model

SearchUnitTest

+ SearchUnitTest():

+ searchProductByPriceTest1(): void
+ searchOrderByDateTest1(): void
+ searchProductByName_EqualsTest3(): voi
+ searchOrderByDateTest3(): void
+ searchProductByPriceTest4(): void
+ searchProductByPriceRangeTest5(): void
+ searchOrderByNameTest2(): void
+ searchOrderByTotalPriceTest1(): void
+ searchOrderByDateTest2(): void
+ searchProductByPriceTest5(): void
+ searchOrderByNameTest4(): void
+ searchProductByPriceTest3(): void
+ searchProductByCategoryTest4(): void
+ searchOrderByTotalPriceTest2(): void
+ searchProductByName_ContainsTest2(): v
+ searchOrderByTotalPriceTest3(): void
+ searchProductByPriceRangeTest2(): void
+ searchProductByName_ContainsTest3(): v
+ searchProductByName_Contains4(): void
+ searchProductByName_ContainsTest(): voi
+ searchProductByPriceTest2(): void
+ searchOrderByNameTest3(): void
+ searchProductByPriceRangeTest4(): void
+ searchProductByPriceRangeTest3(): void
+ searchProductByPriceRangeTest7(): void
+ setup(): void
+ searchProductByCategoryTest2(): void
+ searchProductByName_EqualsTest2(): voi
+ searchProductByPriceRangeTest6(): void
+ searchProductByName_Equals4(): void
+ searchProductByName_EqualsTest1(): voi
+ searchOrderByDateTest4(): void
+ searchProductByCategoryTest1(): void
+ searchOrderByNameTest1(): void
+ searchProductByCategoryTest3(): void
+ searchProductByPriceRangeTest1(): void

1

MercadoLibre

+ MercadoLibre():

- orders: List<Order>
- products: List<Product>

+ deleteProduct(int): void
+ searchProductByTimesBought(int): List<Product>
+ displayProducts(): String
+ deleteOrder(int): void
+ searchProductByName_Contains(String): List<Product>
+ getProductPrice(int): double
+ searchOrderByTotalPrice(double): List<Order>
+ searchOrderByDate(LocalDate): List<Order>
+ addProduct(String, String, double, int, int, int): void
+ searchProductByName_Equals(String): List<Product>
+ displayOrders(): String
+ decreaseProductQuantity(int, int): void
+ getCategory(int): String
+ searchProductByPrice(double): List<Product>
+ increaseProductQuantity(int, int): void
+ searchProductByPriceRange(double, double): List<Produ
+ displaySortedOrderList(List<Order>, String, String): String
+ searchOrderByName(String): List<Order>
+ displaySortedProductList(List<Product>, String, String): S
+ addOrder(List<Integer>, LocalDate, String, double): void
- displayProducts(StringBuilder, List<Product>): StringBuild
+ searchProductByCategory(String): List<Product>
- displayOrders(StringBuilder, List<Order>): StringBuilder
+ increaseTimesBought(int, int): void

orders: List<Order>
products: List<Product>

1

ml

1

mercadoLibre

AddDeleteUnitTest

+ AddDeleteUnitTest():

+ addProductTest8(): void
+ addProductTest2(): void
+ addOrderTest2(): void
+ addProductTest7(): void
+ deleteProductTest1(): void
+ addOrderTest3(): void
+ addOrderTest1(): void
+ addOrderTest5(): void
+ addOrderTest4(): void
+ addOrderTest6(): void
+ deleteOrderTest2(): void
+ deleteProductTest2(): void
+ addProductTest1(): void
+ deleteOrderTest1(): void
+ addProductTest6(): void
+ addProductTest4(): void
+ addProductTest5(): void
+ addProductTest3(): void

1

1

<<enumeration>>
ProductType

- ProductType(String):

- category: String

+ valueOf(String): ProductType
+ values(): ProductType[]

1

type

1

Product

+ Product(String, String, double, int, int, int, int)

- price: double
- timesBought: int
- availableQuantity: int
- name: String
- description: String

+ getCategoryByIndex(int): String
+ toString(): String

name: String
description: String
timesBought: int
price: double
ID: int
category: String
availableQuantity: int

*

*

products

products

<<interface>>
Searchable

<<interface>>
Searcher

+ search(V, U, Function<T, U>): R
+ collectMatches(List<T>, List<Integer>): List<T>
+ searchingByCondition(List<T>, Function<T, U>, BiPredicate<U, U>, U
+ searchingByRange(List<T>, double, double, Function<T, Double>): Lis
+ binarySearchInRange(List<T>, Function<T, Double>, double, double):
+ searchingInt(List<T>, Integer, Function<T, Integer>): List<T>
+ searchingString(List<T>, String, Function<T, String>): List<T>
+ binarySearchByCondition(List<T>, Function<T, U>, BiPredicate<U, U>
+ searchingDouble(List<T>, double, Function<T, Double>): List<T>
+ searchingDate(List<T>, LocalDate, Function<T, LocalDate>): List<T>
+ binarySearch(List<T>, U, Function<T, U>, Comparator<U>): List<Integ

<<record>>
Order

+ Order(List<Product>, LocalDate, String, double, in

+ totalPrice(): double
+ id(): int
+ toString(): String
+ date(): LocalDate
+ products(): List<Product>
+ name(): String

*

orders

FileManager

+ FileManager():

- loadProducts(): List<Product>
+ loadData(List<Product>, List<Order>): voi
- loadOrders(): List<Order>
+ saveData(List<Product>, List<Order>): vo
- saveProducts(List<Product>): void
- saveOrders(List<Order>): void

ui

Main

+ Main():

+ main(String[]): void
+ searchMenu(): void
+ mainMenu(): void
+ setSortParameters(String[], String[]): void
+ productMenu(): void
+ orderMenu(): void
+ displayListValidation(Function<MercadoLibre, String>, String
+ productPriceValidation(double): void

1