**ESTRATEGIAS DE DISEÑO DE ESCENARIOS**

La calidad del proyecto se va a asegurar al realizar las distintas pruebas que buscan evaluar la funcionalidad de las clases Almacén y Categoría, junto con sus métodos. Esto al utilizar las pruebas de JUnit y los assert.

La estrategia general se divide en 3 vías:

* La primera es cuando la prueba quiera verificar un retorno, para verificar estos métodos se introducen valores esperados para los métodos y se verifican con los métodos utilizando un assertEquals o assertSame si son objetos.
* La segunda es para los métodos que crean o eliminan objetos, ara estos se utiliza assertNotNull para verificar que el objeto quedo creado o assertNull para verificar que el objeto fue eliminado
* La tercera es para evaluar valores numéricos o creación de listas, para el primero se crea un valor esperado, se compara en un booleano y se usa assertTrue para ver si el valor era el esperado y para el segundo se usa .size y un booleano que verifica un tamaño mayor a 0 para verificar que la lista fue creada

Plan y Diseño de pruebas:

*@Test*

*void TestDarCategoriaRaiz()*

*{assertEquals("Tecnología",this.almacen.darCategoriaRaiz().buscarNodo("11").darNombre());}*

* Para probar este método ejecutamos un método que incluya la función darCategoriaRaiz y verificamos que este funcione correctamente con assertEquals si darCategoriaRaiz no sirve correctamente el assertEquals debe de fallar.

*@Test*

*void TestCargar () throws AlmacenException*

*{assertNotNull(this.almacen, "No deberia de ser null");}*

* Para verificar este método se verifica que this.almacen no sea nulo, es decir que se cargaron correctamente los datos, si fuera nulo es que no cargaron de manera adecuada

*@Test*

*void TestAgregarNodo() throws AlmacenException*

*{this.almacen.agregarNodo("1", "Marca", "101", "Prueba");*

*assertNotNull(this.almacen.buscarNodo("101"), "No se agrego");*

* Para verificar este método se agrega un nuevo nodo y posteriormente se busca este nodo, al buscar el nodo se utiliza assertNotNull para verificar que se encontró y por ende se creó, si no se encuentra falla el assert y quiere decir que no se creo y fallo el método.

*@Test*

*void TestEliminarNodo () throws AlmacenException*

*{this.almacen.agregarNodo("1", "Marca", "101", "Prueba");*

*assertNotNull(this.almacen.buscarNodo("101"), "No se agrego");*

*this.almacen.eliminarNodo("101");*

*assertNull(this.almacen.buscarNodo("101"), "No se elimino");}*

* Para verificar este método se crea primero un nodo, luego se verifica que el nodo fue creado, se ejecuta el método de eliminar y con assertNull se verifica que el nodo ya no existe usando el método de buscarNodo. Si es diferente de nulo entonces significa que el nodo sigue y que no se borró.

*@Test*

*void TestVenderProducto () throws AlmacenException*

*{int ventasI = this.almacen.darCategoriaRaiz().buscarProducto("31759941").darCantidadUnidadesVendidas();*

*this.almacen.venderProducto("31759941", 10);*

*int ventasF = this.almacen.darCategoriaRaiz().buscarProducto("31759941").darCantidadUnidadesVendidas();*

*boolean prueba = ventasF > ventasI;*

*assertTrue(prueba);}*

* Para verificar este método primero se obtienen el numero de ventas actuales de un producto, luego se ejecuta el método de vender producto y se vuelve a obtener el numero de productos vendidos, finalmente se compara que el numero final sea mayor a la inicial y se prueba ese booleano con un assertTrue, indicando que aumento el numero de ventas y que el método sirve.

*@Test*

*void TestBuscarNodo ()*

*{assertNotNull(this.almacen.buscarNodo("1"), "No sirvio la busqueda");}*

* Para verificar este método se busca el nodo con id “1” el cual es conocido que, si existe, se verifica con un assertNotNull el método ya que como sabemos que el nodo existe, el retorno no debería de ser null.

*@Test*

*void TestAgregarProducto () throws AlmacenException*

*{this.almacen.agregarProducto("1111", "12345", "test", "sirve", 10);*

*assertNotNull(this.almacen.darCategoriaRaiz().buscarProducto("12345"));*

*assertEquals("test", this.almacen.darCategoriaRaiz().buscarProducto("12345").darNombre());}*

* Para verificar este método se agrega un nuevo producto, primero se verifica que esta creado con assertNotNull y el método buscarProducto el cual debería de retornar el producto. Como segundo método de verificación, luego de crear el producto, se saca el valor de su nombre y se compara con el string del nombre correspondiente, verificándolo con un assertEquals.

*@Test*

*void TestEliminarProducto()*

*{this.almacen.eliminarProducto("31759941");*

*assertNull(this.almacen.darCategoriaRaiz().buscarProducto("31759941"));}*

* Para verificar este método utilizando un código conocido, se ejecuta el método de eliminar producto, luego con un assertNull y la función de buscar producto se busca que el resultado sea null ya que ya no debería de existir.

*@Test*

*void TestMetodo1()*

*{assertEquals("Respuesta 1", this.almacen.metodo1());}*

* Para verificar este método se verifica que el retorno del método sea igual a la respuesta conocida “Respuesta 1” con el assertEquals.

*@Test*

*void TestMetodo2()*

*{assertEquals("Respuesta 2", this.almacen.metodo2());}*

* Para verificar este método se verfica que el retorno del método sea igual a la respuesta conocida “Respuesta 2” con el assertEquals.

*//INICIO DE LAS PRUEBAS DE CATEGORIA*

*@Test*

*void TestCategoria ()*

*{Categoria cat = new Categoria ("0101", "Prueba");*

*assertNotNull(cat);}*

* Para verificar este método se crea una nueva categoría y con assertNotNull se verifca que no sea nulo lo que se creo y que si sea una categoría.

*@Test*

*void TesTCategoria2 ()*

*{Categoria cat = this.almacen.darCategoriaRaiz();*

*assertNotNull(cat);}*

* Para verificar este método se obtiene una raíz conocida creada por el buffereader, la cual es la raíz y se verifica que no sea nulo con assertNotNull indicando que si quedo bien creado.

*@Test*

*void TestDarNodos ()*

*{Categoria cat = new Categoria ("0101", "Prueba");*

*List<NodoAlmacen> nodos = cat.darNodos();*

*int tam = nodos.size();*

*boolean p1 = tam==0;*

*assertTrue(p1);*

*List<NodoAlmacen> nodos1 = this.almacen.darCategoriaRaiz().darNodos();*

*int tam1 = nodos1.size();*

*boolean p2 = tam1 > 0;*

*assertTrue(p2);}*

* Para verificar este método se utilizan dos verificaciones, primero se crea un nodo sin hijo y se verifica que su lista de hijos este vacia con un assertTrue y un booleano comparando el tamaño y 0, la segunda verificación es que con un nodo conocido se verifica que su lista de hijos sea mayor a 0 esto de la misma forma que el anterior con un assertTrue y un booleano

*@Test*

*void TestTieneHijo ()*

*{boolean p = this.almacen.darCategoriaRaiz().tieneHijo("11");*

*assertTrue(p);}*

* Para verificar este método se utiliza un nodo que se sabe que tiene hijos, se verifica el booleano de si tiene hijos con un assertTrue

*@Test*

*void TestBuscarPadre ()*

*{NodoAlmacen nodo = this.almacen.darCategoriaRaiz().buscarPadre("11");*

*assertSame(nodo, this.almacen.darCategoriaRaiz());}*

* Para verificar este método se utiliza un nodo cuyo padre ya conocemos y ese padre lo comparamos con el resultado del método utilizando assertSame.

*@Test*

*void TestBuscarCNodo ()*

*{assertNotNull(this.almacen.darCategoriaRaiz().buscarNodo("11"), "No sirvio la busqueda");}*

* Para verificar este método se utiliza la misma lógica de buscar nodo de almacén sino que este caso accediendo desde una categoría y no desde almacén.

*@Test*

*void TestAgregarCNodo () throws AlmacenException*

*{this.almacen.darCategoriaRaiz().agregarNodo("1", "Marca", "101", "Prueba");*

*assertNotNull(this.almacen.darCategoriaRaiz().buscarNodo("101"), "No se agrego"); }*

* Para verificar este método se utiliza la misma lógica de agregar nodo de almacén, sino que este caso accediendo desde una categoría y no desde almacén.

*@Test*

*void TestEliminarCNodo () throws AlmacenException*

*{this.almacen.darCategoriaRaiz().agregarNodo("1", "Marca", "101", "Prueba");*

*assertNotNull(this.almacen.darCategoriaRaiz().buscarNodo("101"), "No se agregó");*

*this.almacen.darCategoriaRaiz().eliminarNodo("101");*

*assertNull(this.almacen.darCategoriaRaiz().buscarNodo("101"), "No se eliminó");}*

* Para verificar este método se utiliza la misma lógica de eliminar nodo de almacén, sino que este caso accediendo desde una categoría y no desde almacén.

*@Test*

*void TestBuscarProducto()*

*{assertNotNull(this.almacen.darCategoriaRaiz().buscarProducto("30747531"));}*

* Para verificar este método utilizando el código de un producto conocido se utiliza el método de buscar producto y con un assertNotNull se verifica que no sea nulo y que por ende se encontró un objeto.

*@Test*

*void TestDarMarcas ()*

*{List<Marca> lista = this.almacen.darCategoriaRaiz().darMarcas();*

*int tam = lista.size();*

*boolean pq = tam > 0;*

*assertTrue(pq); }*

* Para verificar este método se obtiene la lista de una categoría conocida y se compara que su tamaño no sea 0 utlizando un assertTrue para demostrar que si se obtuvo la lista.

*@Test*

*void TestDarPreorden ()*

*{List<NodoAlmacen> lista = this.almacen.darCategoriaRaiz().darPreorden();*

*int tam = lista.size();*

*boolean pq = tam > 0;*

*assertTrue(pq); }*

* Para verificar este método se obtiene la lista de una categoría conocida y se compara que su tamaño no sea 0 utlizando un assertTrue para demostrar que si se obtuvo la lista.

*@Test*

*void TestDarPosorden ()*

*{List<NodoAlmacen> lista = this.almacen.darCategoriaRaiz().darPosorden();*

*int tam = lista.size();*

*boolean pq = tam > 0;*

*assertTrue(pq); }*

* Para verificar este método se obtiene la lista de una categoría conocida y se compara que su tamaño no sea 0 utlizando un assertTrue para demostrar que si se obtuvo la lista.

*@Test*

*void TestDarValorVentas()*

*{this.almacen.venderProducto("31759941", 10);*

*int ventas = this.almacen.darCategoriaRaiz().buscarProducto("31759941").darCantidadUnidadesVendidas();*

*boolean prueba = ventas == 10;*

*assertTrue(prueba);}*

* Para verificar este método se utiliza el método de vender producto y se le pone un valor de vender 10, se sabe que el valor inicial de unidades vendidas es 0. Entonces luego se obtiene la cantidad de las ventas actuales y se compara con 10, lo cual debería ser cierto y se verifica con assertTrue.

Reflexiones Finales:

Muchas veces debido a que no se conocía con exactitud el retorno como por ejemplo las listas o algunos objetos, y encontrar con exactitud el resultado puede ser difícil e improductivo, se buscaba un método alternativo para verificar la creación de estos objetos o listas como al comparar su tamaño. A pesar de que esto no verifique en si la creación exacta si verifica la creación en si y decidimos que este camino era el adecuado. Creemos que hubo un balance adecuado entre las pruebas y la complejidad de estas para que sean útiles y no se conviertan en un obstáculo. También al revisar algunos métodos por defecto se revisa la funcionalidad de otros sin embargo, realizamos las pruebas individuales para estos métodos ya probados.