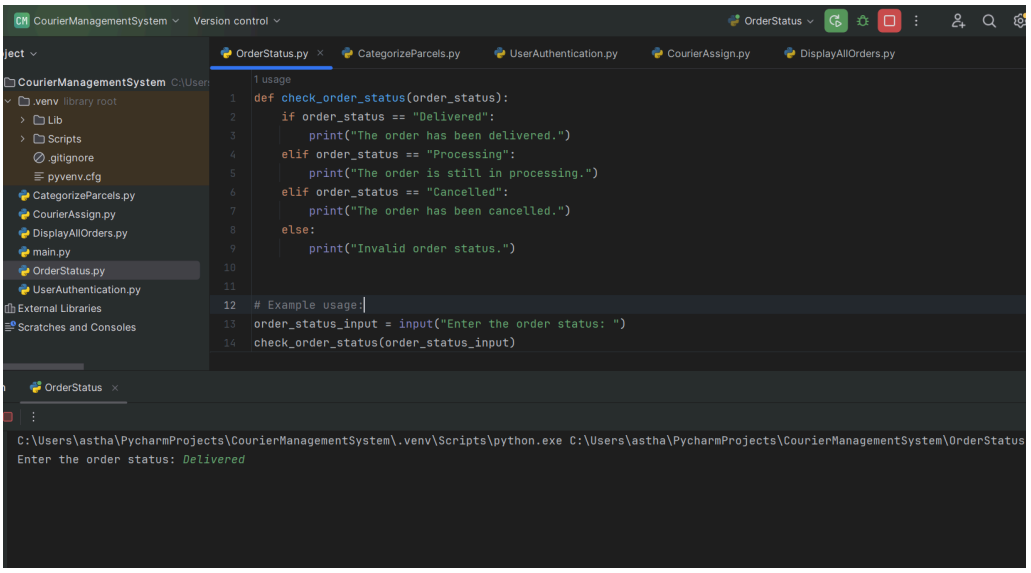
Assignment4: Courier Management System

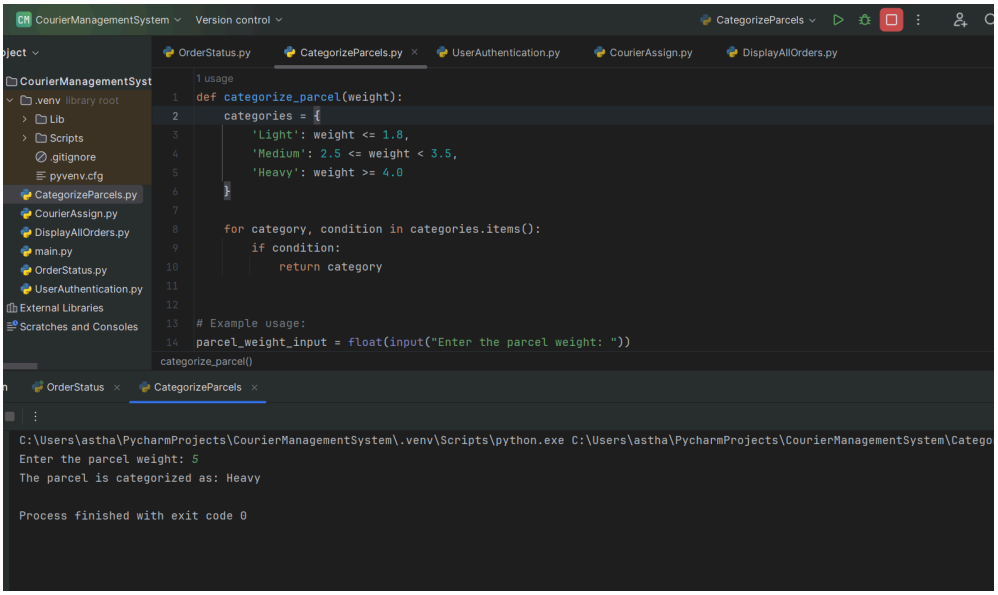
Coding Task 1:

Control Flow Statements

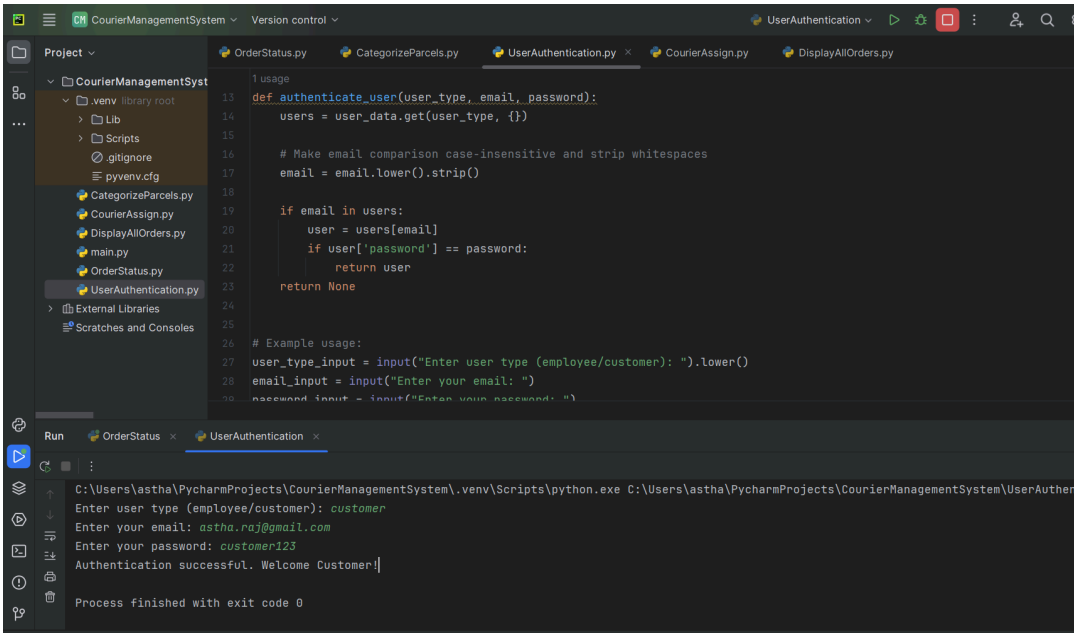
1. Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.

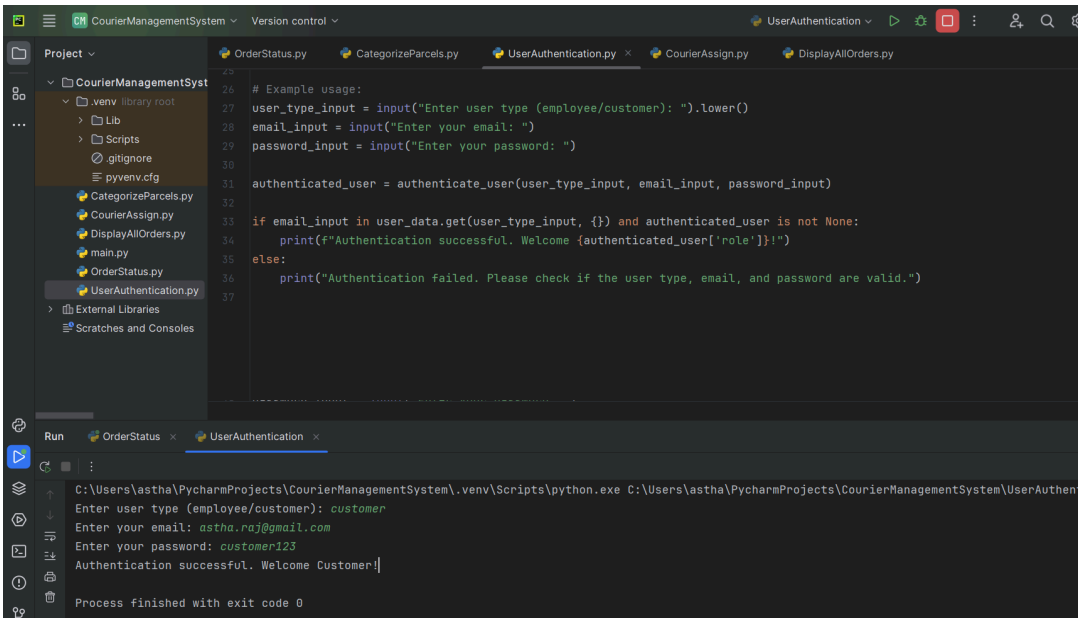


2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

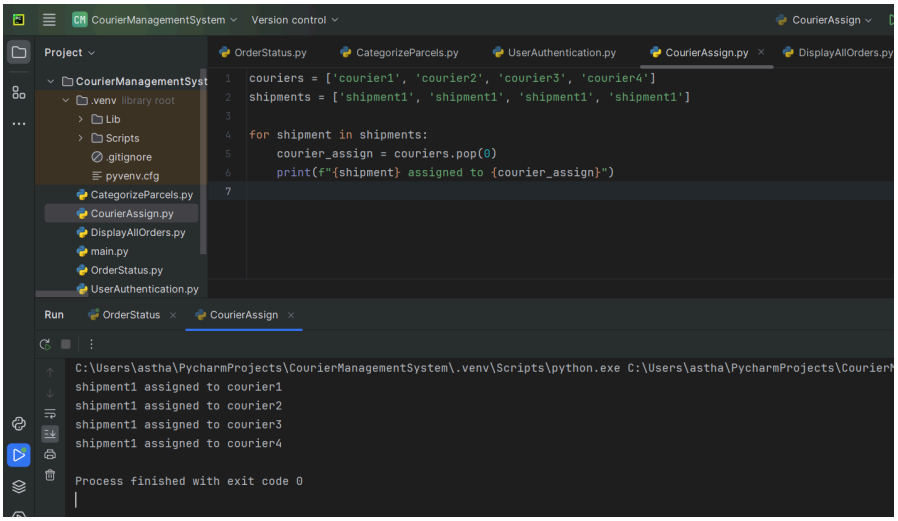


3. Implement User Authentication 1. Create a login system for employees and customers using Java control flow statements.



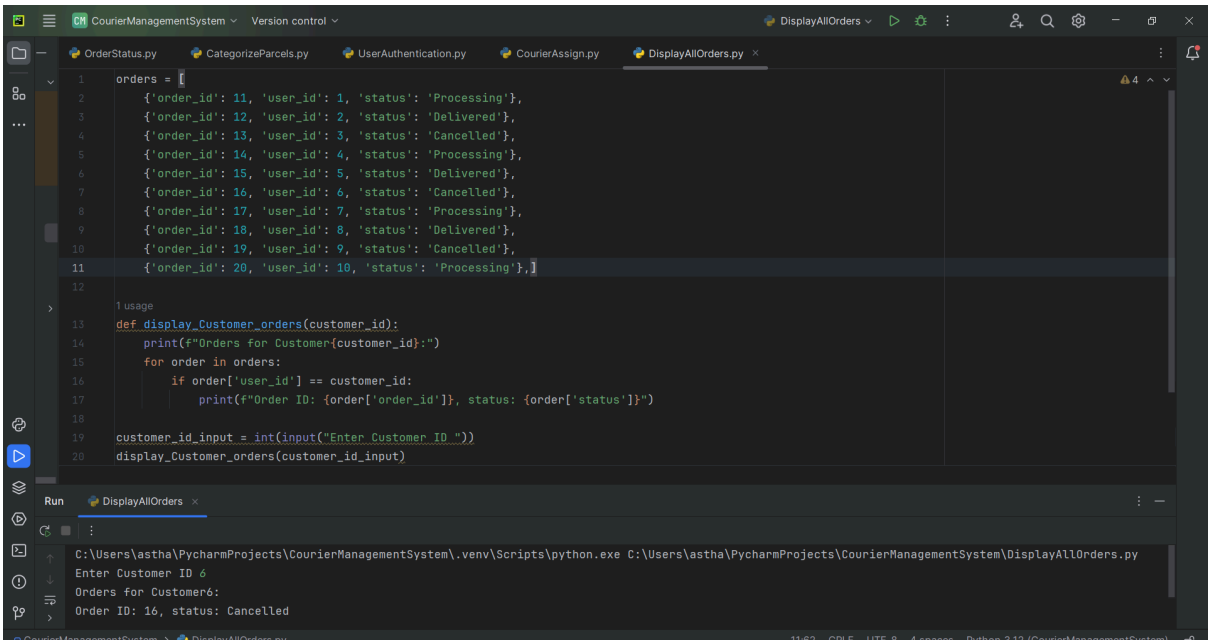


4. Implement Courier Assignment Logic 1. Develop a mechanism to assign couriers to shipments based on predefined criteria (e.g., proximity, load capacity) using loops.

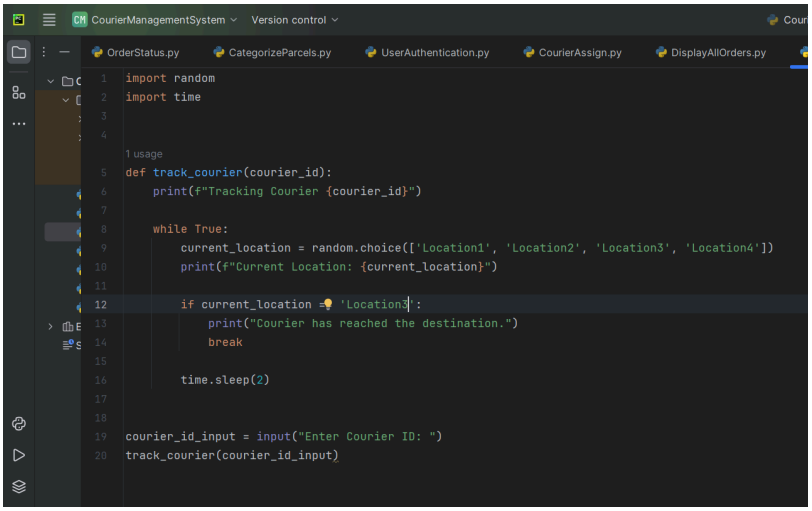


Task 2: Loops and Iteration

5. Write a Java program that uses a for loop to display all the orders for a specific customer.

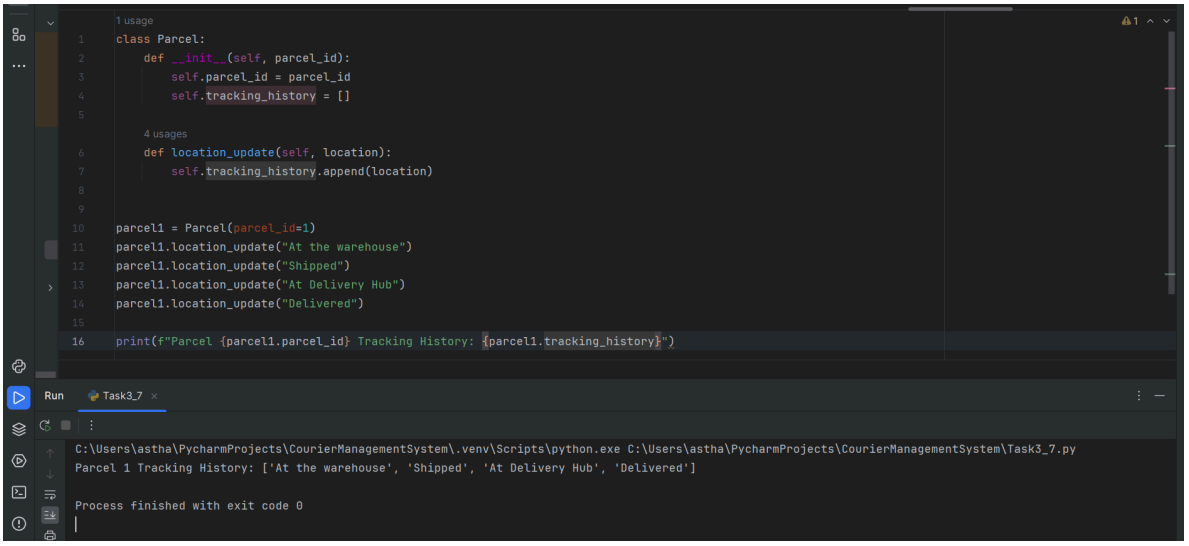


6. Implement a while loop to track the real-time location of a courier until it reaches its destination.

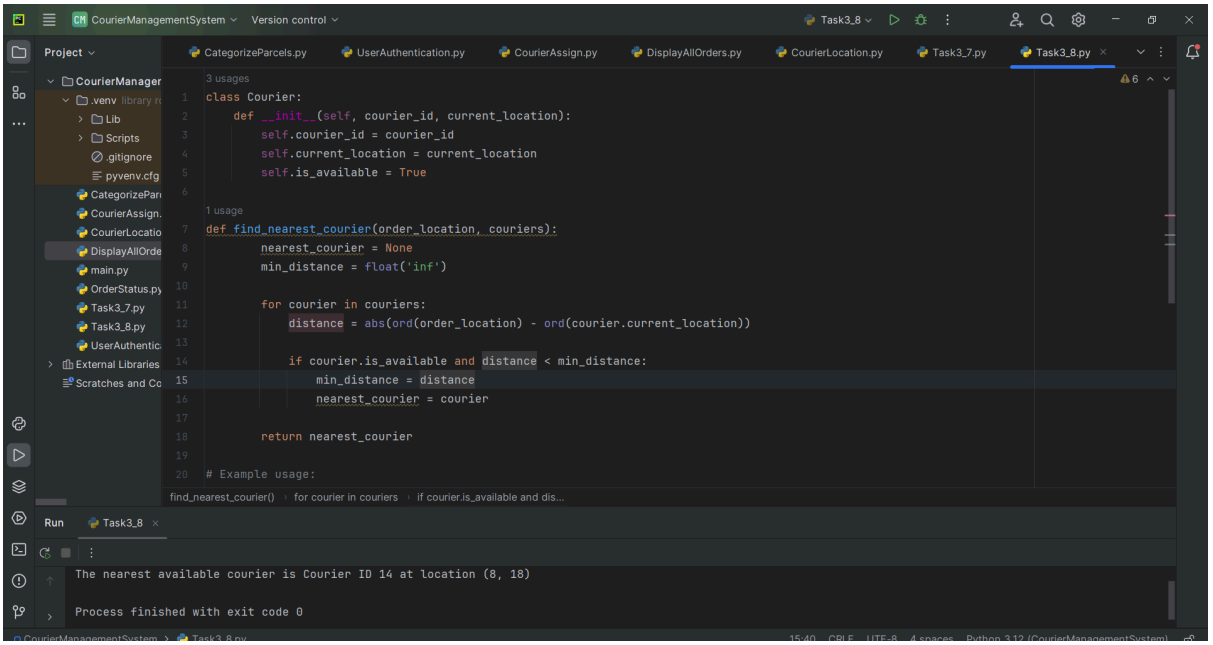


Task 3: Arrays and Data Structures

7. Create an array to store the tracking history of a parcel, where each entry represents a location update.



8. Implement a method to find the nearest available courier for a new order using an array of couriers.

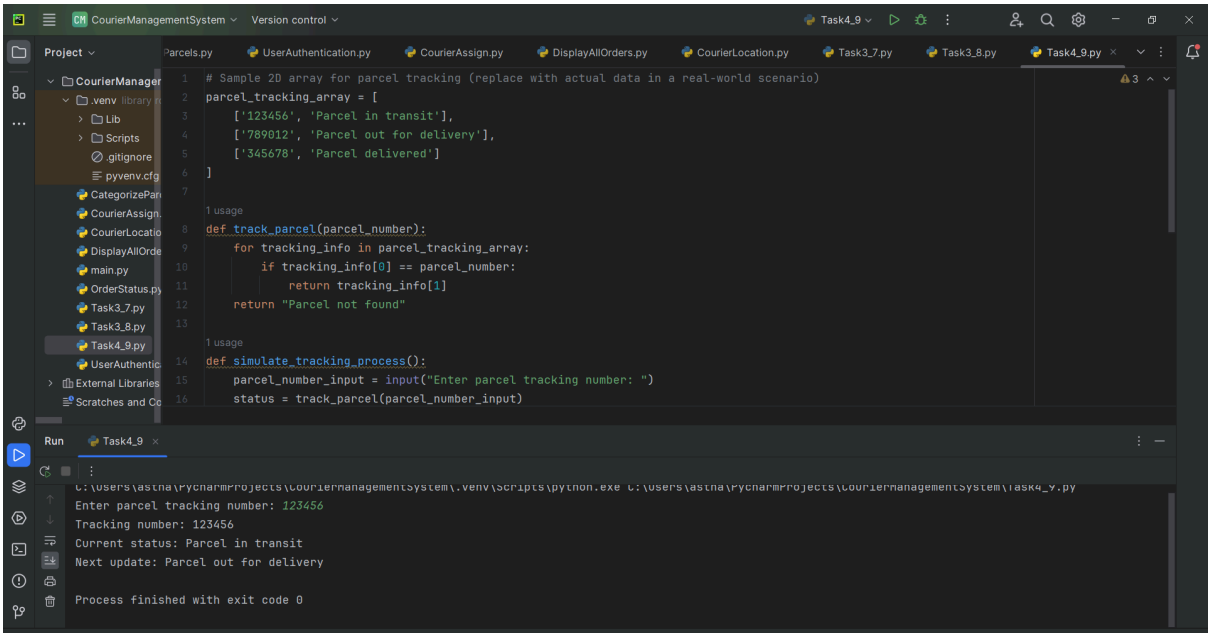


```
class Courier:
    def __init__(self, courier_id, current_location):
        self.courier_id = courier_id
        self.current_location = current_location
        self.is_available = True

def find_nearest_courier(order_location, couriers):
        nearest_courier = None
        min_distance = float('inf')

        for courier in couriers:
            distance = abs(ord(order_location) - ord(courier.current_location))

            if courier.is_available and distance < min_distance:
                min_distance = distance
                nearest_courier = courier

        return nearest_courier

# Example usage:
```

```
The nearest available courier is Courier ID 14 at location (8, 18)

Process finished with exit code 0
```

Task 4: Strings,2d Arrays, user defined functions,Hashmap

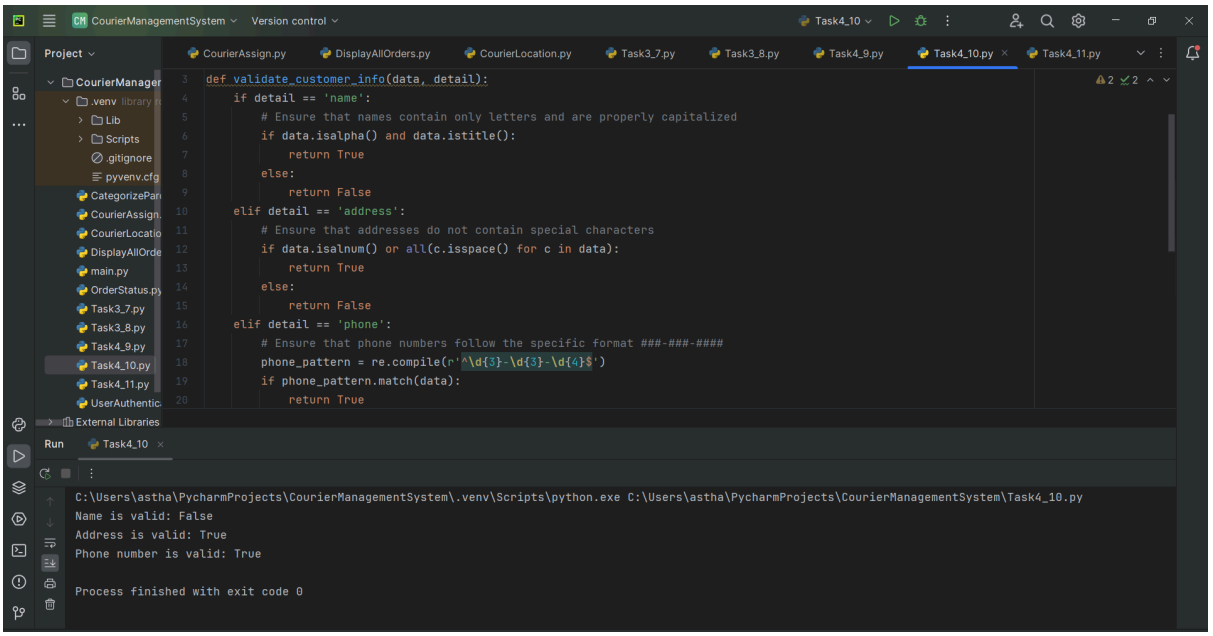
9. Parcel Tracking: Create a program that allows users to input a parcel tracking number.Store the tracking number and Status in 2d String Array. Initialize the array with values. Then, simulate the tracking process by displaying messages like "Parcel in transit," "Parcel out for delivery," or "Parcel delivered" based on the tracking number's status.



```
# Sample 2D array for parcel tracking (replace with actual data in a real-world scenario)
parcel_tracking_array = [
    ['123456', 'Parcel in transit'],
    ['789012', 'Parcel out for delivery'],
    ['345678', 'Parcel delivered']
]

def track_parcel(parcel_number):
    for tracking_info in parcel_tracking_array:
        if tracking_info[0] == parcel_number:
            return tracking_info[1]
    return "Parcel not found"

def simulate_tracking_process():
    parcel_number_input = input("Enter parcel tracking number: ")
    status = track_parcel(parcel_number_input)
```

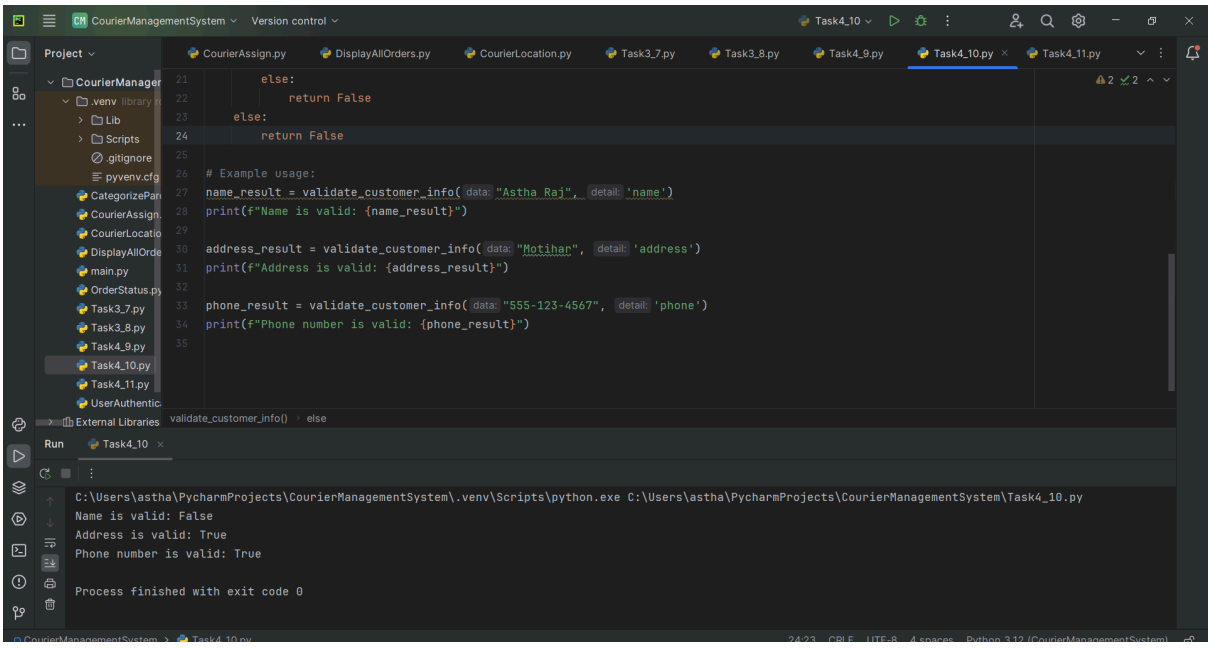
```
Enter parcel tracking number: 123456
Tracking number: 123456
Current status: Parcel in transit
Next update: Parcel out for delivery

Process finished with exit code 0
```

10. Customer Data Validation: Write a function which takes 2 parameters, data-denotes the data and detail-denotes if it is name addtress or phone number.Validate customer information based on following critirea. Ensure that names contain only letters and are properly capitalized, addresses do not contain special characters, and phone numbers follow a specific format (e.g., ###-###-####).



```
def validate_customer_info(data, detail):
    if detail == 'name':
        # Ensure that names contain only letters and are properly capitalized
        if data.isalpha() and data.istitle():
            return True
        else:
            return False
    elif detail == 'address':
        # Ensure that addresses do not contain special characters
        if data.isalnum() or all(c.isspace() for c in data):
            return True
        else:
            return False
    elif detail == 'phone':
        # Ensure that phone numbers follow the specific format ###-###-####
        phone_pattern = re.compile(r'^\d{3}-\d{3}-\d{4}$')
        if phone_pattern.match(data):
            return True
```

```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_10.py
Name is valid: False
Address is valid: True
Phone number is valid: True

Process finished with exit code 0
```

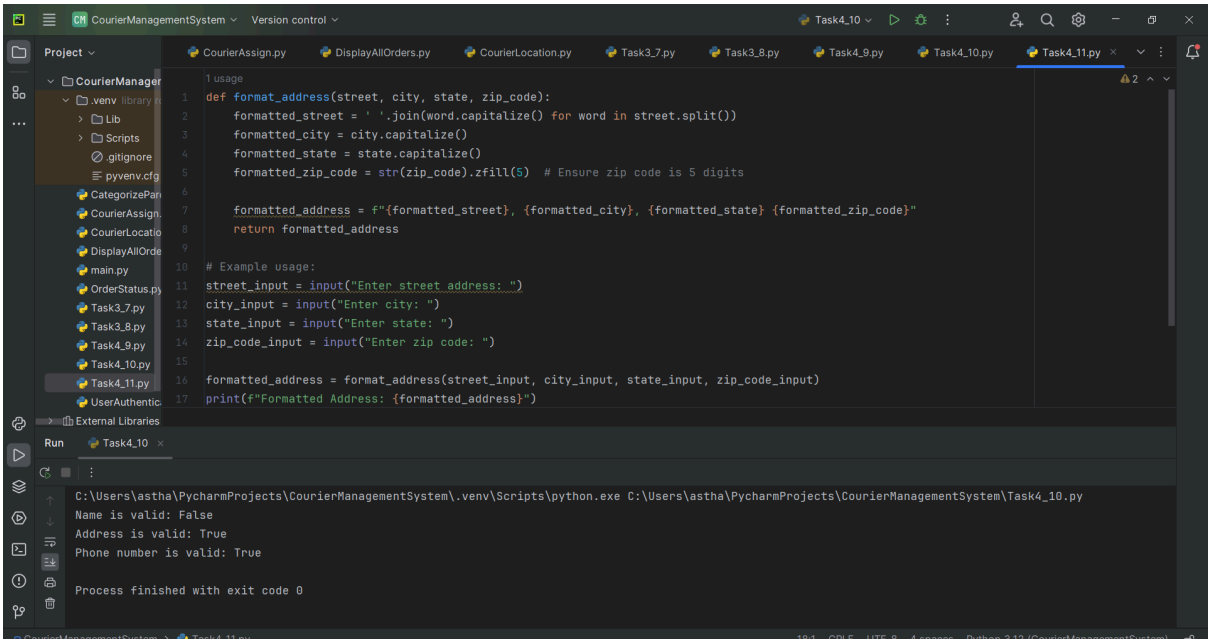


```
        else:
            return False
    else:
        return False

# Example usage:
name_result = validate_customer_info( data: "Astha Raj",  detail: 'name')
print(f"Name is valid: {name_result}")

address_result = validate_customer_info( data: "Motihar",  detail: 'address')
print(f"Address is valid: {address_result}")

phone_result = validate_customer_info( data: "555-123-4567",  detail: 'phone')
print(f"Phone number is valid: {phone_result}")
```

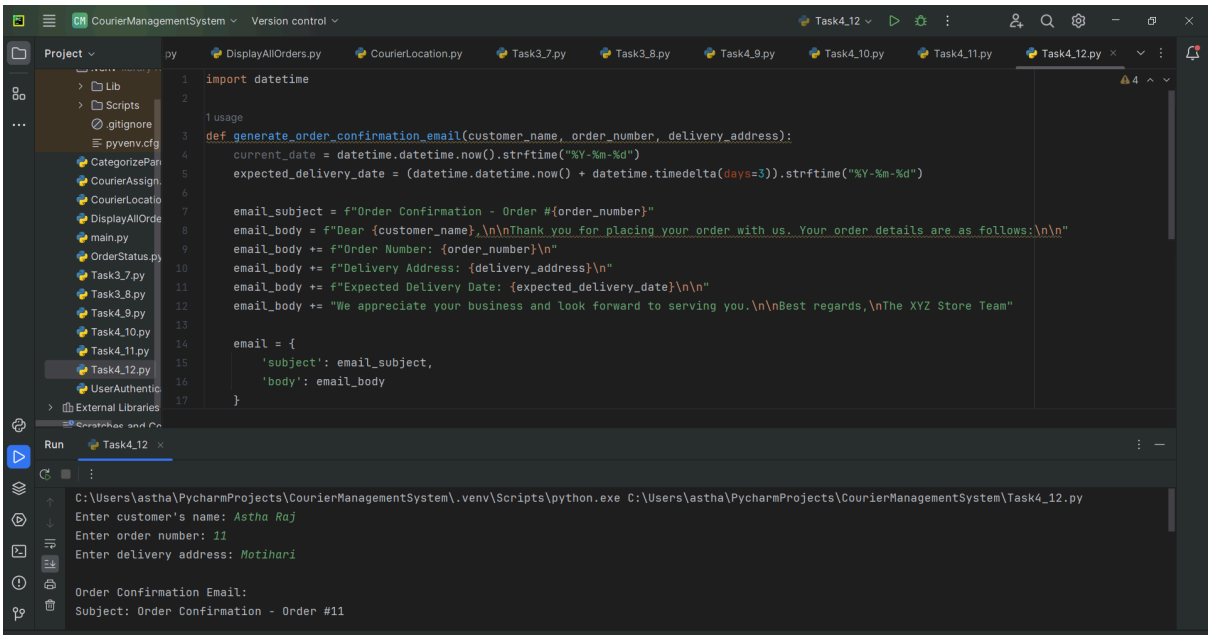
```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_10.py
Name is valid: False
Address is valid: True
Phone number is valid: True

Process finished with exit code 0
```

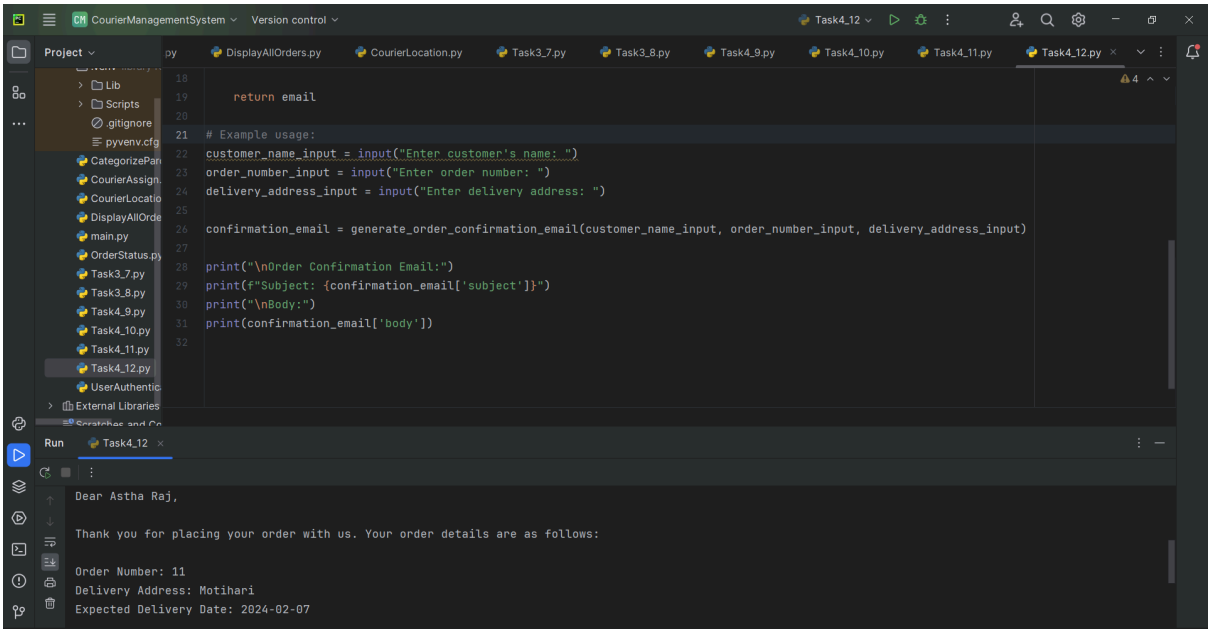
11. Address Formatting: Develop a function that takes an address as input (street, city, state, zip code)

and formats it correctly, including capitalizing the first letter of each word and properly formatting the
zip code.

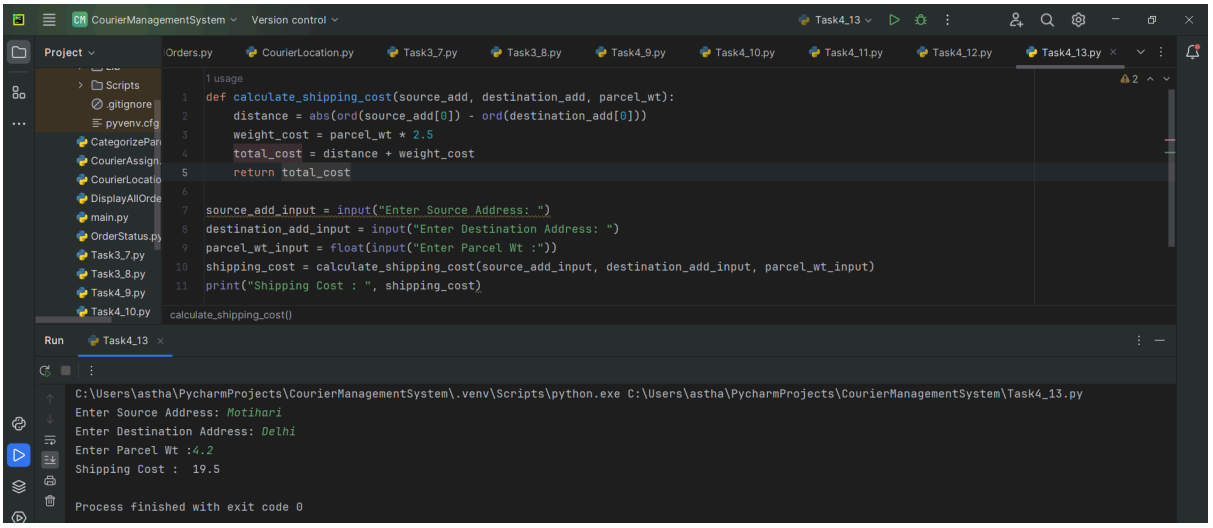


12. Order Confirmation Email: Create a program that generates an order confirmation email. The email should include details such as the customer's name, order number, delivery address, and expected delivery date.

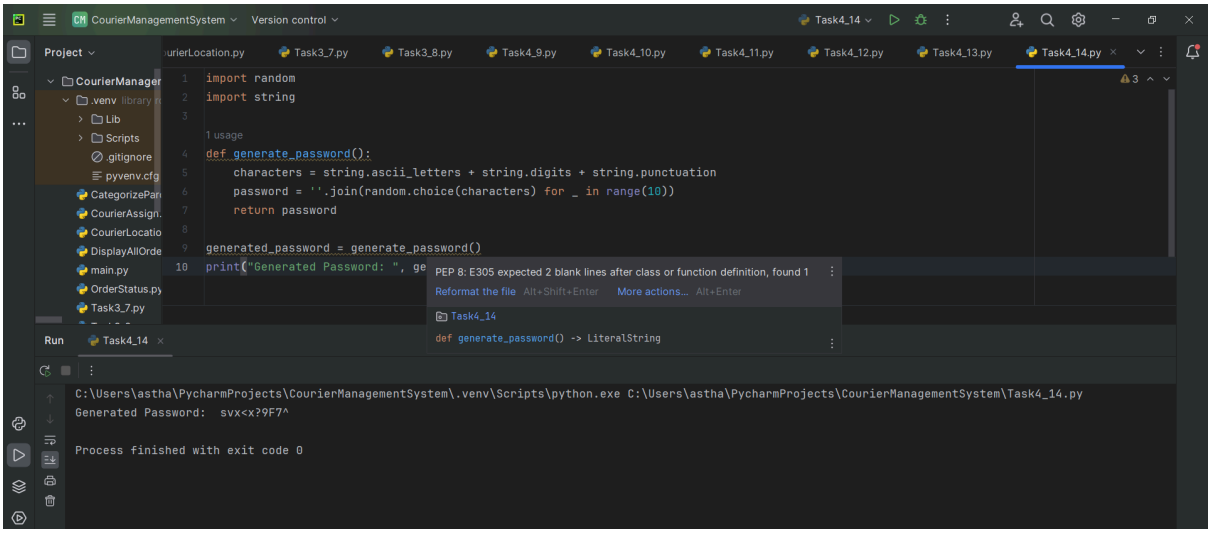




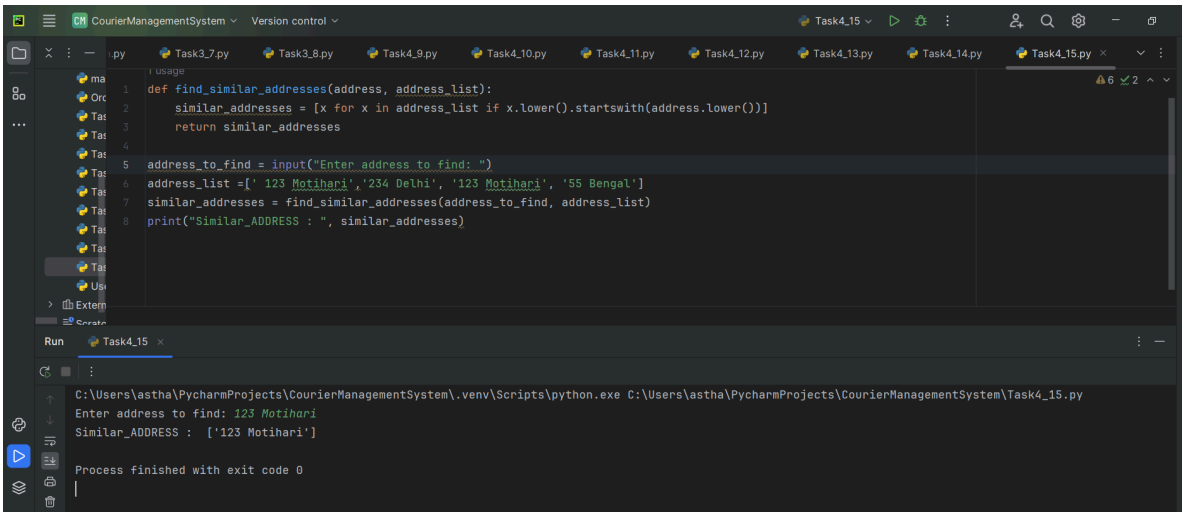
13. Calculate Shipping Costs: Develop a function that calculates the shipping cost based on the distance between two locations and the weight of the parcel. You can use string inputs for the source and destination addresses.



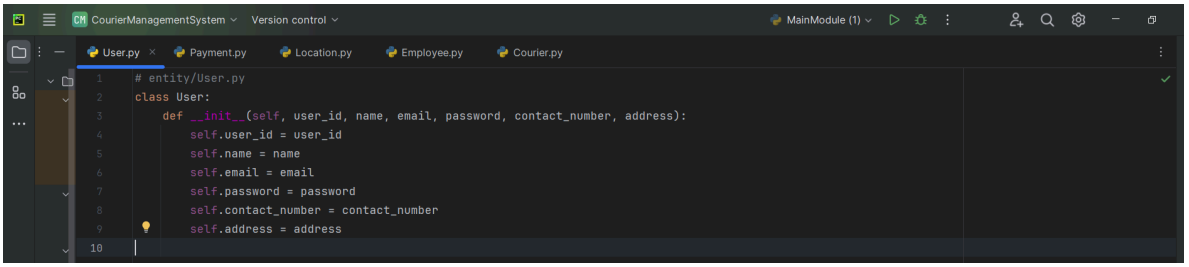
14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.



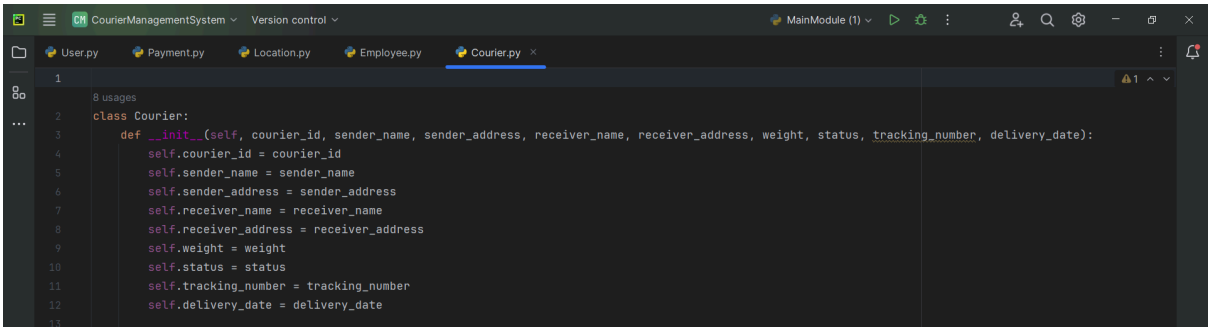
15. Find Similar Addresses: Implement a function that finds similar addresses in the system. This can be useful for identifying duplicate customer entries or optimizing delivery routes.Use string functions to implement this.



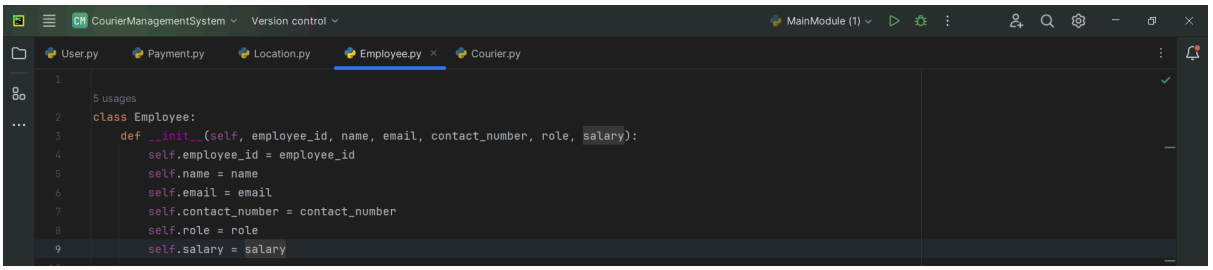
Task 5: Object Oriented Programming Scope :
Entity classes/Models/POJO, Abstraction/Encapsulation
Create the following model/entity classes within package entities with variables declared private, constructors(default and parametrized,getters,setters and toString())
1. User Class: Variables: userID , userName , email , password , contactNumber , address



2. Courier Class Variables: courierID , senderName , senderAddress , receiverName , receiverAddress , weight , status, trackingNumber , deliveryDate ,userId



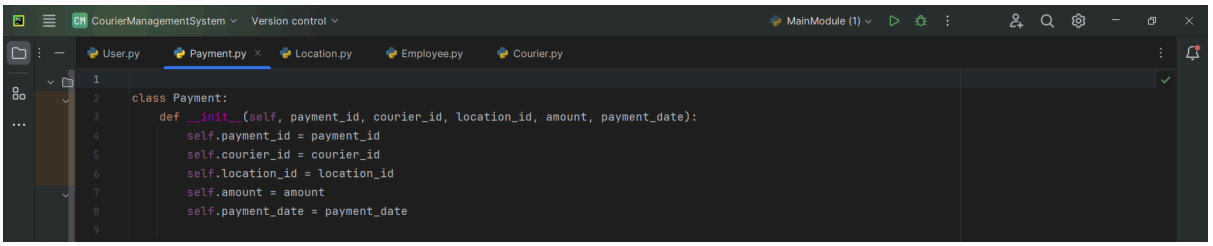
3. Employee Class: Variables employeeID , employeeName , email , contactNumber , role String, salary



4. Location Class Variables LocationID , LocationName , Address

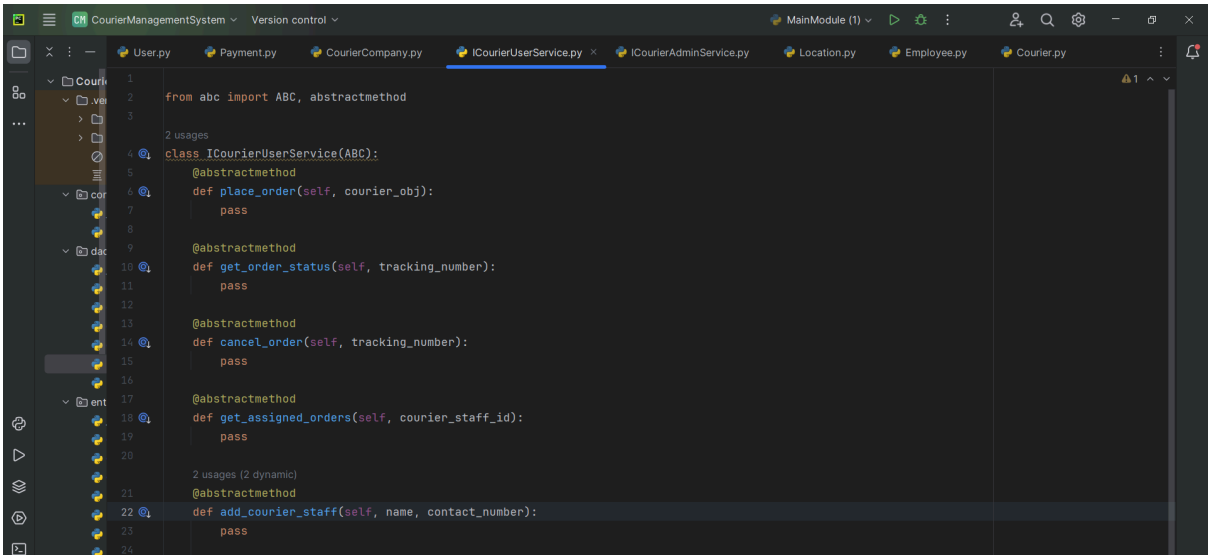
5. CourierCompany Class Variables companyName , courierDetails -collection of Courier Objects, employeeDetails- collection of Employee Objects, locationDetails - collection of Location Objects.

6. Payment Class: Variables PaymentID long, CourierID long, Amount double, PaymentDate Date

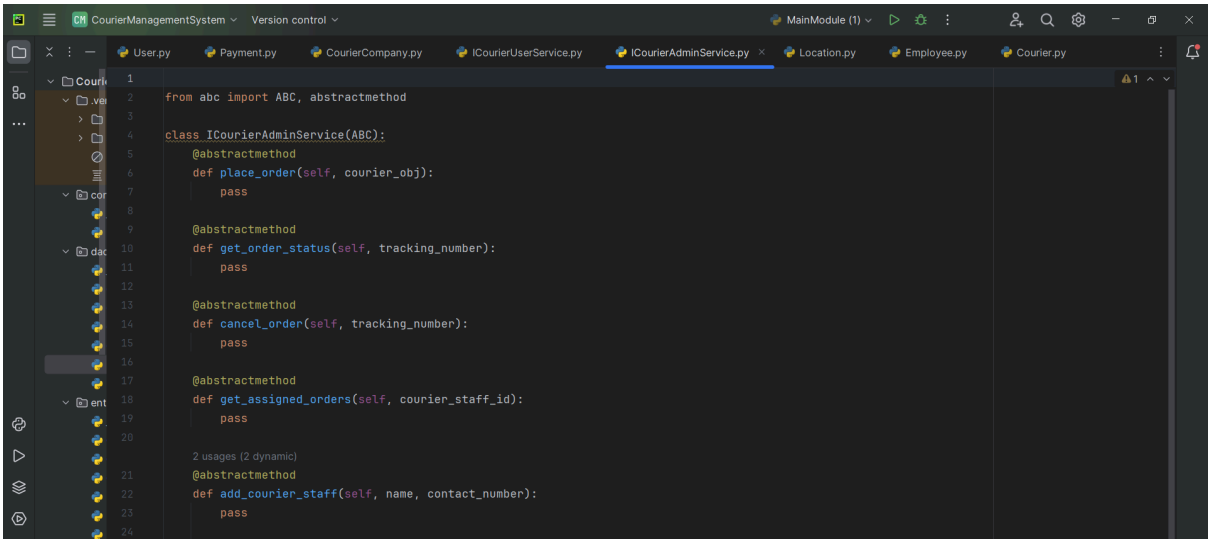


Task 6: Service Provider Interface
/Abstract class Create 2 Interface
/Abstract class ICourierUserService and ICourierAdminService interface
ICourierUserService
{ // Customer-related functions
placeOrder()
/** Place a new courier order. * @param courierObj Courier object created using values entered by users * @return The unique tracking number for the courier order . Use a static variable to generate unique tracking number. Initialize the static variable in Courier class with some random value. Increment the static variable each time in the constructor to generate next values.
getOrderStatus();
/**Get the status of a courier order. *@param trackingNumber The tracking number of the courier order. * @return The status of the courier order (e.g., yetToTransit, In Transit, Delivered). */
cancelOrder()
/** Cancel a courier order. * @param trackingNumber The tracking number of the courier order to be canceled. * @return True if the order was successfully canceled, false otherwise.*/

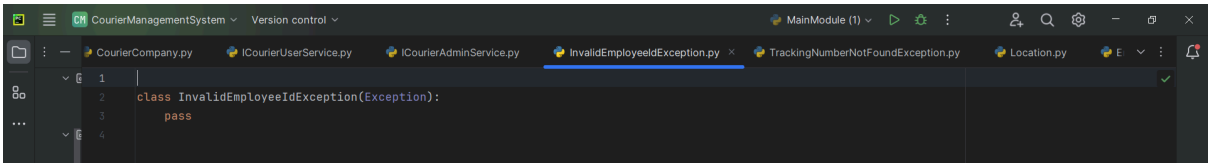
getAssignedOrder()
/** Get a list of orders assigned to a specific courier staff member * @param courierStaffId The ID of the courier staff member. * @return A list of courier orders assigned to the staff member.*/

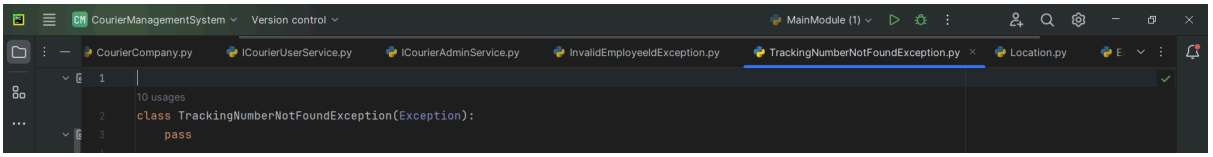


// Admin functions
ICourierAdminService
int addCourierStaff(Employee obj);
/** Add a new courier staff member to the system. * @param name The name of the courier staff member. * @param contactNumber The contact number of the courier staff member. * @return The ID of the newly added courier staff member. */

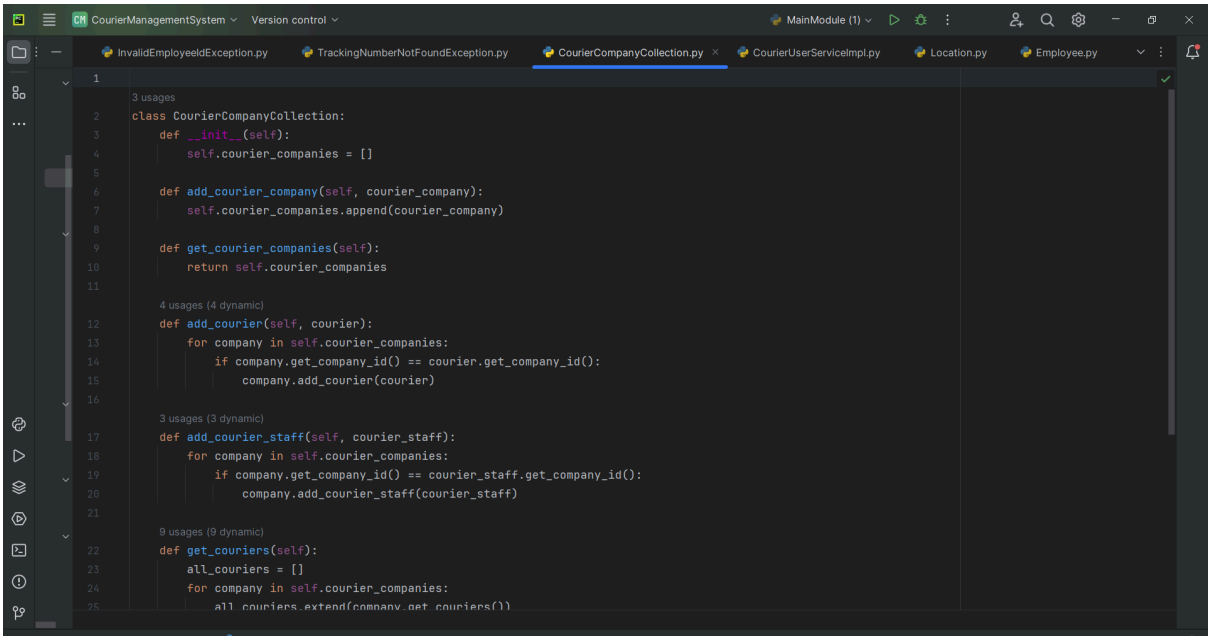


Task 7: Exception Handling
(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally,thow & throws keyword usage)
Define the following custom exceptions and throw them in methods whenever needed . Handle all the excpetions in main method, 1. TrackingNumberNotFoundException :throw this exception when user try to withdraw amount or transfer amount to another account 2. InvalidEmployeeIdException throw this exception when id entered for the employee not existing in the system

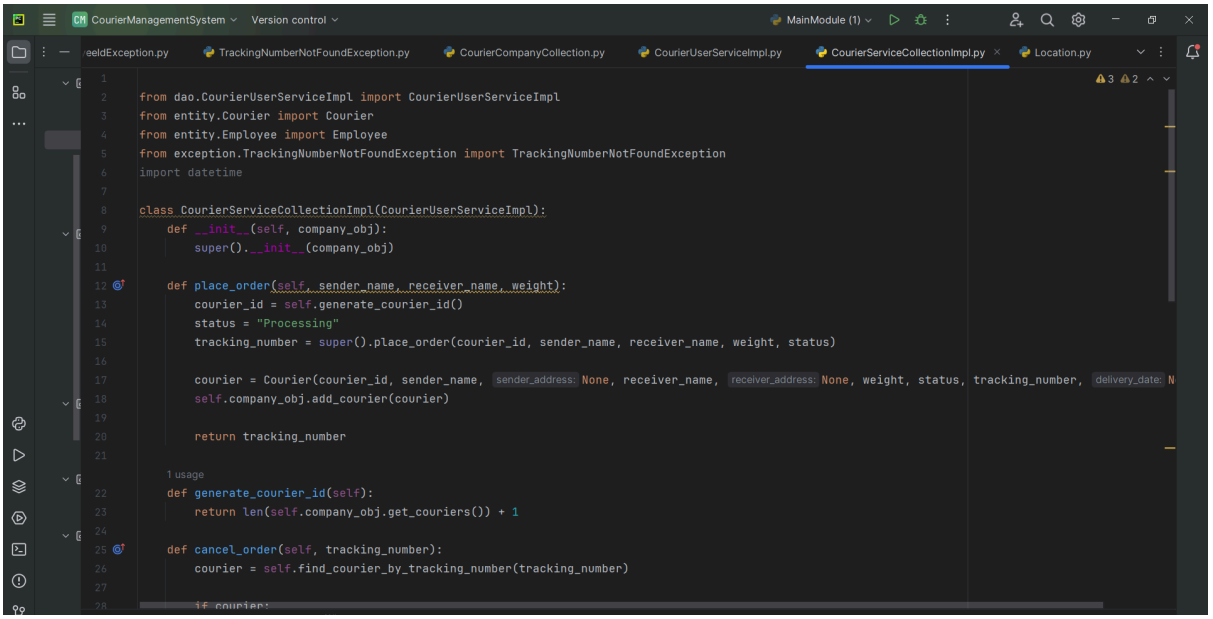




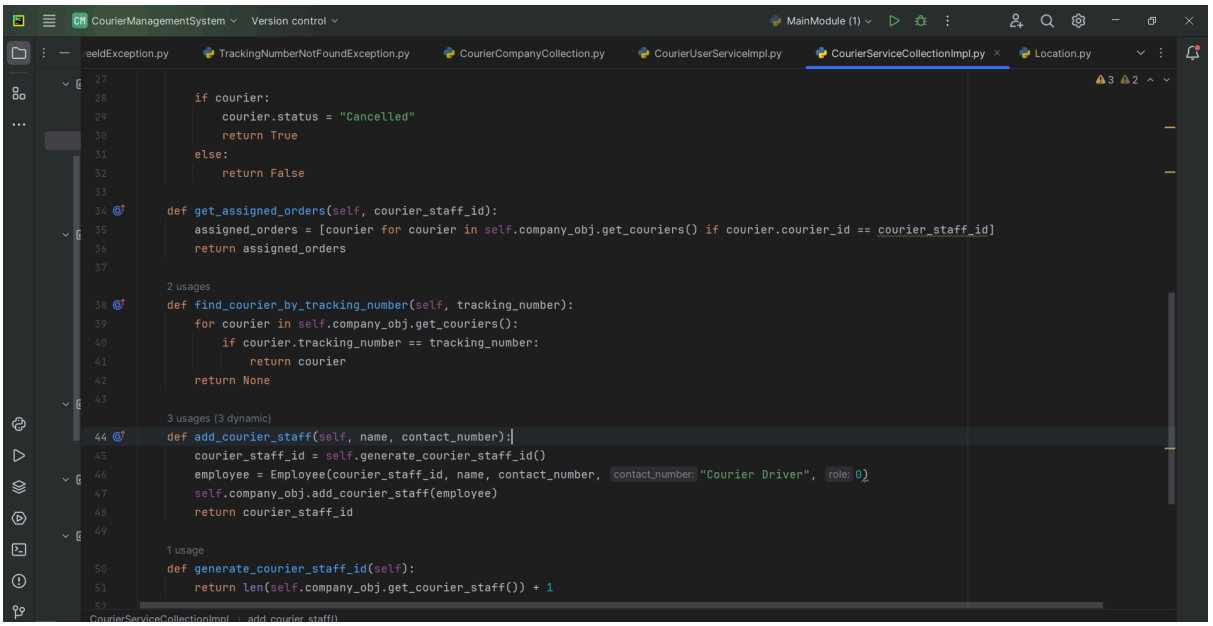
Task 8: Collections Scope: ArrayList/Hashmap Task: Improve the Courier Management System by using Java collections: 1. Create a new model named CourierCompanyCollection in entity package replacing the Array of Objects with List to accommodate dynamic updates in the CourierCompany class



2. Create a new implementation class CourierUserServiceCollectionImpl class in package dao which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompanyCollection

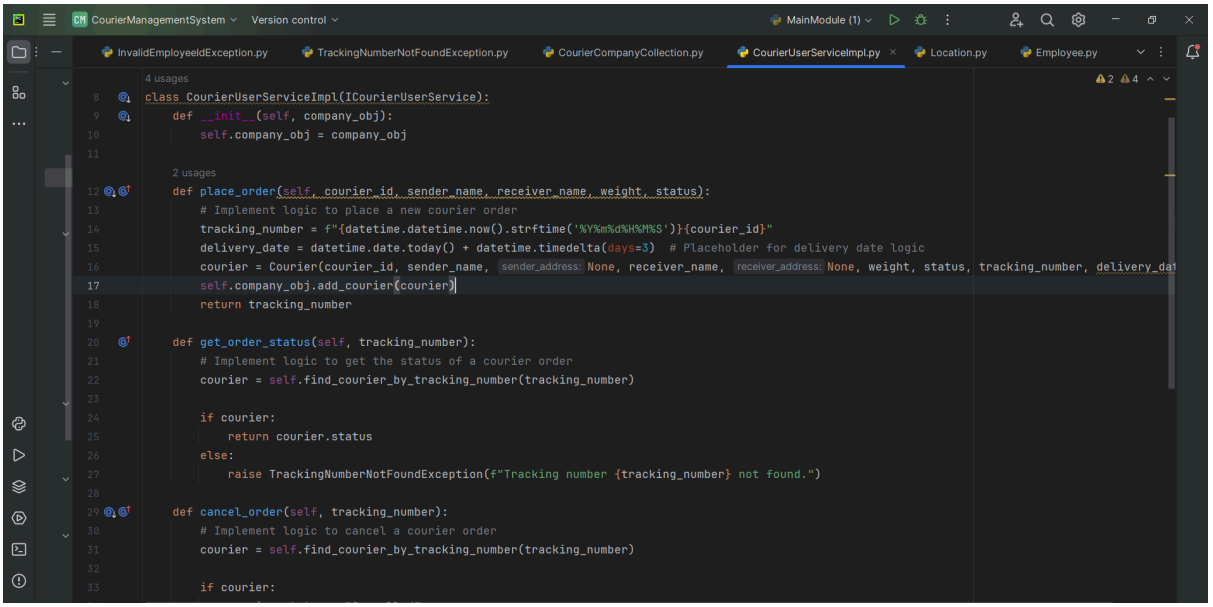


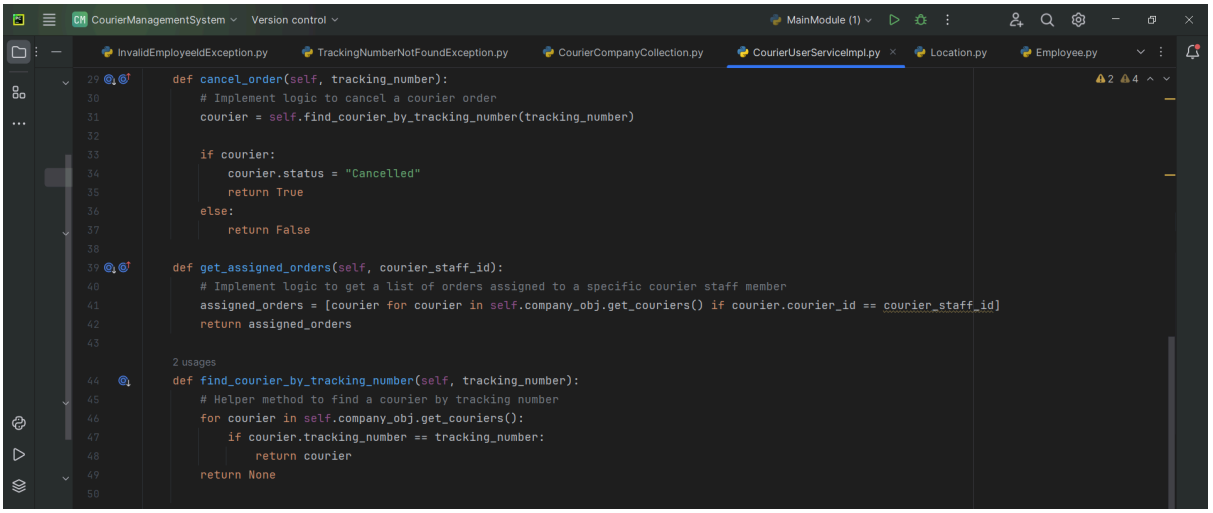
```python
from dao.CourierUserServiceImpl import CourierUserServiceImpl
from entity.Courier import Courier
from entity.Employee import Employee
from exception.TrackingNumberNotFoundException import TrackingNumberNotFoundException
import datetime

class CourierServiceCollectionImpl(CourierUserServiceImpl):
    def __init__(self, company_obj):
        super().__init__(company_obj)

    def place_order(self, sender_name, receiver_name, weight):
        courier_id = self.generate_courier_id()
        status = "Processing"
        tracking_number = super().place_order(courier_id, sender_name, receiver_name, weight, status)

        courier = Courier(courier_id, sender_name, sender_address: None, receiver_name, receiver_address: None, weight, status, tracking_number, delivery_date: N
        self.company_obj.add_courier(courier)

        return tracking_number

    1 usage
    def generate_courier_id(self):
        return len(self.company_obj.get_couriers()) + 1

    def cancel_order(self, tracking_number):
        courier = self.find_courier_by_tracking_number(tracking_number)

        if courier:
```



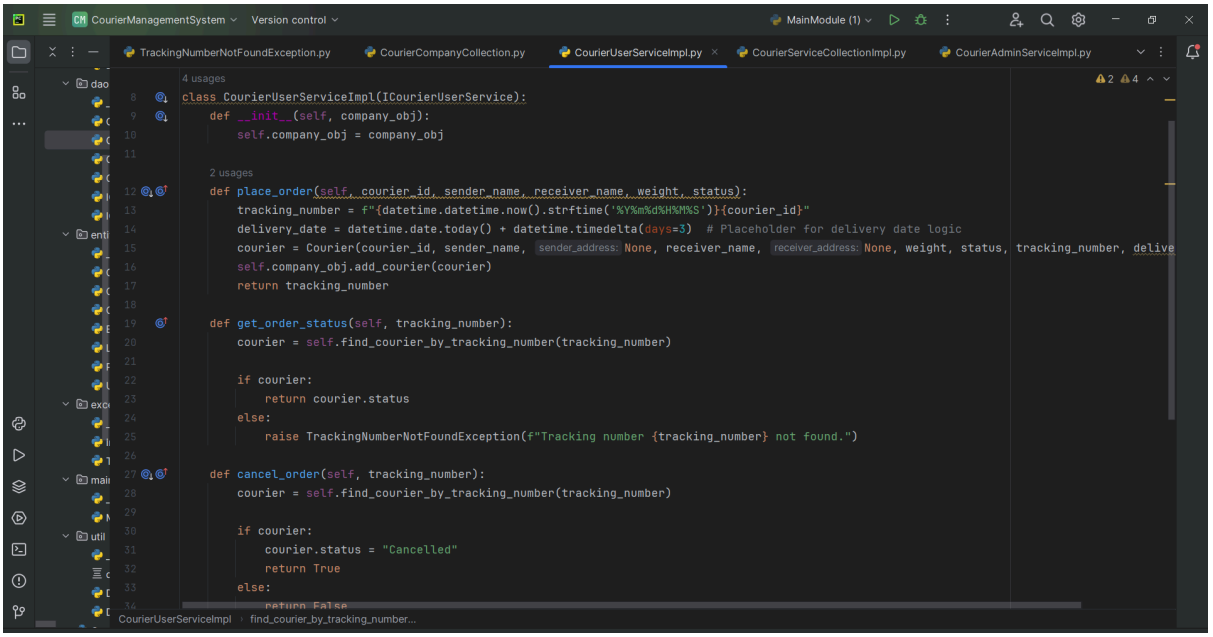
```python
        if courier:
            courier.status = "Cancelled"
            return True
        else:
            return False

    def get_assigned_orders(self, courier_staff_id):
        assigned_orders = [courier for courier in self.company_obj.get_couriers() if courier.courier_id == courier_staff_id]
        return assigned_orders

    2 usages
    def find_courier_by_tracking_number(self, tracking_number):
        for courier in self.company_obj.get_couriers():
            if courier.tracking_number == tracking_number:
                return courier
        return None

    3 usages (3 dynamic)
    def add_courier_staff(self, name, contact_number):
        courier_staff_id = self.generate_courier_staff_id()
        employee = Employee(courier_staff_id, name, contact_number, contact_number: "Courier Driver", role: 0)
        self.company_obj.add_courier_staff(employee)
        return courier_staff_id

    1 usage
    def generate_courier_staff_id(self):
        return len(self.company_obj.get_courier_staff()) + 1
```

Task 8: Service implementation 1.Create CourierUserServiceImpl class which implements ICourierUserService interface which holds a variable named companyObj of type CourierCompany. This variable can be used to access the Object Arrays to access data relevant in method implementations.

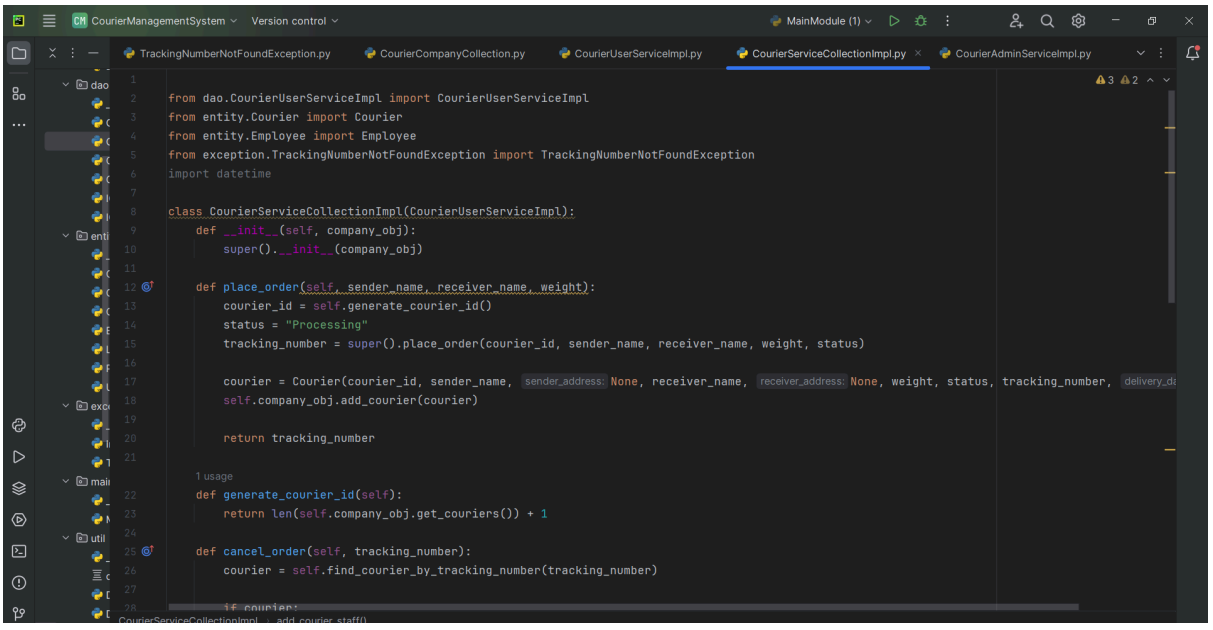


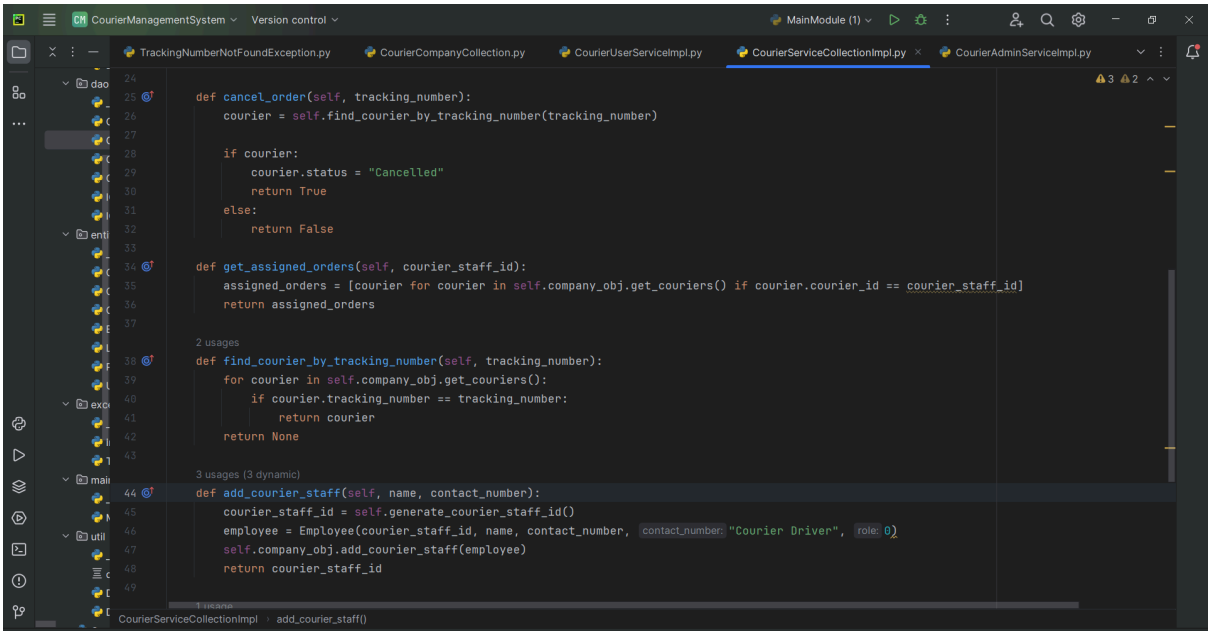


2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and implements ICourierAdminService interface.

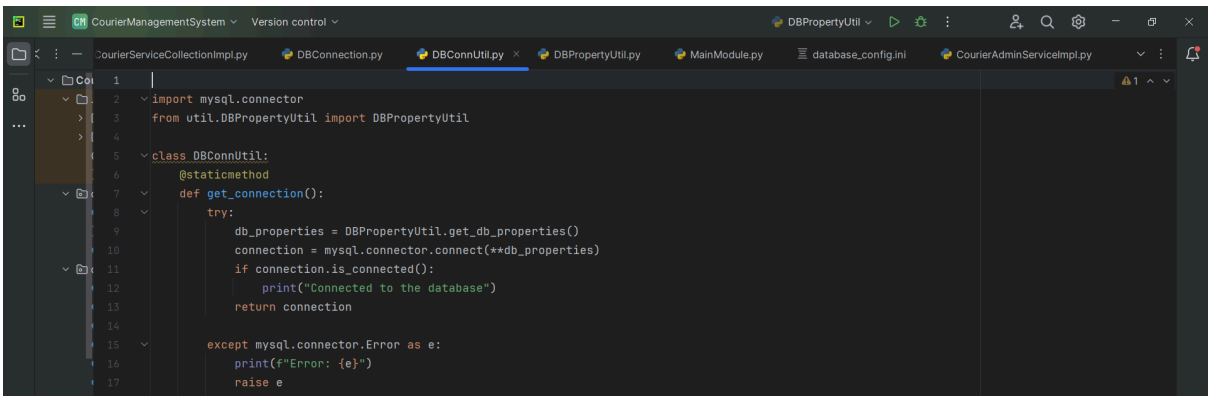


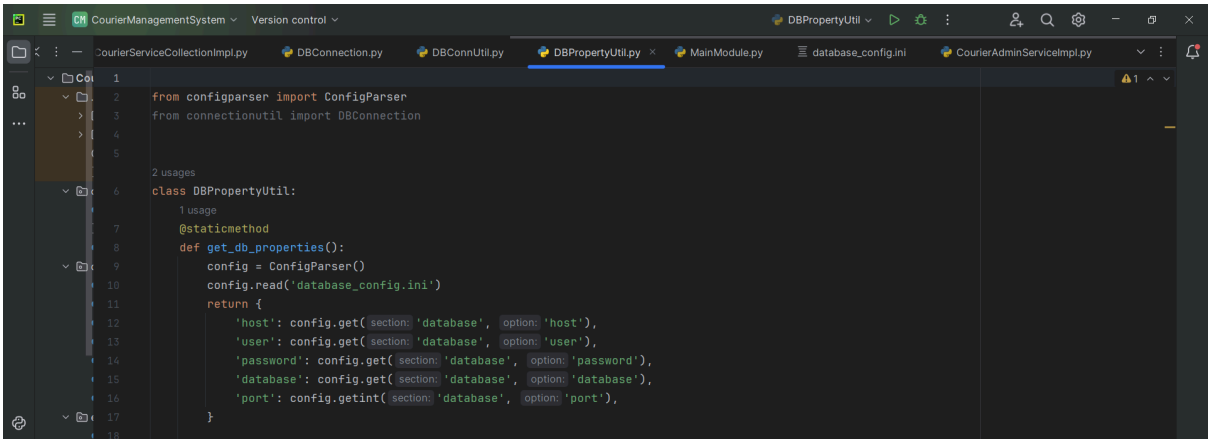
3. Create CourierAdminServiceCollectionImpl class which inherits from CourierUserServiceColectionImpl and implements ICourierAdminService interface.



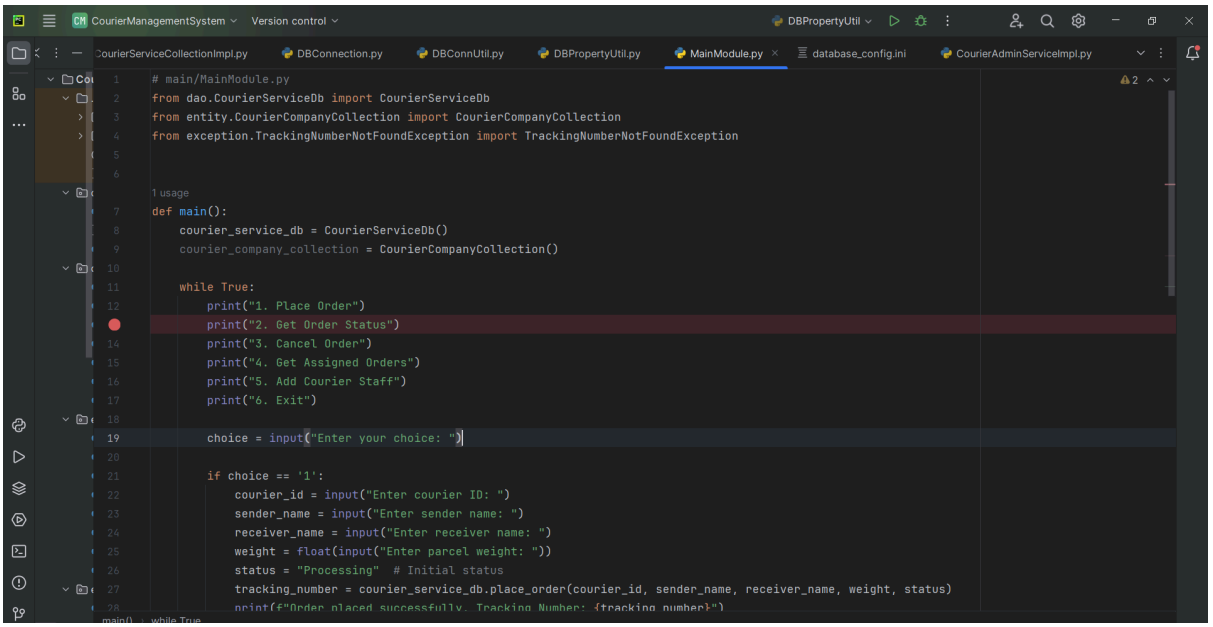


Task 9: Database Interaction Connect your application to the SQL database for the Courier Management System 1. Write code to establish a connection to your SQL database. Create a class DBConnection in a package connectionutil with a static variable connection of Type Connection and a static method getConnection() which returns connection. Connection properties supplied in the connection string should be read from a property file.
2. Create a Service class CourierServiceDb in dao with a static variable named connection of type Connection which can be assigned in the constructor by invoking the method in DBConnection Class.
3. Include methods to insert, update, and retrieve data from the database (e.g., inserting a new order, updating courier status). 4. Implement a feature to retrieve and display the delivery history of a specific parcel by querying the database. 1. Generate and display reports using data retrieved from the database (e.g., shipment status report, revenue report).

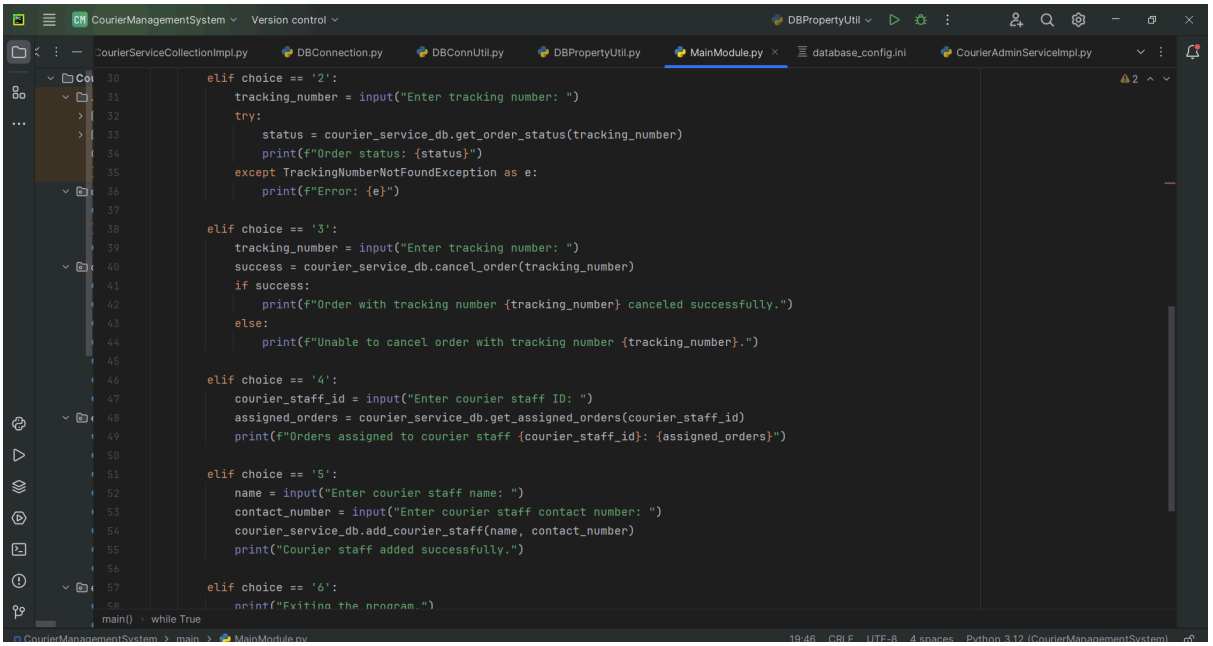




```python
from configparser import ConfigParser
from connectionutil import DBConnection


2 usages
class DBPropertyUtil:
    1 usage
    @staticmethod
    def get_db_properties():
        config = ConfigParser()
        config.read('database_config.ini')
        return {
            'host': config.get( section: 'database', option: 'host'),
            'user': config.get( section: 'database', option: 'user'),
            'password': config.get( section: 'database', option: 'password'),
            'database': config.get( section: 'database', option: 'database'),
            'port': config.getint( section: 'database', option: 'port'),
        }
```



```python
# main/MainModule.py
from dao.CourierServiceDb import CourierServiceDb
from entity.CourierCompanyCollection import CourierCompanyCollection
from exception.TrackingNumberNotFoundException import TrackingNumberNotFoundException


1 usage
def main():
    courier_service_db = CourierServiceDb()
    courier_company_collection = CourierCompanyCollection()

    while True:
        print("1. Place Order")
        print("2. Get Order Status")
        print("3. Cancel Order")
        print("4. Get Assigned Orders")
        print("5. Add Courier Staff")
        print("6. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            courier_id = input("Enter courier ID: ")
            sender_name = input("Enter sender name: ")
            receiver_name = input("Enter receiver name: ")
            weight = float(input("Enter parcel weight: "))
            status = "Processing"  # Initial status
            tracking_number = courier_service_db.place_order(courier_id, sender_name, receiver_name, weight, status)
            print(f"Order placed successfully. Tracking Number: {tracking_number}")
```



```python
        elif choice == '2':
            tracking_number = input("Enter tracking number: ")
            try:
                status = courier_service_db.get_order_status(tracking_number)
                print(f"Order status: {status}")
            except TrackingNumberNotFoundException as e:
                print(f"Error: {e}")

        elif choice == '3':
            tracking_number = input("Enter tracking number: ")
            success = courier_service_db.cancel_order(tracking_number)
            if success:
                print(f"Order with tracking number {tracking_number} canceled successfully.")
            else:
                print(f"Unable to cancel order with tracking number {tracking_number}.")

        elif choice == '4':
            courier_staff_id = input("Enter courier staff ID: ")
            assigned_orders = courier_service_db.get_assigned_orders(courier_staff_id)
            print(f"Orders assigned to courier staff {courier_staff_id}: {assigned_orders}")

        elif choice == '5':
            name = input("Enter courier staff name: ")
            contact_number = input("Enter courier staff contact number: ")
            courier_service_db.add_courier_staff(name, contact_number)
            print("Courier staff added successfully.")

        elif choice == '6':
            print("Exiting the program.")
```