REQUERIMIENTOS FUNCIONALES

RF1: Registrar un restaurante, el cual debe tener como atributos el nombre, nit y nombre del administrador. Debe verificar que el nit sea único.

RF2: Registrar un producto, el cual debe tener como atributos un código, nombre, una descripción, costo y nit del restaurante que ofrece dicho producto. Debe verificar que el código sea único.

RF3: Registrar un cliente, el cual debe tener como atributos un tipo de identificación, un número de identificación, nombre completo, teléfono y dirección. Debe verificar que el número de identificación sea único.

RF4: Registrar un pedido, el cual debe tener como atributos un código de pedido (autogenerado), una fecha y hora (tomada del sistema y de tipo Date), el código del cliente que está haciendo el pedido, el nit del restaurante, y una lista de productos a pedir, cada producto a pedir tiene el código del producto y la cantidad. El código de la orden y del producto agregado debe ser único y deben pertenecer al mismo restaurante.

RF5: Actualizar todos los datos de un restaurante dando su nit.

RF6: Actualizar todos los datos de un producto dado su código.

RF7: Actualizar todos los datos de un cliente dado su número de documento.

RF8: Actualizar todos los datos de un pedido dado su código.

RF9: Insertar ordenadamente siempre la lista de clientes alfabéticamente por nombre y apellido.

RF10: Actualizar el estado de una orden entre SOLICITADO, EN PROCESO, ENVIADO y ENTREGADO en un solo sentido.

RF11: Serializar toda la información del programa.

RF12: Generar un archivo csv de pedidos, incluyendo para cada registro de producto solicitado, los datos del restaurante, del cliente y del producto pedido. El listado debe estar ordenado por los siguientes criterios en este orden: nit del restaurante ascendente, documento del cliente descendente, fecha del pedido ascendente y código del producto ascendente. En el momento en que el usuario elija la opción de exportar esta información, debe preguntársele cuál es el separador que se utilizará. La primera línea del archivo debe tener los nombres de las columnas separadas también por dicho separador.

RF13: Listar en pantalla todos los restaurantes en orden alfabético ascendente.

RF14: Listar en pantalla todos los clientes en orden de su número telefónico descendente, pero dejando de nuevo el arrayList como estaba.

RF15: buscar eficientemente un cliente dado un nombre e indicar el tiempo que tardó la búsqueda.

RF16: Importar datos de un archivo csv con información de restaurantes.

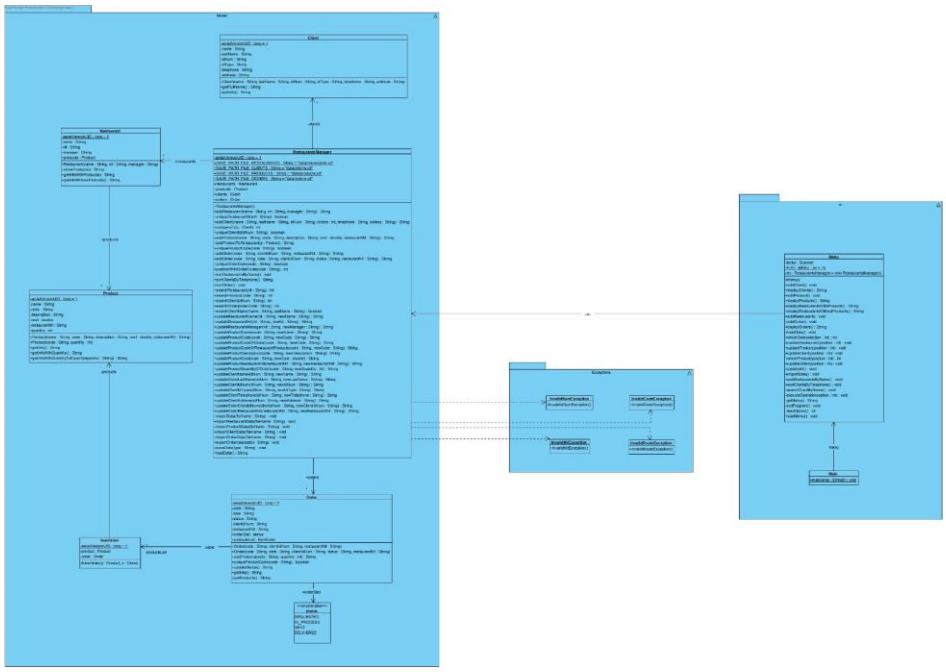
RF17: Importar datos de un archivo csv con información de productos.

RF18: Importar datos de un archivo csv con información de clientes.

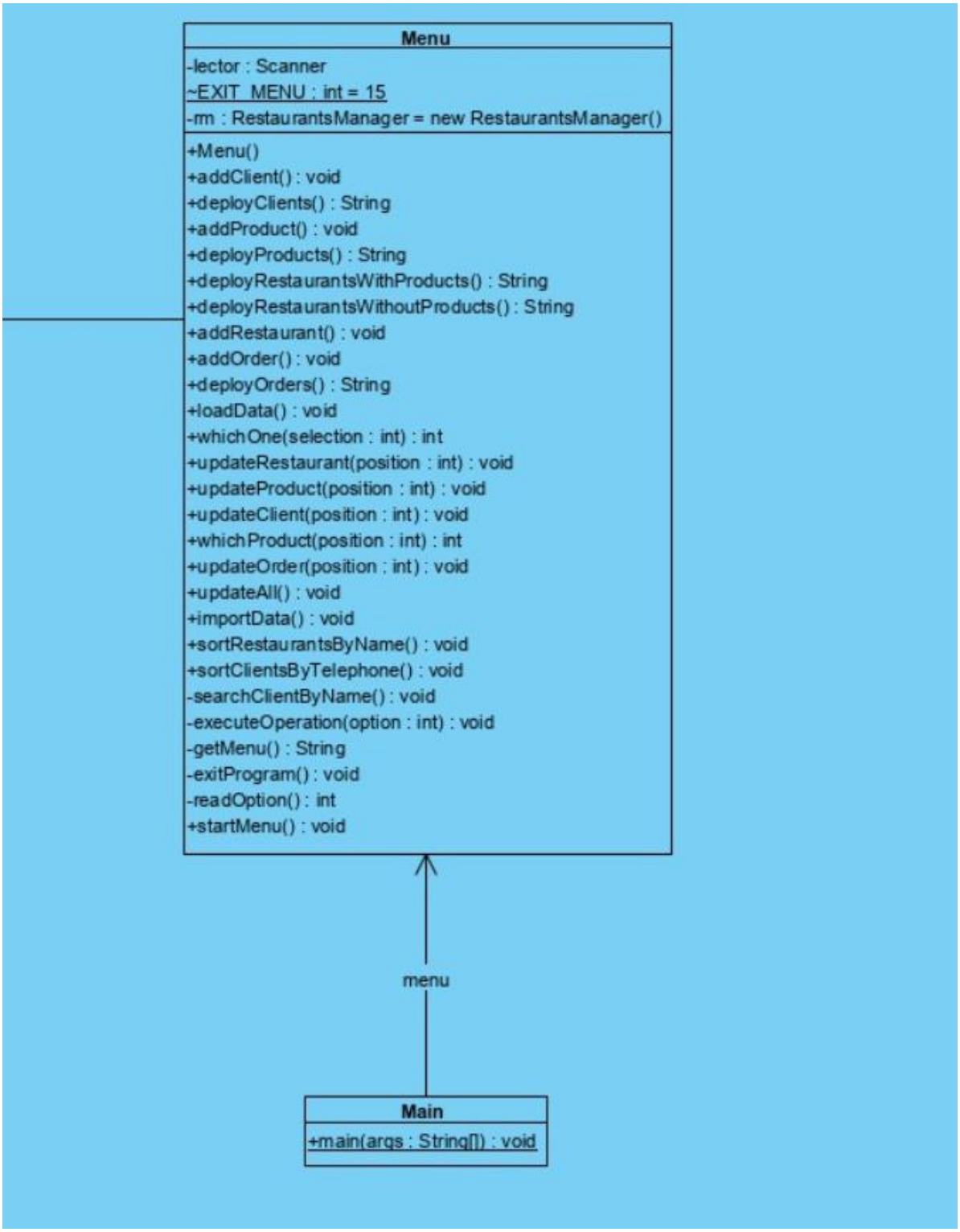
RF19: Importar datos de un archivo csv con información de órdenes.

DIAGRAMA DE CLASES

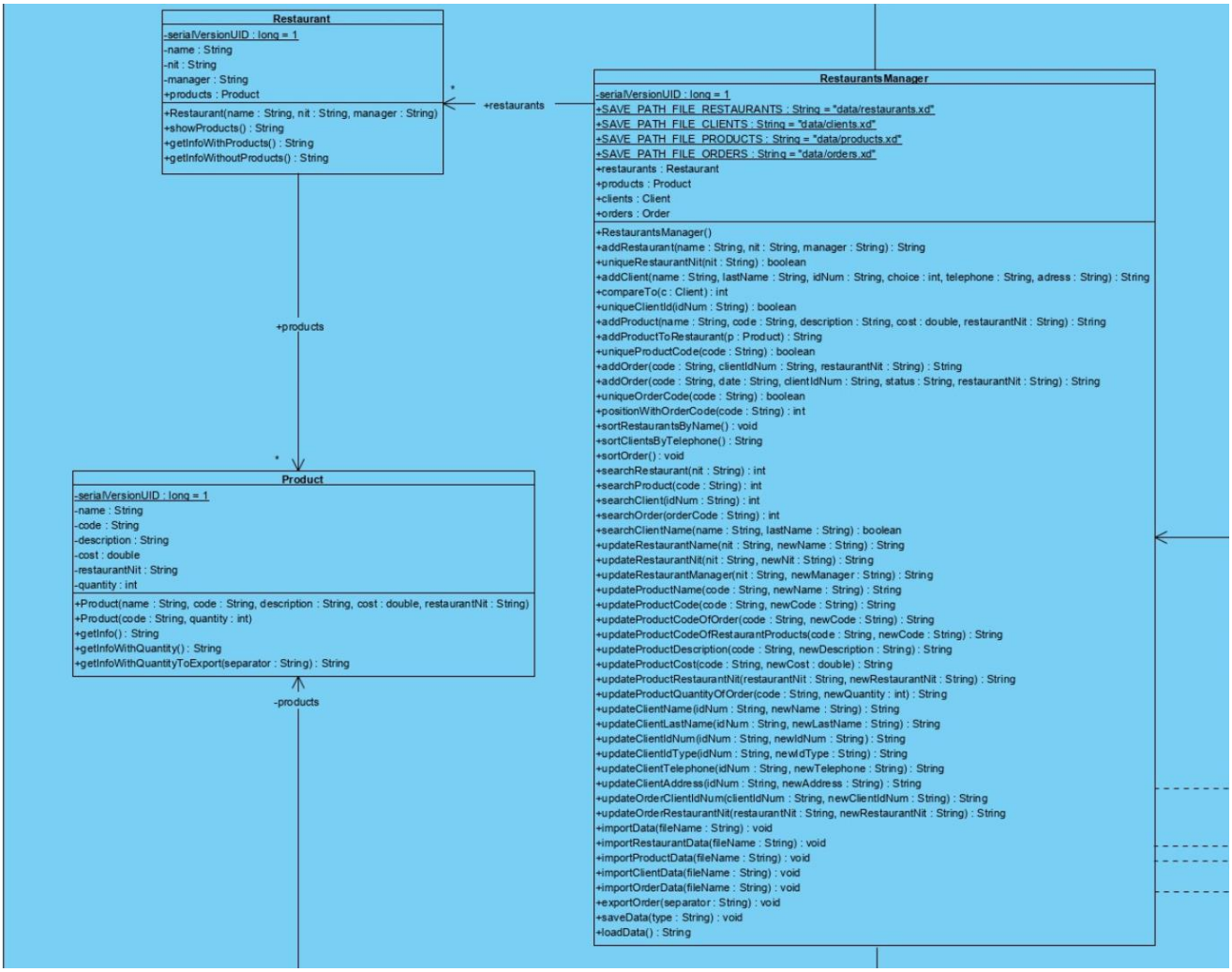
En general

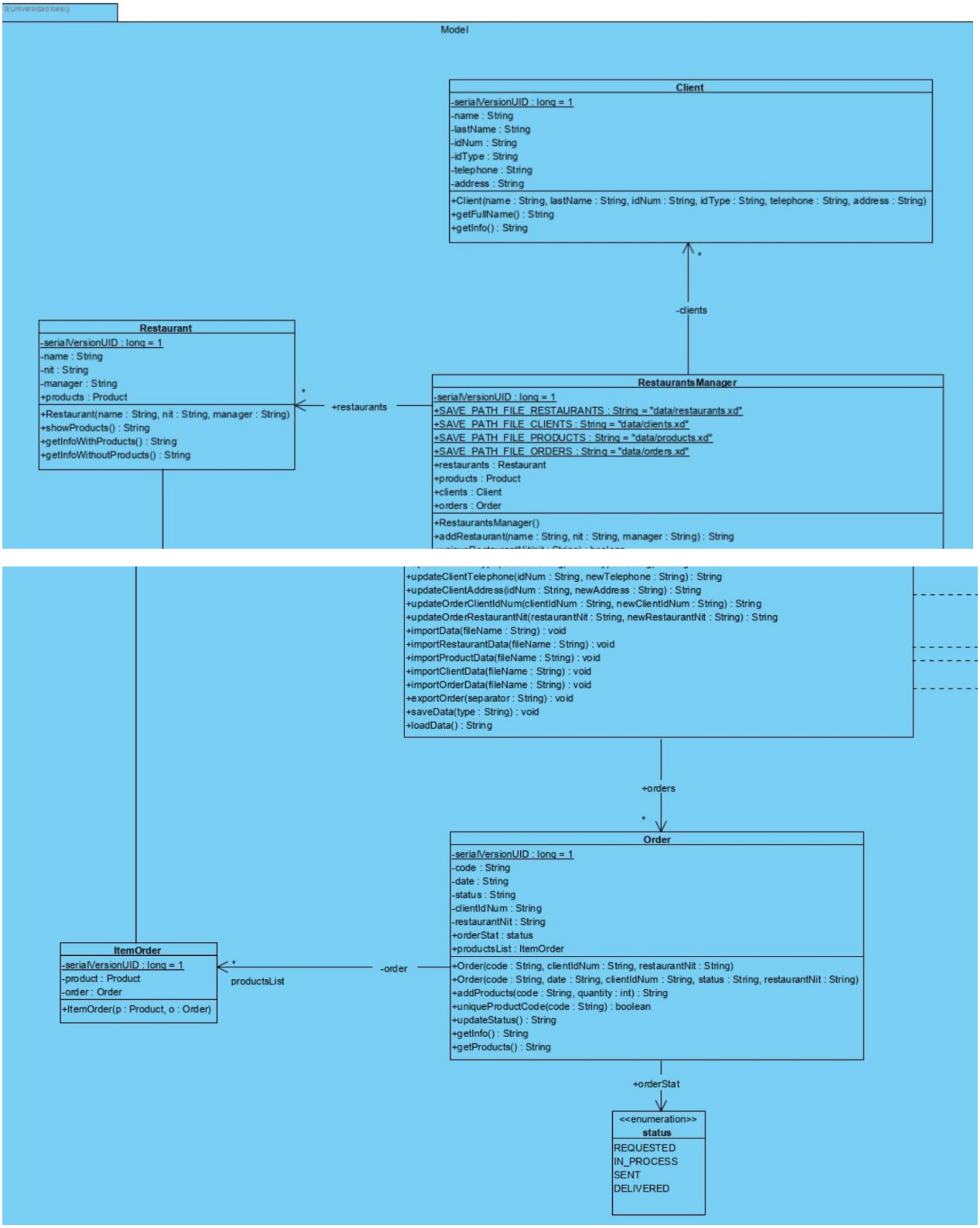


ui

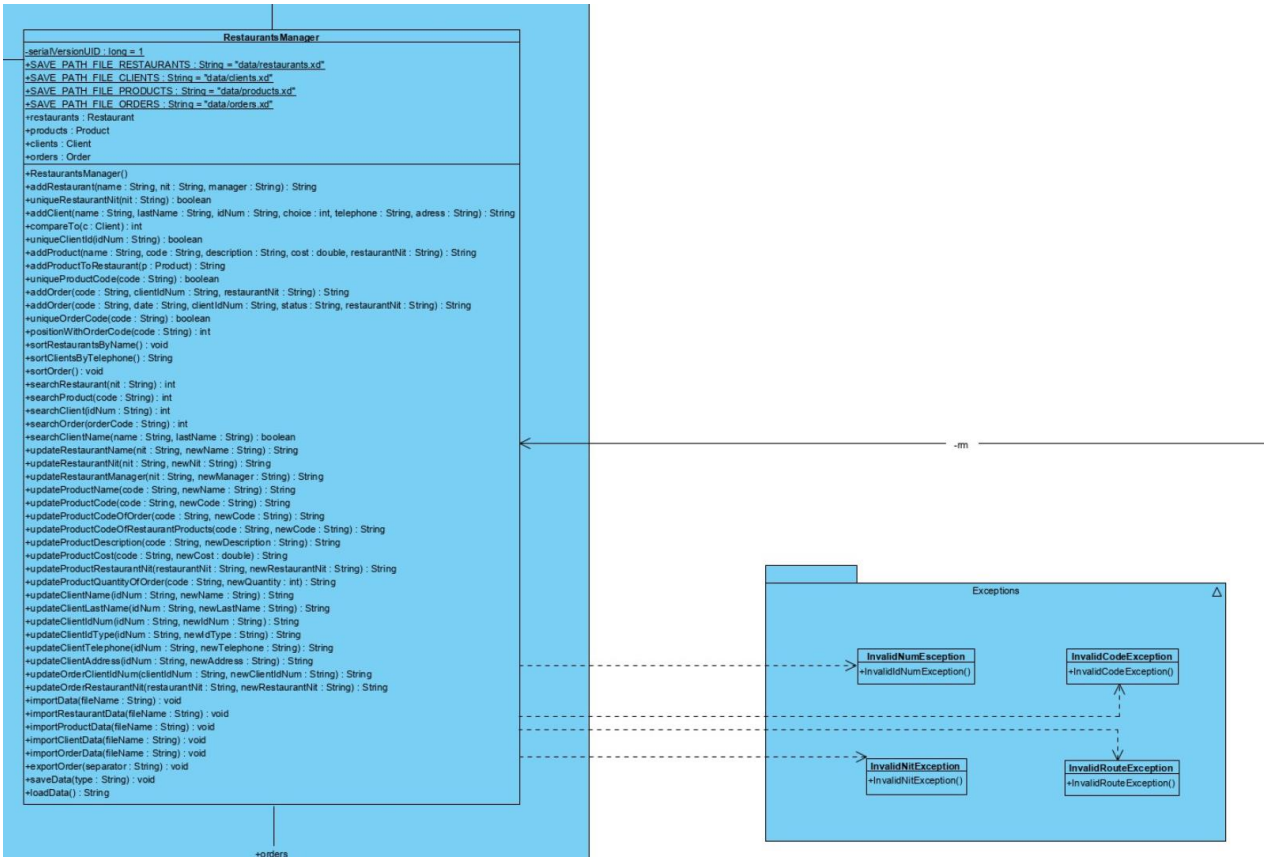

Menu
-lector : Scanner
~EXIT_MENU : int = 15
-rm : RestaurantsManager = new RestaurantsManager()
+Menu()
+addClient() : void
+deployClients() : String
+addProduct() : void
+deployProducts() : String
+deployRestaurantsWithProducts() : String
+deployRestaurantsWithoutProducts() : String
+addRestaurant() : void
+addOrder() : void
+deployOrders() : String
+loadData() : void
+whichOne(selection : int) : int
+updateRestaurant(position : int) : void
+updateProduct(position : int) : void
+updateClient(position : int) : void
+whichProduct(position : int) : int
+updateOrder(position : int) : void
+updateAll() : void
+importData() : void
+sortRestaurantsByName() : void
+sortClientsByTelephone() : void
-searchClientByName() : void
-executeOperation(option : int) : void
-getMenu() : String
-exitProgram() : void
-readOption() : int
+startMenu() : void
menu
Main
+main(args : String[]) : void

model


Restaurant
-serialVersionUID : long = 1
-name : String
-nit : String
-manager : String
+products : Product
+Restaurant(name : String, nit : String, manager : String)
+showProducts() : String
+getInfoWithProducts() : String
+getInfoWithoutProducts() : String
*
+restaurants
RestaurantsManager
-serialVersionUID : long = 1
+SAVE_PATH_FILE_RESTAURANTS : String = "data/restaurants.xd"
+SAVE_PATH_FILE_CLIENTS : String = "data/clients.xd"
+SAVE_PATH_FILE_PRODUCTS : String = "data/products.xd"
+SAVE_PATH_FILE_ORDERS : String = "data/orders.xd"
+restaurants : Restaurant
+products : Product
+clients : Client
+orders : Order
+RestaurantsManager()
+addRestaurant(name : String, nit : String, manager : String) : String
+uniqueRestaurantNit(nit : String) : boolean
+addClient(name : String, lastName : String, idNum : String, choice : int, telephone : String, adress : String) : String
+compareTo(c : Client) : int
+uniqueClientId(idNum : String) : boolean
+addProduct(name : String, code : String, description : String, cost : double, restaurantNit : String) : String
+addProductToRestaurant(p : Product) : String
+uniqueProductCode(code : String) : boolean
+addOrder(code : String, clientIdNum : String, restaurantNit : String) : String
+addOrder(code : String, date : String, clientIdNum : String, status : String, restaurantNit : String) : String
+uniqueOrderCode(code : String) : boolean
+positionWithOrderCode(code : String) : int
+sortRestaurantsByName() : void
+sortClientsByTelephone() : String
+sortOrder() : void
+searchRestaurant(nit : String) : int
+searchProduct(code : String) : int
+searchClient(idNum : String) : int
+searchOrder(orderCode : String) : int
+searchClientName(name : String, lastName : String) : boolean
+updateRestaurantName(nit : String, newName : String) : String
+updateRestaurantNit(nit : String, newNit : String) : String
+updateRestaurantManager(nit : String, newManager : String) : String
+updateProductName(code : String, newName : String) : String
+updateProductCode(code : String, newCode : String) : String
+updateProductCodeOfOrder(code : String, newCode : String) : String
+updateProductCodeOfRestaurantProducts(code : String, newCode : String) : String
+updateProductDescription(code : String, newDescription : String) : String
+updateProductCost(code : String, newCost : double) : String
+updateProductRestaurantNit(restaurantNit : String, newRestaurantNit : String) : String
+updateProductQuantityOfOrder(code : String, newQuantity : int) : String
+updateClientName(idNum : String, newName : String) : String
+updateClientLastName(idNum : String, newLastName : String) : String
+updateClientIdNum(idNum : String, newIdNum : String) : String
+updateClientIdType(idNum : String, newIdType : String) : String
+updateClientTelephone(idNum : String, newTelephone : String) : String
+updateClientAddress(idNum : String, newAddress : String) : String
+updateOrderClientIdNum(clientIdNum : String, newClientIdNum : String) : String
+updateOrderRestaurantNit(restaurantNit : String, newRestaurantNit : String) : String
+importData(fileName : String) : void
+importRestaurantData(fileName : String) : void
+importProductData(fileName : String) : void
+importClientData(fileName : String) : void
+importOrderData(fileName : String) : void
+exportOrder(separator : String) : void
+saveData(type : String) : void
+loadData() : String
+products
*
Product
-serialVersionUID : long = 1
-name : String
-code : String
-description : String
-cost : double
-restaurantNit : String
-quantity : int
+Product(name : String, code : String, description : String, cost : double, restaurantNit : String)
+Product(code : String, quantity : int)
+getInfo() : String
+getInfoWithQuantity() : String
+getInfoWithQuantityToExport(separator : String) : String
-products


Model
Client
-serialVersionUID : long = 1
-name : String
-lastName : String
-idNum : String
-idType : String
-telephone : String
-address : String
+Client(name : String, lastName : String, idNum : String, idType : String, telephone : String, address : String)
+getFullName() : String
+getInfo() : String
*
-clients
Restaurant
-serialVersionUID : long = 1
-name : String
-nit : String
-manager : String
+products : Product
+Restaurant(name : String, nit : String, manager : String)
+showProducts() : String
+getInfoWithProducts() : String
+getInfoWithoutProducts() : String
*
+restaurants
RestaurantsManager
-serialVersionUID : long = 1
+SAVE_PATH_FILE_RESTAURANTS : String = "data/restaurants.xd"
+SAVE_PATH_FILE_CLIENTS : String = "data/clients.xd"
+SAVE_PATH_FILE_PRODUCTS : String = "data/products.xd"
+SAVE_PATH_FILE_ORDERS : String = "data/orders.xd"
+restaurants : Restaurant
+products : Product
+clients : Client
+orders : Order
+RestaurantsManager()
+addRestaurant(name : String, nit : String, manager : String) : String
+updateClientTelephone(idNum : String, newTelephone : String) : String
+updateClientAddress(idNum : String, newAddress : String) : String
+updateOrderClientIdNum(clientIdNum : String, newClientIdNum : String) : String
+updateOrderRestaurantNit(restaurantNit : String, newRestaurantNit : String) : String
+importData(fileName : String) : void
+importRestaurantData(fileName : String) : void
+importProductData(fileName : String) : void
+importClientData(fileName : String) : void
+importOrderData(fileName : String) : void
+exportOrder(separator : String) : void
+saveData(type : String) : void
+loadData() : String
+orders
*
Order
-serialVersionUID : long = 1
-code : String
-date : String
-status : String
-clientIdNum : String
-restaurantNit : String
+orderStat : status
+productsList : ItemOrder
+Order(code : String, clientIdNum : String, restaurantNit : String)
+Order(code : String, date : String, clientIdNum : String, status : String, restaurantNit : String)
+addProducts(code : String, quantity : int) : String
+uniqueProductCode(code : String) : boolean
+updateStatus() : String
+getInfo() : String
+getProducts() : String
ItemOrder
-serialVersionUID : long = 1
-product : Product
-order : Order
+ItemOrder(p : Product, o : Order)
*
productsList
-order
+orderStat
<<enumeration>>
status
REQUESTED
IN_PROCESS
SENT
DELIVERED

model – exceptions



DISEÑOS DE CASOS DE PRUEBA				
JETIVO DE LA PRUEBA: Verificar que un cliente se añada correctamente, ordenadamente y funcione su get id n				
CLASE	MÉTODO	ESCENARIO	VALORES DE ENTRADA	RESULTADO
Client	testClientSortedAdded	setupScenary_1	Cliente: Name = "Mario" LastName = "Casas" IdNum = "123456789" idType = 3 Telephone = "987654321" Address = "Cali"	True. Agrega correctamente y ordenadamente ya que al realizar la prueba en la clase Client y retornar el IdNum de la posición 1 del array corresponde a "123456789"
OBJETIVO DE LA PRUEBA: Verificar que un cliente se pueda añadir correctamente, ordenadamente y funcione su get full name				
Client	testClientFullName	setupScenary_1	Cliente: Name = "Pedro" LastName = "Randonga" IdNum = "1548225645" idType = 4 Telephone = "3124865794" Address = "Cali"	True. Agrega correctamente y ordenadamente ya que al realizar la prueba en la clase Client y retornar el nombre completo de la posición 0 del array corresponde a "Pedro Randonga"
OBJETIVO DE LA PRUEBA: Verificar que una orden se añada correctamente y retorne el tamaño esperado				
Order	testOrder	setupScenary_1	Cliente Mario: igual que el de la prueba anterior. Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Producto: name = "Tequila" Code = "2153" Description = "Directo desde	True. Agrega la orden y devuelve el size esperado
OBJETIVO DE LA PRUEBA: Verificar que el constructor de orden funciona correctamente				
Order	testUpdateStatus	setupScenary_1	Cliente Mario: igual que el de la prueba anterior. Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Producto: name = "Tequila" Code = "2153" Description = "Directo desde méxico" Cost = 20000 RestaurantNit = "123456789". Orden: Client ID="123456789" RestaurantNit = "123456789". Producto en lista de orden: code = 2153 quantity = 2	True. Actualiza el status de la orden de "REQUESTED" a "IN_PROCESS"
OBJETIVO DE LA PRUEBA: Verificar que cuando se agrega un producto, se remita al arrayList de su respectivo				

Product	testAddProduct_1	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Producto: name = "Tequila" Code = "2153" Description = "Directo desde méxico" Cost = 20000 RestaurantNit = "123456789". Producto: name = "Cerveza" Code = "1248" Description = "De trigo" Cost = 3500 RestaurantNit = "123456789"	True. El producto está en el arrayList del restaurante registrado con NIT = "123456789"
OBJETIVO DE LA PRUEBA: Verificar que se añada un solo producto si se intenta añadir dos con el mismo				
Product	testAddProduct_2	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Producto: name = "Tequila" Code = "2153" Description = "Directo desde méxico" Cost = 20000 RestaurantNit = "123456789". Producto: name = "Cerveza" Code = "1248" Description = "De trigo" Cost = 3500 RestaurantNit = "123456789"	True. El producto sólo se añadió la primera vez, el tamaño del array es 1
OBJETIVO DE LA PRUEBA: Verificar que se añada un restaurante correctamente				
RestaurantsManager	testAddRestaurant	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho"	True. El arrayList de restaurantes no es null.
OBJETIVO DE LA PRUEBA: Verificar que actualiza correctamente el nombre del restaurante				

RestaurantsManager	testUpdateRestaurantName	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho"	True. El restaurante cambió su nombre correctamente a "Comidas rápidas Juan"
OBJETIVO DE LA PRUEBA: Verificar que actualiza correctamente el nombre del restaurante				
Restaurant	testSameRestaurantsNit_1	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Restaurante: Name = "Manciladopidira" Nit = "123456789" Manager = "María"	True. El arrayList de restaurantes retornó el size adecuado.
OBJETIVO DE LA PRUEBA: Verificar que actualiza correctamente el nombre del restaurante				
Restaurant	testSameRestaurantsNit_2	setupScenary_1	Restaurante: Name = "Posadero de la esquina" Nit = "123456789" Manager = "Poncho". Restaurante: Name = "Manciladopidira" Nit = "123456789" Manager = "María"	True. El primer elemento del arrayList de restaurantes retornó el nombre que debía retornar
OBJETIVO DE LA PRUEBA: Verificar que el get product funciona adecuadamente				
ItemOrder	testGetProduct	setupScenary_1	Order: Code = "l1hjqmus1f" ClientIdNum = "1654898765" RestaurantNit = "3151598465". With Product: Code = "4898465132" Quantity = 5.	True. El objeto ItemOrder creado no retorna null al llamar al método get product
OBJETIVO DE LA PRUEBA: Verificar que el get order funciona adecuadamente				
ItemOrder	testGetOrder	setupScenary_1	Order: Code = "l1hjqmus1f" ClientIdNum = "1654898765" RestaurantNit = "3151598465". With Product: Code = "4898465132" Quantity = 5.	True. El objeto ItemOrder creado no retorna null al llamar al método get order