**Class Design**

|  |  |  |  |
| --- | --- | --- | --- |
| Order | Product | Customer | File |
| - orderID  - customerID  - productID  - quantity  - status | - productID: int  - name: string  - description: string  - price: double  - stockQuantity :int | - customerID: int  - name: string  - email: string  - address: string  - phone: string | - filename  - content |
| + Order(orderID: int,  customerID: int,  productID: int,  quantity: int,  status: string)  + UpdateStatus(newStatus:string)  + CancelOrder()  + ReturnOrder()  + TrackOrderStatus(): string | + Product(productID: int,  name: string,  description: string  price: double,  stockQuantity: int)  + UpdateStock(newQuantity: int)  + AddStock(quantityToAdd: int)  + RemoveStock(quantityToRemove: int | + Customer(customerID: int,  name: string,  email: string,  address: string,  phone: string)  + UpdateContactInfo(newInfo: string) |  |

**Class Diagram**

File

- filename

- content

Order

- orderID

- customerID

- productID

- quantity

- status

+ Order(orderID: int,

customerID: int,

productID: int,

quantity: int,

status: string)

+ UpdateStatus(newStatus:string)

+ CancelOrder()

+ ReturnOrder()

+ TrackOrderStatus(): string

Product

- productID: int

- name: string

- description: string

- price: double

- stockQuantity :int

- productID: int

- name: string

- description: string

- price: double

- stockQuantity :int

1

1\*

Customer

- customerID: int

- name: string

- email: string

- address: string

- phone: string

+ Customer(customerID: int,

name: string,

email: string,

address: string,

phone: string)

+ UpdateContactInfo(newInfo: string)

1\*

1

1\*

1

1\*

1

**Pseudo Code**

**Define Order class.**

class Order:

- orderID: int

- customerID: int

- productID: int

- quantity: int

- status: string

+ Order(orderID: int, customerID: int, productID: int, quantity: int, status: string)

+ UpdateStatus(newStatus: string): void

+ CancelOrder(): void

+ ReturnOrder(): void

+ TrackOrderStatus(): string

**Define Product class**

class Product:

- productID: int

- name: string

- description: string

- price: double

- stockQuantity: int

+ Product(productID: int, name: string, description: string, price: double, stockQuantity: int)

+ UpdateStock(newQuantity: int): void

+ AddStock(quantityToAdd: int): void

+ RemoveStock(quantityToRemove: int): void

**Define Customer class**

class Customer:

- customerID: int

- name: string

- email: string

- address: string

- phone: string

+ Customer(customerID: int, name: string, email: string, address: string, phone: string)

+ UpdateContactInfo(newInfo: string): void

// Data structures for storing orders, products, and customers

orderList: List of Orders

productMap: Map of ProductID to Product

customerMap: Map of CustomerID to Customer

**Function to add an order to the order list**

function AddOrder(order: Order): void

add order to orderList

**Function to remove an order from the order list**

function RemoveOrder(orderID: int): void

for each order in orderList:

if order.orderID == orderID:

remove order from orderList

break

**Function to search for a product by ID**

function SearchProductByID(productID: int): Product

return productMap[productID]

**Function to search for a customer by ID**

function SearchCustomerByID(customerID: int): Customer

return customerMap[customerID]

**Function to read sample data from a file**

function ReadSampleDataFile(fileName: string): void

open file with fileName

for each line in file:

parse line to extract orderID, customerID, productID, quantity, status,

productName, productDescription, productPrice, customerName, customerEmail, customerAddress, and customerPhone

create Order object with extracted data

create Product object with extracted data

create Customer object with extracted data

add Order object to orderList

add Product object to productMap

add Customer object to customerMap

close file

**Main function**

function main(): void

ReadSampleDataFile("sample\_data.txt")

print "Order list size:", size of orderList

RemoveOrder(1)

print "Order list size after removal:", size of orderList

**Class Justification**

**Order List**:

Since orders are added and removed frequently, a dynamic data structure like a list provides efficient insertion and deletion operations, ensuring that the program can handle orders effectively.

**Product Map** & **Customer Map**:

These data structures are chosen because quick access to product and customer details is essential for various operations such as order processing and status tracking. Using a map data structure ensures that searching for products and customers by their IDs can be done efficiently, meeting the requirement for fast retrieval.

The chosen data structures align with the program's requirements by providing efficient storage and retrieval mechanisms for managing orders, products, and customers.