**Formato de escenarios y casos de uso**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupCorrectlyProduct | ProductTest | In this scenario the test product is created correctly, in order to be used in different tests  Product(name:"computador","description: “Es de carne",Price=100,stock=5) |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setup1IncorrectPriceProduct | ProductTest | In this scenario the test product is created with an incorrect price, for example: -5$. In order to be used in exception testing  Product(name:"computador","description: “acer",Price= -100,stock=5) |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** | |
| setupcorrectStockProduct | ProductTest | | In this scenario the test product is created with an correct inventory, for example: 5. In order to be used in exception testing  Product(name:"computador","description: “Es de carne",Price=100,stock=5) |
|  |  |  | |
|  |  |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** | |
| setupIncorrectStockProduct | ProductTest | | In this scenario the test product is created with an incorrect inventory, for example: -5. In order to be used in exception testing  Product(name:"computador","description: “Es de carne",Price=100,stock=-5) |
|  |  |  | |
|  |  |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verify that the different methods of Product verify that each product that is created makes sense | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| ProductTest | createAproductWithCorrectlyPriceTest | setupCorrectlyProduct | Price=100 | object is added checking that the price makes sense |
| ProductTest | createAProductWithIncorrectPriceTest | setup1IncorrectPriceProduct | price= -100 | The object is not added and throws an exception |
| ProductTest | createAProductWithCorrectlyStockTest | setupcorrectStockProduct | stock= 5 | the object is added correctly |
| ProducTest | notCreateAProductWithIncorrectStockTest | setupIncorrectStockProduct | stock= -5 | Product is not added and throws an exceptionmensaje dice que no se pueden usar números negativos. |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setupProductList | ProducListTest | It is a list of products created correctly to manipulate in the method tests of ProductList  Product:  Name: Tv  Price=100  Stock=5 |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verify that the different methods of Product verify that each product that is created makes sense | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| ProductListTest | addCorrectlyProductToListTest | setupProductList | Product | add the product to the list |
| ProductListTest | removeAproductOfTheListTest | setupProductList | Product | the object is removed from the list |
| ProductListTest | addAmountCorrectlyTest | setupProductList | stock= 5 | stock is updated to 10 |
| ProductListTest | addAmountInvalidTest | setupProductList | stock= -5 | the product is not added and shows an exception. |
| ProductListTest | amountToSellCorrectlyTest | setupProductList | quantityToSell=1 | the product is sold and the available stock is now 4 |
| ProductListTest | amountToSellIncorrectTest | setupProductList | quantityToSell=-51 | The product is not sold and throwing an exception |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1DeliveryList | DeliveryListTest | Products must be sorted by name in alphabetical order.  "Rice" with a price of 150, a category and 5 times bought  "Pasta" with a price of 100, and 3 times bought  "Meat" with a price of 500, and 2 times bought,  "Milk" with a price of 70, a category "dairy" and 8 times bought.  It should be verified that the list of products has the added products. |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup2DeliveryList | DeliveryListTest | Orders must be sorted by date of purchase in chronological order.  Orders need to be added to the list, such as an order for "John" with a total price of 250 and a purchase date of 2023-04-12, an order for "Mary" with a total price of 500 and a purchase date of 2023-04 -14, an order for "Pedro" with a total price of 100 and a purchase date of 2023-04-16. |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup3DeliveryList | DeliveryListTest | Products must be sorted by name in alphabetical order.  A product must be added to the list, such as "Rice" with a price of 150 and 5 times bought.  The binary search algorithm must be implemented to search for a product by name.  You must search for the product "Rice" using the binary search algorithm and verify that it is in the list of products. |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup4DeliveryList | DeliveryListTest | Products must be added to the list, such as "Rice" with a price of 150, and 5 times bought, "Pasta" with a price of 100 and 3 times bought, "Meat" with a price of 500 and 2 times bought, " Milk" with a price of 70 and 8 times bought.  Search by price must be implemented and products must be searched by minimum and maximum price specified by the user.  You must search for products that have a minimum price of 100 and a maximum price of 200, and you must verify that the list of products |
|  |  |  |
|  |  |  |

…

\* El nombre de los escenarios puede ser setupStage1, setupStage2, etc.

\* La clase es la clase de testing correspondiente al modelo donde acontece el escenario. Por ejemplo si usted está probando User, clase será UserTest.

\* El escenario es la descripción de las condiciones iniciales del escenario.

**Diseño de Casos de Prueba**

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

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

…

\* Una prueba se compone de un conjunto de casos de prueba.

\* Cada fila representa un ***caso de prueba*** difente

\* En el objetivo de la prueba debe escribir una descripción sobre qué es lo que específicamente está probando del modelo del programa.

\* La clase es la clase del modelo que está siendo puesto a prueba.

\* El método es específicamente el método de la clase que está siendo puesto a prueba.

\* El escenario se refiere al nombre del escenario que usted definió. Todos los casos de prueba corresponden a escenarios.

\* Los valores de entrada son valores que entran al método puesto a prueba.

\* El resultado esperado es lo que se espera que suceda luego de ejecutar el método.