Pruebas:

Estrategia general: la estrategia usada fue crear para cada método una prueba, buscar la forma de saber si el resultado era correcto o no, ya fuera obtener algo diferente de null, además de verificar las respuestas.

Construcción de las pruebas: los escenarios de prueba iniciales se crearon con base en el documento ya establecido, con base en ese documento se satisfacían algunos requerimientos de otra forma también decidimos crear nodos y productos para casos en especifico.

Implementación: para implementarlas se uso el método de assert equals, para verificar contenido, y de asserttrue y false después de preguntars con ifs, para con estos validar los resultados y hacer mas efectivas y fáciles las comparaciones.

Dificultades: las mayores dificultades que encontramos fue no poder hacer verificaciones mas a fondo, es decir en métodos como los de dar, nuestra mayor posibilidad de verificación era simplemente verificar que no fuera null, pero no podíamos verificar los resultados internos debido a que no vimos la forma.

Plan de pruebas:

Ambiente: Junit test, eclipse, java, estas pruebas ya que no deben conectarse a través de un servidor o a una base de datos, no requieren de el uso o conexión a internet.

Estas pruebas crean unos escenarios de prueba que están previamente configurados, en la carpeta data, por lo que no requiere del ingreso de datos por parte del usuario, y solo se deben correr y ver los resultados que aseguran unos buenos resultados.

Instrucciones, para correr estas pruebas se necesitara eclipse, en donde debe estar alojado el proyecto, para correr cada una de estas pruebas solo será necesario hacerlo una vez por clase, de la siguiente manera: deben situarse en la clase test, de las dos clases participantes en estos test, sobre el archivo, darle click derecho para Windows y darle click con dos dedos para mac, a lo cual saldrá una cinta de opciones, dentro de las posibles opciones, seleccionar la que dice run as, y después Junit Test, y por ultimo verificar que el 100% de los test estén en completados o succesfull.

Pruebas

1. Almacén.
   1. Dar categoría:

Dado un ambiente normal, y un archivo de entrada, con la información de la tienda cargada, la prueba solicita la información y se comprueba que este archivo sea diferente de null, la respuesta debe ser satisfactoria, en tal caso que falle la prueba el error será: no debe retornar null.

* 1. Agregar nodo
     1. Test 1

Dado un ambiente normal, se decide crear un nodo, y después proceder a buscarlo, y así obtener que se encuentra, debido a que se agrego correctamente, después verificar que los datos coincidan con los ingresados en principio.

* + 1. Test 2

Dado un ambiente normal, se decide crear un nodo con datos, que generaran un error, con un padre que no existe y no se puede agregar, por ende este al buscar el nodo no debe encontrar nada ya que gracias a estos datos no se debió agregar.

* 1. Eliminar nodo
     1. Test 1

Dado un ambiente normal, se decide crear un nodo, y después eliminarlo, para con esto proceder a buscarlo y obtener que no se encuentra, debido a que se elimino correctamente.

* + 1. Test 2

Dado un ambiente normal, se decide crear un nodo y eliminar otro nodo que no existe, contrario al anterior método, este debe retornar el elemento agregado y así comprobar que se agrego correctamente y que se logro eliminar correctamente

* 1. Agregar producto
     1. Test 1

Dado un ambiente normal, se decide crear un producto, y después proceder a buscarlo, y así obtener que se encuentra, debido a que se agrego correctamente, después verificar que los datos coincidan con los ingresados en principio.

* + 1. Test 2

Dado un ambiente normal, se decide crear un producto con datos, que generaran un error, con un padre que no existe y no se puede agregar, por ende este al buscar el producto no debe encontrar nada ya que gracias a estos datos no se debió agregar.

* 1. Eliminar producto
     1. Test 1

Dado un ambiente normal, se decide crear un producto, y después eliminarlo, para con esto proceder a buscarlo y obtener que no se encuentra, debido a que se elimino correctamente.

* + 1. Test 2

Dado un ambiente normal, se decide crear un producto y no eliminarlo, contrario al anterior método, este debe retornar el elemento agregado y así comprobar que se agrego correctamente y que se logró eliminar correctamente.

* 1. Buscar nodo
     1. Test1

Dado un ambiente normal, Se procede a buscar un nodo existente y creado en el setup, verificamos la información obtenida, con la información con la que se creo ese nodo si el nodo es diferente de null y los datos coinciden el test es exitoso.

* + 1. Test 2

Dado un ambiente normal, Se busca un nodo que no existe, este debe retornar un null, ya que no está en el arreglo puesto que no existe.

* 1. Vender producto
     1. Test 1

Dado un ambiente normal, se decide crear un producto, y después proceder a buscarlo, para así registrar ventas sobre el, para después obtener el producto nuevamente y así obtener que se encuentra, y que las ventas coinciden con las registradas, debido a que se agrego correctamente la venta.

* + 1. Test 2

Dado un ambiente normal, se decide crear un producto con datos, correctos, pero con ventas menores varias veces, para después obtener el producto y así ver sus ventas y comprobar que todas estas se registraron correctamente, así que se compara el resultado de ventas con el experimental.

1. Categoría.
   1. Dar nodos
      1. Test 1

Dado un ambiente normal, y Debido a que se configura el ambiente de prueba 1, que tiene todos los elementos, este debe retornar una lista con elementos.

* + 1. Test 2

Dado un ambiente normal, y Debido a que se toma el ambiente de configuración dos, que solo tiene un nodo, el actual, este no tiene hijos por lo que debe retornar una lista vacía.

* 1. Buscar padre
     1. Test 1

Dado un ambiente normal, Se procede a buscar un nodo existente, no debe ser hijo necesariamente y creado en el setup, si el nodo es diferente de null el test es exitoso.

* + 1. Test 2

Dado un ambiente normal, Se busca un nodo que no existe, este debe retornar un null, ya que no está en el arreglo puesto que no existe.

* 1. Buscar nodo
     1. Test 1

Dado un ambiente normal, Se procede a buscar un nodo hijo existente y creado en el setup, verificamos la información obtenida, con la información con la que se creó ese nodo si el nodo es diferente de null y los datos coinciden el test es exitoso.

* + 1. Test 2

Dado un ambiente normal, Se busca un nodo que no existe, este debe retornar un null, ya que no está en el arreglo puesto que no existe.

* 1. Agregar nodo
     1. Test 1

Dado un ambiente normal, se decide crear un nodo, y después proceder a buscarlo, y así obtener que se encuentra, debido a que se agregó correctamente, después verificar que los datos coincidan con los ingresados en principio.

* + 1. Test 2

Dado un ambiente normal, se decide crear un nodo con datos, que generaran un error, con un padre que no existe y no se puede agregar, por ende este al buscar el nodo no debe encontrar nada ya que gracias a estos datos no se debió agregar.

* 1. Eliminar nodo
     1. Test 1

Dado un ambiente normal, se decide crear un producto, y después eliminarlo, para con esto proceder a buscarlo y obtener que no se encuentra, debido a que se eliminó correctamente.

* + 1. Test 2

Dado un ambiente normal, se decide crear un producto y no eliminarlo, contrario al anterior método, este debe retornar el elemento agregado y así comprobar que se agregó correctamente y que se logró eliminar correctamente

* 1. Buscar producto
     1. Test 1

Dado un ambiente normal, Este test busca un producto que existe y debe devolver el producto, así que este debe ser diferente de null.

* + 1. Test 2

Dado un ambiente normal, Este test intenta buscar un producto que no existe por consiguiente, debe ser null el resultado de la busqueda

* 1. Dar productos
     1. Test 1

Dado un ambiente normal, Para esta prueba se piden una lista de productos, que están inicializados y tienen una buena cantidad, por lo que se espera una lista con más de 1 producto.

* + 1. Test 2

Dado un ambiente normal, Para esta prueba debe retornar una lista vacía, pues se elige la configuración que solo tiene una categoría y que esta vacía.

* 1. Dar marcas
     1. Test 1

Dado un ambiente normal, En este caso, se crea en el escenario un conjunto de información con diversas marcas, como respuesta se espera que esta lista no este vacía y contenga las marcas.

* + 1. Test 2

Dado un ambiente normal, Para este se usa una configuración de un escenario que no tiene marcas, por lo que este debe retornar una lista vacía.

* 1. Dar preorden
     1. Test 1

Dado un ambiente normal, Debido a que se quiere verificar una lista dada, para el valor pre orden, sabiendo como deberían quedar ordenados, dependiendo de la configuración del escenario de prueba se valida que estos estén en pre orden y que este ordenamiento se haga correctamente.

* 1. Dar posorden
     1. Test 1

Dado un ambiente normal, Debido a que se quiere verificar una lista dada, para el valor pos orden, sabiendo como deberían quedar ordenados, dependiendo de la configuración del escenario de prueba se valida que estos estén en pos orden y que este ordenamiento se haga correctamente.

* 1. Dar valor ventas
     1. Test 1

Dado un ambiente normal, Debido a que no se ha generado ninguna venta, el valor de ventas debe ser igual a 0