Group Project 2

Business Rules

Rule number	Rule
Rule 1	Each member ID is unique to a single member
Rule 2	No two products have the same name
Rule 3	Each item is ordered separately and received as a separate shipment
Rule 4	When a product's level reaches the reorder level or below, an order will be made for twice the minimum reorder amount (unless an unfulfilled order has already been placed)
Rule 5	When inventory of an item is below the threshold, reorder the item at double the threshold value
Rule 6	A user may visit the store multiple times on a single day.
Rule 7	There cannot be more than 1 unfulfilled order for a product

Use Case Diagrams

1) Enroll a member: **Ayden Sinn**

Actions performed by the actor	Responses from the system
1. The customer fills out an application form containing the customer's name, address, phone number, date joined, and paid fee amount. The customer gives this to the employee	
2. The employee issues a request to add a new member	
	3. The system asks for data about the new member
4. The employee enters the data into the system	
	5. The system reads the data and generates an identification number if the member can be added. Information is remembered about the member. The
	system asks the clerk if the member was added and outputs the member's name, address, phone number, date joined, paid fee amount, and id number.
6. The new member is given their id number by the employee	

4) Add products: Ayden Sinn

Actions performed by the actor	Responses from the system
1. The employee issues a request to add a new product	
	2. The system asks for the product name, id, current price, and a minimum reorder level for the product
3. The employee inputs a product name, id, price, and minimum reorder level	

	4. If the input is valid, the system remembers information about the product. The system automatically generates an order for twice the minimum reorder level of the product

5. Check out a member’s cart: John Quinlan

Actions performed by the actor	Responses from the system
1. The member comes to the checkout counter with a cart of grocery items and the employee issues a request to check out a member	
	2. The system requests for the member ID
3. The employee inputs the member ID into the system	
	4. If the member ID is valid, the system asks for the product ID and quantity of the item;If invalid, it prints an appropriate message and exits the use case
5. The employee inputs the product ID of the product and enters the quantity of the product	
	6. If the product ID is valid, the system updates the cart with the quantity of the item; then asks if there are more grocery items
7. The employee responds positive or negative in regard to input of additional items	
	8.If there are more items for checking out, the system goes back to step 5; Otherwise the system displays the individual items, the number of units, unit prices, and price for the item and computes the total price and requests payment.
9. The employee takes payment from the member	

	10. The payment is collected, sale closed and transaction recorded. Inventory levels are updated.
11. The employee collects the groceries and bags and puts them in the member’s cart. The member leaves the co-op with their groceries	

7. Process Shipment: Qaalib Farah

Actor	System
1. The employee issues a request to process a new shipment.	
	2. System prompts user for product ID and quantity.
3. The employee enters the product ID and the quantity of the shipment.	

	4. If the product ID is valid the system updates the stock of the product and returns the product id, name and the new stock amount. It then asks if the user wants to process other shipments.
5. The employee answers in the affirmative or negative and sets the shipment aside to the processed section.	
	6. If the answer is affirmative, the system goes to step 2. Otherwise, the system exits.

9. Print transactions: Nate Goetsch

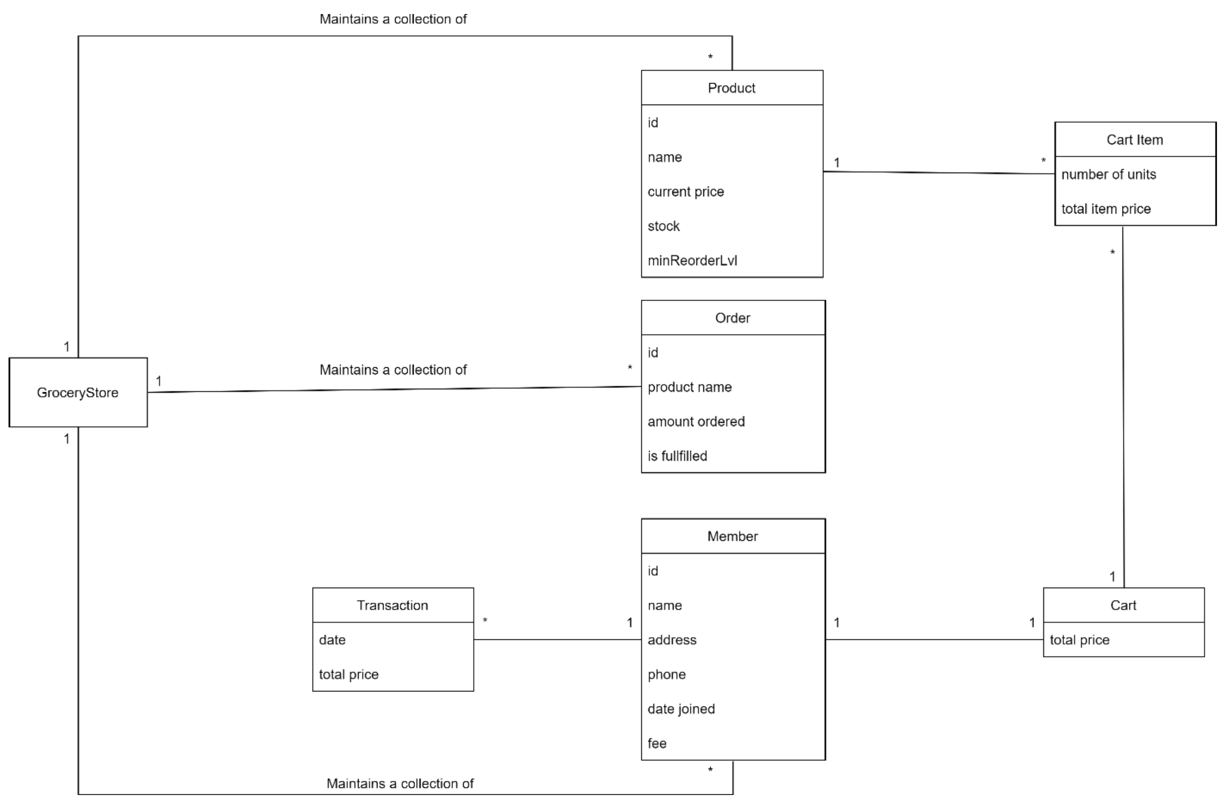
Actions performed by the actor	Response from the system
1. A request is issued to get a member's transaction.	
	2. The system asks for the member ID and two dates (input in mm/dd/yyyy)
3. The employee enters the member id and two dates(in the format mm/dd/yyyy)..	
	4. If the member id is valid, and that the first date is not after the second date, then the system prints out the information about each visit by the member between the two input dates: a. The date on which the user visited the store b. The total price paid during that visit If the information entered is not valid, then an appropriate error message is returned. Resulting in the employee(clerk) to re-input the member id and the two dates.

5.The employee gives the customer a transaction that includes the date when the user visited the store, and the total price paid during that visit.	

12. List all order: Leng Vang

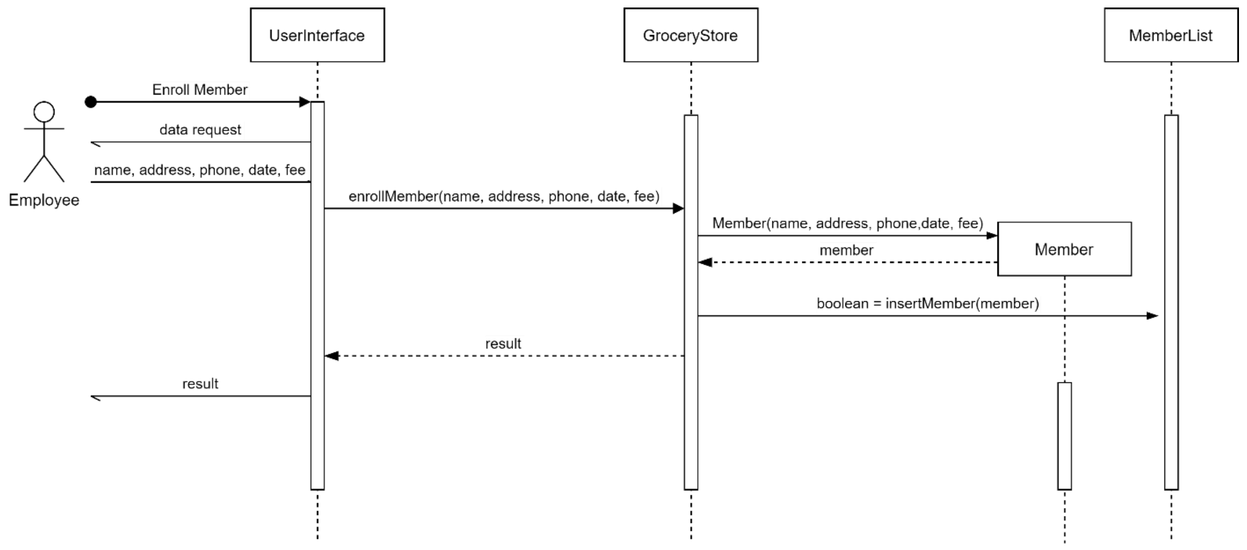
Actions performed by actor	Responses from the system
1. Employee issues a request for all outstanding orders.	
	2. System displays all outstanding products by names, id, and quantity.

Conceptual Class Diagram



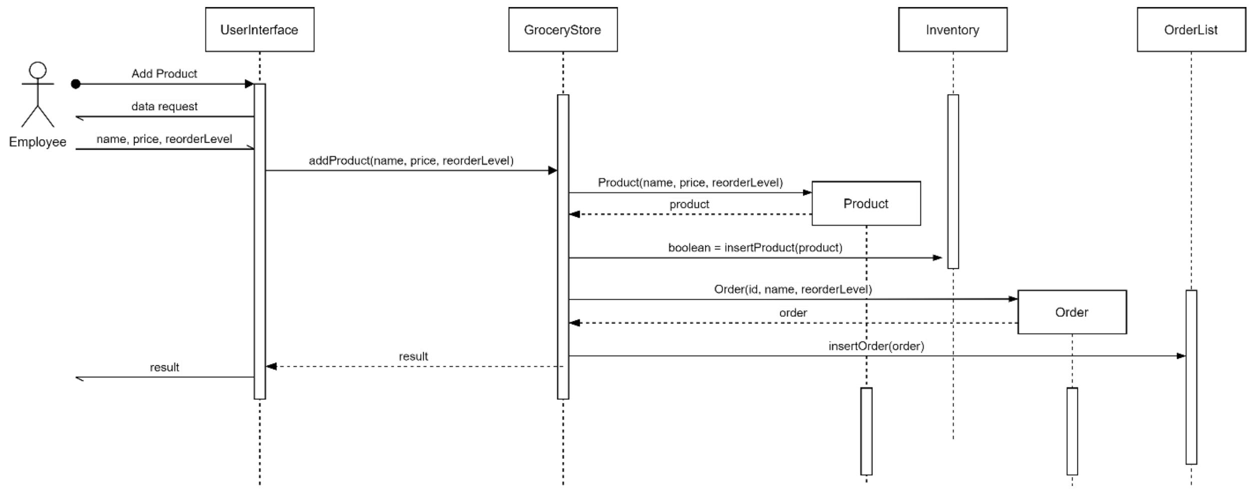
Sequence Diagrams

Enroll Member: Ayden Sinn



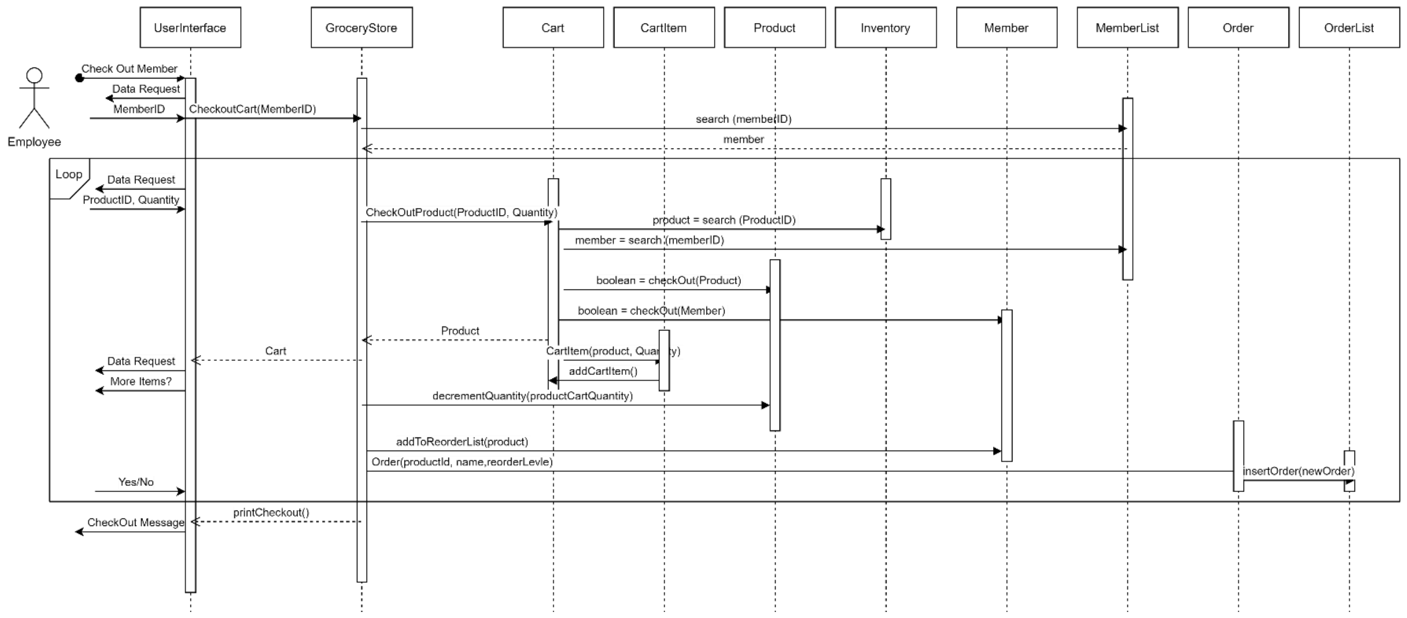
UserInterface
GroceryStore
MemberList
Enroll Member
data request
name, address, phone, date, fee
Employee
enrollMember(name, address, phone, date, fee)
Member(name, address, phone,date, fee)
member
Member
boolean = insertMember(member)
result
result

Add Product: Ayden Sinn

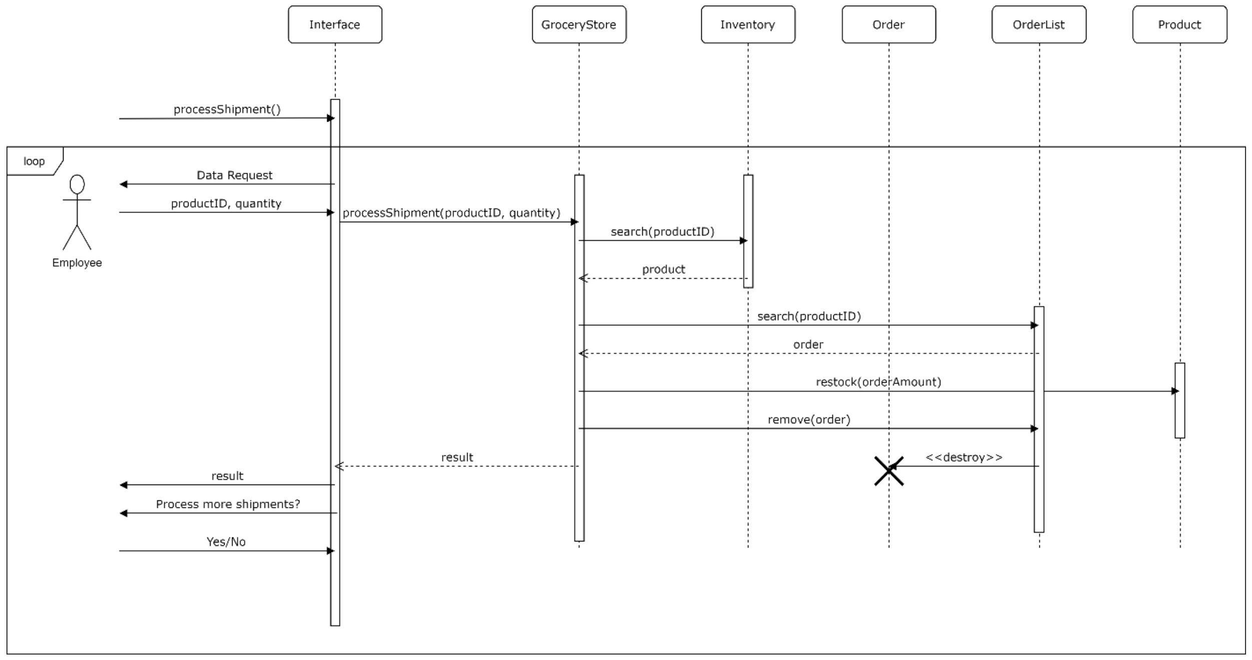


UserInterface
GroceryStore
Inventory
OrderList
Add Product
data request
name, price, reorderLevel
Employee
addProduct(name, price, reorderLevel)
Product(name, price, reorderLevel)
product
Product
boolean = insertProduct(product)
Order(id, name, reorderLevel)
order
Order
insertOrder(order)
result
result

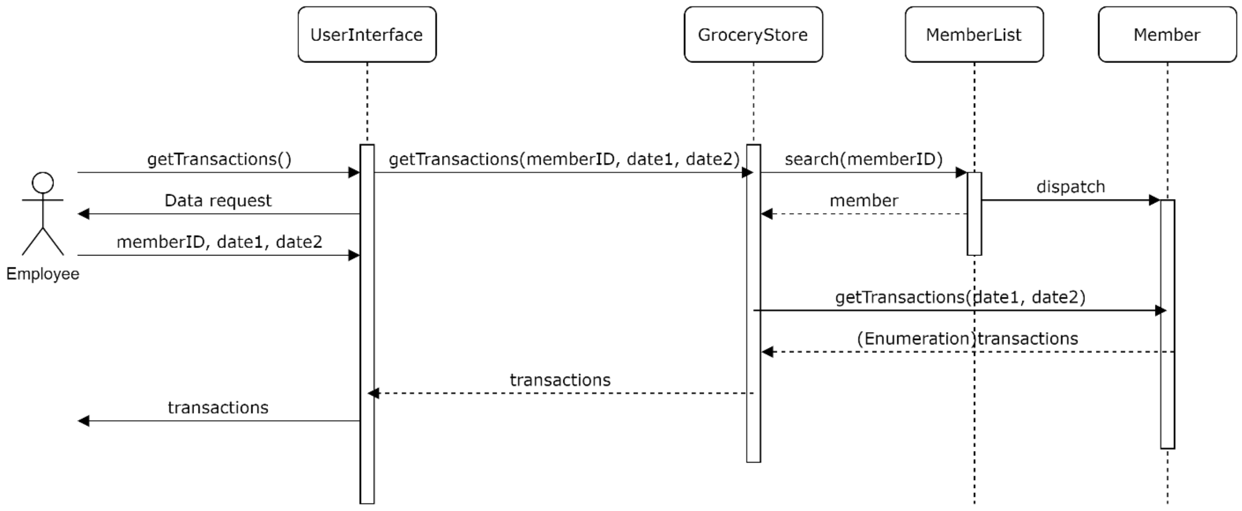
Checkout Cart: John Quinlan



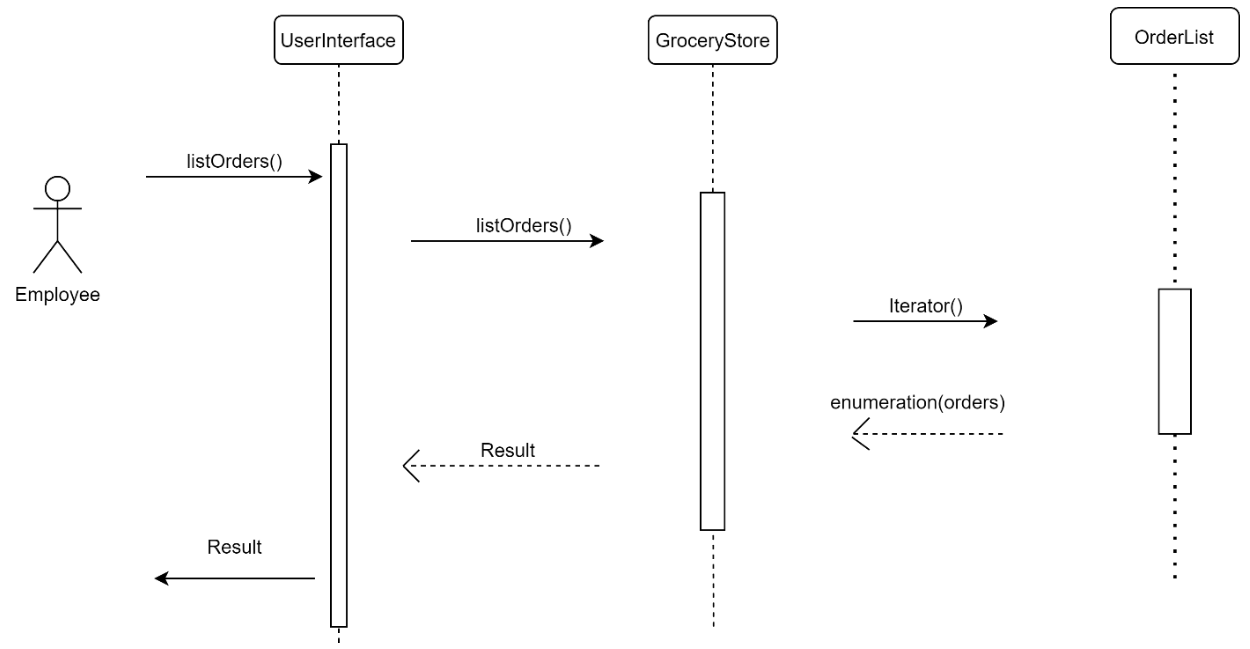
Process Shipment: Qaalib Farah



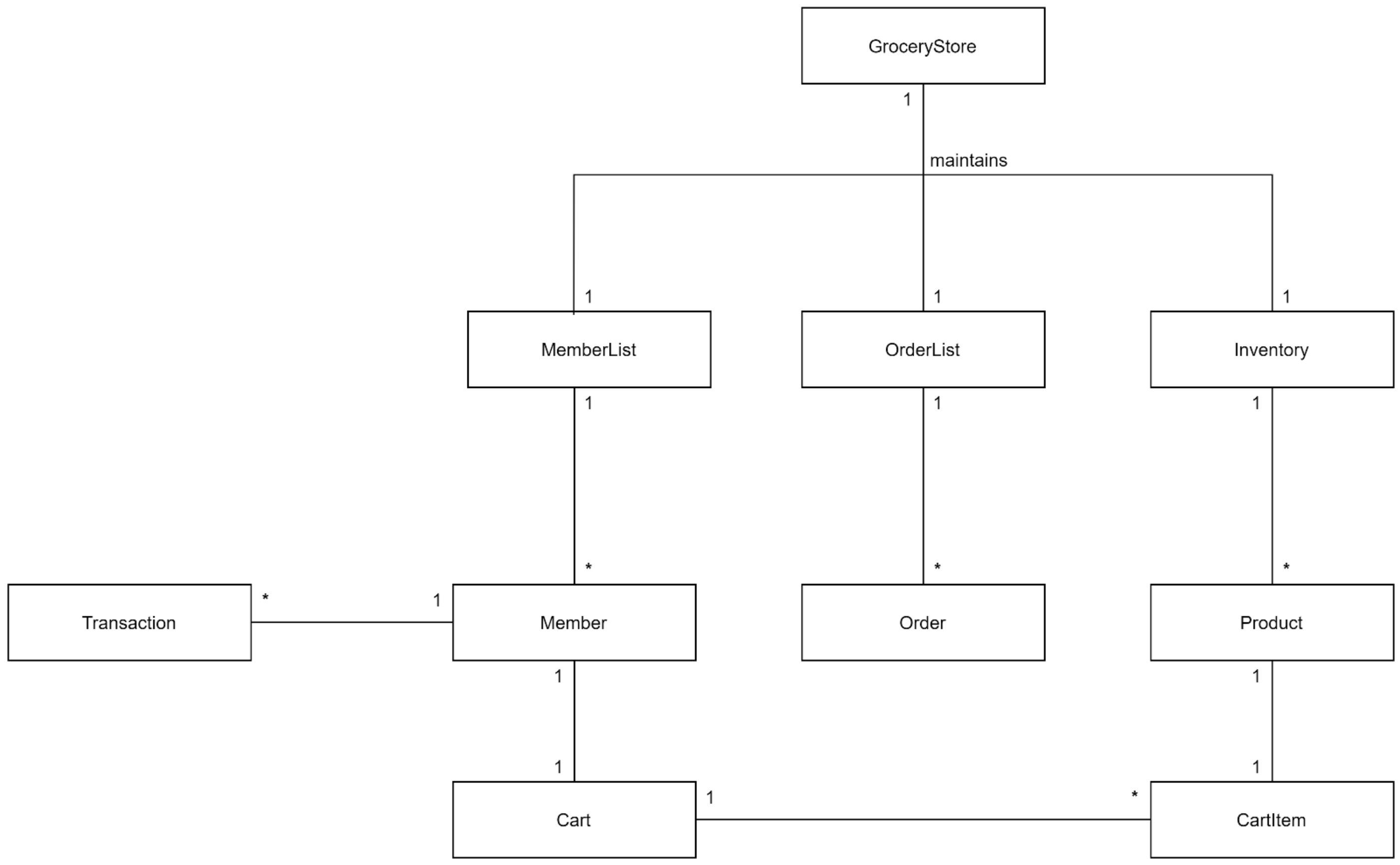
Print Transactions: Nate Goetsch



List Orders: Leng Vang



Class Diagrams



GroceryStore

- members: MemberList

- inventory: Inventory

- orders: OrderList

+ enrollMember(name: String, address: String, phone: int, joinDate: calendar, fee: int): Membe

+ removeMember(memberID: String): int

+ retrieveMember(memberID: String): Member

+ addProduct(name: String, price: double, reorderLevel: int): Product

+ searchMember(memberID: String): Member

+ retrieveProduct(productID: String): Product

+ processShipment(productID: String): Product

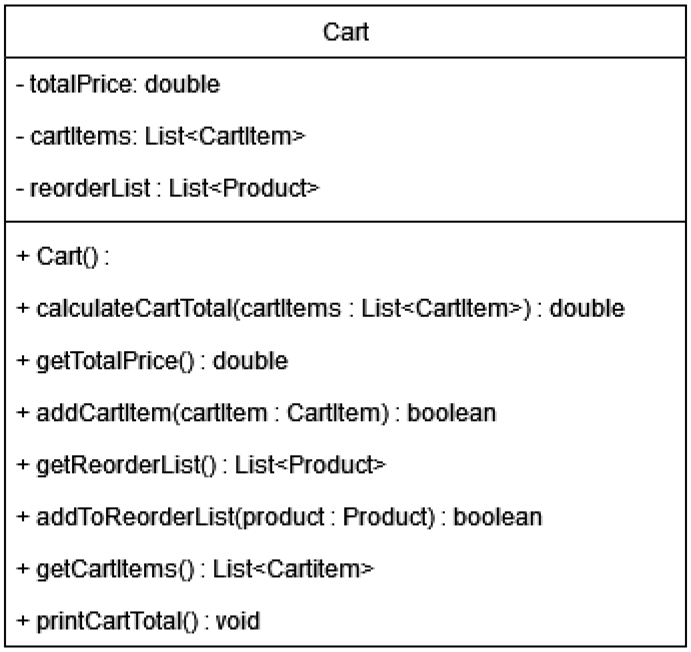
+ changePrice(productID: String, price: double): Product

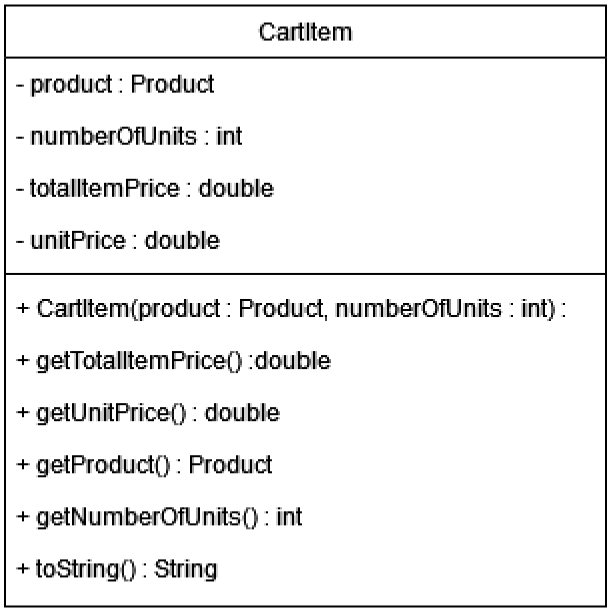
+ getTransactions(memberID: String): Iterator

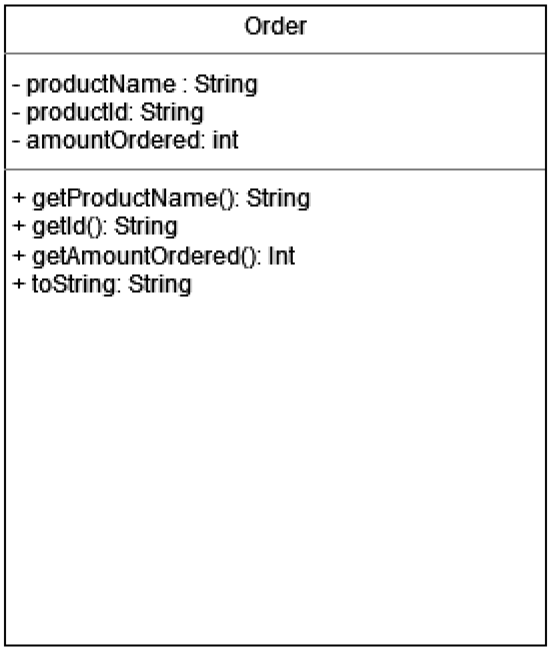
+ listMembers(): Iterator

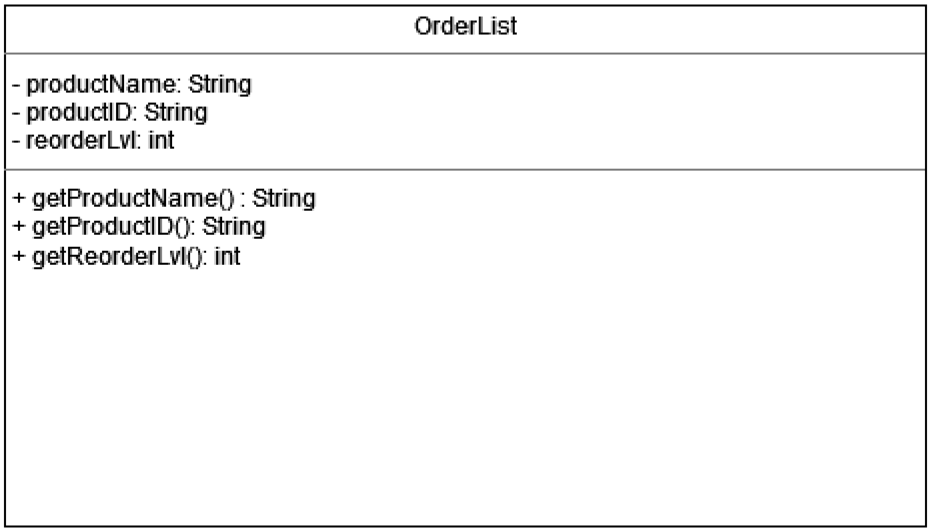
+ listProducts(): Iterator

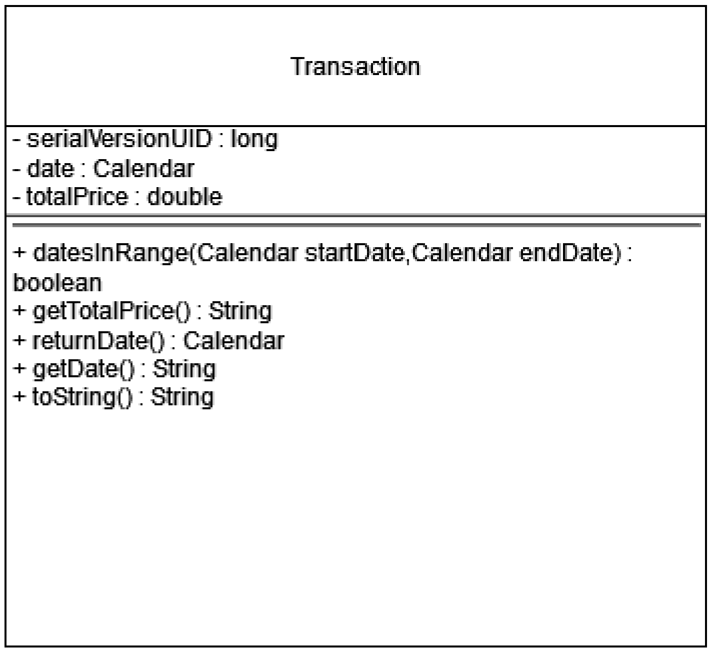
+ listOrders(): Iterator

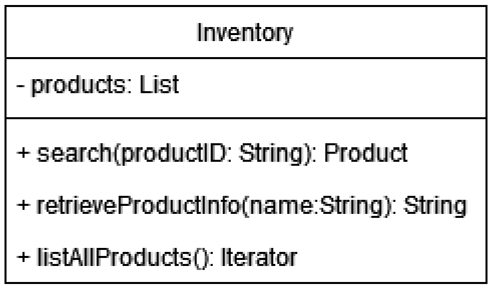

Cart
- totalPrice: double
- cartItems: List<CartItem>
- reorderList : List<Product>
+ Cart() :
+ calculateCartTotal(cartItems : List<CartItem>) : double
+ getTotalPrice() : double
+ addCartItem(cartItem : CartItem) : boolean
+ getReorderList() : List<Product>
+ addToReorderList(product : Product) : boolean
+ getCartItems() : List<Cartitem>
+ printCartTotal() : void

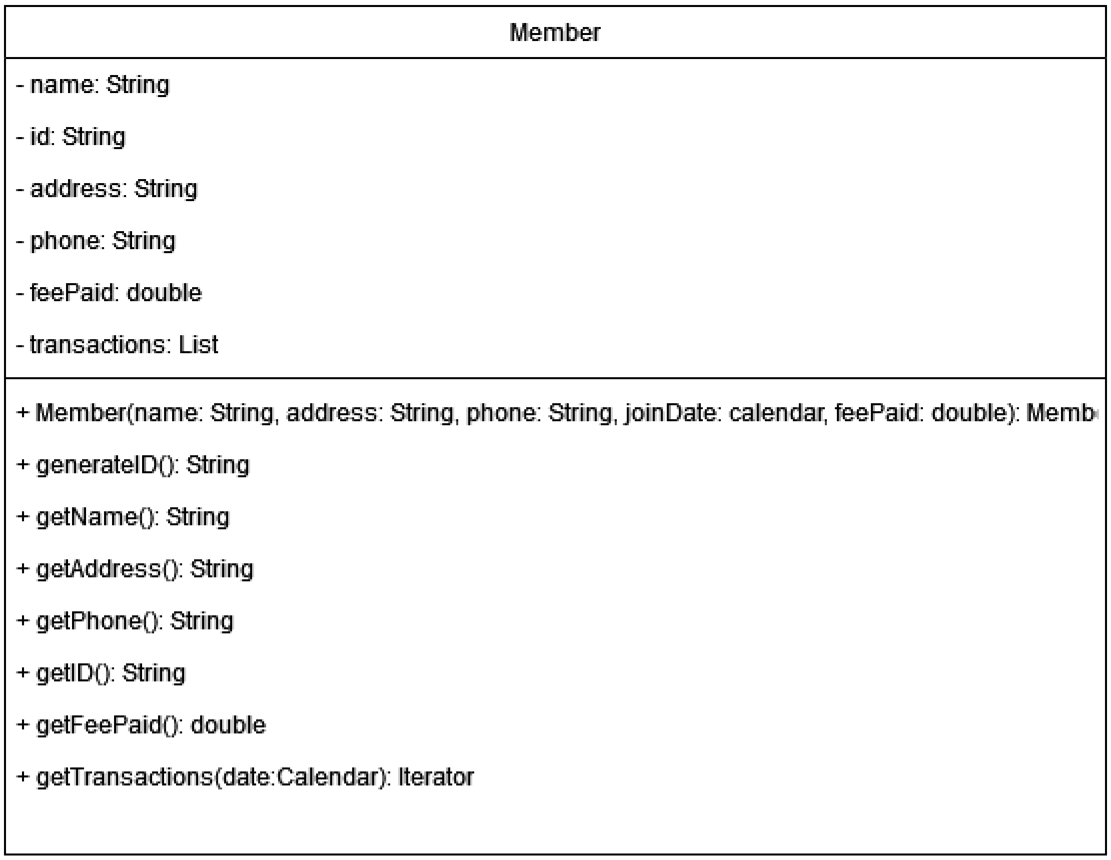

CartItem
- product : Product
- numberOfUnits : int
- totalItemPrice : double
- unitPrice : double
+ CartItem(product : Product, numberOfUnits : int) :
+ getTotalItemPrice() :double
+ getUnitPrice() : double
+ getProduct() : Product
+ getNumberOfUnits() : int
+ toString() : String

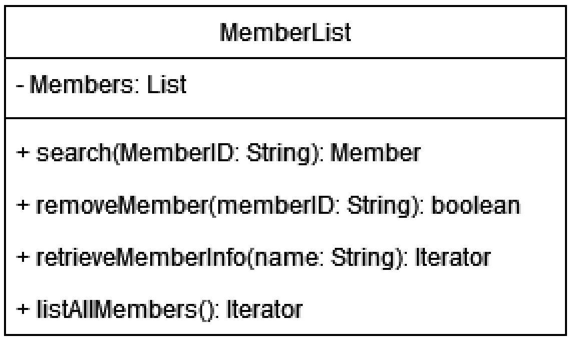

Order
- productName : String
- productId: String
- amountOrdered: int
+ getProductName(): String
+ getId(): String
+ getAmountOrdered(): Int
+ toString: String

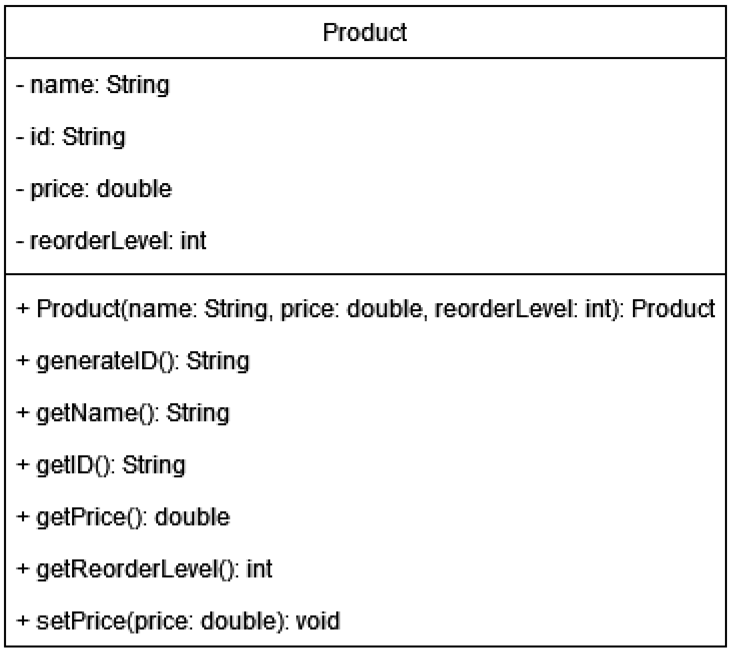

OrderList
- productName: String
- productID: String
- reorderLvl: int
+ getProductName() : String
+ getProductID(): String
+ getReorderLvl(): int


Transaction
- serialVersionUID : long
- date : Calendar
- totalPrice : double
+ datesInRange(Calendar startDate,Calendar endDate) : boolean
+ getTotalPrice() : String
+ returnDate() : Calendar
+ getDate() : String
+ toString() : String


Inventory
- products: List
+ search(productID: String): Product
+ retrieveProductInfo(name:String): String
+ listAllProducts(): Iterator


Member
- name: String
- id: String
- address: String
- phone: String
- feePaid: double
- transactions: List
+ Member(name: String, address: String, phone: String, joinDate: calendar, feePaid: double): Memb
+ generateID(): String
+ getName(): String
+ getAddress(): String
+ getPhone(): String
+ getID(): String
+ getFeePaid(): double
+ getTransactions(date:Calendar): Iterator


MemberList
- Members: List
+ search(MemberID: String): Member
+ removeMember(memberID: String): boolean
+ retrieveMemberInfo(name: String): Iterator
+ listAllMembers(): Iterator


Product
- name: String
- id: String
- price: double
- reorderLevel: int
+ Product(name: String, price: double, reorderLevel: int): Product
+ generateID(): String
+ getName(): String
+ getID(): String
+ getPrice(): double
+ getReorderLevel(): int
+ setPrice(price: double): void