INTEGRATIVE TASK 2

Nicolás Cuéllar Molina – A00394970

Davide Flamini Cazarán - A00381665

Andres Felipe Cabezas - A00394772

Tarea integradora 2.docx - Statement of Integrative Task 2

PROBLEM SPECIFICATION TABLE

CLIENT	MercadoLibre
USER	MercadoLibre Administrator
FUNCTIONAL REQUIREMENTS	- R1: Register and edit products - R2: Create Order - R3: Search for products - R4: Search Orders - R5: Edit products
CONTEXT OF THE PROBLEM	Build a program that allows the administrator of MercadoLibre can register both products and orders, in addition to keeping a database with their information.
NON-FUNCTIONAL REQUIREMENTS	- RN1: The program must handle exceptions - RN2: The project must be developed using TDD - RN3: The project must be uploaded to the Github platform and must have changes that allow you to see its evolution - RN4: The project must have quality indicators that allow to see its evolution, through reliability and completeness. - RN5: The program must have data persistence through JSON serialization.

Functional Requirements Analysis Table

<table>
<tr><td>Name or identifier</td><td colspan="3">R1: Register products</td></tr>
<tr><td>Summary</td><td colspan="3">The program allows you to register a product with all its corresponding data (name, description, price, quantity available, category and number of times purchased). After that, a confirmation of the product creation or an error message will be displayed.</td></tr>
<tr><td rowspan="7">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>name</td><td>String</td><td></td></tr>
<tr><td>description</td><td>String</td><td></td></tr>
<tr><td>price</td><td>double</td><td></td></tr>
<tr><td>amount</td><td>int</td><td></td></tr>
<tr><td>category</td><td>Category</td><td></td></tr>
<tr><td>number of purchases</td><td>int</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. The program asks for information about the product.
2. Verify that the product isn't registered before
3. Shows a confirmation or an error message</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">A product registered/edited or an error message</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>confirmation</td><td>String</td><td></td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R2: Create Order</td></tr>
<tr><td>Summary</td><td colspan="3">The program allows you to register an order, by entering information such as the buyer's name, list of products, total price and date of purchase. After that, a confirmation of the creation of the order or an error message will be displayed.</td></tr>
<tr><td rowspan="5">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>name of the buyer</td><td>String</td><td></td></tr>
<tr><td>products</td><td>Product[]</td><td></td></tr>
<tr><td>price</td><td>double</td><td></td></tr>
<tr><td>date</td><td>Calendar</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">1. Ask for the info of the order
2. Allows the user to search and select the product of the order
3. Shows a confirmation or an error message</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">A product Order created or an error message</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>confirmation</td><td>String</td><td></td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R3: Search Product</td></tr>
<tr><td>Summary</td><td colspan="3">The program has a product search engine which should allow you to search for products byname, price, category, number of times purchased and units available(This search can also be done in intervals or ranges).In addition to allowing the user to choose the order in which the products are displayed on the screen and a sort criteria.</td></tr>
<tr><td rowspan="10">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>searchedAttribute</td><td>String</td><td></td></tr>
<tr><td>rangeOption</td><td>boolean</td><td></td></tr>
<tr><td>minRange</td><td>int</td><td></td></tr>
<tr><td>maxRange</td><td>int</td><td></td></tr>
<tr><td>letter</td><td>String</td><td></td></tr>
<tr><td>startPrefix</td><td>String</td><td></td></tr>
<tr><td>endPrefix</td><td>String</td><td></td></tr>
<tr><td>sortingAttribute</td><td>int</td><td></td></tr>
<tr><td>ascendingOption</td><td>boolean</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">It shows a submenu to the user that allows him to select one of the product attributes to search for it and selects if he wants to search by ranges or directly by what he enters himself
If you select search by ranges depending on the Attribute you can select a maximum and minimum value. Otherwise you can select a letter, starting prefix and letter or ending prefix.
Allows the user to select the ordering attribute and the order in which the products will appear on the screen
Shows the products that match the attributes selected by the user</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">An ordered list of objects with the searched attributes</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>productsList</td><td>String</td><td></td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td colspan="3">R4: Fetch Order</td></tr>
<tr><td>Summary</td><td colspan="3">The program has an order search engine which should allow you to search for products bybuyer name, total price anddate of purchase(This search can also be done in intervals or ranges). In addition to allowing the user to choose the order in which the orders are displayed on the screen and a sort criteria.</td></tr>
<tr><td rowspan="10">Inputs</td><td>input name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>searchedAttribute</td><td>String</td><td></td></tr>
<tr><td>rangeOption</td><td>boolean</td><td></td></tr>
<tr><td>minRange</td><td>int</td><td></td></tr>
<tr><td>maxRange</td><td>int</td><td></td></tr>
<tr><td>letter</td><td>String</td><td></td></tr>
<tr><td>startPrefix</td><td>String</td><td></td></tr>
<tr><td>endPrefix</td><td>String</td><td></td></tr>
<tr><td>sortingAttribute</td><td>int</td><td></td></tr>
<tr><td>ascendingOption</td><td>boolean</td><td></td></tr>
<tr><td>General activities necessary to obtain the results</td><td colspan="3">It shows a submenu to the user that allows him to select one of the product attributes to search for it and selects if he wants to search by ranges or directly by what he enters himself
If you select search by ranges depending on the Attribute you can select a maximum and minimum value. Otherwise you can select a letter, starting prefix and letter or ending prefix.
Allows the user to select the ordering attribute and the order in which the orders will appear on the screen
Shows orders that match the attributes selected by the user</td></tr>
<tr><td>Result or postcondition</td><td colspan="3">An ordered list of orders with the attributes searched for</td></tr>
<tr><td rowspan="2">Outputs</td><td>output name</td><td>Datatype</td><td>Selection or repetition condition</td></tr>
<tr><td>ordersList</td><td>String</td><td></td></tr>
</table>

<table>
<tr><td>Name or identifier</td><td>R5: Edit products</td></tr>
<tr><td>Summary</td><td>The program allows you to edit the number of available quantities of a product. To edit the product, the program must previously allow you to search for the product (see R3). After that, a confirmation of the correct edition of the product or an error message will be displayed.

Note: What is highlighted in yellow is what concerns the search for product requirement, a requirement that we will use to edit the products.</td></tr>
<tr><td>Inputs</td><td>
<table>
<tr><th>input name</th><th>Datatype</th><th>Selection or repetition condition</th></tr>
<tr><td>searchedAttribute</td><td>String</td><td></td></tr>
<tr><td>rangeOption</td><td>boolean</td><td></td></tr>
<tr><td>minRange</td><td>int</td><td></td></tr>
<tr><td>maxRange</td><td>int</td><td></td></tr>
<tr><td>letter</td><td>String</td><td></td></tr>
<tr><td>startPrefix</td><td>String</td><td></td></tr>
<tr><td>endPrefix</td><td>String</td><td></td></tr>
<tr><td>sortingAttribute</td><td>int</td><td></td></tr>
<tr><td>ascendingOption</td><td>boolean</td><td></td></tr>
<tr><td>amount</td><td>int</td><td></td></tr>
<tr><td>productPosition</td><td>int</td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>General activities necessary to obtain the results</td><td>
1. It shows a submenu to the user that allows him to select one of the product attributes to search for it and selects if he wants to search by ranges or directly by what he enters himself
2. If you select search by ranges depending on the Attribute you can select a maximum and minimum value. Otherwise you can select a letter, starting prefix and letter or ending prefix.
3. Allows the user to select the ordering attribute and the order in which the products will appear on the screen
4. Shows the products that match the attributes selected by the user
5. From the list that appears on the screen, it allows the user to select the product that they want to edit from an index
6. The user enters the new quantity of the selected product
7. Shows a confirmation or an error message
</td></tr>
</table>

Result or postcondition	A product edited or an error message		
Outputs	**output name**	**Datatype**	**Selection or repetition condition**
	confirmation	String	

TESTS

Name	Class	Scenery
setupStage1	ProductTest OrderTest	The product list is empty. The order list is empty.
setupStage2	ProductTest OrderTest	The product list has registered products. *name:* TXL *description:* Truck with excellent features in this segment *price:* 340.000.000 *Quantity Available:* 13 *category:* Toys *number of times purchased:* 38 *name:* Picanto *description:* Car with high efficiency in fuel consumption *price:* 53.000.000 *Quantity Available*: 4 *category:*Toys *number of times purchased*: 74 *name:* Twingo *description*: Car hated by Shakira

		*price:*18.000.000 *Quantity Available*: 7 *category:*Toys *number of times purchased:*47 *name:* Spark *description:* Ideal car for the city *price:* 41.900.000 *Quantity Available*: 2 *category:*Toys *number of times purchased*: 35 *name:*Raptor *description*: Truck necessary to get you out of the routine *price:*320.000.000 *Quantity Available*: 1 *category:*Toys *number of times purchased:*2 The order list has registered orders: *name of the buyer:* Enrique Suarez *list of products:*TXL *total price:*340.000.000 *purchase date:*04/08/2016 *name of the buyer*: Humberto Cazaran *list of products:*Spark, Twingo *total price:*59.900.000 *purchase date*: 01/05/2021

Add (Product and order)

Test Objective: Validate that a new Product was successfully registered				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Product Controller	addProduct()	setupStage1	*name:* Hp *description*: High speed computer processing *price*: 3.500.000 *Quantity Available:* 12 *category*: Electronics *number of times purchased*: 24	The new product is expected to be successfully added to the empty list.
Product Controller	addProduct()	setupStage2	*name:* Lenovo *description:* gaming computer *Quantity Available*: 4 *category:* electronics *number of times purchased:* 15	It is expected that the new product will be successfully added to the end of the list that already contained products.

Test Objective: Validate that it is not allowed to register a new order that does not contain all the corresponding information				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Order Controller	addOrder()	setupStage1	*name of the buyer*: Davide Cazarán *list of products:*Spark, Twingo *total price:* *purchase date*: 01/05/2021	Expect to throw an exception and not allow adding it by not having a total price specified
Order Controller	addOrder()	setupStage2	*name of the buyer:* *list of products:*TXL *total price:*340.000.000 *purchase date:*04/08/2020	Expected to throw exception and disallow add due to not having specified name

Search (Product and order)

Test Objective: Validate that the program searches for products or orders within a price range and orders them from their name in ascending order.				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Product Controller	searchProduct()	setupStage2	*rangeOption: True* *minRange:* 0 maxRange: 42000000 sortingAttribute: 1 (name) ascendingOption: True	Products are expected to be listed as follows: *name:* Spark *description:* Ideal car for the city *price:* 41.900.000 *Quantity Available*: 2 *category:*Toys *number of times purchased*: 35 *name:* Twingo *description*: Car hated by Shakira *price:*18.000.000 *Quantity Available*: 7 *category:*Toys *number of times purchased:*47
Order Controller	searchOrder()	setupStage2	*rangeOption: True* *minRange:* 0 maxRange: 42000000 sortingAttribute: 1 (buyer name) ascendingOption: True	Orders are expected to be listed as follows: *name of the buyer*: Humberto Cazaran *list of products:*Spark, Twingo *total price:*59.900.000 *purchase date*: 01/05/2021 .

Test Objective: Validate that the program does not allow searching for products or orders when the list is empty.				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Product Controller	searchProduct()	setupStage1	*rangeOption: True* *minRange:* 0 maxRange: 42000000 sortingAttribute: 1 (name) ascendingOption: True	It is expected that an exception will be thrown and it will not allow to search for the product since the product list is empty.
Order Controller	searchOrder()	setupStage1	*rangeOption: True* *minRange:* 0 maxRange: 42000000 sortingAttribute: 1 (buyer name) ascendingOption: True	It is expected that an exception will be thrown and it will not allow to search for the order since the order list is empty.

edit product

Test Objective: Validate that the program allows you to edit the quantity of a product already registered.				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Product Controller	editProduct()	setupStage2	*rangeOption: True* *minRange:* 0 maxRange: 100000000000 sortingAttribute: 1 (name) ascendingOption: True quantity: 100 productPosition: 1	It is expected that the product with index 1 of the list will be published, which according to the organization criteria should be: *name:* Picanto *description:* Car with high efficiency in fuel consumption *price:* 53.000.000 *Quantity Available*: 4 *category:*Toys *number of times purchased*: 74 The Kia Picanto would be left with an available quantity of 100.
Product Controller	editProduct()	setupStage2	*rangeOption: True* *minRange:* 0 maxRange: 340000001 sortingAttribute: 1 (name) ascendingOption: True quantity: 5 productPosition: 4	It is expected that the product with index 4 of the list will be published, which according to the organization criteria should be: *name:* Twingo *description*: Car hated by Shakira *price:*18.000.000 *Quantity Available*: 7 *category:*Toys *number of times purchased:*47 The Twingo would be left with an available quantity of 5.

Test Objective: Validate that the program does not allow placing negative values in the quantity of a product when editing it.				
Class	**Method**	**Scenery**	**Entry Values**	**Expected result**
Product Controller	searchProduct()	setupStage1	*rangeOption: True* *minRange:* 0 maxRange: 100000000000 sortingAttribute: 1 (name) ascendingOption: True quantity: -24 productPosition: 1	An exception is expected to be thrown and it doesn't allow to edit the product as you are putting in a negative quantity.
Order Controller	searchOrder()	setupStage1	*rangeOption: True* *minRange:* 0 maxRange: 350000000 sortingAttribute: 1 (name) ascendingOption: True Quantity 1 productPosition: 4	An exception is expected to be thrown and not allows edit the product since you are placing a negative amount.