Testeig mètodes de les classes

Mostrarem la documentació dels tests de tots els mètodes de les classes del sistema. Tota la documentació es troba també al codi en format commentari Javadoc a sobre de cada mètode

Test de la classe ProductList

testAddProduct

- **Objecte de la prova**: Comprova que el mètode addProduct() afegeix correctament un producte a la llista.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el producte afegit es troba dins de la llista.
- **Operativa**:
    1. Es crea una instància de ProductList i un objecte Product.
    2. Es crida al mètode addProduct() amb el producte com a paràmetre.
    3. Es comprova que el mètode retorna true i que el producte es troba dins de la llista.

testAddMultipleProducts

- **Objecte de la prova**: Comprova que el mètode addProduct() permet afegir múltiples productes a la llista.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que tots els productes afegits es troben dins de la llista.
- **Operativa**:
    1. Es generen diversos productes amb dades aleatòries.
    2. Es crida al mètode addProduct() per cada producte.
    3. Es comprova que cada producte afegit es troba dins de la llista.

testAddProductWithDifferentCategory

- **Objecte de la prova**: Comprova que el mètode addProduct() permet afegir un producte amb una categoria diferent.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el producte es pot afegir encara que la seva categoria sigui diferent.
- **Operativa**:
    1. Es crea un producte amb una categoria diferent a la de la llista.
    2. Es crida al mètode addProduct() amb aquest producte.
    3. Es comprova que el producte es troba dins de la llista.

testRemoveProduct

- **Objecte de la prova**: Comprova que el mètode removeProduct() elimina correctament un producte existent de la llista.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el producte eliminat ja no es troba a la llista.
- **Operativa**:
    1. Es crea una instància de ProductList amb un producte afegit.
    2. Es crida al mètode removeProduct() amb el nom del producte.
    3. Es comprova que el mètode retorna true i que el producte ha estat eliminat.

testRemoveNonExistentProduct

- **Objecte de la prova**: Comprova que el mètode removeProduct() retorna false quan s'intenta eliminar un producte inexistent.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que no es produeix cap canvi quan el producte no existeix.
- **Operativa**:
    1. Es crida al mètode removeProduct() amb el nom d'un producte inexistent.
    2. Es comprova que el mètode retorna false.

testRemoveMultipleProducts

- **Objecte de la prova**: Comprova que el mètode removeProduct() elimina correctament múltiples productes de la llista.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que cada producte eliminat ja no es troba a la llista.
- **Operativa**:
    1. Es generen diversos productes i s'afegeixen a la llista.
    2. Es crida al mètode removeProduct() per cada producte.
    3. Es comprova que cada producte ha estat eliminat correctament.

testApplyDiscount

- **Objecte de la prova**: Comprova que el mètode applyDiscount() aplica un descompte percentual correctament als productes de la llista.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el preu de cada producte és actualitzat amb el descompte aplicat.
- **Operativa**:
    1. S'afegeix un producte amb un preu conegut a la llista.
    2. Es crida al mètode applyDiscount(10) i es calcula el preu esperat.
    3. Es comprova que el preu resultant coincideix amb el preu esperat.

testApplyDiscountToMultipleProducts

- **Objecte de la prova**: Comprova que el mètode applyDiscount() aplica correctament un descompte a múltiples productes de la llista.

- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el preu de tots els productes és actualitzat correctament.
- **Operativa**:
    1. Es generen diversos productes i s'afegeixen a la llista.
    2. Es crida al mètode applyDiscount(10) per a tota la llista.
    3. Es comprova que els preus dels productes coincideixen amb els valors esperats.

testToString

- **Objecte de la prova**: Comprova que el mètode toString() retorna una representació textual correcta de la llista de productes.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que la representació inclou tots els productes de la llista en l'ordre esperat.
- **Operativa**:
    1. Es creen diversos productes i s'afegeixen a la llista.
    2. Es genera manualment la sortida esperada de toString().
    3. Es compara la sortida del mètode amb el resultat esperat

Test de la classe Product

testGetName

- **Objecte de la prova**: Comprova que el mètode getName() retorna el nom correcte del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el valor retornat coincideix amb el nom assignat durant la creació de l'objecte.
- **Operativa**:
    1. Es crea una instància de Product amb el nom "Test Product".
    2. Es crida al mètode getName() i es comprova que el valor retornat és "Test Product".

testGetCategory

- **Objecte de la prova**: Comprova que el mètode getCategory() retorna la categoria correcta del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el valor retornat coincideix amb la categoria assignada durant la creació de l'objecte.
- **Operativa**:
    1. Es crea una instància de Product amb la categoria "Category".
    2. Es crida al mètode getCategory() i es comprova que el valor retornat és "Category".

testSetCategory

- **Objecte de la prova**: Verifica que el mètode setCategory() actualitza correctament la categoria del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es comprova que el valor retornat per getCategory() coincideix amb el valor establert.
- **Operativa**:
    1. Es crea una instància de Product.
    2. Es crida a setCategory("New Category").
    3. Es comprova que getCategory() retorna "New Category".

testSetNullCategory

- **Objecte de la prova**: Comprova que el mètode setCategory() llença una excepció quan la categoria és null.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que es llença una excepció de tipus IllegalArgumentException.
- **Operativa**:
    1. Es crida a setCategory(null).
    2. Es comprova que es llença l'excepció esperada.

testSetEmptyCategory

- **Objecte de la prova**: Comprova que el mètode setCategory() llença una excepció quan la categoria és una cadena buida.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que es llença una excepció de tipus IllegalArgumentException.
- **Operativa**:
    1. Es crida a setCategory("").
    2. Es comprova que es llença l'excepció esperada.

testGetPrice

- **Objecte de la prova**: Comprova que el mètode getPrice() retorna el preu correcte del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el valor retornat coincideix amb el preu assignat durant la creació de l'objecte.
- **Operativa**:
    1. Es crea una instància de Product amb un preu de 10.0.
    2. Es crida al mètode getPrice() i es comprova que el valor retornat és 10.0.

testSetPrice

- **Objecte de la prova**: Verifica que el mètode setPrice() actualitza correctament el preu del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es comprova que el valor retornat per getPrice() coincideix amb el valor establert.
- **Operativa**:
    1. Es crea una instància de Product.
    2. Es crida a setPrice(20.0).
    3. Es comprova que getPrice() retorna 20.0.

testSetNegativePrice

- **Objecte de la prova**: Comprova que el mètode setPrice() llença una excepció quan el preu és negatiu.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que es llença una excepció de tipus IllegalArgumentException.
- **Operativa**:
    1. Es crida a setPrice(-5.0).
    2. Es comprova que es llença l'excepció esperada.

testGetAmount

- **Objecte de la prova**: Comprova que el mètode getAmount() retorna la quantitat correcta del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el valor retornat coincideix amb la quantitat assignada durant la creació de l'objecte.
- **Operativa**:
    1. Es crea una instància de Product amb una quantitat de 5.
    2. Es crida al mètode getAmount() i es comprova que el valor retornat és 5.

testSetAmount

- **Objecte de la prova**: Verifica que el mètode setAmount() actualitza correctament la quantitat del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es comprova que el valor retornat per getAmount() coincideix amb el valor establert.
- **Operativa**:
    1. Es crea una instància de Product.
    2. Es crida a setAmount(10).
    3. Es comprova que getAmount() retorna 10.

testSetNegativeAmount

- **Objecte de la prova**: Comprova que el mètode setAmount() llença una excepció quan la quantitat és negativa.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que es llença una excepció de tipus IllegalArgumentException.
- **Operativa**:
    1. Es crida a setAmount(-3).
    2. Es comprova que es llença l'excepció esperada.

testApplyDiscount

- **Objecte de la prova**: Verifica que el mètode applyDiscount() aplica correctament un descompte percentual al preu.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que el preu resultant correspon al percentatge aplicat.
- **Operativa**:
    1. Es crea una instància de Product amb un preu de 10.0.
    2. Es crida a applyDiscount(10).
    3. Es comprova que getPrice() retorna 9.0.

testApplyInvalidDiscount

- **Objecte de la prova**: Comprova que el mètode applyDiscount() llença una excepció si el descompte és superior al 100%.

- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que es llença una excepció de tipus IllegalArgumentException.
- **Operativa**:
    1. Es crida a applyDiscount(110).
    2. Es comprova que es llença l'excepció espera

testToString

- **Objecte de la prova**: Comprova que el mètode toString() retorna la representació textual correcta del producte.
- **Fitxers de dades necessaris**: No calen.
- **Valors estudiats**: Estratègia caixa gris. Es compara la sortida del mètode amb el resultat esperat.
- **Operativa**:
    1. Es crea una instància de Product amb atributs coneguts.
    2. Es crida a toString() i es compara amb una cadena generada manualment.

Test de la classe BruteForceAlgorithm

testOrderProductList

- **Objecte de la prova**: Comprova el correcte funcionament del mètode orderProductList() de la classe BruteForceAlgorithm.
- **Fitxers de dades necessaris**: No calen. Les dades són simulades amb mock objects.
- **Valors estudiats**: Estratègia caixa grisa. Es verifica que el resultat de l'ordenació dels productes compleix amb les expectatives predefinides.
- **Operativa**:
    1. Es crea una llista simulada de productes (mockProducts) on cada producte té un nom i puntuacions de similitud configurades manualment amb altres productes.
    2. Es configura un ProductList simulat (mockProductList) que retorna aquest conjunt de productes.
    3. Es crida al mètode orderProductList() amb el mockProductList i un valor límit.
    4. Es verifica que la sortida del métode és:
        - No nul·la.
        - Conté 3 grups de productes.
        - Els grups tenen les següents mides: 3 productes en el primer grup, 3 en el segon i 4 en el tercer.

Test de la classe HillClimbingAlgorithm

testOrderProductList

- **Objecte de la prova**: Comprova el correcte funcionament del mètode orderProductList() de la classe HillClimbingAlgorithm.
- **Fitxers de dades necessaris**: No calen. Les dades són generades i simulades amb mocks.
- **Valors estudiats**: Estratègia caixa gris. Es verifica que la llista de productes generada compleix amb els requisits esperats: que els productes estan agrupats correctament en funció de les seves similituds.
- **Operativa**:
    1. Es configura una llista de productes (mockProducts) on cada producte té una similitud definida amb altres productes de la llista.
    2. Es crea un conjunt simulat de productes (mockProductList) que retorna aquest conjunt quan es crida al mètode getProducts().
    3. Es crida al mètode orderProductList() amb el conjunt simulat i un límit de 10 productes.
    4. Es verifica que la sortida del mètode és:
        - No nul·la (assertNotNull).
        - Conté 3 grups de productes.
        - Els grups tenen les següents mides: 3 productes en el primer grup, 3 en el segon i 4 en el tercer.

SimilarityTest

testGetScore()

- **Objecte de la prova:** Test del mètode getSimilarityScore() de la classe Similarity.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna el valor correcte de la puntuació de similitud.
- **Operativa:** Es crea una instància de la classe Similarity amb una puntuació inicial de 5.0. S'invoca el mètode getSimilarityScore() i es comprova que el resultat és 5.0 amb un marge de tolerància de 0.01.

---

testGetProduct1()

- **Objecte de la prova:** Test del mètode getProduct1() de la classe Similarity.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna l'objecte del primer producte passat al constructor.
- **Operativa:** Es crea una instància de la classe Similarity amb un mock del primer producte. S'invoca el mètode getProduct1() i es comprova que retorna l'objecte mockProduct1.

---

testGetProduct2()

- **Objecte de la prova:** Test del mètode getProduct2() de la classe Similarity.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna l'objecte del segon producte passat al constructor.
- **Operativa:** Es crea una instància de la classe Similarity amb un mock del segon producte. S'invoca el mètode getProduct2() i es comprova que retorna l'objecte mockProduct2.

---

testGetOtherProduct()

- **Objecte de la prova:** Test del mètode getOtherProduct(Product product) de la classe Similarity.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna l'altre producte en funció del que es passa com a paràmetre.
- **Operativa:**
    1. Es crea una instància de la classe Similarity amb dos mocks de productes (mockProduct1 i mockProduct2).
    2. Es passa mockProduct1 com a paràmetre al mètode getOtherProduct() i es comprova que retorna mockProduct2.
    3. Es passa mockProduct2 com a paràmetre al mètode getOtherProduct() i es comprova que retorna mockProduct1.

ShelfIntegrationTest

testDistribute()

- **Objecte de la prova:** Test del mètode distribute(String name) de la classe Shelf.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode crea una distribució amb el nom proporcionat i que es pot canviar l'algoritme d'assignació per generar una altra distribució.
- **Operativa:**
    1. Es crida al mètode distribute() amb el nom "Test Distribution" i es comprova que el nom de la distribució retornada és correcte.
    2. Es canvia l'algoritme utilitzat a BruteForceAlgorithm i es crida novament a distribute() amb el nom "Test Distribution 2".
    3. Es comprova que el nom de la nova distribució coincideix amb el passat per paràmetre.

---

testGetNameLastDistribution()

- **Objecte de la prova:** Test del mètode getNameLastDistribution() de la classe Shelf.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna el nom de l'última distribució creada.
- **Operativa:**
    1. Es crida al mètode distribute() amb el nom "Test Distribution".
    2. Es comprova que getNameLastDistribution() retorna el nom "Test Distribution".

---

testGetLastDistribution()

- **Objecte de la prova:** Test del mètode getLastDistribution() de la classe Shelf.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.

- **Valors estudiats:** Estratègia caixa gris. Es comprova que el mètode retorna correctament l'última distribució creada.
- **Operativa:**
    1. Es crea una distribució utilitzant el mètode distribute() amb el nom "Test Distribution".
    2. Es comprova que getLastDistribution() retorna la distribució creada.

---

testChangeAlgorithm()

- **Objecte de la prova:** Test del mètode changeAlgorithm(AbstractAlgorithm algorithm) de la classe Shelf.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que l'algoritme de distribució es pot modificar.
- **Operativa:**
    1. Es crea un nou algoritme de tipus BruteForceAlgorithm.
    2. Es crida al mètode changeAlgorithm() amb el nou algoritme.
    3. Es comprova que l'algoritme actual retornat per getAbstractAlgorithm() és l'objecte passat.

---

testChangeProductList()

- **Objecte de la prova:** Test del mètode changeProductList(ProductList productList) de la classe Shelf.
- **Fitxers de dades necessaris:** Dades introduïdes manualment. No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la llista de productes es pot canviar correctament.
- **Operativa:**
    1. Es crea una nova llista de productes anomenada "Test Product List 2".
    2. Es crida al mètode changeProductList() amb la nova llista.
    3. Es comprova que el resultat de getProductList() és la nova llista passada per paràmetre i que no és igual a la llista original.

Testeig dels mètodes de la classe DriverProductManager

---

testCreateProductList()

- **Objecte de la prova:**
  Provar el mètode createProductList() per verificar que es pot crear una llista de productes amb nom i categoria vàlids.

- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que una llista de productes es crea correctament al sistema.
- **Operativa:**
  1. Es demanen el nom i la categoria de la llista.
  2. Es crida al mètode createProductList() de ProductManager.
  3. Es comprova si la llista es crea amb èxit.

---

testAddProductToList()

- **Objecte de la prova:**
  Provar el mètode addProductToList() per afegir un producte existent del catàleg a una llista de productes.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es verifica que el producte es pot associar correctament a una llista de productes.
- **Operativa:**
  1. Es demanen el nom de la llista i del producte.
  2. Es crida al mètode addProductToList().
  3. Es comprova si el producte s'afegeix correctament.

---

testRemoveProductFromList()

- **Objecte de la prova:**
  Provar el mètode removeProductFromList() per eliminar un producte d'una llista.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es verifica que el producte es pot eliminar correctament d'una llista de productes.
- **Operativa:**
  1. Es demanen el nom de la llista i del producte a eliminar.
  2. Es crida al mètode removeProductFromList().
  3. Es comprova si el producte s'elimina correctament.

---

testAddProductToCatalog()

- **Objecte de la prova:**
  Provar el mètode addProductToCatalog() per verificar que un producte amb atributs vàlids es pot afegir al catàleg.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.

- **Valors estudiats:**
  Es comprova que el producte s'afegeix correctament al catàleg.
- **Operativa:**
  1. Es demanen el nom, la categoria, el preu i la quantitat del producte.
  2. Es crida al mètode addProductToCatalog().
  3. Es comprova si el producte s'afegeix amb èxit.

---

testGetProductList()

- **Objecte de la prova:**
  Provar el mètode getProductList() per obtenir una llista de productes per nom.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que el nom i la categoria de la llista retornada són correctes.
- **Operativa:**
  1. Es demana el nom de la llista.
  2. Es crida al mètode getProductList().
  3. Es comprova que la llista es recupera correctament.

---

testApplyDiscountToList()

- **Objecte de la prova:**
  Provar el mètode applyDiscountToList() per aplicar un descompte a una llista de productes.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que el descompte aplicat s'actualitza correctament.
- **Operativa:**
  1. Es demana el nom de la llista i el percentatge de descompte.
  2. Es crida al mètode applyDiscountToList().
  3. Es comprova que el descompte s'aplica correctament.

---

testIncreaseProductQuantity()

- **Objecte de la prova:**
  Provar el mètode increaseProductQuantity() per incrementar la quantitat d'un producte.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que el nombre d'unitats del producte s'incrementa correctament.

- **Operativa:**
    1. Es demanen el nom del producte i la quantitat a incrementar.
    2. Es crida al mètode increaseProductQuantity().
    3. Es verifica que la quantitat final és correcta.

---

testDecreaseProductQuantity()

- **Objecte de la prova:**
  Provar el mètode decreaseProductQuantity() per disminuir la quantitat d'un producte.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que el nombre d'unitats del producte disminueix correctament.
- **Operativa:**
    1. Es demanen el nom del producte i la quantitat a disminuir.
    2. Es crida al mètode decreaseProductQuantity().
    3. Es verifica que la quantitat final és correcta.

---

testAddSimilarity()

- **Objecte de la prova:**
  Provar el mètode addSimilarity() per afegir semblances entre productes.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es verifica que es poden associar dos productes amb una semblança determinada.
- **Operativa:**
    1. Es demanen els noms dels productes i el valor de semblança.
    2. Es crida al mètode addSimilarity().
    3. Es comprova que la semblança es registra correctament.

---

testRemoveSimilarity()

- **Objecte de la prova:**
  Provar el mètode removeSimilarity() per eliminar una semblança entre productes.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs; dades introduïdes manualment.
- **Valors estudiats:**
  Es comprova que la semblança s'elimina correctament.
- **Operativa:**
    1. Es demanen els noms dels productes.
    2. Es crida al mètode removeSimilarity().
    3. Es comprova que ja no hi ha semblança registrada entre els productes.

---

testShowSimilarities()

- **Objecte de la prova:**
  Provar el mètode showSimilarities() per llistar totes les semblances registrades entre productes.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs.
- **Valors estudiats:**
  Es comprova que totes les semblances es llisten correctament.
- **Operativa:**
  1. Es crida al mètode showSimilarities().
  2. Es comprova que les semblances es mostren correctament.

---

testShowAllProducts()

- **Objecte de la prova:**
  Provar el mètode getAllProducts() per mostrar tots els productes registrats al catàleg.
- **Fitxers de dades necessaris:**
  No es requereixen fitxers externs.
- **Valors estudiats:**
  Es comprova que tots els productes es llisten correctament.
- **Operativa:**
  1. Es crida al mètode getAllProducts().
  2. Es comprova que tots els productes es mostren.

DriverControllerDomain

showMenu()

- **Objecte de la prova:** Test del menú principal, incloent la navegació per les opcions relacionades amb fitxers, productes i prestatgeries.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que les opcions del menú es mostren correctament i que es navega a altres menús segons l'opció seleccionada.
- **Operativa:**
  1. Es presenta un menú amb opcions numerades.
  2. Es comprova que les opcions 1, 2, i 3 dirigeixen correctament als menús de fitxers, productes i prestatgeries respectivament.
  3. Es verifica que l'opció 0 permet sortir del programa.
  4. Es gestionen entrades no vàlides amb missatges d'error.

---

showShelfMenu()

- **Objecte de la prova:** Test del menú per gestionar prestatgeries, incloent la creació i modificació de distribucions i algoritmes.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que les opcions del menú es processen adequadament.
- **Operativa:**
    1. Es presenta un menú amb opcions per crear prestatgeries, canviar algoritmes, mostrar prestatgeries, gestionar distribucions, entre altres.
    2. Es comprova que cada opció seleccionada executa el mètode corresponent: createShelf, changeShelfAlgorithm, showAllShelfs, etc.
    3. Es verifica la gestió d'entrades no vàlides amb missatges d'error.

---

showParserMenu()

- **Objecte de la prova:** Test del menú per a operacions amb fitxers, incloent la lectura de dades i la creació de fitxers de distribució.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que les opcions es processen correctament.
- **Operativa:**
    1. Es presenta un menú amb opcions per llegir dades d'un fitxer extern o crear un fitxer de distribució.
    2. Es comprova que l'opció 1 crida el mètode getDataThroughFile i l'opció 2 executa crateNewDistributionFile.
    3. Es verifica la gestió d'errors per entrades no vàlides.

---

showProductMenu()

- **Objecte de la prova:** Test del menú per a operacions amb productes i llistes de productes, incloent l'addició, eliminació, modificació i visualització.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que cada opció del menú executa la funcionalitat esperada.
- **Operativa:**
    1. Es presenta un menú amb múltiples opcions per gestionar productes i llistes de productes.
    2. Es comprova que les opcions corresponents criden els mètodes adequats, com ara addProductToCatalog, removeProductFromCatalog, createNewProductList, etc.
    3. Es verifica la gestió de casos d'ús no vàlids amb missatges d'error.

---

modifyDistribution()

- **Objecte de la prova:** Test del mètode modifyDistribution() per modificar una distribució intercanviant la posició de dos productes.

- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que es realitza l'intercanvi correcte entre productes especificats.
- **Operativa:**
    1. Es sol·licita el nom de la distribució i dels dos productes que s'intercanviaran.
    2. Es crida al mètode modifyDistribution del controlador amb els valors introduïts.
    3. Es verifica que es mostren missatges d'èxit o d'error segons sigui el cas.

---

createNewDistribution()

- **Objecte de la prova:** Test del mètode createNewDistribution() per crear una nova distribució associada a una prestatgeria.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la distribució es crea correctament amb el nom i l'ID de prestatgeria proporcionats.
- **Operativa:**
    1. Es sol·liciten l'ID de la prestatgeria i el nom de la distribució.
    2. Es crida al mètode createNewDistribution del controlador amb els valors introduïts.
    3. Es verifica que es mostren missatges d'èxit o d'error segons sigui el cas.

removeList()

- **Objecte de la prova:** Test del mètode removeList() per eliminar una llista de productes del sistema pel seu nom.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la llista especificada s'elimina correctament.
- **Operativa:**
    1. Es demana a l'usuari el nom de la llista a eliminar.
    2. Es crida al mètode removeProductList() del controlador de domini.
    3. Es comprova que es mostra un missatge d'èxit o un missatge d'error en cas d'excepció (ShelfException o ProductListException).

---

changeProductListAtShelf()

- **Objecte de la prova:** Test del mètode changeProductListAtShelf() per associar una nova llista de productes a una prestatgeria especificada.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que es canvia correctament la llista associada a la prestatgeria.
- **Operativa:**
    1. Es demanen a l'usuari l'ID de la prestatgeria i el nom de la nova llista a associar.
    2. Es crida al mètode changeProductListAtShelf() del controlador.
    3. Es mostra un missatge d'èxit o d'error segons el resultat de l'operació (ShelfException, ProductListException, o IllegalArgumentException).

---

showListChanges()

- **Objecte de la prova:** Test del mètode showListChanges() per mostrar els canvis realitzats a totes les llistes de productes.
- **Fitxers de dades necessaris:** No calen fitxers addicionals.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que es mostren correctament els canvis registrats.
- **Operativa:**
    1. Es crida al mètode showListChanges() del controlador de domini.
    2. Es verifica que la informació retornada es mostra correctament.

---

increaseProductQuantity()

- **Objecte de la prova:** Test del mètode increaseProductQuantity() per augmentar la quantitat d'un producte especificat.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la quantitat del producte augmenta correctament.
- **Operativa:**
    1. Es demana el nom del producte i la quantitat a augmentar.
    2. Es crida al mètode increaseProductQuantity() del controlador amb aquests valors.
    3. Es verifica que es mostren missatges d'èxit o d'error segons sigui el cas (ProductException o IllegalArgumentException).

---

decreaseProductQuantity()

- **Objecte de la prova:** Test del mètode decreaseProductQuantity() per reduir la quantitat d'un producte especificat.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la quantitat del producte es redueix correctament.
- **Operativa:**
    1. Es demana el nom del producte i la quantitat a reduir.
    2. Es crida al mètode decreaseProductQuantity() del controlador amb aquests valors.
    3. Es verifica que es mostren missatges d'èxit o d'error segons sigui el cas (ProductException o IllegalArgumentException).

---

applyDiscountToList()

- **Objecte de la prova:** Test del mètode applyDiscountToList() per aplicar un descompte a una llista de productes especificada.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el descompte s'aplica correctament a la llista.

- **Operativa:**
    1. Es demanen el nom de la llista i el percentatge de descompte.
    2. Es crida al mètode applyDiscountToList() del controlador.
    3. Es mostren missatges d'èxit o d'error segons el resultat (ProductListException o IllegalArgumentException).

---

createNewProductList()

- **Objecte de la prova:** Test del mètode createNewProductList() per crear una nova llista de productes.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que la llista es crea correctament.
- **Operativa:**
    1. Es demanen el nom i la categoria de la nova llista.
    2. Es crida al mètode createProductList() del controlador.
    3. Es mostra un missatge d'èxit o d'error segons el resultat (ProductListException).

---

addProductToCatalog()

- **Objecte de la prova:** Test del mètode addProductToCatalog() per afegir un producte al catàleg.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.
- **Valors estudiats:** Estratègia caixa gris. Es comprova que el producte s'afegeix correctament al catàleg i que s'estableixen similituds amb altres productes.
- **Operativa:**
    1. Es demanen el nom, la categoria, el preu, la quantitat, i similituds amb altres productes existents.
    2. Es crida al mètode addProductToCatalog() del controlador.
    3. Es mostren missatges d'èxit o d'error segons sigui el cas (ProductException o IllegalArgumentException).

---

createShelf()

- **Objecte de la prova:** Test del mètode createShelf() per crear una prestatgeria nova.
- **Fitxers de dades necessaris:** No calen fitxers addicionals, dades introduïdes manualment.

- **Valors estudiats:** Estratègia caixa gris. Es comprova que la prestatgeria es crea correctament amb els paràmetres especificats.
- **Operativa:**
    1. Es demanen l'ID, la mida, el nom de la llista i el tipus d'algoritme a associar.
    2. Es crida al mètode createShelf() del controlador.
    3. Es mostren missatges d'èxit o d'error segons sigui el cas (ShelfException, ProductListException, o IllegalArgumentException).