Integrative task – APO

Juan Manuel Díaz Moreno – A00394477

The functional requirements for the application are the following:

1. **Allow the entry of product data**, including its name, description, price, quantity available, category and number of times purchased.

Name or identifier	R1: Product registration.		
Abstract	The system allows the user to enter the information of a product and the category to which it belongs to register it.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	Name	String	
	Description	String	
	Price	double	
	Category	String	
	Amount	int	
	Number of times purchased	int	
General activities	1. The system allows the user to enter product information. 2. The system validates the information entered. 3. If the information is correct, the product is registered and the validation is sent on the screen. 2. 4. If the information is not correct, an error is generated, the product is not registered and the user is notified.		
Result or postcondition	Product registered successfully.		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	Validation message.	String	Prior information entry.

Caso de prueba:

Objetivo de la Prueba: Verify that the application allows the user to enter product data correctly.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Product	addProduct()	Enter a product with complete information	- Name: “Cotton Jacket.” - Description: “Jacket in blue, red and white colors, made of 100% high quality cotton.” - Price: 230000 - Quantity available: 100 - Category: Clothes and accessories. - Number of times purchased: 0	The product should be successfully added to the database and the app displays a confirmation message.
Product	addProduct()	Enter an unnamed product	- Name: “” - Description: “New Balance shoes, collection inspired by the 80s.” - Price: 789000 - Quantity available: 150 - Category: Clothes and accessories. - Number of times purchased: 0	The system should display an error message indicating that the "name" field is required.

2. **Allow order registration, including buyer's name, product list,** total price, and purchase date. When generating an order, the quantity of the inventory must decrease. In addition, the user must be able to increase the quantity of each product already registered.

Name or identifier	R2: Orders register.		
Abstract	The system must allow entering the information to generate the order and reduce the quantities of the inventory. Finally the user can increase the amount of a product already registered.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	Buyer name	String	
	Product list	Arraylist	
	Total price	Double	
	date	date	

General activities	1. Allow the user to enter order information. 2. Validate the information entered by the user. 3. Check if the products requested by the user are available. 4. Create the order on behalf of the user. 5. Decrease the products chosen from the order. 6. Allow adding products to the order if required by the user.		
Result or postcondition	Order created successfully.		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	Validation message	String	
	Decrease in inventory	int	

Caso de prueba:

Objetivo de la Prueba: Verify that an order can be registered correctly, that the quantity of a product in an already registered order can be increased, and that the inventory quantity decreases.

Clase	Método	Escenario	Valores de Entrada	Resultado esperado
Order	registerOrder()	Order successfully registered.	- Buyer name: “Diego” - Product List: [{id: Shoes45, quantity: 2}, {id: Sweatshirt00, quantity: 1}] - Total price: 300 - Purchase date: "2022-05-11"	The order is successfully posted and the quantity of the products in inventory is updated.
Order	increaseQuantityProduct()	Increase the quantity of a product in an order	- orderID: 001 - idProduct: Shoes45 - quantity to be increased: 2	The quantity of the product in the order is updated correctly.

3. **Allow the elimination of products already entered.**

Name or identifier	R3: Delete products
Abstract	The system must search for a product by its name or identifier and remove it from the platform.

Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	Product name	String	The product must exist on the platform
General activities	1. Allow the user to enter product information. 2. The system searches for the product and removes it from the database. 3. Send the verification message.		
Result or postcondition	Product removed successfully.		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	Validation message	String	Prior input of information

Caso de prueba:

Objetivo de la Prueba: Verify the functionality of removing a product from the database.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Product	deleteProductByIdentifier()	The product exists in the database	-Product name: “Camisa roja”	The product "Camisa roja" is removed from the database and a success message is displayed.
Product	deleteProductByIdentifier()	The product it is not found in the database.	-Product name: “Zapatos nike”	Error message indicating that the product is not found in the database.

4. **Search for products and orders by any of their attributes.** The product search engine should allow searching by name, price, category and number of times purchased. The order finder allows you to search by buyer's name, total price and date of purchase. Both search engines must use the binary search algorithm and allow range or interval searches for numeric or String attributes. In addition, the user must be able to specify the ascending or descending order of the results and the ordering variable.

Name or identifier	R4: Search products or orders by some attribute.		
Abstract	Searches for an order or a product by some system attribute. Both search engines must use the binary search algorithm and allow range or interval searches for numeric or string attributes. In addition, the user must be able to specify the ascending or descending order of the results and the sort variable.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	Identifier	(String, date, Price, etc)	
General activities	1. The system receives the conditions to search for the client. 2. The system validates the information. 3. Taking into account the range or attribute or character by which the user wants to search for the product or order, it searches the database to see if it is found. 4. Once the results related to the search are found, the system asks for the order (ascending or descending) in which the user wants the results. 5. The system displays the final results of the search.		
Result or postcondition	the product is in the found database according to the user's conditions.		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	products found	Product, Order	Prior input of information

Caso de prueba:

Objetivo de la Prueba: Verify that both products and orders can be searched with the conditions given by the user.

Clase	Método	Escenario	Valores de Entrada	Resultado esperado
Product	searchByPriceRange()	There are several products in the database with different prices.	Minimum price: 10.00 Maximum price: 50.00 Sorting variable: Number of times purchased Descending order	The search engine finds all the products with prices between 10.00 and 50.00 and displays them in the console ordered by number of times purchased in descending order.

Order	searchOrByTotalPrice()	The order exists in the database.	Total order price: 100.50	The finder finds the order with a total price of 100.50 in the database and displays it in the console.

5. **Data persistence using JSON serialization,** to guarantee the integrity of the data and to be able to access it even after closing the program.

Name or identifier	R5: Data persistence		
Abstract	To guarantee the integrity of the data and to be able to access it even after closing the program.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	Products to save	Products	The products need must exist.
General activities	1. Collect product data through the user interface or through integration with other systems. 2. Serialize the data in JSON format and store it in a persistent file on the local file system. 3. Read the data stored in the persistent file and deserialize it into the appropriate data structure when the application starts. 1. Provide the necessary functions to update, delete and query the data stored in the persistent file.		
Result or postcondition	Product data is stored persistently		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	System persistent file	Archive	

Caso de prueba:

Objetivo de la Prueba: Verify that product information is stored correctly in a JSON file.

Clase	Método	Escenario	Valores de Entrada	Resultado esperado

Application	storeDataInFile()	There are several products in the database.	File name: "products.json"	The method stores the information of all the products in the file "products.json" in JSON format and returns a confirmation.
Application	updateDataInFile()	There is a JSON file with the information of various products.	File name: "products.json"	The method updates the product information in the "products.json" file with the latest data from the application's database and returns a confirmation.

6. **Handle exceptions to avoid unexpected states during product entry and search.** You must provide informative error messages to guide the user in the event of an unexpected error.

Name or identifier	R6: Throw exceptions		
Abstract	The system must provide informative error messages to guide the user in the event of an unexpected error.		
Inputs	**Input name**	**Datatype**	**Selection or repetition condition**
	User inputs	(Any type used in the program)	
General activities	1. Identify and handle exceptions that may arise during product entry and search. 2. Provide informative error messages to guide the user in the event of an unexpected error. 3. Record exceptions for further analysis and correction.		
Result or postcondition	The system handles exceptions appropriately and provides informative error messages to the user.		
Outputs	**Output name**	**Datatype**	**Selection or repetition condition**
	exceptions	depends on the situation	prior input of information.

Caso de prueba:

Objetivo de la Prueba: Verify exception handling during entry and product search.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado esperado**
Application	addProduct()	Try to add a product with a negative Price.	Name: "Test Product" Description: "This is a test product" Price: -10 Quantity Available: 20 Category: "Electronics" Number of times purchased: 0	You must throw an exception indicating that the price must be a positive value.
Applicaation	addProduct()	Try to add a product without specifying the quantity available.	Name: "Test Product" Description: "This is a test product" Price: 10 Category: "Electronics" Number of times purchased: 0	It should throw an exception indicating that the available quantity should be specified.