Domingo 29 de mayo del 2022

Integrantes:

- Nicolas Merchan Cuestas - n.merchan@uniandes.edu.co – 202112109
- Julián Camilo Rivera Monroy - jc.riveram1@uniandes.edu.co - 202013338
- Juan Felipe Serrano - j.serrano@uniandes.edu.co - 201921654

Taller 7

Introducción

En las presentes pruebas se hará énfasis en los métodos individuales en las clases “Almacen” y “Categoria” de la aplicación en cuestión. Ello se da porque dichas clases están compuestas en su mayoría por métodos y pocos atributos. En ese orden de ideas, se procurará explicar las pruebas de manera individual por método y casos de prueba para dichos métodos. Primero se realizarán las pruebas de los métodos de la clase “Categoria”, dado que atributos y métodos de la clase “Almacen” dependen de la misma. Vale la pena resaltar que se asume que todas las demás clases, excepto “Almacen” funcionan correctamente.

Clase Categoria

- **Categoria**
    1. Retorna una categoría no nula cuando el archivo existe y está en el formato correcto (ConstructorCategoriaCargaExiste()).
    2. Retorna una excepción NullPointerException cuando el archivo existe, pero está vacío (ConstructorCategoriaCargaVacio()).
    3. Retorna una excepción AlmacenException cuando el archivo existe, pero está en el formato incorrecto (ConstructorCategoriaCargaIncorrecta()).
    4. Retorna una categoría no nula cuando se hace uso del constructor sin carga y con pIdentificador != null, pIdentificador != "", pNombre != null y pNombre != "" (ConstructorCategoriaNoCargaCorrecto).

  No se consideró el caso en el cual el archivo no existe porque aquel error se daría en la clase “Almacen”.

- **darNodos**
    1. Retorna una lista vacía con elementos de la clase nodosHijos cuando se crea un categoría vacía (darNodosCorrectoNoCarga()).
    2. Retorna una lista no vacía cunado se crea una categoría no vacía (darNodosCorrectoCarga()).

    Solo se verifica que retorne una lista con elementos de la clase deseada. Verificar que los elementos sean correctos y estén completos se verificará cuando se pruebe agregarNodo().

- **buscarNodo**
    1. Retorna un objeto de clase NodoAlmacen cuando pIdentificador ==”1” (buscarNodoCorrecto()).
    2. Retorna null cuando el arreglo nodosHijos está vacio.
    3. Retorna null cuando pIdentificador ==”2” (no existe) (buscarNodoNoExiste()).
    4. Retorna una excepción NullPointerException cuando se ingresa pIdentificador == null (buscarNodoNull()).
    5. Retorna null cuando se ingresa pIdentificador == “” (buscarNodoVacio()).

    En todas las pruebas se utilizó el constructor que carga el archivo. Así mismo, no se verifica de que clase es el objeto de retorno, ya que de los contrario no se podría ejecutar el método.

- **agregarNodo**
    1. Retorna un nodo con el nombre “Prueba” cuando se ingresa pIdPadre=”1”, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (agregarNodoExiste()).
    2. Retorna null cuando se ingresa pIdPadre=””, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (agregarNodoIncorrecto1()).
    3. Retorna un nodo con el nombre “Prueba” y con identificador “” cuando se ingresa pIdPadre=”1”, pTipo==”Test”, pIdentificador==”” y pNombre==”Prueba” (agregarNodoIncorrecto2()).
    4. Retorna null cuando se ingresa pIdPadre=null, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (agregarNodoIncorrecto3()).

En todas las pruebas se utilizó el constructor que carga el archivo y se dice que “retorna” cuando se busca el nodo indicado por medio del método “buscarNodo()”.

- **buscaPadre**
    1. Retorna un nodo con el identificador “1” cuando se ingresa pIdNodo=”2” para un nodo con pIdPadre=”1”, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (buscarPadreExiste()).
    2. Retorna null cuando se ingresa pIdNodo=”2” para un nodo con pIdPadre=null, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (buscarPadreNoExiste()).

    En todas las pruebas se utilizó el constructor que carga el archivo. Solo se consideraron los casos si el nodo padre existe o no. Así mismo, no se verifica de que clase es el objeto de retorno, ya que de los contrario no se podría ejecutar el método.

- **eliminarNodo**
    1. Retorna un nodo con el identificador “2” cuando se ingresa pIdNodo=”2” para un nodo con pIdPadre=”1”, pTipo==”Test”, pIdentificador==”2” y pNombre==”Prueba” (eliminarNodoExiste()).
    2. Retorna null cuando se ingresa pIdNodo=”1” y el arreglo nodosHijos está vacío (eliminarNodoNoExiste()).
    3. Retorna null cuando se ingresa pIdNodo=null y el arreglo nodosHijos no está vacío (eliminarNodoNull()).
    4. Retorna null cuando se ingresa pIdNodo=”” y el arreglo nodosHijos no está vacío (eliminarNodoVacio()).

    En todas las pruebas se utilizó el constructor que carga el archivo. Solo se consideraron los casos posibles de la única entrada. Así mismo, no se verifica de que clase es el objeto de retorno, ya que de los contrario no se podría ejecutar el método.

- **buscarProducto**
    1. Retorna un objeto diferente de null cuando se ingresa pCodigo=” 24881271” (buscarNodoExiste()).
    2. Retorna null cuando se ingresa pIdNodo=”24883241” el producto con dicho código no existe (buscarNodoNoExiste ()).
    3. Retorna null cuando se ingresa pIdNodo=null (buscarNodoNull()).
    4. Retorna null cuando se ingresa pIdNodo=”” (buscarNodoVacio()).

En todas las pruebas se utilizó el constructor que carga el archivo. Solo se consideraron los casos posibles de la única entrada. Así mismo, no se verifica de que clase es el objeto de retorno, ya que de los contrario no se podría ejecutar el método.

- **darMarcas**
    1. Retorna una lista de marcas con tamaño diferente de cero cuando el arreglo nodosHijos no está vacía.
    2. Retorna una lista de marcas con tamaño igual a cero cuando el arreglo nodosHijos está vacía.

    En todas las pruebas se utilizó el constructor que carga el archivo. Así mismo, no se verifica de que clase es el objeto de retorno, ya que de los contrario no se podría ejecutar el método.

A consideración del grupo, los demás métodos que no se probaron dependían enteramente en clases diferentes a “Categoria” y “Almacen”. Por lo tanto, el testeo de dichos métodos salía del alcance del presente taller.

Clase Almacen

- **Almacen**
    1. Retorna una categoría no nula cuando el archivo existe (ConstructorAlmacenCargaExiste()).
    2. Retorna una excepción de tipo AlmacenException cuando el archivo no existe (ConstructorAlmacenCargaNoExiste()).

El grupo considero que los demás métodos de la clase dependen directamente de métodos de otras clases. Por lo tanto, solo se testeo el constructor de la clase en cuestión.