Juan Pablo Sanin
A00296776
Mr. Delivery

FR1. The program must allow restaurants to register with their respective data, which includes: a name, tax ID, and the manager's name.

FR2. The program must allow product registering using a code, name, description, price, and the tax ID of the restaurant it belongs to.

FR3. The program must allow client registering, a client must give the following information: Their id type(Passport, ID card, or driver's license), document number, full name, phone number, and address.

FR4. The program must allow order registering, the order must have an autogenerated code, date and hour (taken from the system using Date type), the code of the client that is ordering, the tax ID of the restaurant, and a list of products in the order, each product has a code and quantity.

FR5. The program has to be able to update a restaurant's information given the restaurant's tax ID.

FR6. The program has to be able to update a product's information given the product code.

FR7. The program has to be able to update a client's information given the ID type and number

FR8. The program has to be able to update an order's information given the order code.

FR9. The program has to have a list of clients that is always descending in alphabetical order using their first names and last names. If a new client is added, he must be added into his corresponding position, not added then ordered.

FR10. The program must allow the changing of order status in between requested, in process, shipped, and delivered. The status cannot go backward, only forward. Ej. (Shipped to in process is wrong, Shipped to Delivered is correct)

FR11. The program has to verify that all products selected by the client in the order are from the same restaurant.

FR12. The program must save all its information using serialization of its objects in files, the saving of the information must be transparent to the final user. The information must be saved each time something is registered or updated.

Juan Pablo Sanin
A00296776
Mr. Delivery

FR13. The program must generate a CSV file of orders including for each order its restaurant's data, client's data, and the products ordered. The file must be ordered using the following criteria:

1. **Ascendent** restaurant tax ID (0-9)
2. **Descendent** client id Number (9-0)
3. **Ascendent** order date (0-31)
4. **Ascendent** product code (0-9)

FR14. When the file is exported the program must ask for the separator of the information and include a first row indicating the information in each column.

FR15. The program must have an option that shows all of the registered restaurants ordered in **ascendent** alphabetical order.

FR16. The program must have an option that shows all of the registered clients ordered **descendingly** using their phone numbers.

FR17. The program must have an option that searches for a client efficiently using their name. The option must show the time it took to find it.

FR18. The program has to have at least 2 automatic unit test cases for each class in the model.

FR.19 The program has to use 2 out of the 3 sorting algorithms (bubble, insertion, or selection).

FR20. The program has to do 1 sorting using Comparable and 1 sorting using Comparator, using in both cases the sort method from Collections or Arrays.

FR21. The program must allow data to be imported into the program from a CSV file with restaurant information.

FR22. The program must allow data to be imported into the program from a CSV file with client information.

FR23. The program must allow data to be imported into the program from a CSV file with product information.

FR24. The program must allow data to be imported into the program from a CSV file with order information.

Juan Pablo Sanin
A00296776
Mr. Delivery

Test Case Design

Restaurant Test:

Scenario Configuration

Name	Class	Scenario
setUpStage1	RestaurantTest	Object from class Restaurant with name= Burger King, TaxID=9852564, manager name= Bobby Pujols

Test Objective: Verify that the class's constructor is working properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Restaurant	Restaurant	*none*	name=”McDonalds” taxId=889456 managerName= "Ricky Porter"	An object of the Restaurant class is created successfully with the parameters that are given.

Test Objective:.Verify that the class is adding products correctly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Restaurant	AddProduct	*setUpStage1*	int code=1547; String name= "Cheese Burger" String description= "150g of angus meat with cheese" double price = 6.99	A product is added successfully to the restaurant's menu items with the given parameters.

Juan Pablo Sanin
A00296776
Mr. Delivery

Client Test:

Scenario Configuration

Name	Class	Scenario
setUpStage1	ClientTest	Object from class Client with IdType.LICENSE, id=564654, name="Pedro Salazar",phone=3187284546L,address="421 Rocky rd"

Test Objective:.Verify that the class's constructor is working properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**

Client	Client	*none*	idType=PASSPORT;
			id= 457812;
			name= "Ricky Rubio";
			phone=5648881234;
			address="634 Buckingham St";
				An object of the Client class is created successfully with the parameters that are given.

Test Objective:.Verify that the class is adding orders correctly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**

Client	AddOrder	*setUpStage1*	int clientID=564654;
			
			int taxId=547623;
				An order is added successfully to the client's orders with the given parameters.

Juan Pablo Sanin
A00296776
Mr. Delivery

Product Test:

Scenario Configuration

Name	Class	Scenario
setUpStage1	ProductTest	Object from class Product with code=962, name="French Fries", description="Classic French Fries",price= 3.99, restaurant tax id=72703

Test Objective:.Verify that the class's constructor is working properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**

Product	Product	*none*	code=1547;
			name= "Cheese Burger";
			description= "150g of angus meat with cheese";
			price = 6.99;
			resTaxID=421077
				An object of the Product class is created successfully with the parameters that are given.

Test Objective:.Verify that the class is displaying information correctly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Product	ToString	*setUpStage1*	none	Product information is displayed correctly.

Juan Pablo Sanin
A00296776
Mr. Delivery

Order Test:

Scenario Configuration

Name	Class	Scenario
setUpStage1	OrderTest	Object from class Order with client id=141422, restaurant tax id= 665544

Test Objective:.Verify that the class's constructor is working properly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Order	Order	*none*	clientID=564654; taxId=547623;	An object of the Order class is created successfully with the parameters that are given.

Test Objective:.Verify that the class is adding products correctly				
Class	**Method**	**Scenario**	**Entry Values**	**Result**
Order	addProduct	*setUpStage1*	int code=1547; name= "Cheese Burger"; description= "150g of angus meat with cheese"; price = 6.99; quantity=4;	The product is added successfully with the correct quantity and parameters given.