**Formato de escenarios y casos de uso**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupStageProduct1 | ProductTest | Un objeto de la clase Product con productName = "PS5", description = "VideoGame", price = 2000000.0, quantityAvailable = 4, category = "Toys and Games", purchasedNumber = 2. |
| setupStageOrder1 | OrderTest |  |
| setupStageProductList1 | ProductListTest | Un objeto de la clase ProductList sin productos. |
| setupStageProductList2 | ProductListTest | Un objeto de la clase ProductList con tres objetos de la clase Product:   * Primero con: productName = “PS5”, description = “VideoGame”, price = 2000000.0, quantityAvailable = 4, category = “Toys and Games”, purchasedNumber = 2 * Segundo con: productName = “Hamburguesa”, description = “Pan con carne”, price = 25000.0, quantityAvailable = 50, category = “Foods and Drinks”, purchasedNumber = 3 * Tercero con: productName = “Camiseta”, description = “Camiseta oversize”, price = 70000.0, quantityAvailable = 20, category = “Clothes and\_accesories”, purchasedNumber = 10 |
| setupStageProductList3 | ProductListTest | Un objeto de la clase ProductList con cinco objetos de la clase Product:   * Primero con: productName = “PS5”, description = “VideoGame”, price = 2000000.0, quantityAvailable = 4, category = “Toys and Games”, purchasedNumber = 2 * Segundo con: productName = “Hamburguesa”, description = “Pan con carne”, price = 25000.0, quantityAvailable = 50, category = “Foods and Drinks”, purchasedNumber = 3 * Tercero con: productName = “Camiseta”, description = “Camiseta oversize”, price = 70000.0, quantityAvailable = 20, category = “Clothes and\_accesories”, purchasedNumber = 10 * Cuarto con: productName = “XBOX”, description = “VideoGame”, price = 2500000.0, quantityAvailable = 7, category = “Toys and Games”, purchasedNumber = 3 * Quinto con: productName = “Balón”, description = “Balón de fútbol”, price = 50000.0, quantityAvailable = 20, category = “Sports”, purchasedNumber = 5 |
| setupStageOrderList1 | OrderListTest | Un objeto de la clase OrderList sin pedidos. |

|  |  |  |
| --- | --- | --- |
| setupStageOrderList2 | OrderListTest | Un objeto de la clase OrderList con dos objetos de la clase Order:   * Primero con: buyerName = “Santiago”, ProductList = “PS5, PSP, Audífonos”, productQuantity = 5, 3, 10, totalPrice = 2000000.0, purchaseDate = 2023-04-05 * Segundo con: buyerName = “James”, ProductList = “Camiseta, Pantaloneta, Guayos”, productQuantity = 2, 2, 1, totalPrice=250000.0, purchaseDate = 2023-04-12 |
| setupStageOrderList3 | OrderListTest | Un objeto de la clase OrderList con tres objetos de la clase Order:   * Primero con: buyerName = “Santiago”, ProductList = “PS5, PSP, Audífonos”, productQuantity = 5, 3, 10, totalPrice = 2000000.0, purchaseDate = 2023-04-05 * Segundo con: buyerName = “James”, ProductList = “Camiseta, Pantaloneta, Guayos”, productQuantity = 2, 2, 1, totalPrice=250000.0, purchaseDate = 2023-04-12 * Tercero con: buyerName = “Luis”, ProductList = “XBOX, Mando inalámbrico, Audífonos”, productQuantity = 3, 5, 15, totalPrice = 3000000.0, purchaseDate = 2023-04-05 |

**Diseño de Casos de Prueba**

**RF2 – El programa debe permitir al usuario ingresar información sobre cada producto.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método addProduct de la clase ProductList funcione correctamente, creando con éxito el objeto Product. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| ProductList | addProduct | setupStageProductList1 | productName = “PS5”, description = “VideoGame”, price = 2000000.0, quantityAvailable = 4, category = “Toys and Games”, purchasedNumber = 2 | El producto es añadido a la lista de productos. |
| ProductList | addProduct | setupStageProductList1 | productName = “PS5”, description = “VideoGame”, price =  -2000000.0, quantityAvailable = 4, category = “Toys and Games”, purchasedNumber = 2 | El producto no es añadido a la lista de productos. Adicionalmente, se lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |
| ProductList | addProduct | setupStageProductList1 | productName = “PS5”, description = “VideoGame”, price = 2000000.0, quantityAvailable = -4, category = “Toys and Games”, purchasedNumber = 2 | El producto no es añadido a la lista de productos. Adicionalmente, se lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |
| ProductList | addProduct | setupStageProductList1 | productName = “PS5”, description = “VideoGame”, price = 2000000.0, quantityAvailable = 4, category = “Toys and Games”, purchasedNumber = -2 | El producto no es añadido a la lista de productos. Adicionalmente, se lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |

**RF3 – El programa debe permitir al usuario eliminar un producto del inventario.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método deleteProduct de la clase ProductList funcione correctamente, removiendo de la lista el objecto Product deseado. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| ProductList | deleteProduct | setupStageProductList2 | productName = “Hamburguesa” | La lista de productos ahora tiene 2 productos: PS5 y Camiseta. |
| ProductList | deleteProduct | setupStageProductList2 | productName = “Coca cola” | La lista de productos conserva sus 3 productos. Adicionalmente, lanza una excepción y un mensaje de error. |

**RF4 – El programa debe permitir al usuario aumentar la cantidad de cada producto ya registrado.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método addQuantityAvailable de la clase Product funcione correctamente, adicionando la cantidad pasada por parámetro a la cantidad actual del producto. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Product | addQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Add = 4 | La cantidad del producto se ha actualizado a 8. |
| Product | addQuantityAvailable | setupStageProduct1 | productName = “Xbox”, quantity2Add = 3 | Lanza una Excepción y el mensaje dice que el producto no ha sido registrado. |
| Product | addQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Add = -4 | La cantidad del producto se mantiene en 4. Adicionalmente, lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |
| Product | addQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Add = 2.5 | La cantidad del producto se mantiene en 4. Adicionalmente, lanza una Excepción y el mensaje dice que no se pueden usar números decimales. |

**RF5 – El programa debe permitir al usuario ingresar información sobre cada pedido.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método addOrder de la clase OrderList funcione correctamente, creando con éxito el objeto Order. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| OrderList | addOrder | setupStageOrderList1 | buyerName = “James”, ProductList = “Camiseta, Pantaloneta, Guayos”, productQuantity = 2, 2, 1, totalPrice = 250000.0, purchaseDate = 2023-04-12 | El pedido es añadido a la lista de pedidos. |
| OrderList | addOrder | setupStageOrderList1 | buyerName = “James”, ProductList = “Camiseta, Pantaloneta, Guayos”, productQuantity = 2, 2, 1, totalPrice = -250000.0, purchaseDate = 2023-04-12 | El pedido no es añadido a la lista de pedido. Adicionalmente, se lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |

**RF6 – El programa debe permitir disminuir las cantidades del inventario al momento de generar un pedido.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método subtractQuantityAvailable de la clase Product funcione correctamente, disminuyendo la cantidad pasada por parámetro a la cantidad actual del producto. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| Product | subtractQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Subtract = 4 | La cantidad del producto se ha actualizado a 0. |
| Product | subtractQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Subtract = 7 | La cantidad del producto se mantiene en 4. Adicionalmente, lanza una Excepción y el mensaje dice que la cantidad a comprar es mayor que las unidades disponibles. |
| Product | subtractQuantityAvailable | setupStageProduct1 | productName = “Xbox”, quantity2Subtract = 3 | Lanza una Excepción y el mensaje dice que el producto no ha sido registrado. |
| Product | subtractQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Subtract = -4 | La cantidad del producto se mantiene en 4. Adicionalmente, lanza una Excepción y el mensaje dice que no se pueden usar números negativos. |
| Product | subtractQuantityAvailable | setupStageProduct1 | productName = “PS5”, quantity2Subtract = 2.5 | La cantidad del producto se mantiene en 4. Adicionalmente, lanza una Excepción y el mensaje dice que no se pueden usar números decimales. |

**RF7 – El programa debe permitir al usuario eliminar un pedido.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método deleteOrder de la clase OrderList funcione correctamente, removiendo de la lista el objecto Order deseado. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| OrderList | deleteOrder | setupStageOrderList2 | buyerName = “Santiago” | El pedido es eliminado de la lista de pedidos. Ahora tiene 1 pedido. |
| OrderList | deleteOrder | setupStageOrderList2 | buyerName = “David” | La lista de productos conserva sus 2 pedidos. Además, lanza una excepción y un mensaje de error. |

**RF8 – El programa debe contar con un buscador de productos que permita buscar productos por sus características principales (*nombre, precio, categoría y número de veces comprado*).**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método searchProduct de la clase ProductList funcione correctamente, permitiendo buscar un producto por alguna de sus características principales. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| ProductList | searchProduct | setupStageProductList3 | productName = “Hamburguesa” | El producto “Hamburguesa” es mostrado mediante un mensaje que contiene su información. |
| ProductList | searchProduct | setupStageProductList3 | price = 70000.0 | El producto “Camiseta” es mostrado mediante un mensaje que contiene su información. |
| ProductList | searchProduct | setupStageProductList3 | category = “Toys and Games” | Los productos “PS5” y “XBOX” son mostrados mediante un mensaje que contiene su información. |
| ProductList | searchProduct | setupStageProductList3 | purchasedNumber = 5 | El producto “Balón” es mostrado mediante un mensaje que contiene su información. |

**RF9 - El programa debe contar con un buscador de pedidos que permita buscar por sus características principales (*nombre de comprador*, *precio total y fecha de compra)*.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verificar que el método searchOrder de la clase OrderList funcione correctamente, permitiendo buscar una orden por alguna de sus características principales. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| OrderList | searchOrder | setupStageProductList3 | buyerName = “James” | El pedido del cliente “James” es mostrado mediante un mensaje que contiene su información. |
| OrderList | searchOrder | setupStageProductList3 | totalPrice = 2000000.0 | El pedido del cliente “Santiago” es mostrado mediante un mensaje que contiene su información. |
| OrderList | searchOrder | setupStageProductList3 | purchaseDate = 2023-04-05 | Los pedidos de los clientes “Santiago” y “Luis” son mostrados mediante un mensaje que contiene su información. |

**R10 – El programa debe permitir las búsquedas por rango para los atributos numéricos, como el precio, la cantidad disponible o el número de veces comprado.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**RF11 – El programa debe permitir las búsquedas por intervalo para los atributos de tipo String, como el nombre.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**RF12 – El buscador debe permitir al usuario escoger el orden de los datos que se muestran y la variable de ordenamiento.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |