Nalongsone Danddank
Rich Fritz
Ryan Kinsella
Gilbert Ponsness
Marc Wedo
ICS372-02
21-MAR-2021

Grocery Store Design Document

USE CASES:

Remove a Member – prepared by Richard Fritz

Action Performed by the Actor	Response from the system
1) The Employee initiates remove a Member	
	2) The System asks for the Members ID
3) The Employee enters the Member ID	
	4) The System checks the member list to see if it is a valid Member. If valid the member is removed, if not return an error

Add a Product – prepared by Richard Fritz

Action Performed by the Actor	Responses from the system
1) Employee initiates adding a new product	
	2) The System asks for the product name, id, stock on hand and current price,
3) Employee enters the product name, id, stock on hand, current price, and reorder level	
	4) The system checks the product name. If product does not exist new product is added with product name, product id, stock on hand, current price, and reorder level. An Order is generated with qty = 2 * reorder level If product exists, the system does not add product and returns an error that product <name> already exists

Check Out Items – prepared by Gilbert Ponsness

Actions performed by the actor	*Responses from the system*
1. The member comes to the check-out counter and removes grocery items from cart.	
2. The cashier issues a request to check out items, and asks member for identification.	
	3. The system asks for the member id.
4. The cashier inputs member id into the system.	
	5. The system checks the member list for the member id, and starts a record of transaction if member is found.
	6. The system asks for the product id and quantity of items.
7. The cashier grabs item(s) of one product type, then inputs the product id and quantity into the system.	
	8. The system checks the inventory for the product id, and starts the checkout process for that item if product is found.
	9. The system fetches the price for that product and computes the price times quantity. The system then adds this amount to the running transaction total.
	10. The system confirms product is scanned, and asks if there are more items to be purchased.
11. The cashier returns the item(s) to the member, then enters yes if there are more items on the counter.	
	12. If the cashier enters yes, the system returns to Step 6. Otherwise, the system displays every line item (name, price, amount, and total) along with grand total, and asks the cashier if payment is recieved.
13. The cashier asks the member for cash. If the member pays fully, the cashier enters yes.	

	14. If the cashier enters yes, the system passes the transaction record to be finalized according to the **helper case**. Otherwise, the transaction is dropped and the system exits.
15. The cashier closes the main terminal, or issues another check-out request.	
	Helper Case: Finalize Transaction
	Actions performed by the system
	A. The system reduces amount on hand for all products on transaction according to quantity purchased.
	B. If a product has an amount on hand equal to or below the reorder level, the system places an order for twice the reorder level.
	C. If an order is placed for a product, the system displays a message saying that the item will be reordered, how much was reordered, and what the order number is.
	D. If there is another product that meets the criteria, the system returns to Step B.

Process Shipment – prepared by Marc Wedo

Actions Performed by the Actor	Responses From the System
1. A delivery of products arrives from a supplier.	
2. The clerk issues a request to Process Shipment.	
	3. The system asks for the order number.
4. The clerk enters the order number into the system.	
	5. The system retrieves the order information.
	6. The system locates the product in the order and uses the current quantity in stock and the quantity listed on the order to calculate the new total quantity.
	7. The system updates the status of the order from outstanding to complete.
	8. The system displays the product ID, product name, and new total quantity.
	9. The system asks if the clerk wants to process another order.
10. The clerk answers in the affirmative or in the negative.	
	11. If the answer is in the affirmative, system goes to step 3. Otherwise, it exits.

Retrieve Member Info – prepared by Ryan Kinsella

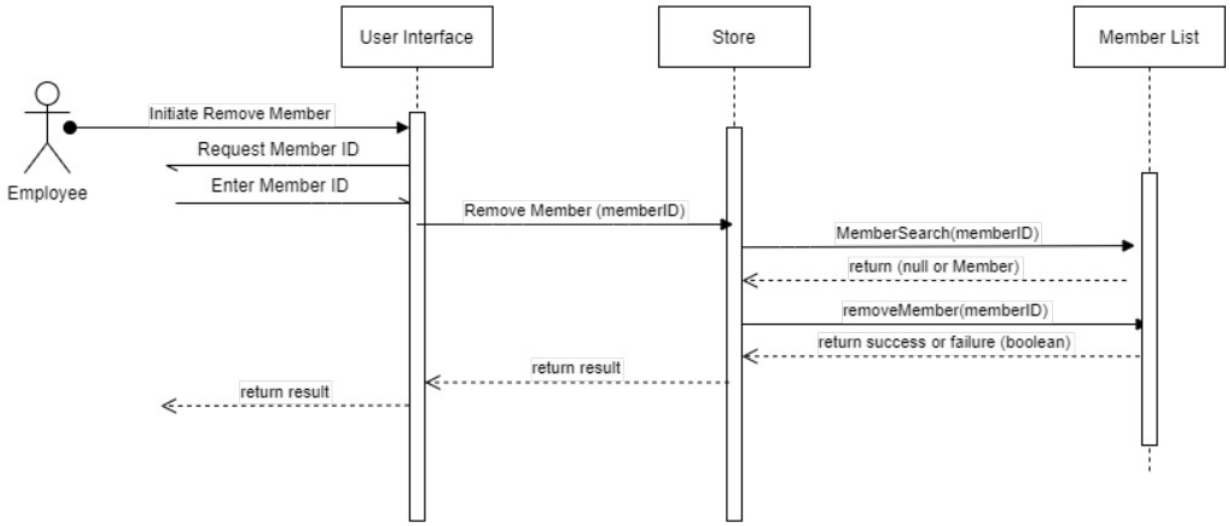
Action Performed by the Actor	Responses from the System
1. Employee initiates retrieve member info	
	2. System requests string input
3. Employee enters string	
	4. System checks for matching string
	5. If match found, system displays address, ID, and fee paid for all members whose name begins with the input string
	6. If no match found, system displays a Member Not Found message.

Print Transactions - prepared by Nalongsone Danddank

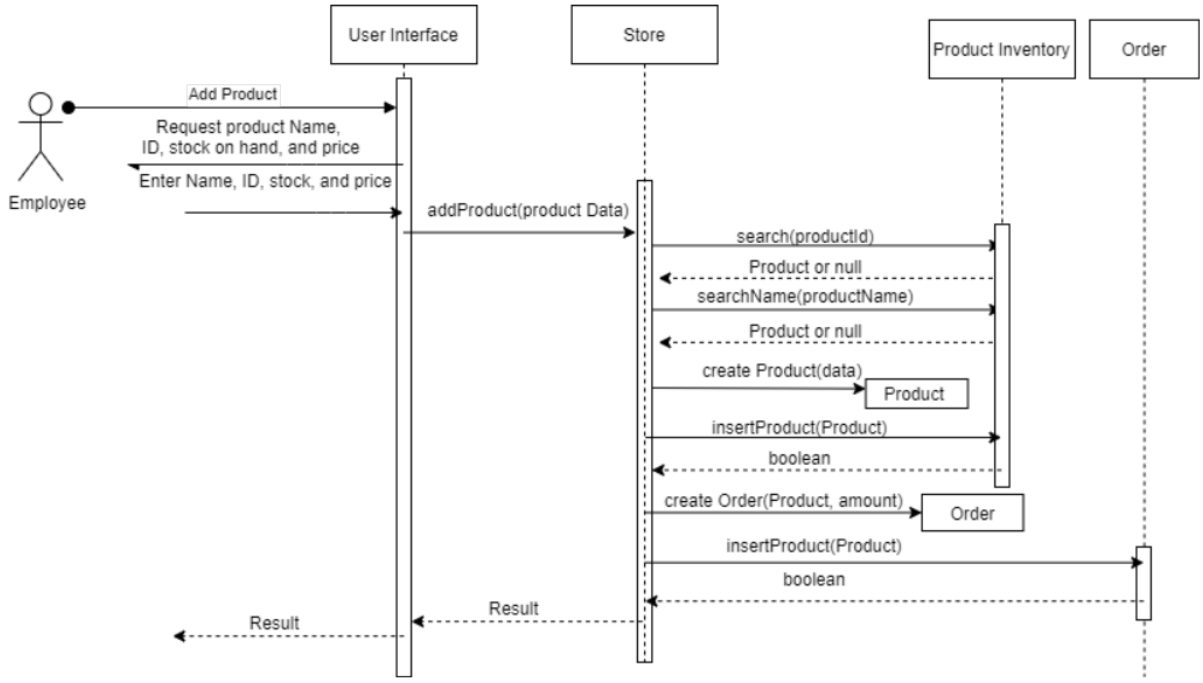
Action Performed by the Actor	Responses from the system
1) The clerk issues a request to get member transactions.	
	2) The system asks for the user ID of the member and the 2 dates (beginning and ending) for which the transactions are in of the user.
3) The clerk enters the identity of the user and the 2 date (beginning and ending).	
	4) If the ID is valid, then check date format for 2 date and also check beginning date must be before ending date. If all correct. the system outputs information about all transactions completed by the user on the given between 2 date. For each transaction, it shows the information of transaction like product name, price, total price and items.
5) Clerk prints out the transactions and hands them to the user.	

Sequence Diagrams:

Remove a member – prepared by Richard Fritz

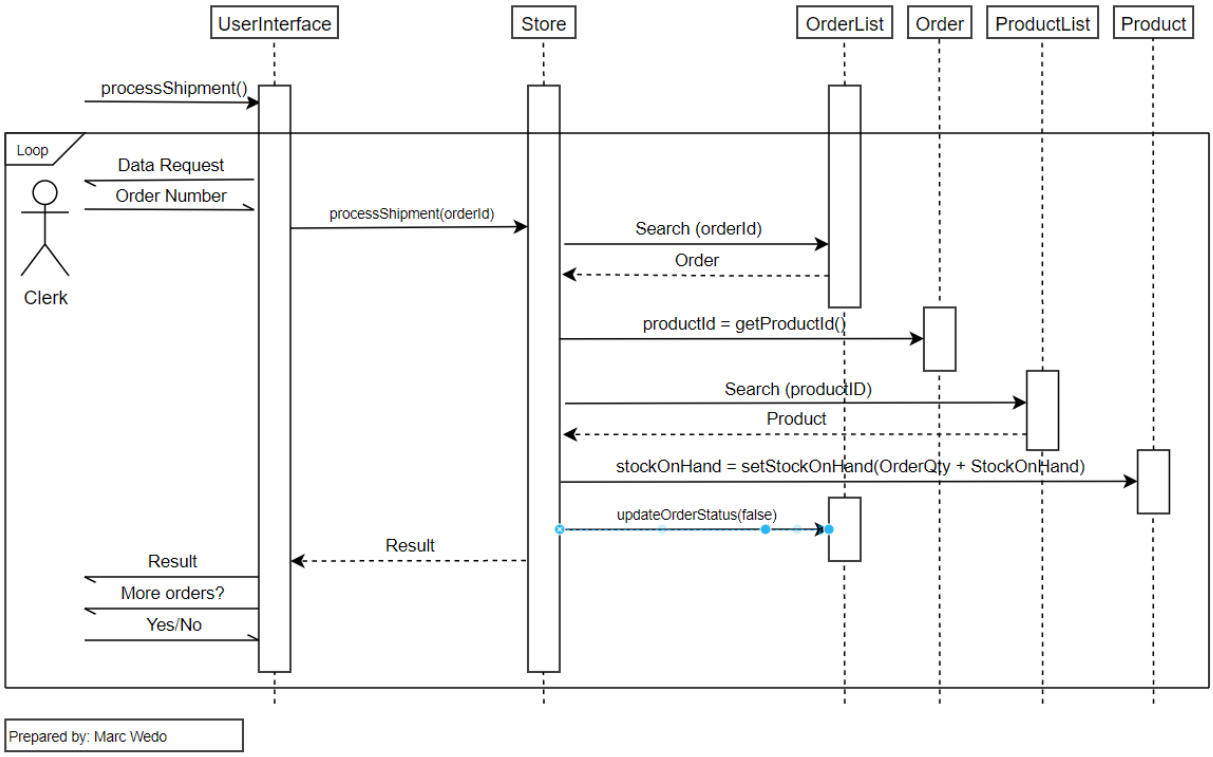


Add a product – prepared by Richard Fritz

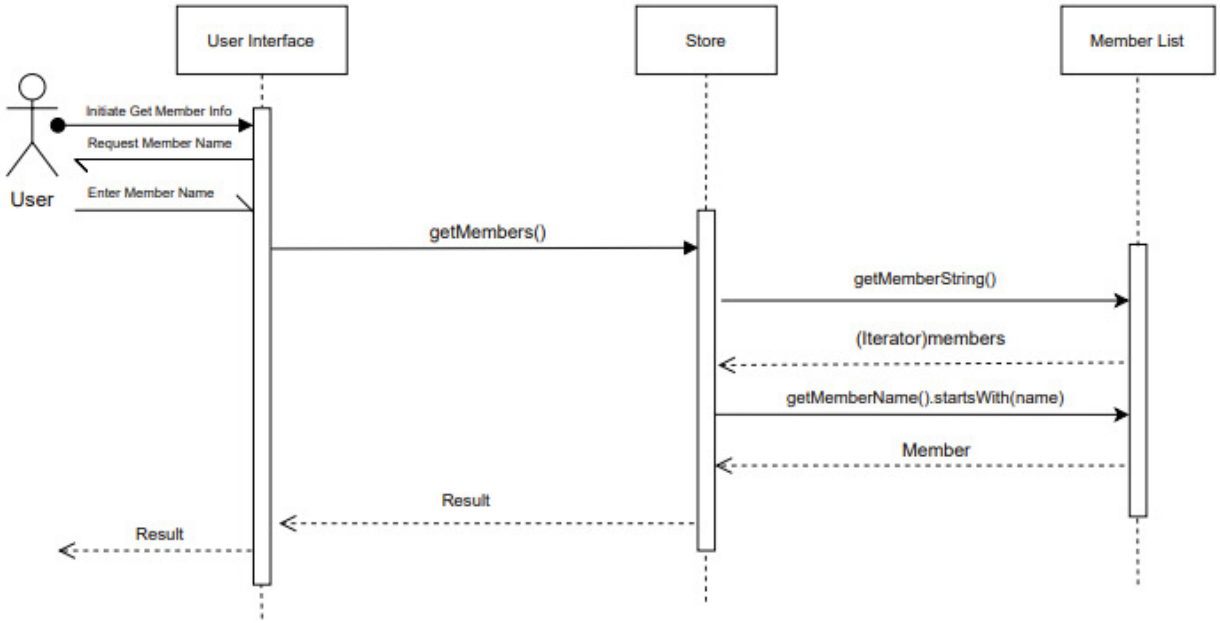


Check Out Items – prepared by Gilbert Ponsness

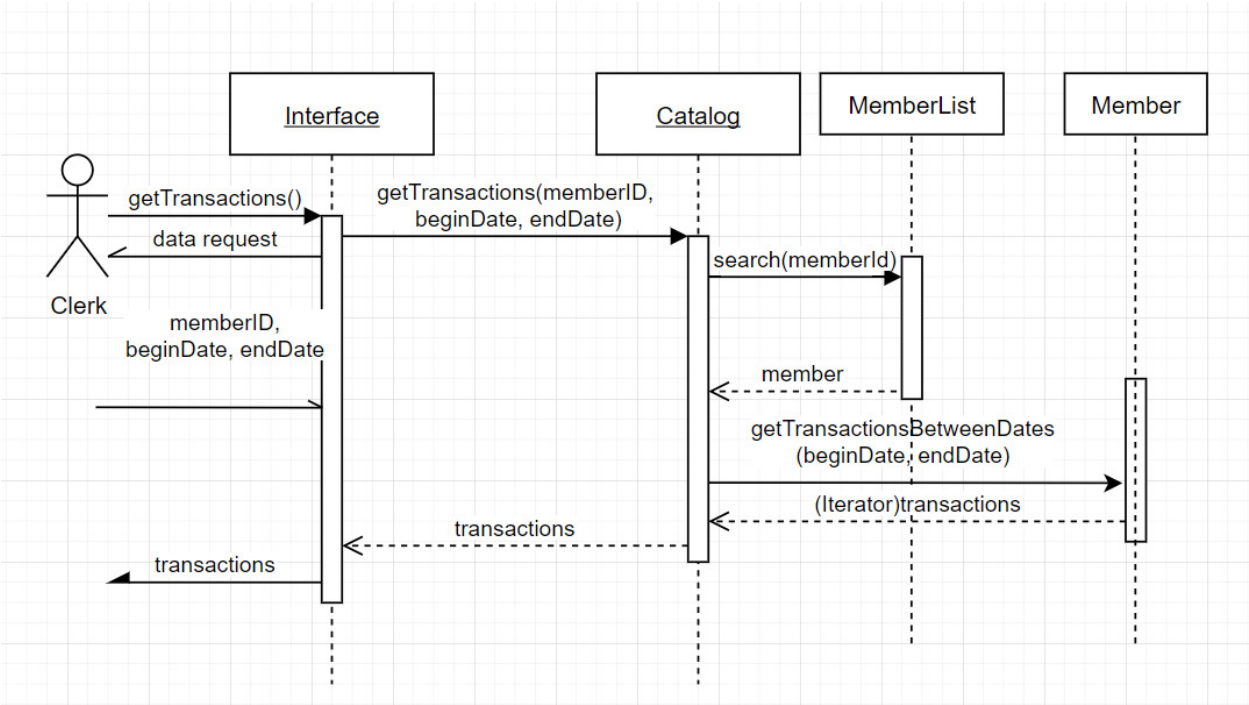
Process shipment – prepared by Marc Wedo



Retrieve Member Info – prepared by Ryan Kinsella

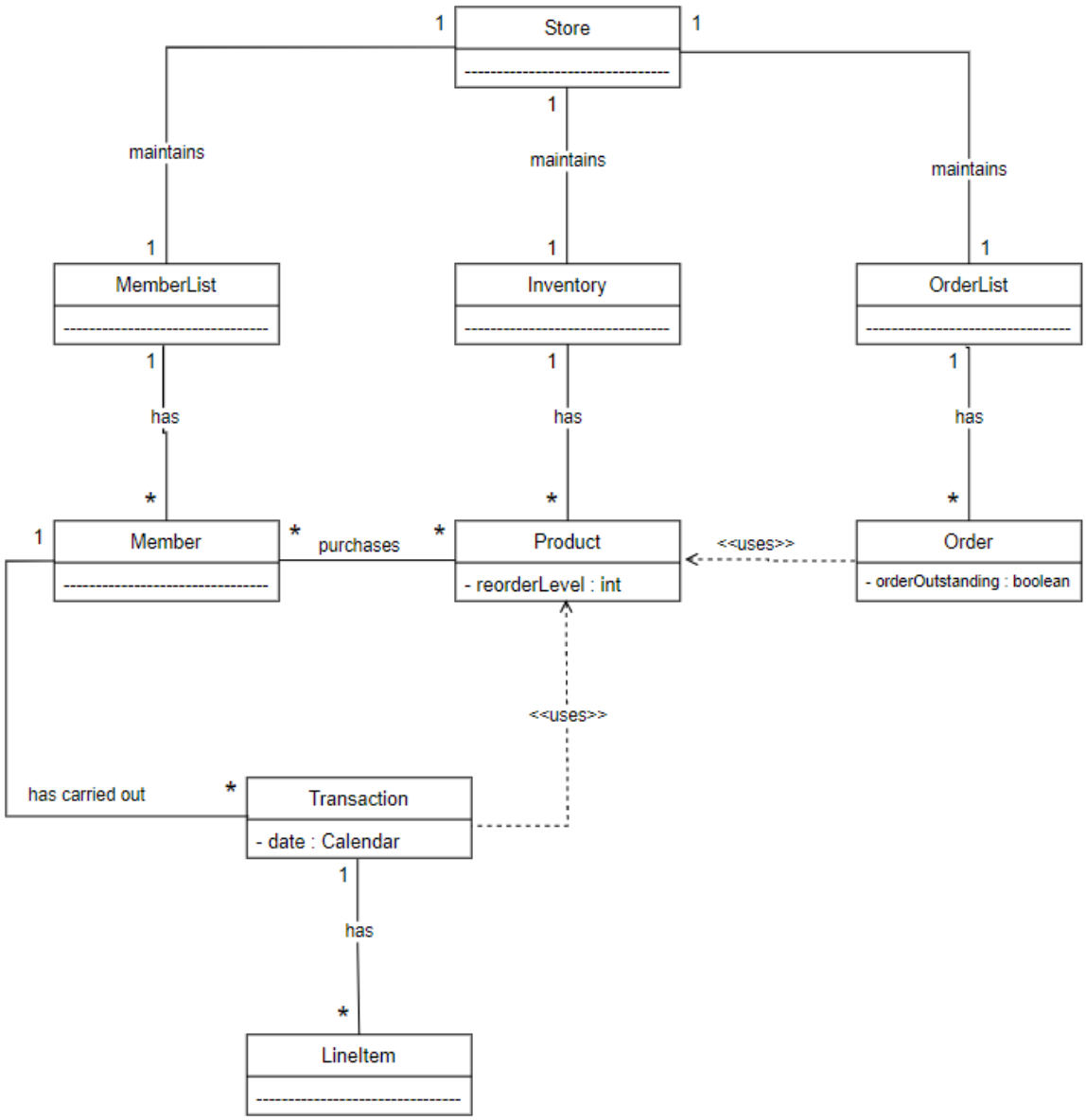


Print Transactions – prepared by Nalongsone Danddank


Interface
Catalog
MemberList
Member
Clerk
getTransactions()
data request
getTransactions(memberID,
beginDate, endDate)
search(memberId)
memberID,
beginDate, endDate
member
getTransactionsBetweenDates
(beginDate, endDate)
(Iterator)transactions
transactions
transactions

Class Diagrams:

Conceptual Class Diagram – prepared by Marc Wedo



Physical Class Diagrams

UserInterface – prepared by Richard Fritz

UserInterface
- userInterface: UserInterface - reader : BufferedReader - store: Store
-UserInterface():void +instance():UserInterface +getToken(prompt:String):String +getName(prompt:String):String +getNumber(prompt:String):int +getDate(prompt:String):Calendar +getCommand():int +help():void +enrollMember():void +addProduct():void +changePrice():void +checkoutItems():void +removeMember():void +getProductInfo():void +getMemberInfo():void +getTransactions():void +getOutstandingOrders():void +getMembers():void +getProduct():void +processShipment():void +save():void +retrieve():void +process():void

Store – Prepared by Marc Wedo

Store
- inventory : Inventory - members : MemberList - orders: OrderList
+ addProduct(productName : String, productId : String, productStockOnHand : int, productPrice : String, productReorderLevel : int) : Product + changePrice(productId : String, productPrice: String) : Product + beginTransaction() : Transaction + checkOutItem(memberId: String, productId: String, purchaseAmount: int) : Product + displayPurchases(currentTransaction : Transaction) : String + finalizeTransaction(memberId : String, currentTransaction : Transaction) : Member + adjustInventory(lineItems : Iterator) : String + enrollMember(memberName: String, memberAddress: String, memberPhone : String) : Member + removeMember(memberId : String) : String + searchMembership(memberId : String) : Member + processShipment(orderId : String) : Product + getTransactions(memberId : String, date : String) : Iterator + getOutstandingOrders() : Iterator + getMembers() : Iterator + getProduct() : Iterator + save() : boolean + retreive() : Store

Member – Prepared by Richard Fritz

Member
- name : String - address : String - phone : String - dateJoined : Calendar - fee : String - id : String - transactions: List - idCounter: int
+Member(name:String,address:String,phone:String,fee:String):Member +getTransactionsOnDate(date:Calander):Iterator +hashcode():int +equals(object:Object):boolean +save(output:ObjectOutputStream):void +retrieve(input:ObjectOutputStream):void

MemberList – Prepared by Richard Fritz

MemberList
- members: List
+search(memberId:String):Member +removeMember(memberId:String):boolean +insertMember(member:Member):boolean +iterator():Iterator +toString():String

Order – Prepared by Marc Wedo

Order
- id : String - productOrdered : Product - dateOrdered : Calendar - qtyOrdered : int - orderOutstanding : boolean
+ Order(productOrdered : Product, qtyOrdered : int) : Order + isOutstanding() : boolean + updateStatus(status : boolean) : void + toString() : String + hashCode() : int + equals(object : Object) : boolean + save(output : ObjectOutputStream) : void + retrieve(input : ObjectInputStream) : void

OrderList – Prepared by Marc Wedo

OrderList
- orders : List
+ search(orderId : String) : Order + insertOrder(order : Order) : boolean + getOutstandingOrders() : Iterator + iterator() : Iterator + toString() : String

Product – Prepared by Richard Fritz

Product
- name : String - id : String - price : String - reorderLevel : int - stockOnHand : int
+ Product(name:String,id:String,price:String,reorderLevel:int, stockOnHand:int) : Product + getName() : String + getId() : String + getPrice() : String + getReorderLevel() : int + getStockOnHand() : int + setName(name : String) : void + setId(String : id) : void + setPrice(String : price) : void + setReorderLevel(int : reorderLevel) : void + setStockOnHand(int : stockOnHand) : void + toString() : String + hashCode() : int + equals(object : Object) : boolean

Inventory – Prepared by Richard Fritz

Inventory
- Product : List
+ search(productId : String) : Product + searchName(productId : String) : Product + insertProduct(product : Product) : boolean +iterator():Iterator +toString():String

Transaction – Prepared by Gilbert Ponsness

Transaction
- date: Calendar - purchaseTotal: double - groceryItems: List
+ Transaction() + addItem(itemForPurchase: Product, purchaseAmount int) : String + getPurchaseTotal() : String + betweenDates(beginDate: Calendar, endDate: Calendar) : boolean + getDate() : String + getLineItems() : Iterator + buildReceipt() : String +toString() : String

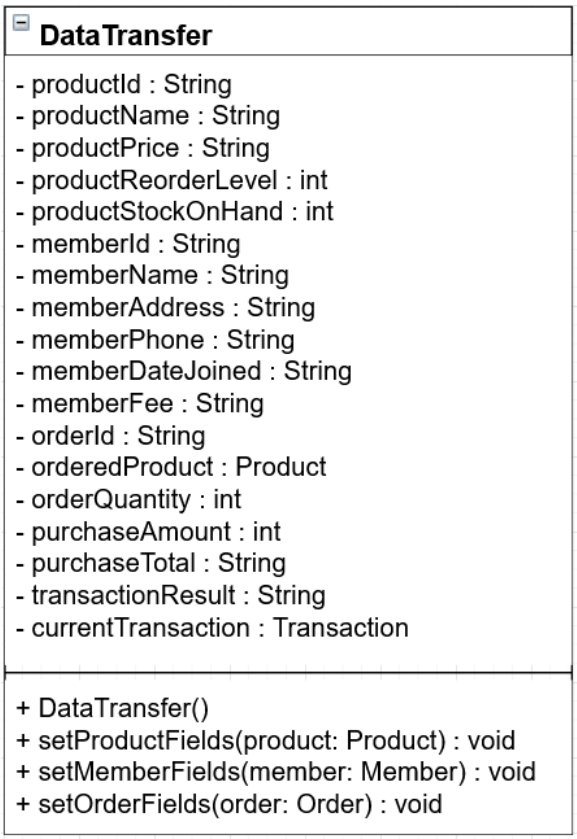
LineItem – Prepared by Gilbert Ponsness

LineItem
- product : Product - purchaseAmount : int - purchasePrice : double
+ LineItem(product Product, purchaseAmount: int) + toString() : String

Print transactions – Prepared by Nalongsone Danddank

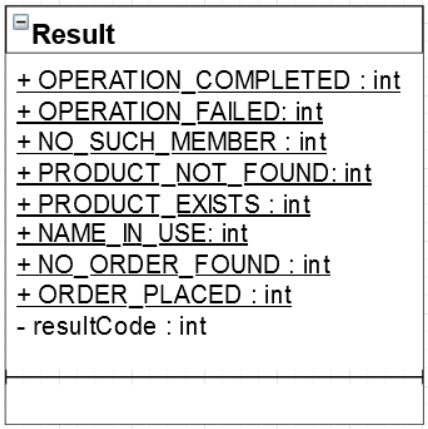
Transactions
-date: Calendar +purchaseTotal: double +groceryItems: List<LineItem>
+onDate(date:Calendar): boolean +betweenDates(Calendar beginDate, Calendar endDate): boolean +buildReceipt(): String +addItem(Prodect itemForPurchase, int purchaseAmount):String

DataTransfer – Prepared by Gilbert Ponsness



Request – Prepared by Gilbert Ponsness

Result – Prepared by Gilbert Ponsness



SafeIterator – Prepared by Ryan Kinsella

SafeIterator
- iterator : Iterator - type : Type - result : Result
+ hasNext() : boolean + next() : Result

FilteredOrderIterator – Prepared by Ryan Kinsella

FilteredOrderIterator
- item : Order - predicate : Predicate - iterator : Iterator
+ hasNext() : boolean + next() : Order - getNextItem() : void

FilteredTransactionIterator – Prepared by Ryan Kinsella

FilteredTransactionIterator
- item : Transaction - predicate : Predicate - iterator : Iterator
+ hasNext() : boolean + next() : Transaction - getNextItem() : void

AutomatedTester – Prepared by Ryan Kinsella

AutomatedTester
- memberNames : String[5] - addresses : String[5] - phones : String[5] - fee : String[5] - members : Member[5] - productName : String[20] - productid : String[20] - stockOnHand : int[20] - currentPrice : String[20] - reorderLevel : int[20] - products : Product[20]
+ testEnrollMember() : void + testRemoveMember() : void + testAddProduct() : void + testCheckOutItems() : void + testProcessShipment() : void + testChangePrice() : void + testAll() : void