|  |  |
| --- | --- |
| **Client** | Mercadolibre |
| **User** | Administrator and the Client |
| **Functional requirements** | 1. Enter information of a product. 2. Enter information of an order. 3. Increase the quantity in stock of a registered product. 4. Eliminate a product by name. 5. The program must count with a products search engine and a simple order search engine which allows:   a. Search products by name, price, category and number of times purchased.  b. Search orders by buyer’s name, total Price and purchase date.   1. The program should count with a product search engine that allows to:    1. Search a product by numeric attributes(Price, available amount or the number of times purchased) given a range of values(minimum and maximum value).    2. Search for a product by it´s name . The user can define a starting letter or prefix and an ending letter or prefix inside that alphabetic range. 2. The simple search engine and by range has to allow the user choose the kind of sort(ascendant or descendant) for the data that is shown and the sort variable. 3. Serialize the system information   Non-functional - restrictions   1. The search engines have to use the binary search algorithm to find coincidences when products are registered or ordered. They should also show the results by console. 2. Serialize the information using Json files. |
| **Problem context** | Mercado Libre wants to create an application that allows to sell their products online. Your task is to create a program that allows you to enter the inventory of the online store and allows the search and elimination of products. Also, the program has to be able to register orders that the users of the store make. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R1. Enter the information of each product. | | |
| **Abstract** | The user enters information of a product to have it registered in the system. The product should have a unique name including a brief description of what it does, its price, the category in which it belongs and the quantity of units that have been sold. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| name | String | The products should not have the same name |
| description | String |  |
| price | Double | It should not have a negative number |
| category | String | The category should belong to the existing list. |
| purchasedAmount | int | It cannot be a negative value. |
| **General activities necessary to obtain the results** | The ArralyList that contains the products should be initialized so that new products can be added. | | |
| **Result or postcondition** | A new added product object to the ArralyList of products of the system. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| msg | String |  |

Enter information to make an order

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R2. Enter information to make an order | | |
| **Abstract** | The user can make an order entering his name and accessing the product list to see each products price and select the product to make the purchase | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| buyerName | String | The buyers name should not exist previously |
| productsList | ArrayList<Product> | The list should not be null |
| totalPrice | double | The price must not be zero nor negative. |
| datePurchase | LocalDateTime | The date should not exist previously |
| **General activities necessary to obtain the results** | 1. The Arraylist of the products must be initialized and different from null.  2. The array must be sorted with the value given.(In this case the name is used to make the search.)  3. The binary search algorithm is used to look for the names that start or end with the given prefix.  4. Coincidences are stored in an auxiliar ArrayList.  5. The auxiliar Arraylist is ordered with the sort variable and the type of sort(ascendant or descendant) stablished by the user. | | |
| **Result or postcondition** | An object or a list of objects that match with the search made by the user. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| productsWithPrefixCoincidences | ArrayList<Product> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R3. Increase the quantity in stock of a registered product | | |
| **Abstract** | The user can increase the product units in the inventory. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| productIndex | int | The number has to be greater than zero. |
| amount | int | The number has to be greater than zero. |
| **General activities necessary to obtain the results** | 1. El ArrayList de Productos debe estar inicializado (Debe ser diferente a nulo)  2. Look for the product with the given index  3. Once the product is found the units are increased using the set method. | | |
| **Result or postcondition** | An elemento from the list has been modified. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| msg | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R4. Delete a product by name | | |
| **Abstract** | The user can delete a product from the inventory. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| productIndex | int | The number has to be greater than zero. |
| **General activities necessary to obtain the results** | 1. The ArrayList of Products have to be initialized(It has to be different from null).  2. The product is deleted from the list with the given index | | |
| **Result or postcondition** | An element from the list has been deleted. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| msg | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R5A. Search a product using name, Price, category or number of times purchased. | | |
| **Abstract** | The user can search a product or an order using the name, price, category and number of purchases. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| name | String | The name should not have line breaks. |
| category | String | The category should not have line breaks. |
| price | Double | Should not have symbols nor letters. |
| purchaseAmount | Double | Should not contain symbols nor letters. |
| sortVariable | String | Should belong to the options of a product to be sorted. |
| sort | String | Should have ascendant or descendent. |
| **General activities necessary to obtain the results** | 1. The ArrayList of Products have to be initialized (It has to be different from null).  2. The ArrayList has to be sorted with the given search value.  3. The binary search algorithm is used to look for the product.  4. The result Arraylist is ordered with the sort variable and the type of sort(ascendant or descendant) stablished by the user. | | |
| **Result or postcondition** | Un objeto o una lista de objetos que coinciden con la busqueda realizada por el usuario. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| productsMatch | ArrayList<Products> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R5B. Search an order using the name of the buyer, total price or purchase date of an order. | | |
| **Abstract** | The user can search an order using the name of the buyer, total price and the date of purchase of an order | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| BuyerName | String | Should not have line brakes |
| totalPrice | double | Should not have symbols nor letters. |
| datePurchase | String | Should not have line brakes |
| sortVariable | String | Should belong to the options of a product or order to be sorted. |
| sort | String | Should be ascendant or descendent |
| **General activities necessary to obtain the results** | 1. The ArrayList of orders have to be initialized (It has to be different from null).  2. The ArrayList should be sorted with the search value given(To execute the search algorithm).  3. The binary search algorithm is used to look for the order.  4. It is ordered with the sort variable and the sort(ascendant and descendent) stablished by the user. | | |
| **Result or postcondition** | An object or a list of objects that match with the search made by the user. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| ordersMatch | ArrayList<Order> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R6A. Search an amount of products using a range of numeric values. | | |
| **Abstract** | The user can search an amount of products using a range of values (Minimum value and maximum value) associated to the price, amount sold, or the available amount. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| valueMin | double | Should not have symbols nor letters. |
| valueMax | double | Should not have symbols nor letters. |
| searchVariable | String | Should belong to the product options((Price, amount, purchaseAmount). |
| sortVariable | String | Should belong to the sorting types for products and orders |
| sort | String | Should be ascendant or descendent |
| **General activities necessary to obtain the results** | 1. The ArrayList of orders have to be initialized (It has to be different from null).  2. The ArrayList has to be sorted with the given search value.( To execute the search algorithm).  3. The binary search algorithm is used to search the index of the product that has the minimum range value.  4. The binary search algorithm is used to search the index of the product that has the maximum range value  5. A new Arralylist is created that contains the elements that are withtin the index range of the elements found. The original Arraylist is partitioned to create a new Arraylist with the products that are within the range.  4. It is ordered with the sort variable and the sort type(ascendent or descendent) stablished by the user | | |
| **Result or postcondition** | An object or a list of objects that match with the search made by the user. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| productsMatch | ArrayList<Product> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R6B. Search an amount of products using a starting or finishing prefix of the user´s name. | | |
| **Abstract** | The user can look for an amount of products using a prefix or suffix of a product name. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| prefix | String | The data cannot be null. |
| typePrefix | String | Should specify if its staring or finishing |
| sortVariable | String | Should belong to the sorting types for products and orders |
| sort | String | Should be ascendent or descendent |
| **General activities necessary to obtain the results** | 1. The ArrayList of orders have to be initialized (It has to be different from null).  2. The ArrayList has to be sorted with the given search value.( In this case by name).  3. The binary search algorithm is used to search the names that start or end with the prefix given.  4. Las coincidencias se guardan en un ArrayList auxiliar. The matches are stored in an axiliar ArralyList.  5. The auxiliar ArrayList is sorted with a sort variable and the sorting type(ascendent or descendent) stablished by the user. | | |
| **Result or postcondition** | An object or list of objects that match with the search made by the user. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| productsWithPrefixCoincidences | ArrayList<Product> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R7. The simple search by range has to allow the user to choose the variable and the type of sort( ascendant or descendant) to show the result. | | |
| **Abstract** | Organizes the results with the parameters given by the user. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| sortVariable | String | The data cannot be null |
| order | boolean | Should specify if its starting or finishing. |
| **General activities necessary to obtain the results** | 1. The ArrayList of prodcuts have to be initialized (It has to be different from null).  2. The result ArrayList has to sorted with the given variable stablished by the user.  3. The result is returned. | | |
| **Result or postcondition** | The ordered list of products with the variable and type of sort given by the user. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| resultProducts | ArrayList<Product> |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name or identifier** | R8. Serialize the system information using Json. | | |
| **Abstract** | Saves information in Json file to upload it when the system is closed. | | |
| **Inputs** | **Input name** | **Datatype** | **Selection or repetition condition** |
| **General activities necessary to obtain the results** | 1. When the system closes, the ArrayList of the products and orders are transformed to text using the Gson library.  2. When the system initiates, the files are read and are transformed in Arraylists of products and orders also using the Gson library. | | |
| **Result or postcondition** | The files are saved in the local disc. | | |
| **Outputs** | **Output name** | **Datatype** | **Selection or repetition condition** |
| ProductList | ArrayList<Product> |  |
| orderList | ArrayList<Order> |  |