzado y contener solo categorías.

testDarMarcas: se espera una marca para calzado, que es Nike.

testDarPreorden: se espera calzado por ser la categoría raíz.

testDarPosorden: se espera Nike por ser el último hijo directo de calzado.

testDarValorVentas: se espera un 0.0 por no haber producto y por ende 0 ventas.

Then (Cambios esperados después de ejecutar el comportamiento esperado)

La misma estructura con la que se inició.

2) Ejecución del sistema

Las pruebas se implementan y realizan en la clase TestCategoria, se un escenario de prueba, el cual puede

cambiarse, y se crea un método para probar cada método de la clase Categoría.

Las pruebas se realizan con assertEquals, esperando simplemente que se retorne algo igual a lo que se espera, para así, hacer fallar la prueba en caso de un resultado inesperado.

3) Comparación de los resultados obtenidos con los esperados

Todos los resultados obtenidos fueron los esperados.

4) Destruir la estructura

La estructura definida para las pruebas se destruye automáticamente una vez se haya cerrado el programa ya que no se están guardando nuevos registros en la carpeta data.

Información adicional

Dificultades del proceso

En principio, es difícil entender un proyecto realizado por alguien más y, como se dice en el enunciado, tan acoplado. Específicamente, saber para qué es cada método y también el hecho de que había métodos con el mismo nombre que al menos uno de ellos no se entendió para qué era y por ende no se probó.

Aspectos cubiertos

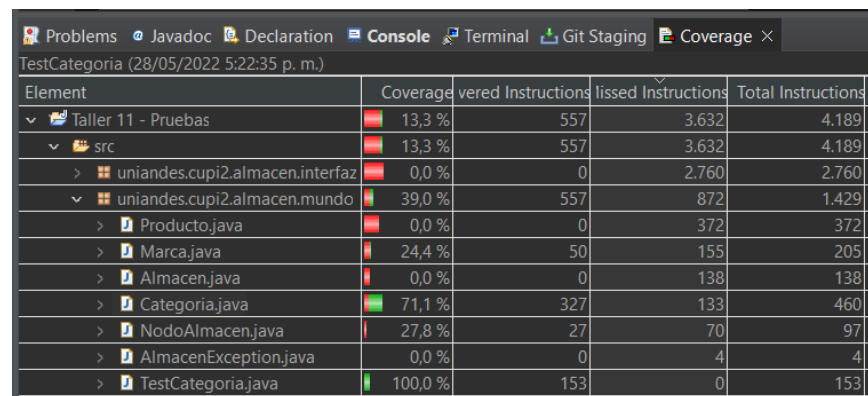
Retornos esperados y corner cases como categorías sin productos.

Aspectos no cubiertos y razones

Habían dos constructores para Categoria, uno de ellos no se usó porque no se entendió, el siguiente: `public Categoria(String pLinea, BufferedReader pLector) throws AlmacenException`

Demás líneas de código no cubiertas fueron excepciones y condicionales a los que se pudo entrar, pero con otros escenarios. El escenario con el que más se cubrió la clase de pruebas es el que se dejó en `TestCategoria`.

Reporte de cubrimiento



Element	Coverage	Covered Instructions	Uncovered Instructions	Total Instructions
▼ Taller 11 - Pruebas	13,3 %	557	3.632	4.189
▼ src	13,3 %	557	3.632	4.189
> uniandes.cupi2.almacen.interfaz	0,0 %	0	2.760	2.760
▼ uniandes.cupi2.almacen.mundo	39,0 %	557	872	1.429
> Producto.java	0,0 %	0	372	372
> Marca.java	24,4 %	50	155	205
> Almacen.java	0,0 %	0	138	138
> Categoria.java	71,1 %	327	133	460
> NodoAlmacen.java	27,8 %	27	70	97
> AlmacenException.java	0,0 %	0	4	4
> TestCategoria.java	100,0 %	153	0	153

Clase Almacen

Patrón de cuatro fases

1) Definición de la estructura

Given (Diseño de escenario para las pruebas)

El escenario inicial que se crea para las pruebas es la carga de datos. Esta carga de datos se hace con los mismos datos del txt original, pero se cargan desde una copia, así no habrá problemas si el archivo a usar en la aplicación cambia.

When (Comportamiento esperado de las pruebas)

testCargar: Se espera que se carguen los datos sin que se arroje AlmacenException

testDarCategoriaRaiz: Se espera que se retorne un string con el nombre de la categoría raíz "Cupi2"

testAgregarNodo: Se espera que se agregue el nodo "TOSHIBA" tipo marca en el nodo tecnología y no se retorne un null al consultar

testEliminarNodo: Se espera que se elimine el nodo "Tecnología" y se retorne un null al consultar

testVenderProducto: Se espera que se venda un producto de id "34089951" y se retorne un número distinto a 0 al comprobar el número de ventas del nodo

testBuscarNodo: Se espera que no se retorne null al buscar el nodo "11" o el nodo de tecnología

testAgregarProducto: Se espera que se agregue un producto "PCGamer" de precio 10 a la marca ASUS, se venda este producto y se compruebe que el valor de las ventas sea 10

testEliminarProducto: Se espera que la lista que contiene los productos del nodo haya disminuido su tamaño en 1

testMetodo1: Se espera que se retorne el string
"Respuesta 1"

testMetodo2: Se espera que se retorne el string
"Respuesta 2"

Then (Cambios esperados despues de ejecutar el comportamiento esperado)

La misma estructura con la que se inició.

2) Ejecución del sistema

Las pruebas se implementan y realizan en la clase TestAlmacen, se un escenario de prueba, el cual puede cambiarse, y se crea un método para probar cada método de la clase Almacen.

Las pruebas se realizan con assertEquals, assertNull, assertNotNull y assertTrue dependiendo de lo que retorne el metodo, ya que hay algunos que retornan un String o un null por defecto. En el caso de assertTrue, este se usa en metodos que puedan dar AlmacenException sí fallan.

3) Comparación de los resultados obtenidos con los esperados

No todos los resultados fueron los esperados.

TestAgregarProducto no pudo ejecutarse de manera correcta y testAgregarNodo no da el resultado esperado aunque si finaliza la prueba

4) Destruir la estructura

La estructura definida para las pruebas se destruye automáticamente una vez se haya cerrado el programa ya que no se están guardando nuevos registros en la carpeta data.

Información adicional

Dificultades del proceso

En principio, es difícil entender un proyecto realizado por alguien más y, como se dice en el enunciado, tan acoplado.

Específicamente, saber para qué es cada método y también el saber la clase en la que se debían pasar los parámetros puede ser complicado, ya que también se debió trabajar con la clase `NodoAlmacen` para lograr comprobar el resultado de las pruebas así como una lista con los productos del nodo.

Aspectos cubiertos

Retornos esperados

Aspectos no cubiertos y razones

Hay un método el cual no se logró completar la prueba, que es `testAgregarProducto`. El proceso falla al momento de agregar el producto inventado. Se cree que el problema sea al momento de digitar la marca al que pertenece el producto.

Otro método, `testAgregarNodo`, logra completar la prueba pero no ofrece el resultado esperado, ya que se debe comprobar que no retorne un valor null siendo que, al parecer, el resultado de la operación es null.

Reporte de cubrimiento

TestAlmacen (30/05/2022 4:54:04 p. m.)				
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ Taller 11 - Pruebas	22,4 %	980	3.389	4.369
▼ src	22,4 %	980	3.389	4.369
> uniandes.cupi2.almacen.interfaz	0,0 %	0	2.760	2.760
▼ uniandes.cupi2.almacen.mundo	60,9 %	980	629	1.609
> Categoria.java	64,6 %	297	163	460
> Producto.java	58,9 %	219	153	372
> TestCategoria.java	0,0 %	0	153	153
> Marca.java	76,6 %	157	48	205
> Almacen.java	66,7 %	92	46	138
> TestAlmacen.java	82,2 %	148	32	180
> NodoAlmacen.java	69,1 %	67	30	97
> AlmacenException.java	0,0 %	0	4	4

TestAlmacen (30/05/2022 4:54:04 p. m.)				
Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
> TestCategoria.java	0,0 %	0	153	153
> Marca.java	76,6 %	157	48	205
> Almacen.java	66,7 %	92	46	138
▼ TestAlmacen.java	82,2 %	148	32	180
▼ testAgregarProducto()	7,1 %	2	26	28
testAgregarNodo()	82,4 %	14	3	17
testCargar()	81,2 %	13	3	16
testescenario()	100,0 %	8	0	8
testBuscarNodo()	100,0 %	10	0	10
testDarCategoriaRaiz()	100,0 %	13	0	13
testEliminarNodo()	100,0 %	15	0	15
testEliminarProducto()	100,0 %	21	0	21
testMetodo1()	100,0 %	12	0	12
testMetodo2()	100,0 %	10	0	10
testVenderProducto()	100,0 %	21	0	21
> NodoAlmacen.java	69,1 %	67	30	97