Assignment 1

Mansour Abdulla Alfalahi

202205486

Program Fund

Prof Areej Abdulfattah

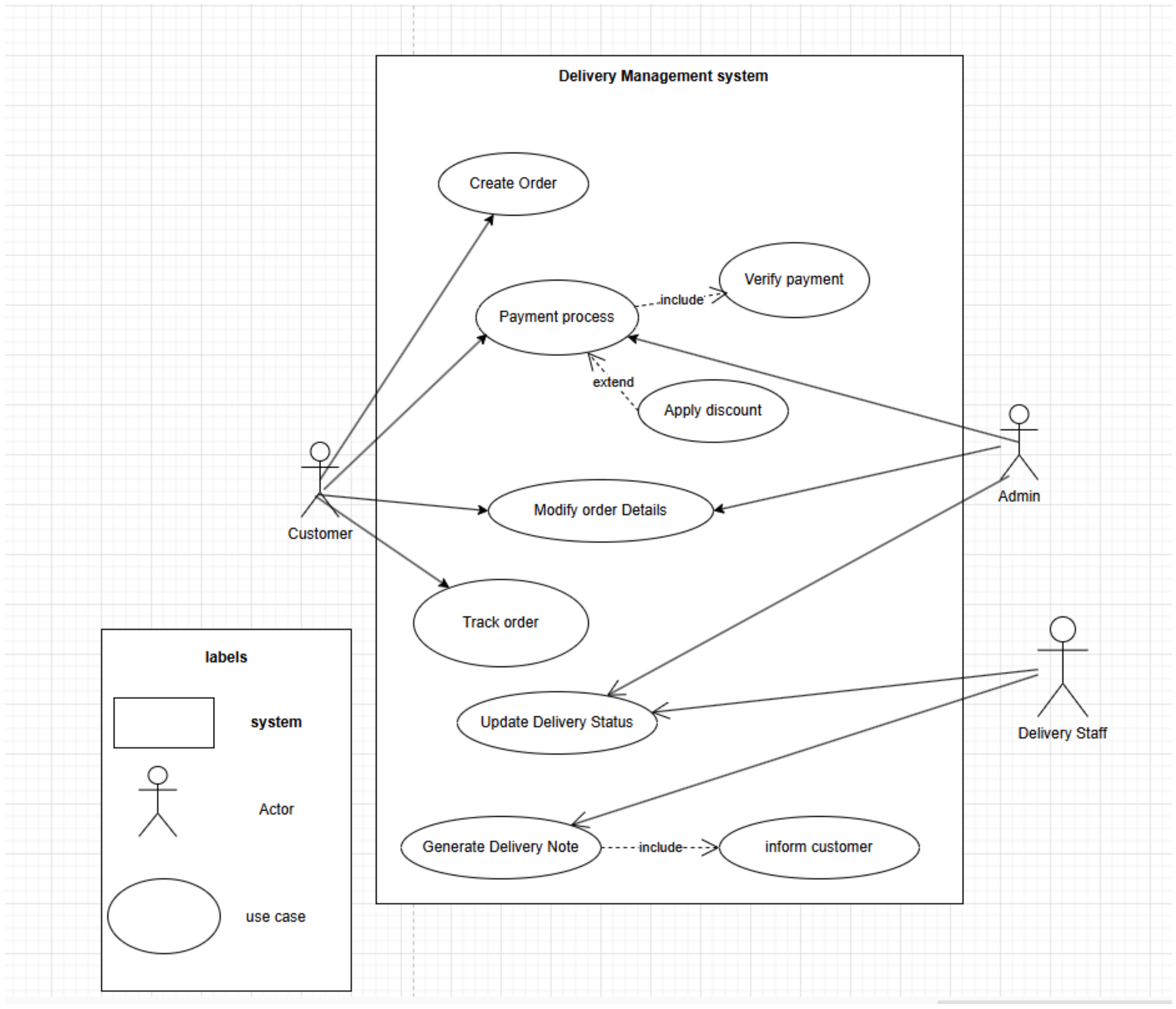
Date 28/2/2025

Use case diagram for the delivery management system.

1) Create Order
2) Modify Order Details
3) Process Payment
4) Verify Payment
5) Apply Discount (Optional)
6) Track Order
7) Update Delivery Status
8) Generate Delivery Note
9) Inform Customer
10) Cancel Order
11) Assign Order to Delivery Staff

The actors are:

1) Customer
2) Admin
3) Delivery



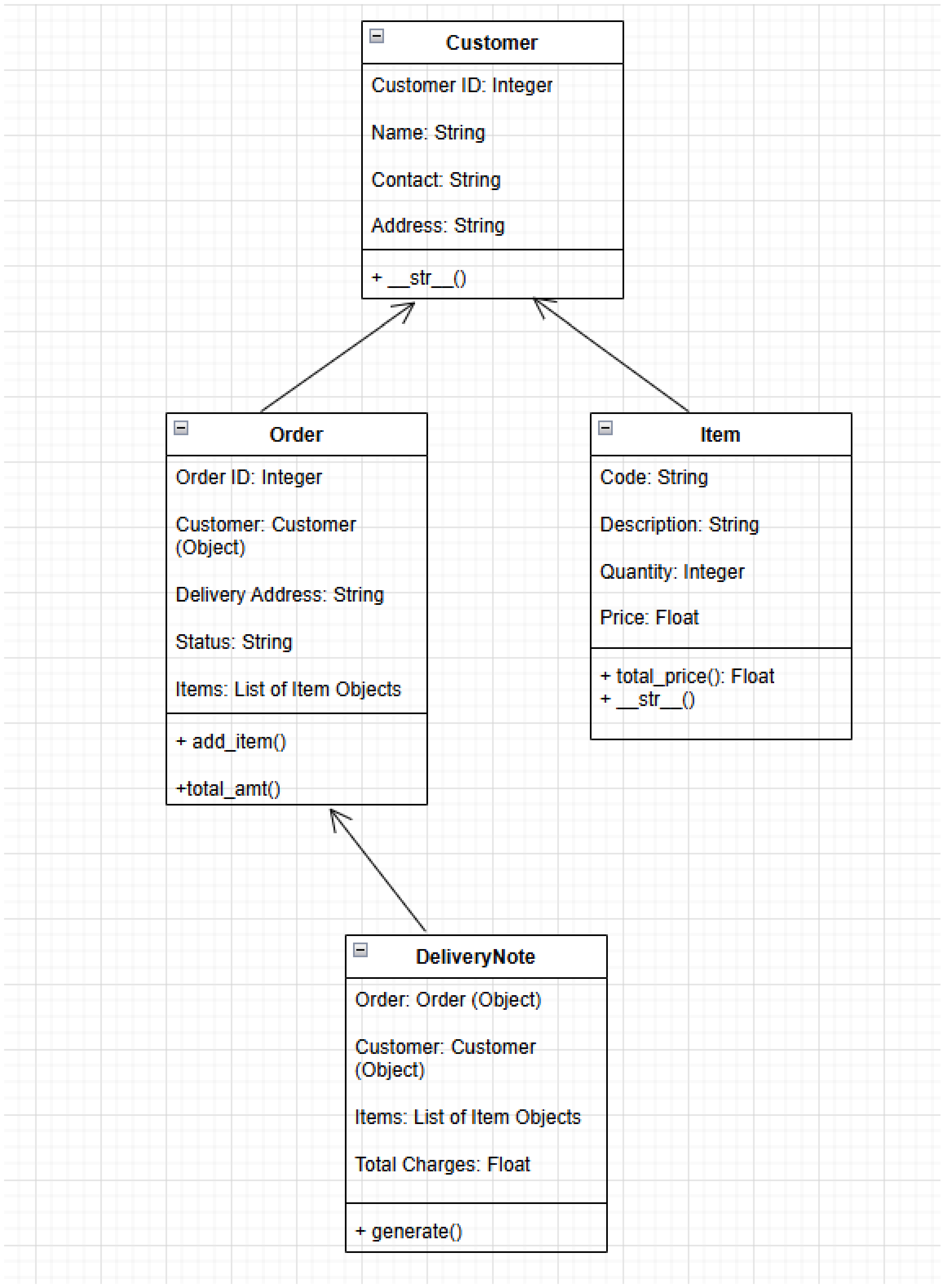
Use case descrition tables:

Use Case	Create Delivery Order
Trigger	The customer places a new delivery request.
Actors	Customer, Admin
Preconditions	The customer must be registered in the system. The system must be operational.
Main Scenario	1. The client inputs delivery information. 2. An order number is generated by the system. 3. Order information is stored in the database by the system. 4. The order confirmation is sent to the customer by the system.
Exceptions	The entered details are incomplete. The system prompts the customer to provide missing details.

Use Case	Generate Delivery Note
Trigger	A new delivery order is processed.
Actors	Admin, System
Preconditions	The order must be confirmed and assigned.
Main Scenario	1. Order details are retrieved by the system. 2. A structured delivery note is produced by the system. 3. The item summary, delivery information, and recipient data are all included in the delivery note. 4. The delivery note is stored and shown by the system.
Exceptions	Order details are missing. The system prompts an admin to verify missing data.

Use Case	Track Delivery Status
Trigger	The customer requests to track their order.
Actors	Customer, System
Preconditions	The customer must have an active order.
Main Scenario	1. The customer enters the order number. 2. The system checks the latest order status. 3. The system displays the real-time order status.
Exceptions	Order not found. The system notifies the customer that the order does not exist.

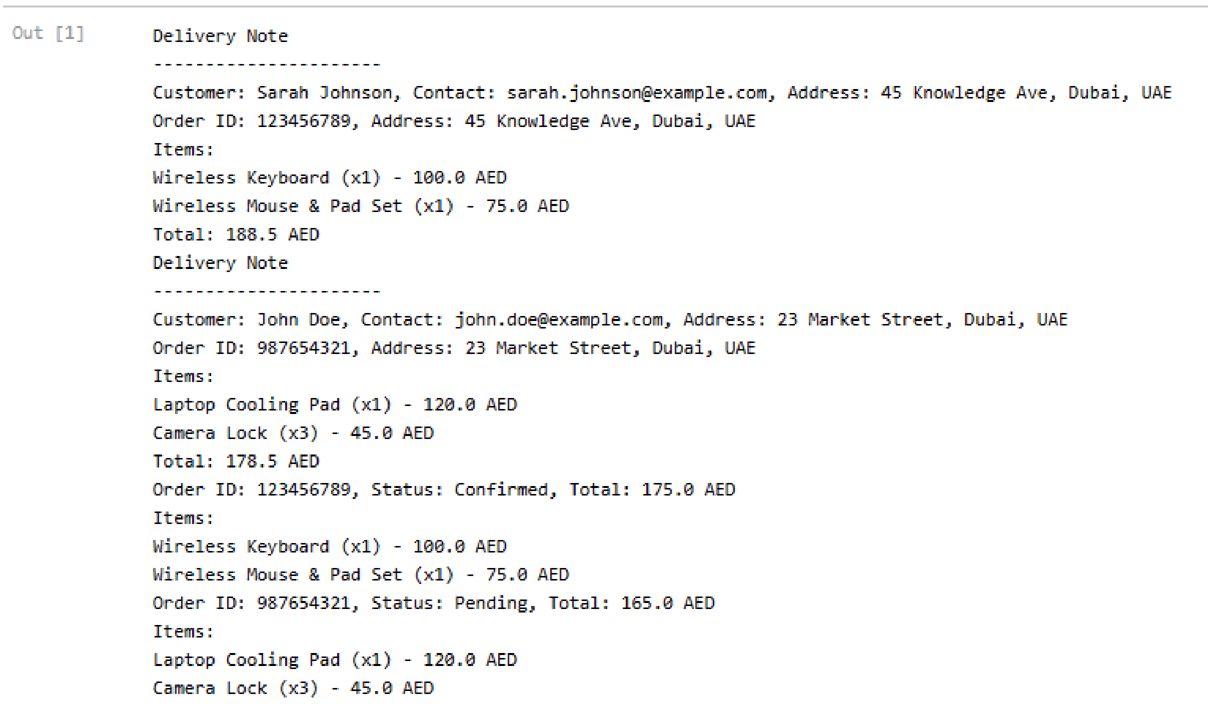
Uml Class diagram:



Python code:

```
from enum import Enum

# Enum for Order Status
class OrderStatus(Enum):
    PENDING = "Pending"
    CONFIRMED = "Confirmed"
    DELIVERED = "Delivered"

class Customer:
    def __init__(self, customer_id, name, contact, address):
        self.__customer_id = customer_id
        self.__name = name
        self.__contact = contact
        self.__address = address

    def get_name(self):
        return self.__name

    def set_name(self, name):
        self.__name = name

    def display(self):
        print(f"Customer: {self.__name}, Contact: {self.__contact}, Address: {self.__address}")

class Item:
    def __init__(self, code, description, quantity, price):
        self.__code = code
        self.__description = description
        self.__quantity = quantity
        self.__price = price

    def total_price(self):
        return self.__quantity * self.__price

    def display(self):
        print(f"{self.__description} (x{self.__quantity}) - {self.total_price()} AED")

class Order:
    def __init__(self, order_id, customer, delivery_address, status, delivery_date):
        self.__order_id = order_id
        self.__customer = customer
        self.__delivery_address = delivery_address
        self.__status = OrderStatus(status)
        self.__delivery_date = delivery_date
        self.__items = []
    def add_item(self, item):
        self.__items.append(item)

    def total_amt(self):
        return sum(item.total_price() for item in self.__items)

    def display(self):
        print(f"Order ID: {self.__order_id}, Status: {self.__status.value}, Total: {self.total_amt()}
AED")
        print("Items:")
        for item in self.__items:
            item.display()

class DeliveryNote:
    def __init__(self, order):
        self.__order = order
        self.__customer = order._Order__customer
        self.__items = order._Order__items
        self.__total_charges = order.total_amt() + 13.50

    def generate(self):
        print("Delivery Note")
        print("----------------------")
        self.__customer.display()
        print(f"Order ID: {self.__order._Order__order_id}, Address:
{self.__order._Order__delivery_address}")
        print("Items:")
        for item in self.__items:
            item.display()
        print(f"Total: {self.__total_charges} AED")

class DeliverySystem:
    def __init__(self):
        self.__orders = []

    def add_order(self, order):
        self.__orders.append(order)

    def display_orders(self):
        for order in self.__orders:
            order.display()

# Example Usage
customer1 = Customer(1, "Sarah Johnson", "sarah.johnson@example.com", "45 Knowledge
Ave, Dubai, UAE")
customer2 = Customer(2, "John Doe", "john.doe@example.com", "23 Market Street, Dubai,
UAE")
```

```
order1 = Order(123456789, customer1, "45 Knowledge Ave, Dubai, UAE", "Confirmed",
"2025-01-25")
order2 = Order(987654321, customer2, "23 Market Street, Dubai, UAE", "Pending",
"2025-02-01")

order1.add_item(Item("ITM001", "Wireless Keyboard", 1, 100.00))
order1.add_item(Item("ITM002", "Wireless Mouse & Pad Set", 1, 75.00))
order2.add_item(Item("ITM003", "Laptop Cooling Pad", 1, 120.00))
order2.add_item(Item("ITM004", "Camera Lock", 3, 15.00))

delivery_system = DeliverySystem()
delivery_system.add_order(order1)
delivery_system.add_order(order2)

delivery_note1 = DeliveryNote(order1)
delivery_note2 = DeliveryNote(order2)

delivery_note1.generate()
delivery_note2.generate()

delivery_system.display_orders()

pass  # Placeholder to ensure script runs without error
```



```
Out [1]    Delivery Note
           ----------------------
           Customer: Sarah Johnson, Contact: sarah.johnson@example.com, Address: 45 Knowledge Ave, Dubai, UAE
           Order ID: 123456789, Address: 45 Knowledge Ave, Dubai, UAE
           Items:
           Wireless Keyboard (x1) - 100.0 AED
           Wireless Mouse & Pad Set (x1) - 75.0 AED
           Total: 188.5 AED
           Delivery Note
           ----------------------
           Customer: John Doe, Contact: john.doe@example.com, Address: 23 Market Street, Dubai, UAE
           Order ID: 987654321, Address: 23 Market Street, Dubai, UAE
           Items:
           Laptop Cooling Pad (x1) - 120.0 AED
           Camera Lock (x3) - 45.0 AED
           Total: 178.5 AED
           Order ID: 123456789, Status: Confirmed, Total: 175.0 AED
           Items:
           Wireless Keyboard (x1) - 100.0 AED
           Wireless Mouse & Pad Set (x1) - 75.0 AED
           Order ID: 987654321, Status: Pending, Total: 165.0 AED
           Items:
           Laptop Cooling Pad (x1) - 120.0 AED
           Camera Lock (x3) - 45.0 AED
```

Summary of what i have learned:

I gained a better understanding of object-oriented programming and UML modeling during this project. I acquired the ability to deconstruct a practical delivery system into use cases and logical classes.

I was able to see the advantages of modular, reusable code by putting encapsulation (private attributes, public methods) and linkages between classes into practice. Using OOP principles to dynamically generate delivery notes strengthened practical software development methods.

Because I built a structured repository using software engineering best practices, this project also improved my GitHub version control abilities.