ui

main
+main(args : string) : void

Menu
-REGISTER : int = 1
-UPDATE_DATA : int = 2
-CHANGE_STATE : int = 3
-SAVE : int = 4
-IMPORT : int = 5
-EXPORT : int = 6
-SHOW : int = 7
-FIND_CLIENT : int = 8
-EXIT : int = 9
-PRINCIPAL : string = "1.REGISTER" + "\n"+ "2.UPDATE DATA" + "\n"+ "3.CHANGE STATE" + "\n" + "4.SAVE" + "\n" + "5.IMPORT" + "\n" +   "6.EXPORT" + "\n" + "7.SHOW IN SCREEN" + "\n" + "8.FIND CLIENT " + "\n" + "9.EXIT"
+showMenu() : void

Model

ProductTest
+setupScenary1() : void
+testUpdateAll() : void
+testGetName() : void

RestaurantTest
+setupScenary1() : void
+testAddProductWithSameCode() : void
+testSearchProduct() : void

RestaurantAssociation
+RestaurantAssociation() : void
+registerRestaurant(nit : int, name : string, nameAdmin : string) : void
+saveRestaurant() : void
+importRestaurants(filename : string, separator : string) : void
+showRestaurants() : void
+searchRestaurant(nit : int) : Restaurant

RestaurantAssociationTest
+setupScenary1() : void
+testSearchRestaurant() : void
+testRegisterRestaurant() : void

ClientTest
+setupStage1() : void
+setupStage2() : void
+testIdentificationException() : void
+testSearchException() : void

exceptions

<<Exception>> IdentificationException
+IdentificationException() : void

<<Exception>> ChangeStateException
+ChangeStateException() : void

<<Exception>> SearchException
+SearchException() : void

Product
-code : int
-restaurantNit : int
-cost : int
-name : string
-description : string
+Product(code : int, restaurantNit : int, cost : int, name : string, description : string) : void
+setCode(code : int) : void
+getCode() : int
+setRestaurantNit(restaurantNit : int) : void
+getRestaurantNit() : int
+setCost(cost : int) : void
+getCost() : int
+setName(name : string) : void
+getName() : string
+setDescription(description : string) : void
+getDescription() : string
+updateAllData(code : int, cost : int, name : string, description : string) : void
+compareTo(product : Product) : int

Restaurant
-name : string
-nit : int
-nameAdmin : string
+Restaurant(nit : int, name : string, nameAdmin : string) : void
+setNit(nit) : void
+getNit() : int
+setName(name : string) : void
+getName() : string
+setNameAdmin(nameAdmin : string) : void
+getNameAdmin() : string
+getProducts() : Product
+getClients() : Client
+getOrders() : Order
+setProducts(products : Product) : void
+addProduct(code : int, cost : int, name : string, description : string) : void
+addClient(n : int, p : int, t : string, f : string, l : string, a : string) : void
+importProducts(filename : string, separator : string) : void
+importClients(filename : string, separator : string) : void
+showClients() : void
+searchProduct(code : int) : Product
+searchClient(number : int) : Client
+updateAllData(nit : int, name : string, nameAdmin : string, products : Product, clients : Client) : void

Client
-identificationNumber : int
-phone : int
-identificationType : string
-firstName : string
-address : string
-lastName : string
+Client(identificationNumber : int, phone : int, identificationType : string, firstName : string, lastName : string, addres : string) : void
+setIdentificationNumber(identificationNumber : int)
+getIdentificationNumber() : int
+setPhone(phone : int) : void
+getPhone() : int
+setIdentificationType(identificationType : string) : void
+getIdentificationType() : string
+setFirstName(firstName : string) : void
+getFirstName() : string
+setAddres(addres : string) : void
+getAddres() : string
+setLastName(lastName : string) : void
+getLastName() : string
+addOrder(clientCode : int, restaurantNit : int, products : Product) : void
+getOrders() : Order
+changeState(state : string, index : int) : void
+showOrders() : void
+importOrders(filename : string, separator : string, restaurantNit : int, r : Restaurant, clientCode : int) : void
+searchOrder(code : int) : void
+updateAllData(identificationNumber : int, phone : int, identificationType : string, firstName : string, lastName : string, addres : string) : void
+compareTo(o : Client) : int
+sortOrder() : void
+export(filename : string, separator : string) : void

Order
-code : int
-clientCode : int
-restaurantNit : int
-date : Date
-state : int
+Order(code : int, clientCode : int, restaurantNit : int, products : Product) : void
+setCode(code : int) : void
+getCode() : int
+setClientCode(clientCode : int) : void
+getClientCode() : int
+setRestaurantNit(restaurantNit : int) : void
+getRestaurantNit() : int
+getDate() : Date
+getProducts() : Product
+getState() : string
+setState(state : string) : void
+setProducts(products : Product) : void
+compareTo(order : Order) : int

OrderTest
+setupScenary1() : void
+testConstructor() : void
+testSetState() : void

Relationships / multiplicities:
--restaurantAssociation 0..1
--client
--restaurant 0..1
--product 0..1
--order 0..1
products 0..*
restaurants 0..*
clients 0..*
orders 0..*
products on order 0..*

FUNCTIONAL REQUERIMENTS:

R#1.REGISTER: The program must be able to register restaurants, clients, orders, or a product.

R#2.UPDATE DATA: program must have the possibility to update the data of a restaurant, client, product, or order. To find the object to update they count with an identificator like code, number, etc.

R#3.SORT: The client list must always be ordered alphabetically descending by surname and first name.

R#4.CHANGE STATE: The program must allow changing the status of an order between REQUESTED, IN PROCESS, SHIPPED and DELIVERED. You can change the order status forward but not backward.

R#5.VERIFICATION: it must be verified that all the products of an order belong to the restaurant that the client chooses to place the order.

R#6.SAVE: The program must save all its information through the serialization of its objects in files. Each time information is registered or updated it will be saved.

R#7.EXPORT: A csv file of orders must be generated, including for each requested product record, the data of the restaurant, the client, and the requested product. The user exporting this file must specify the separator. The csv file must be sort.

R#8.IMPORT: The program must allow importing data from a csv file with information of restaurants, clients, products, or orders.

R#9:SHOW IN SCREEN: The program must have an option that allows all restaurants or clients to be listed on the screen.

List of restaurants: they will be in ascending alphabetical order.

List of clients: in order of their descending phone number.

R#10: FIND CLIENT: The program must have an option that allows you to efficiently search for a client given a name and indicate the time that the search took.

JUNIT TEST

CONFIGURATION OF THE STAGES:

Name	Class	Stage
setupStage1	ClientTest	Create a client
setupStage2	ClientTest	Create a client with orders and set the state of the order in delivered
setupScenary1	OrderTest	Create an order
setupScenary1	ProductTest	Create a product
setupScenary1	RestaurantAssociationTest	Create a restaurant association with a restaurant
setupScenary1	RestaurantTest	Create a restaurant with a product

TEST CASE:

TEST OBJETIVE: verifies if the identification exception works				
CLASS	METHOD	SCENARY	INPUT	RESULT
Client	Client	setupStage1	IdentificationNumber:21212 Phone:32343 identificationType:PO firstName:JUAN LastName: GOMEZ Addres: CIUDAD JARDIN	The Identification exception throws

TEST OBJETIVE: Verifies if the changeStateException works				
CLASS	METHOD	SCENARY	INPUT	RESULT
Client	chageState	setupStage2	State: "REQUESTED"	The changeStateException throws

TEST OBJETIVE:Verifies if the order makes with state of "REQUESTED"				
CLASS	METHOD	SCENARY	INPUT	RESULT
Order	Order	setupScenary1	CODE: 2121324 CLIENTCODE:232432 RESTAURANTNIT:23131 LIST<PRODUCT>	The new instance of order is correct

TEST OBJETIVE:Verifies if the setState method works				
CLASS	METHOD	SCENARY	INPUT	RESULT
Order	setState	setupScenary1	State: "IN PROCCES"	The state was change

TEST OBJETIVE:Verifies the correct update of all data				
CLASS	METHOD	SCENARY	INPUT	RESULT
Product	updateAllData	setupScenary1	CODE: 1212 COST:4343 NAME:"NOAJI" DESCRIPTION:"NOPICA"	All data was update

TEST OBJETIVE:Verifies de get name method works				
CLASS	METHOD	SCENARY	INPUT	RESULT
Product	getName	setupScenary1		The name of the product

TEST OBJETIVE: verifies if the searchRestaurant method works				
CLASS	METHOD	SCENARY	INPUT	RESULT
RestaurantAssociation	searchRestaurant	setupScenary1	NIT: 23131	Restaurant found
TEST OBJETIVE: verifies if the restaurant was correct register				
CLASS	METHOD	SCENARY	INPUT	RESULT
RestaurantAssociation	register Restaurant	setupScenary1	A restaurant	the restaurant was added

TEST OBJETIVE: Verifies if a product with a same code can't be added				
CLASS	METHOD	SCENARY	INPUT	RESULT
Restaurant	addProduct	setupScenary1	A product with same code	The product doesn't was added successfully

TEST OBJETIVE: Verifies the search of a product				
CLASS	METHOD	SCENARY	INPUT	RESULT
Restaurant	searchProduct	setupScenary1	Code: 23232	Product found