Use cases and scenarios

Configuration of scenarios

Name	**Class**	**Scenario**
setupStage1	InventoryTest	An empty inventory object
setupStage2	InventoryTest	An inventory object with a product object. The product with: - Name: Harry Potter 1 - Description: The first installment of the saga - Price: 10.0 - Amount: 10 - Category: 0
setupStage1	ProductSearch EngineTest	An inventory object with ten product objects. Product 1: - Name: Harry Potter collection - Description: A book - Price: 10.0 - Amount: 10 - Category: 0 Product 2: - Name: VacuumCleaner - Description: VacuumCleaner - Price: 5.0 - Amount: 20 - Category: 1 Product 3: - Name: Red T-Shirt - Description:Red T-Shirt - Price: 15.0 - Amount: 100 - Category: 2 Product 4: - Name: Beer barrel - Description:Beer barrel - Price: 100.0 - Amount: 15 - Category: 3 Product 5: - Name: Pencil box - Description:Pencil box - Price: 2.0 - Amount: 200 - Category: 4 Product 6: - Name: Golty ball - Description:To play soccer - Price: 20.0 - Amount: 60 - Category: 5 Product 7: - Name: Red lip - Description:Red lip - Price: 2.0 - Amount: 500 - Category: 6

		Product 8: - Name: Play station 5 - Description:Ps is better than xbox - Price: 500.0 - Amount: 100 - Category: 7 Product 9: - Name: How to make money - Description: To make money - Price: 5.0 - Amount: 20 - Category: 0 Product 10: - Name: Fridge - Description:Fridge - Price: 500.0 - Amount: 20 - Category: 1
setupStage1	OrderStorageTest	An Order object with the following data OrderStorage: • OrdeObject: ○ Pedro Pascal • ProductsArray: ○ Product 1: ▪ Name: ProductoUno ▪ Descroption: DescripcionUno ▪ Price: 100 ▪ Amount: 10 ▪ Category: 1 ○ Product 2: ▪ Name: ProductoDos ▪ Descroption: DescripcionDos ▪ Price: 200 ▪ Amount: 20 ▪ Category: 2 ○ Product 3: ▪ Name: ProductoTres ▪ Descroption: DescripcionTres ▪ Price: 330 ▪ Amount: 35 ▪ Category: 3 ○ Product 4: ▪ Name: ProductoCuatro ▪ Descroption: DescripcionCuatro ▪ Price: 430 ▪ Amount: 43 ▪ Category: 4 ○ Product 5: ▪ Name: ProductoCinco ▪ Descroption: DescripcionCinco ▪ Price: 560 ▪ Amount: 70 ▪ Category: 60 • AmountArray: ○ 10 ○ 14 ○ 20 ○ 31

		○ 54
setupStage1	OrderSearchEngineTest	- products1: - Product: - Name: Harry Potter collection - Description: A book - Price: 10.0 - Amount: 10 - Category: 0 - Product: - Name: VacuumCleaner - Description: VacuumCleaner - Price: 5.0 - Amount: 20 - Category: 1 - products2: - Product: - Name: Red T-Shirt - Description:Red T-Shirt - Price: 15.0 - Amount: 100 - Category: 2 - Product: - Name: Beer barrel - Description:Beer barrel - Price: 100.0 - Amount: 15 - Category: 3 - Product: - Name: Pencil box - Description:Pencil box - Price: 2.0 - Amount: 200 - Category: 4 - products3: - Product: - Name: Golty ball - Description:To play soccer - Price: 20.0 - Amount: 60 - Category: 5 - Product: - Name: Red lip - Description:Red lip - Price: 2.0 - Amount: 500 - Category: 6 - products4: - Product: - Name: Play station 5 - Description:Ps is better than xbox - Price: 500.0 - Amount: 100 - Category: 7 - Product:

		- Name: How to make money - Description: To make money - Price: 5.0 - Amount: 20 - Category: 0 - Product: - Name: Fridge - Description:Fridge - Price: 500.0 - Amount: 20 - Category: 1 Amout1: - 5 - 20 Amout2: - 50 - 7 - 120 Amout3: - 30 - 250 Amout4: - 1 - 5 - 2 Order1: • Angelica • 2016,211 Order2: • Angela • 2018, 11, 8 Order3: • Federico • 2021, 5, 18 Order4: • Fernando • 2023, 7, 27 orderStorage: - Order1, products1, amount1 - Order2, products2, amount2 - Order3, products3, amount3 - Order4, products4, amount4 - Order5, products5, amount5

Tests design

Objective of the test: Verify that the elemental methods of the inventory class work. If this tests works, an inventory object will can contains and save products				
Class	**Method**	**Scenario**	**input values**	**Expected result**
InventoryTest	saveProductMethodCanConstructsAndSavesANewElementCorrectlyTest	setupStage1	A new Product object with random values	The inventory object must contain the created product. For check that, use the contains() method
InventoryTest	saveProductMethodThrowsNonNaturalNumberExceptionExceptionToANegativeAmountTest	setupStage1	A random product with a negative amount	The inventory object must throw an exception.Because a product cannot have amount minor than cero
InventoryTest	saveProductMethodThrowsNonNaturalNumberExceptionExceptionToThePriceTest	setupStage1	A random product with price 0.0 and another random product with price -1.0	The inventory object must throw an exception.Because a product cannot cost cero or less
InventoryTest	addToInventoryMethodCanAddMoreUnitsToASavedProductTest	setupStage2	A product with: -Name:Harry Potter 1 -Description: The first installment of the saga -Price: 10.0 -Amount: 10 -Category: 0	How the product must be saved in the inventory object by the stage, the amount of this product must increase ten units
InventoryTest	addToInventoryMethodCanThrowNonNaturalNumberExceptionExceptionTest	setupStage1	A product with: -Name:Harry Potter 1 -Description: The first installment of the saga -Price: 10.0 -Amount: -10 -Category: 0	The inventory object must throw an exception because the increased amount has a negative value

Objective of the test: Verify that the inventory class can search an filter products correctly

Class	Method	Scenario	input values	Expected result
ProductSearchEngineTest	searchAnElementCanReturnTheObjectAmountWhenTheObjectIsSavedInTheInventoryTest	setupStage1	This method doesn't have input values.But the method must search for the amount of two products	The founded amount and the entered amount at the stage must be equals
ProductSearchEngineTest	searchAnElementCanThrowProductIsNotRegisteredExceptionExceptionTest	setupStage1	A random non-existent product	The Inventory class must throw an exception because is trying to find an non existent product
ProductSearchEngineTest	filterByRangeMethodCanGetAllElementsWhenTheUserFilterThemByPriceTest	setupStage1	A range of prices by 5.0 to 15.0 to the method be able to filter the expected products	The inventory must return the objects: 1. Harry Potter collection 2. How to make money 3. Red T-shirt 4. VacuumCleaner
ProductSearchEngineTest	filterByRangeMethodCanGetAllElementsWhenTheUserFilterThemBySalesTest	setupStage1	A range of sales by 0 to 2 to the method be able to filter the expected products	The inventory must return all Products saved
ProductSearchEngineTest	filterByRangeMethodCanGetAllElementsWhenTheUserFilterThemByAmountTest	setupStage1	A range of amount by 10 to 60 units to the method be able to filter the expected products	The inventory must return the objects: 1. Beer barrel 2. Fridge 3. Golty ball 4. Harry Potter collection 5. How to make money? 6. VaccumCleaner
ProductSearchEngineTest	filterByRangeMethodCanThrowThereIsNotProductsByTheFilterExceptionExceptionTest	setupStage1	A range of sales by 10 to 100	The inventory object must throw an exception because there aren't products in the interval
ProductSearchEngineTest	filterByIntervalMethodCanFilterTheProductsByLettersTest	setupStage1	The letter "F" to the beginning of the interval and letter "e" to the end of the interval	The system must return the object: fridge
ProductSearchEngineTest	filterByIntervalMethodCanFilterTheProductsByPrefixTe	setupStage1	The prefix "Har" and the suffix "ion" to the beginning and the end of interval respectively	The system must return the object Harry Potter collection

	st			
ProductSearchEngineTest	filterByIntervalMethodCanThrowThereIsNotProductsByTheFilterExceptionExceptionTest	setupStage1	The prefix “NON” and the suffix “NON” to the interval	The system must throw an exception because there aren’t products according to this interval.

Objective of the test: Verify that the elemental methods of the OrderStorage class work. If this tests works, an inventory object will can contains and save orders

Class	Method	Scenario	input values	Expected result
OrderStorageTest	searchTotalPriceIntervalTest1	setupStage1	A string “Pedro Pascal”	The first object in orderStorage must be the same
OrderStorageTest	searchTotalPriceIntervalTest2	setupStage1	The name of the first products array	The first productname of the first orderStorage must be the same
OrderStorageTest	searchCustomerNameIntervalTest1	setupStage1	The first int of the amount class	The amount of the first OrderProduct of the first orderStorage must be the same
OrderStorageTest	searchCustomerNameIntervalTest2	setupStage1	The fourt int of the amount class	The amount of the four OrderProduct of the first orderStorage must be the same
OrderStorageTest	searchDateIntervalTest1	setupStage1	A int 0	To check if the subtraction of quantity is being done correctly, the quantity of the first products array should be 0.
OrderStorageTest	searchDateIntervalTest2	setupStage1	A int 12	To check if the subtraction of quantity is being done correctly, the quantity of the fourth products array should be 12.
OrderStorageTest	orderTotalPriceTest	setupStage1	A sum of all total prices of the orderStorage array	To verify that the 'orderStorage' object has the total price of all the products it stores, the sum of the total price of all the products it stores should be the same as its 'totalPrices' variable.

Objective of the test: Verify that the inventory class can filter products correctly				
Class	**Method**	**Scenario**	**input values**	**Expected result**
OrderSearchEngineTest	searchTotalPriceIntervalTest1	setupStage1	A range of prices by 10 to 200	The system must return the object: order1
OrderSearchEngineTest	searchTotalPriceIntervalTest2	setupStage1	A range of prices by 1000 to 1200	The system must return the object: order3
OrderSearchEngineTest	searchCustomerNameIntervalTest1	setupStage1	The prefix “Fe” and the suffix “o” to the beginning and the end of interval respectively	The system must return the objects: order3 and order4
OrderSearchEngineTest	searchCustomerNameIntervalTest2	setupStage1	The prefix “A” and the suffix “a” to the beginning and the end of interval respectively	The system must return the objects: order1 and order2
OrderSearchEngineTest	searchDateIntervalTest1	setupStage1	A range of dates by 2015/01/01 to 2019/01/01	The system must return the objects: order1and order 2
OrderSearchEngineTest	searchDateIntervalTest2	setupStage1	A range of dates by 2020/01/01 to 2025/01/01	The system must return the objects: order3 and order 4