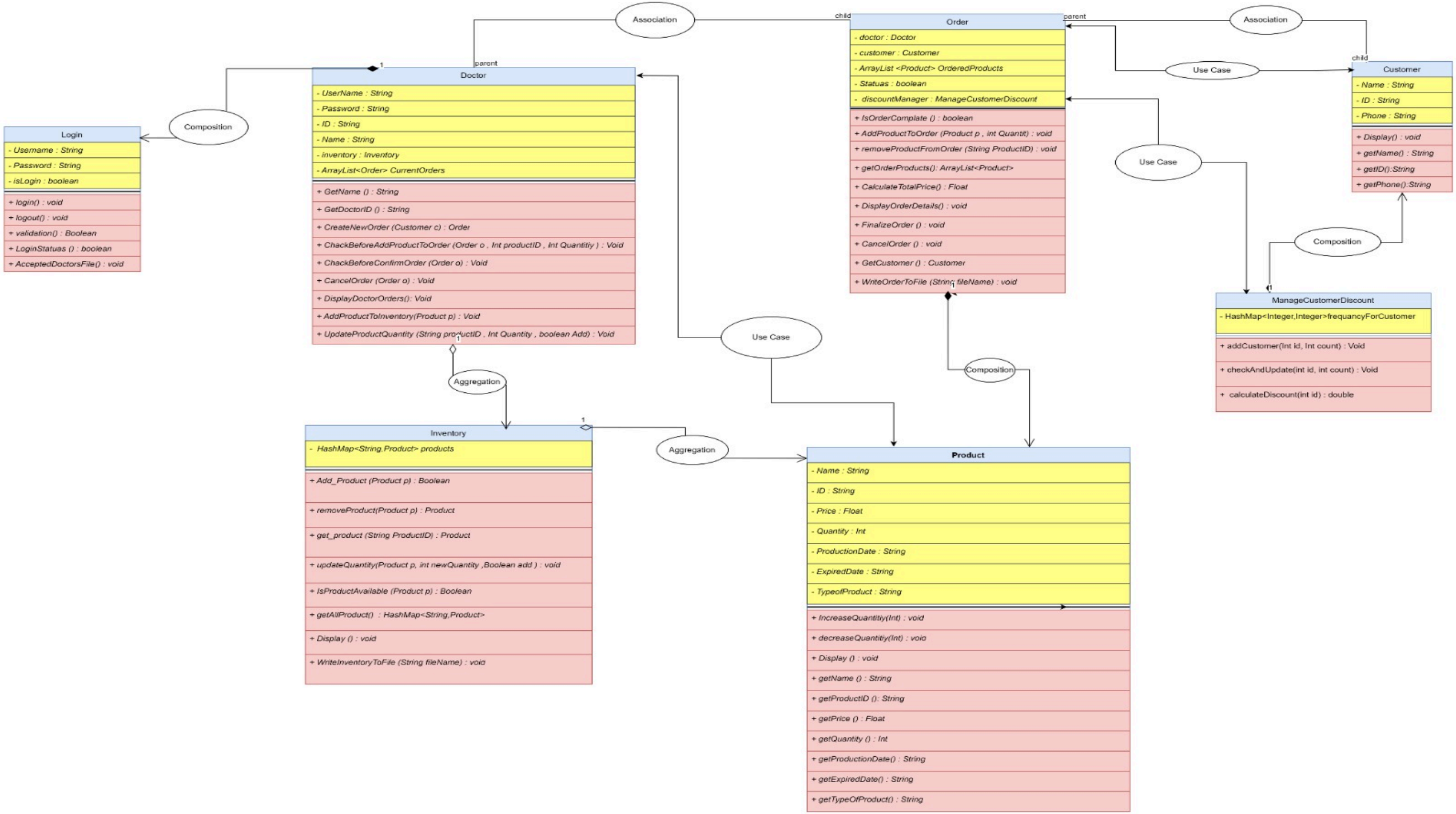
Project : Pharmacy System

Members:
1.Abdelrahman Hamada Al Sayed Mohamed
2.Kareem Mohamed Badr Eldin
3.Mohamed Gamal Salah Ismail
4.Ahmed Mohamed Ahmed Abdelaziz


Association
child
parent
Order
- doctor : Doctor
- customer : Customer
- ArrayList <Product> OrderedProducts
- Statuas : boolean
- discountManager : ManageCustomerDiscount
+ IsOrderComplate () : boolean
+ AddProductToOrder (Product p , int Quantit) : void
+ removeProductFromOrder (String ProductID) : void
+ getOrderProducts(): ArrayList<Product>
+ CalculateTotalPrice() : Float
+ DisplayOrderDetails() : void
+ FinalizeOrder () : void
+ CancelOrder () : void
+ GetCustomer () : Customer
+ WriteOrderToFile (String fileName) : void
Association
Use Case
child
Customer
- Name : String
- ID : String
- Phone : String
+ Display() : void
+ getName() : String
+ getID():String
+ getPhone():String
Use Case
Composition
1
parent
Doctor
- UserName : String
- Password : String
- ID : String
- Name : String
- inventory : Inventory
- ArrayList<Order> CurrentOrders
+ GetName () : String
+ GetDoctorID () : String
+ CreateNewOrder (Customer c) : Order
+ ChackBeforeAddProductToOrder (Order o , Int productID , Int Quantitiy) : Void
+ ChackBeforeConfirmOrder (Order o) : Void
+ CancelOrder (Order o) : Void
+ DisplayDoctorOrders(): Void
+ AddProductToInventory(Product p) : Void
+ UpdateProductQuantity (String productID , Int Quantity , boolean Add) : Void
Composition
Login
- Username : String
- Password : String
- isLogin : boolean
+ login() : void
+ logout() : void
+ validation() : Boolean
+ LoginStatuas () : boolean
+ AcceptedDoctorsFile() : void
Use Case
Composition
1
ManageCustomerDiscount
- HashMap<Integer,Integer>frequancyForCustomer
+ addCustomer(Int id, Int count) : Void
+ checkAndUpdate(int id, int count) : Void
+ calculateDiscount(int id) : double
1
Aggregation
1
Inventory
- HashMap<String,Product> products
+ Add_Product (Product p) : Boolean
+ removeProduct(Product p) : Product
+ get_product (String ProductID) : Product
+ updateQuantity(Product p, int newQuantity ,Boolean add) : void
+ IsProductAvailable (Product p) : Boolean
+ getAllProduct() : HashMap<String,Product>
+ Display () : void
+ WriteInventoryToFile (String fileName) : void
Aggregation
Product
- Name : String
- ID : String
- Price : Float
- Quantity : Int
- ProductionDate : String
- ExpiredDate : String
- TypeofProduct : String
+ IncreaseQuantitiy(Int) : void
+ decreaseQuantitiy(Int) : void
+ Display () : void
+ getName () : String
+ getProductID (): String
+ getPrice () : Float
+ getQuantity () : Int
+ getProductionDate() : String
+ getExpiredDate() : String
+ getTypeOfProduct() : String

Summary :

this system allows the purchase of product (both cosmetic and medicines) while keeping track of the inventory , user's data , receipts. The system also checks whether the user has a discount or not to calculate the receipts.

Short description of classes :

product class :

the product class represents a product in the pharmacy inventory system.
it includes attributes such as the product's name , ID , price , and available quantity.

the class provides methods to :

Getters: Retrieve the product's name ,Id , price and quantity.
Quantity Management : increase or decrease the product quantity in stock.
Display Information: print the product details in a readable format.

Attributes :

name: the name of the product.
productid: A unique identifier for the product.
price: the cost of the product.
quantity: the number of units available in inventory

Customer Class:

The Customer class represents a customer in the pharmacy management system. It encapsulates essential customer information and provides methods to access this data.

Attributes:
name: The name of the customer.
customerId: A unique identifier for the customer.
phone: The contact phone number of the customer.

Methods:
Getters:Returns the customer's name , ID , discount percentage and phone number.

display(): Prints the customer's details in a formatted manner.

Login Class :

The Login class provides security for the pharmacy system and to enter the system , you must enter the username and password.

Attributes:

username: username for the doctor
password: the pass for the doctor to login the system.
isLigin: check if the doctor is login or logout

Methods:

login(): to login the system.
logout(): to logout the system.
validation(): to check the username and password (true or false).
loginStatus(): return isLogin.
acceptedDoctorFile(): contains the validate doctors.

Inventory Class :

The Inventory class manages the collection of Product objects in the pharmacy management system.
It provides functionality to add, remove, and update product quantities while ensuring efficient inventory management.

Attributes:
products: A HashMap that stores products, using the product ID as the key for quick access.
Methods:
addProduct(Product product): Adds a new product to the inventory.
removeproduct(Product product): Removes a product from the inventory.
getAllProducts(): Returns a HashMap of all products in the inventory.
updateQuantity(Product product, int quantity, boolean add): Increases or decreases the quantity of a specified product based on the provided flag.
getProduct(int productId): Retrieves a product by its ID.
isProductAvailable(int productId, int quantity): Checks if the specified quantity of a product is available in the inventory.
writeInventoryToFile(String fileName): Writes the current inventory to a specified file, appending product details.
display(): Displays the details of all products in the inventory.

Doctor Class :

The Doctor class represents a doctor in the pharmacy management system, providing functionalities to manage patient orders, inventory, and products.

Attributes:

name: The name of the doctor.

doctorId: A unique identifier for the doctor.

userName: The username for the doctor's account.

password: The password for the doctor's account

.

currentOrders: A list of orders currently managed by the doctor.

inventory: an object from class inventory to allow the doctor manage it.

Methods:

createNewOrder(Customer customer): Creates and adds a new order for a specified customer.

addProductToOrder(Order order, int productId, int quantity): Adds a specified product to a given order after checking inventory availability.

//finalizeOrder(Order order): Finalizes an order if it contains products.

cancelOrder(Order order): Cancels a specified order and removes it from current orders.

displayOrders(): Displays details of all current orders.

addProductToInventory(Product product): Adds a new product to the inventory.

updateProductQuantity(int productId, int quantity, boolean add): Updates the quantity of an existing product in the inventory (either increases or decreases).

Order Class :

The Order class represents a customer's order in the pharmacy management system.
It encapsulates details about the doctor who created the order, the customer placing the order, and the products included in the order.

Attributes:

doctor: The doctor associated with the order.

customer: The customer placing the order
.
orderedProducts: A list of products that have been ordered.

status: A boolean indicating whether the order is completed (true) or incomplete (false).

Methods:

addProductToOrder(Product product, int quantity): Adds a specified product and its quantity to the order.

removeProductFromOrder(int productId): Removes a product from the order based on its product ID.

getOrderedProducts(): Returns a list of all products in the order.

calculateTotalPrice(): Computes the total price of the order, considering the customer's discount.

//finalizeOrder(): Marks the order as completed.

cancelOrder(): Clears all products from the order and sets its status to incomplete.
isOrderComplete(): Checks whether the order is complete.

displayOrderDetails(): Displays the details of the order, including the doctor, customer, status, ordered products, and total price.

ManageCustomerDiscount Class:

the ManageCustomerDiscount is responsible for managing customer discount based on their purchase frequency. It tacks hoe many purchases each customer has made and calculates the applicable discount for future order.

Attributes:

frequencyForCustomer: A HashMap that stores details of customers, using the customer ID as the key for quick access.

Methods:

addCustome():add customer in HashMap if not exist.
checkAndUpdate():to cheak if the customer exists in the system or not, update their purchase count.
calculateDiscount(): determine the discount rate based on the customer's purchase freqancy.