

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.

The screenshot shows the PyCharm IDE interface with the 'CourierManagementSystem' project open. The 'OrderStatus.py' file is the active editor. The code defines a function 'check_order_status' that takes an order status as input and prints a message indicating whether it is delivered, processing, cancelled, or invalid. A terminal window below shows the script being run with the input 'Delivered'.

```
def check_order_status(order_status):
    if order_status == "Delivered":
        print("The order has been delivered.")
    elif order_status == "Processing":
        print("The order is still in processing.")
    elif order_status == "Cancelled":
        print("The order has been cancelled.")
    else:
        print("Invalid order status.")

# Example usage:
order_status_input = input("Enter the order status: ")
check_order_status(order_status_input)
```

```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\OrderStatus
Enter the order status: Delivered
```

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

The screenshot shows the PyCharm IDE interface with the 'CourierManagementSystem' project open. The 'CategorizeParcels.py' file is the active editor. The code defines a function 'categorize_parcel' that takes a weight as input and returns a category ('Light', 'Medium', or 'Heavy') based on the weight range. A terminal window below shows the script being run with the input '5'.

```
def categorize_parcel(weight):
    categories = {
        'Light': weight <= 1.8,
        'Medium': 2.5 <= weight < 3.5,
        'Heavy': weight >= 4.0
    }

    for category, condition in categories.items():
        if condition:
            return category

# Example usage:
parcel_weight_input = float(input("Enter the parcel weight: "))
categorize_parcel()
```

```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\CategorizeParcels
Enter the parcel weight: 5
The parcel is categorized as: Heavy

Process finished with exit code 0
```

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

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure under 'CourierManagementSyst'. The main editor window shows the 'UserAuthentication.py' file with the following code:

```
def authenticate_user(user_type, email, password):
    users = user_data.get(user_type, {})

    # Make email comparison case-insensitive and strip whitespaces
    email = email.lower().strip()

    if email in users:
        user = users[email]
        if user['password'] == password:
            return user
    return None

# Example usage:
user_type_input = input("Enter user type (employee/customer): ").lower()
email_input = input("Enter your email: ")
password_input = input("Enter your password: ")

authenticated_user = authenticate_user(user_type_input, email_input, password_input)

if email_input in user_data.get(user_type_input, {}) and authenticated_user is not None:
    print(f"Authentication successful. Welcome {authenticated_user['role']}!")
else:
    print("Authentication failed. Please check if the user type, email, and password are valid.")
```

The bottom terminal window shows the execution results:

```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\UserAuthentication.py
Enter user type (employee/customer): customer
Enter your email: astha.raj@gmail.com
Enter your password: customer123
Authentication successful. Welcome Customer!
```

The screenshot shows the PyCharm IDE interface. The left sidebar displays the project structure under 'CourierManagementSyst'. The main editor window shows the 'UserAuthentication.py' file with the following code:

```
def authenticate_user(user_type, email, password):
    users = user_data.get(user_type, {})

    # Make email comparison case-insensitive and strip whitespaces
    email = email.lower().strip()

    if email in users:
        user = users[email]
        if user['password'] == password:
            return user
    return None

# Example usage:
user_type_input = input("Enter user type (employee/customer): ").lower()
email_input = input("Enter your email: ")
password_input = input("Enter your password: ")

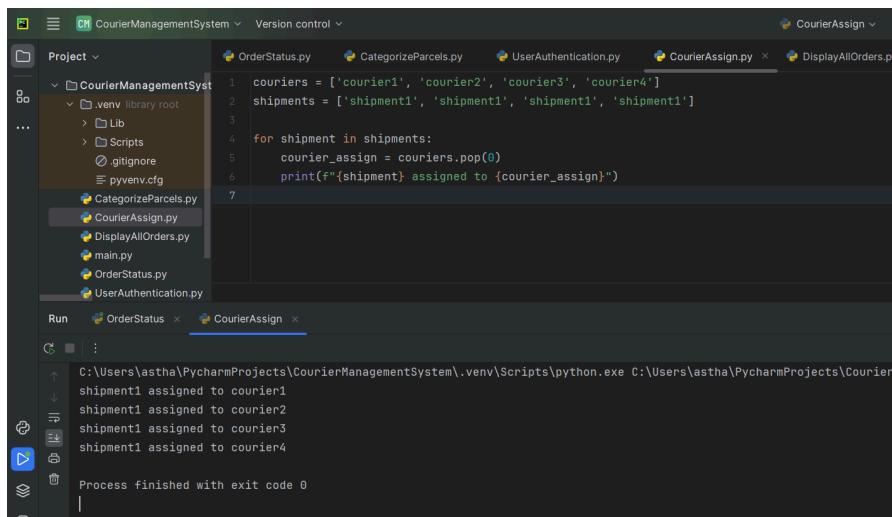
authenticated_user = authenticate_user(user_type_input, email_input, password_input)

if email_input in user_data.get(user_type_input, {}) and authenticated_user is not None:
    print(f"Authentication successful. Welcome {authenticated_user['role']}!")
else:
    print("Authentication failed. Please check if the user type, email, and password are valid.")
```

The bottom terminal window shows the execution results:

```
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\UserAuthentication.py
Enter user type (employee/customer): customer
Enter your email: astha.raj@gmail.com
Enter your password: customer123
Authentication successful. Welcome Customer!
```

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.



```
couriers = ['courier1', 'courier2', 'courier3', 'courier4']
shipments = ['shipment1', 'shipment1', 'shipment1', 'shipment1']

for shipment in shipments:
    courier_assign = couriers.pop(0)
    print(f"{shipment} assigned to {courier_assign}")
```

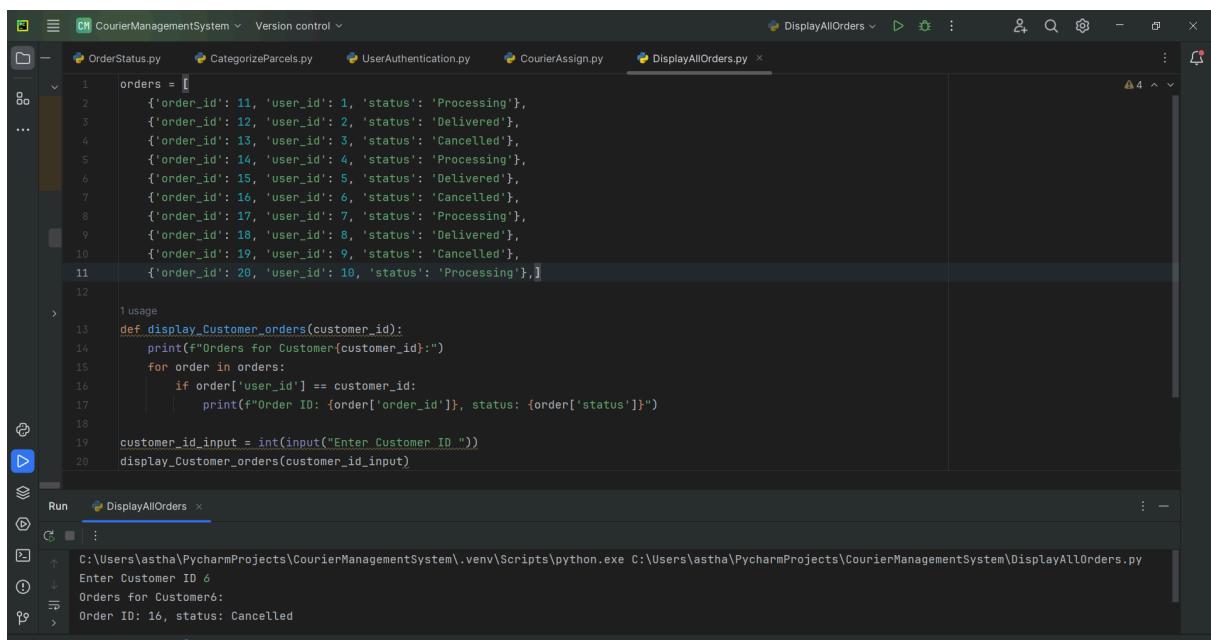
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\CourierAssign.py

shipment1 assigned to courier1
shipment1 assigned to courier2
shipment1 assigned to courier3
shipment1 assigned to courier4

Process finished with exit code 0

Task 2: Loops and Iteration

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



```
orders = [
    {'order_id': 1, 'user_id': 1, 'status': 'Processing'},
    {'order_id': 2, 'user_id': 2, 'status': 'Delivered'},
    {'order_id': 3, 'user_id': 3, 'status': 'Cancelled'},
    {'order_id': 4, 'user_id': 4, 'status': 'Processing'},
    {'order_id': 5, 'user_id': 5, 'status': 'Delivered'},
    {'order_id': 6, 'user_id': 6, 'status': 'Cancelled'},
    {'order_id': 7, 'user_id': 7, 'status': 'Processing'},
    {'order_id': 8, 'user_id': 8, 'status': 'Delivered'},
    {'order_id': 9, 'user_id': 9, 'status': 'Cancelled'},
    {'order_id': 10, 'user_id': 10, 'status': 'Processing'}
]

def display_Customer_orders(customer_id):
    print(f"Orders for Customer{customer_id}:")
    for order in orders:
        if order['user_id'] == customer_id:
            print(f"Order ID: {order['order_id']}, status: {order['status']}")

customer_id_input = int(input("Enter Customer ID "))
display_Customer_orders(customer_id_input)
```

Enter Customer ID 6

Orders for Customer6:

Order ID: 16, status: Cancelled

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

The screenshot shows a PyCharm interface with a dark theme. The code editor displays a Python script named 'track_courier.py'. The script contains a function 'track_courier' that prints a tracking message, updates a current location (randomly chosen from four locations), and checks if the courier has reached the destination. If so, it breaks the loop. It then sleeps for 2 seconds before prompting the user for a courier ID and calling the function again. The code uses f-strings and the random module.

```
1 usage
2
3
4
5     def track_courier(courier_id):
6         print(f"Tracking Courier {courier_id}")
7
8         while True:
9             current_location = random.choice(['Location1', 'Location2', 'Location3', 'Location4'])
10            print(f"Current Location: {current_location}")
11
12            if current_location == 'Location3':
13                print("Courier has reached the destination.")
14                break
15
16            time.sleep(2)
17
18
19 courier_id_input = input("Enter Courier ID: ")
20 track_courier(courier_id_input)
```

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.

The screenshot shows a PyCharm interface with a dark theme. The code editor displays a Python script named 'Task3_7.py'. It defines a class 'Parcel' with an __init__ method to initialize a parcel ID and an empty tracking history list. It also includes a location_update method to append a location to the tracking history. The script creates a 'parcel1' object, updates its location three times, and then prints the tracking history. The run tab at the bottom shows the command run and the output: 'Parcel 1 Tracking History: ['At the warehouse', 'Shipped', 'At Delivery Hub', 'Delivered']'.

```
1 usage
2
3
4
5
6     class Parcel:
7         def __init__(self, parcel_id):
8             self.parcel_id = parcel_id
9             self.tracking_history = []
10
11
12         def location_update(self, location):
13             self.tracking_history.append(location)
14
15
16 parcel1 = Parcel(parcel_id=1)
17 parcel1.location_update("At the warehouse")
18 parcel1.location_update("Shipped")
19 parcel1.location_update("At Delivery Hub")
20 parcel1.location_update("Delivered")
21
22 print(f"Parcel {parcel1.parcel_id} Tracking History: {parcel1.tracking_history}")
```

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

```

1 usages
2 class Courier:
3     def __init__(self, courier_id, current_location):
4         self.courier_id = courier_id
5         self.current_location = current_location
6         self.is_available = True
7
8     1 usage
9     def find_nearest_courier(order_location, couriers):
10        nearest_courier = None
11        min_distance = float('inf')
12
13        for courier in couriers:
14            distance = abs(ord(order_location) - ord(courier.current_location))
15
16            if courier.is_available and distance < min_distance:
17                min_distance = distance
18                nearest_courier = courier
19
20        return nearest_courier
21
22    # Example usage:
23    find_nearest_courier()  # for courier in couriers : if courier.is_available and dis...

```

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.

```

1 # Sample 2D array for parcel tracking (replace with actual data in a real-world scenario)
2 parcel_tracking_array = [
3     ['123456', 'Parcel in transit'],
4     ['789012', 'Parcel out for delivery'],
5     ['345678', 'Parcel delivered']
6 ]
7
8 1 usage
9 def track_parcel(parcel_number):
10     for tracking_info in parcel_tracking_array:
11         if tracking_info[0] == parcel_number:
12             return tracking_info[1]
13     return "Parcel not found"
14
15 1 usage
16 def simulate_tracking_process():
17     parcel_number_input = input("Enter parcel tracking number: ")
18     status = track_parcel(parcel_number_input)

```

L:\users\astha\pycharmprojects\couriermanagementsystem\.venv\scripts\python.exe L:\users\astha\pycharmprojects\couriermanagementsystem\task4_9.py

Enter parcel tracking number: 123456

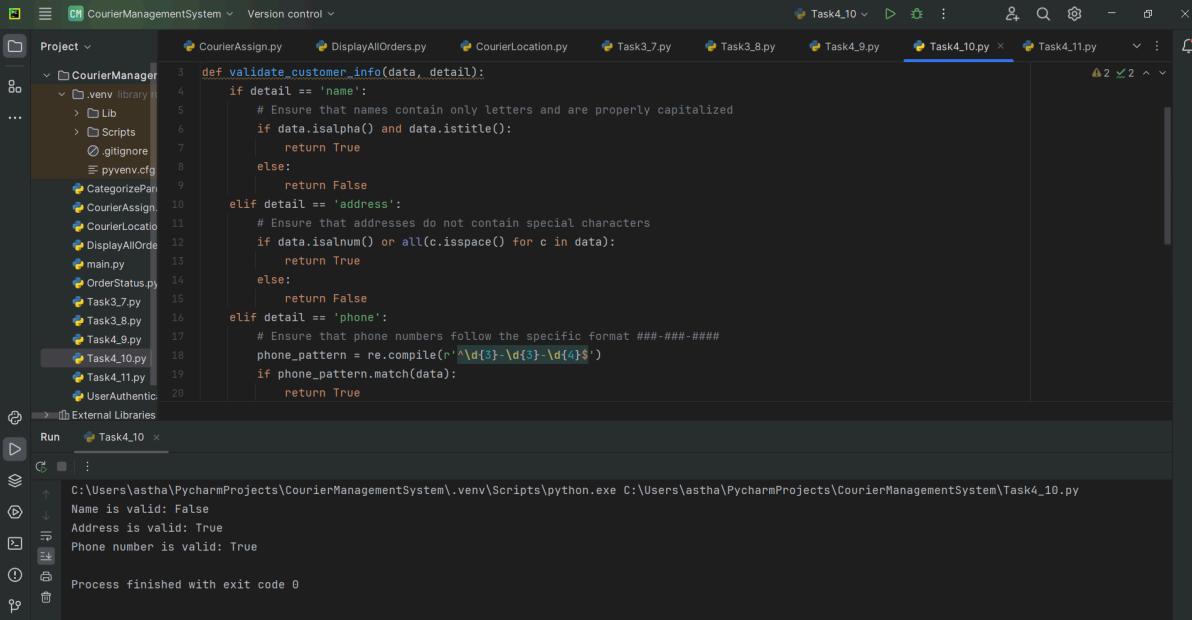
Tracking number: 123456

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 address 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., ####-####-####).



```

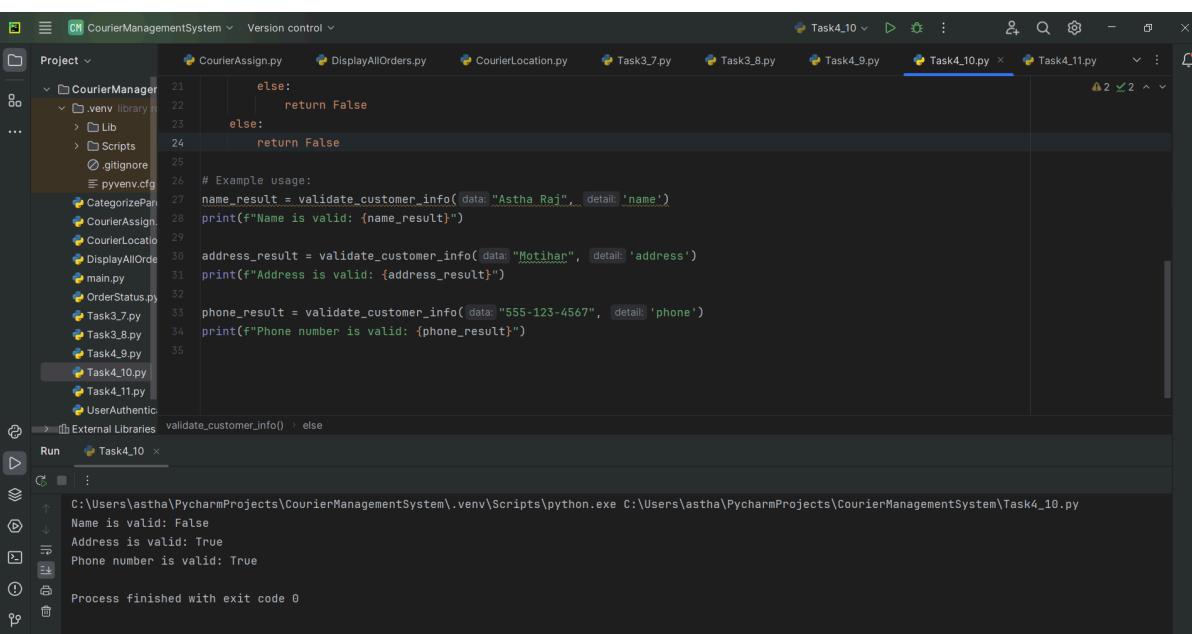
def validate_customer_info(data, detail):
    if detail == 'name':
        # Ensure that names contain only letters and are properly capitalized
        if data.isalpha() and data.istitle():
            return True
        else:
            return False
    elif detail == 'address':
        # Ensure 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
        else:
            return False

```

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



```

def validate_customer_info(data, detail):
    if detail == 'name':
        return True
    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}")

```

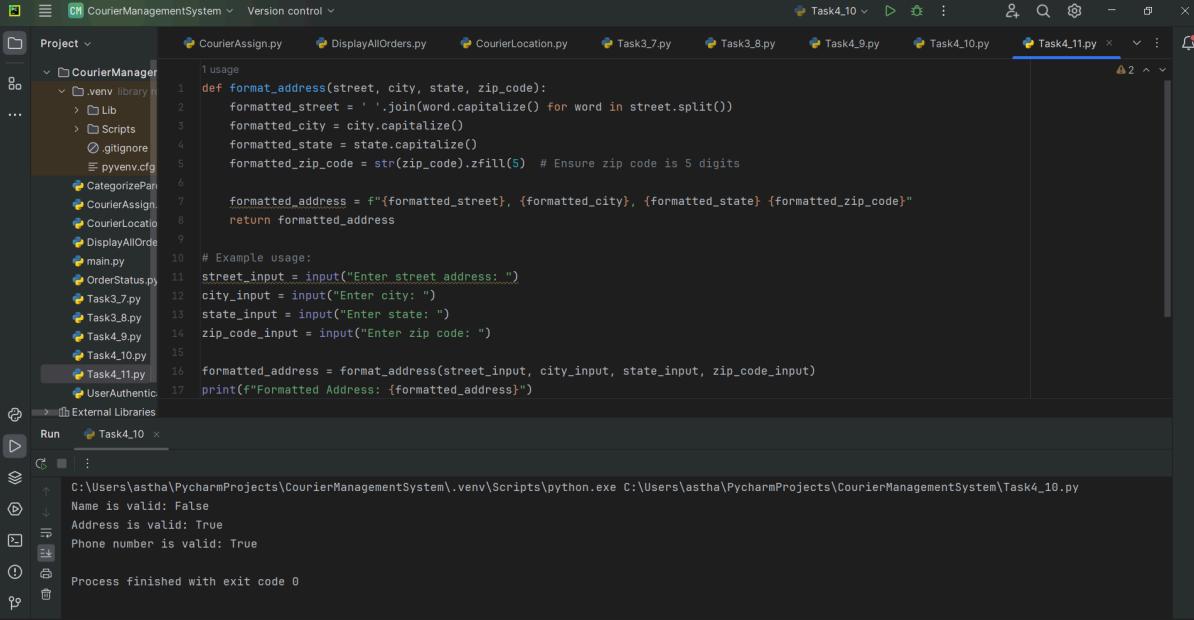
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_10.py

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)

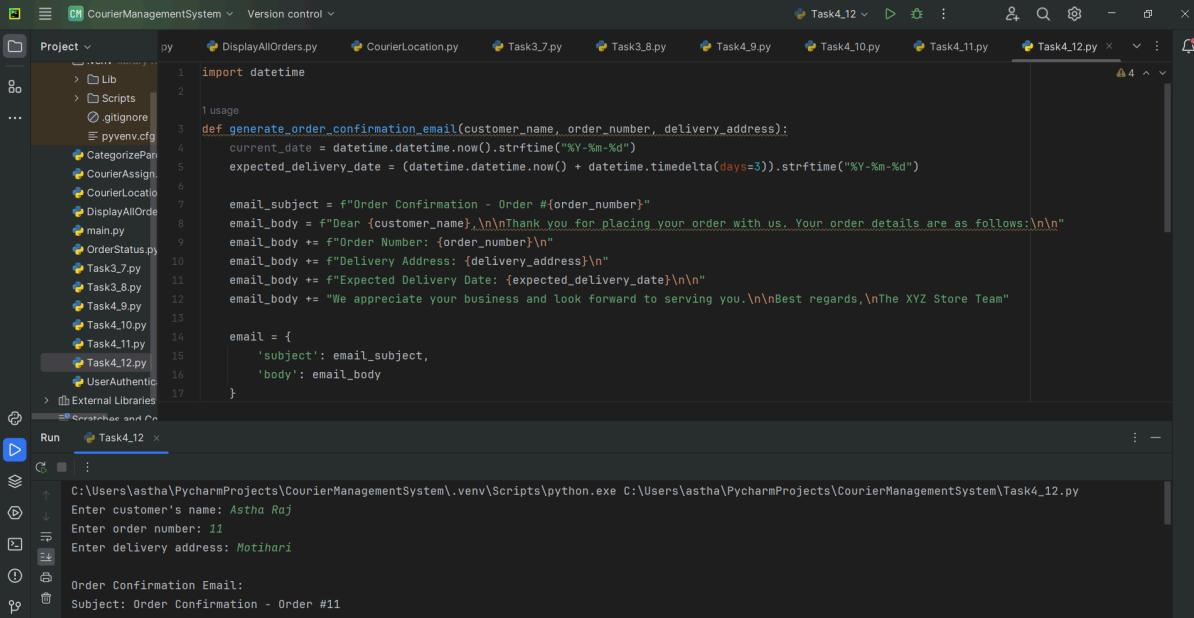
and formats it correctly, including capitalizing the first letter of each word and properly formatting the zip code.



```
usage
1 def format_address(street, city, state, zip_code):
2     formatted_street = ' '.join(word.capitalize() for word in street.split())
3     formatted_city = city.capitalize()
4     formatted_state = state.capitalize()
5     formatted_zip_code = str(zip_code).zfill(5) # Ensure zip code is 5 digits
6
7     formatted_address = f"{formatted_street}, {formatted_city}, {formatted_state} {formatted_zip_code}"
8     return formatted_address
9
10 # Example usage:
11 street_input = input("Enter street address: ")
12 city_input = input("Enter city: ")
13 state_input = input("Enter state: ")
14 zip_code_input = input("Enter zip code: ")
15
16 formatted_address = format_address(street_input, city_input, state_input, zip_code_input)
17 print(f"Formatted Address: {formatted_address}")

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
```

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.



```
import datetime

1 usage
2
3 def generate_order_confirmation_email(customer_name, order_number, delivery_address):
4     current_date = datetime.datetime.now().strftime("%Y-%m-%d")
5     expected_delivery_date = (datetime.datetime.now() + datetime.timedelta(days=3)).strftime("%Y-%m-%d")
6
7     email_subject = f"Order Confirmation - Order #{order_number}"
8     email_body = f"Dear {customer_name},\n\nThank you for placing your order with us. Your order details are as follows:\n\n"
9     email_body += f"Order Number: {order_number}\n"
10    email_body += f"Delivery Address: {delivery_address}\n"
11    email_body += f"Expected Delivery Date: {expected_delivery_date}\n\n"
12    email_body += f"We appreciate your business and look forward to serving you.\n\nBest regards,\nThe XYZ Store Team"
13
14    email = {
15        'subject': email_subject,
16        'body': email_body
17    }

C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_12.py
Enter customer's name: Astha Raj
Enter order number: 11
Enter delivery address: Motihari
Order Confirmation Email:
Subject: Order Confirmation - Order #11
```

The screenshot shows the PyCharm IDE interface with the project 'CourierManagementSystem' open. The code editor displays Task4_12.py, which contains Python code for generating an order confirmation email. The run output window shows the generated email content:

```
Dear Astha Raj,  
Thank you for placing your order with us. Your order details are as follows:  
Order Number: 11  
Delivery Address: Motihari  
Expected Delivery Date: 2024-02-07
```

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.

The screenshot shows the PyCharm IDE interface with the project 'CourierManagementSystem' open. The code editor displays Task4_13.py, which contains Python code for calculating shipping costs. The run output window shows the execution results:

```
Enter Source Address: Motihari  
Enter Destination Address: Delhi  
Enter Parcel Wt : 4.2  
Shipping Cost : 19.5  
Process finished with exit code 0
```

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.

The screenshot shows the PyCharm IDE interface with the project 'CourierManagementSystem' open. The code editor displays Task4_14.py, which contains a function to generate a password. A PEP 8 error is shown in the status bar. The terminal below shows the execution of the script and its output.

```

import random
import string

def generate_password():
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(10))
    return password

generated_password = generate_password()
print("Generated Password: ", generated_password)

```

C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_14.py
Generated Password: svx<?9F7^
Process finished with exit code 0

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.

The screenshot shows the PyCharm IDE interface with the project 'CourierManagementSystem' open. The code editor displays Task4_15.py, which contains a function to find similar addresses. The terminal below shows the execution of the script and its output.

```

def find_similar_addresses(address, address_list):
    similar_addresses = [x for x in address_list if x.lower().startswith(address.lower())]
    return similar_addresses

address_to_find = input("Enter address to find: ")
address_list = ['123 Motihari', '234 Delhi', '123 Motihari', '55 Bengal']
similar_addresses = find_similar_addresses(address_to_find, address_list)
print("Similar_ADDRESS : ", similar_addresses)

```

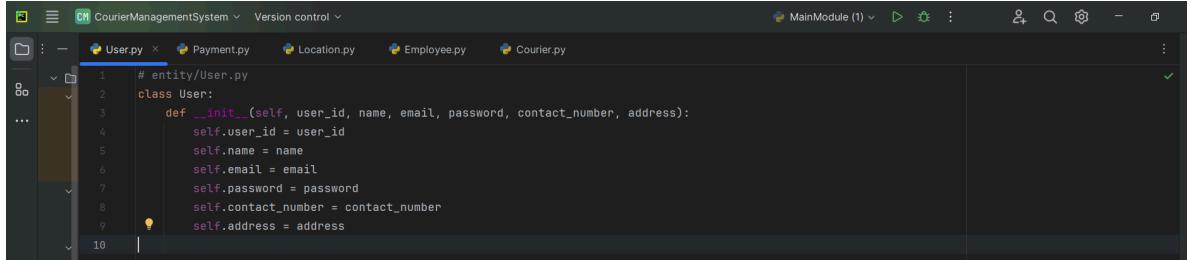
C:\Users\astha\PycharmProjects\CourierManagementSystem\.venv\Scripts\python.exe C:\Users\astha\PycharmProjects\CourierManagementSystem\Task4_15.py
Enter address to find: 123 Motihari
Similar_ADDRESS : ['123 Motihari']
Process finished with exit code 0

Task 5: Object Oriented Programming Scope :

Entity classes/Models/POJO, Abstraction/Encapsulation

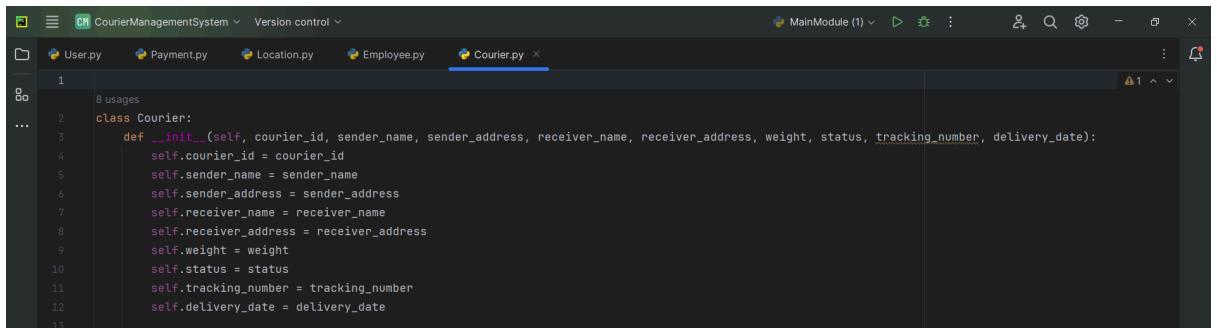
Create the following model/entity classes within package entities with variables declared private, constructors(default and parametrized),getters, setters and `toString()`

1. User Class: Variables: `userID` , `userName` , `email` , `password` , `contactNumber` , `address`



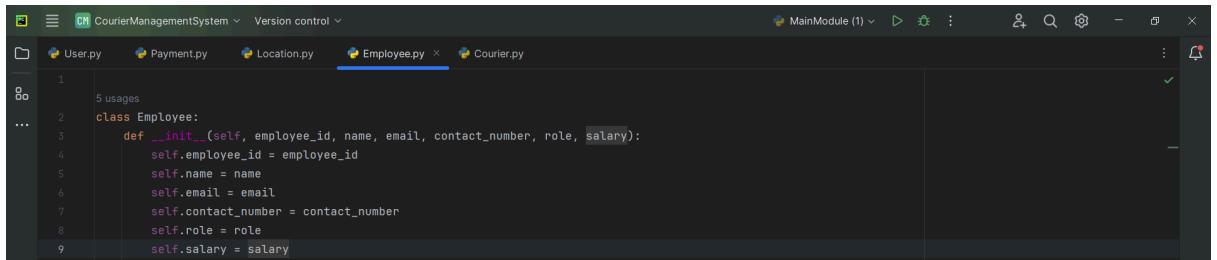
```
# entity/User.py
class User:
    def __init__(self, user_id, name, email, password, contact_number, address):
        self.user_id = user_id
        self.name = name
        self.email = email
        self.password = password
        self.contact_number = contact_number
        self.address = address
```

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



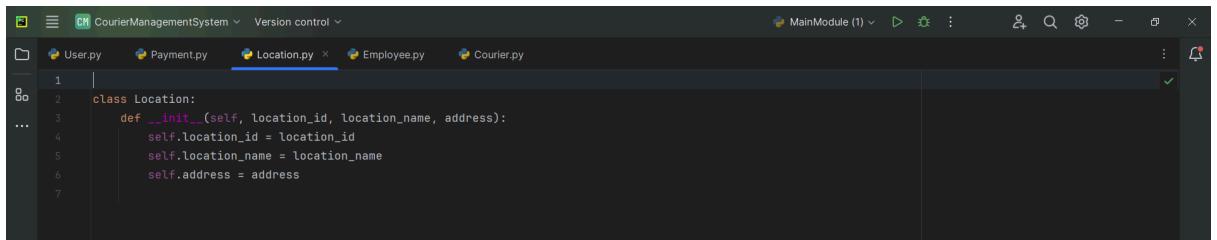
```
8 usages
class Courier:
    def __init__(self, courier_id, sender_name, sender_address, receiver_name, receiver_address, weight, status, tracking_number, delivery_date):
        self.courier_id = courier_id
        self.sender_name = sender_name
        self.sender_address = sender_address
        self.receiver_name = receiver_name
        self.receiver_address = receiver_address
        self.weight = weight
        self.status = status
        self.tracking_number = tracking_number
        self.delivery_date = delivery_date
```

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



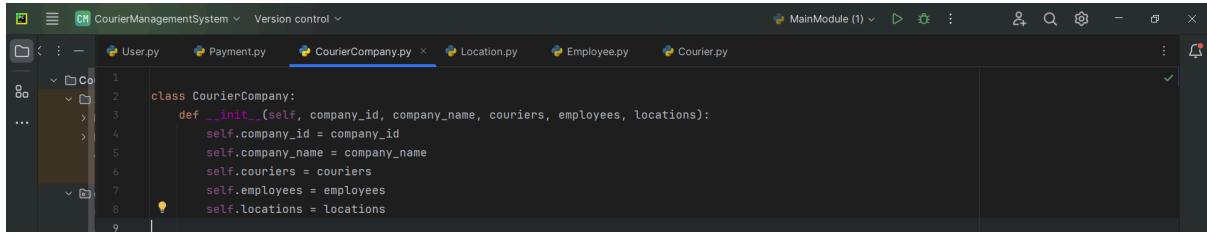
```
5 usages
class Employee:
    def __init__(self, employee_id, name, email, contact_number, role, salary):
        self.employee_id = employee_id
        self.name = name
        self.email = email
        self.contact_number = contact_number
        self.role = role
        self.salary = salary
```

4. Location Class Variables LocationID , LocationName , Address



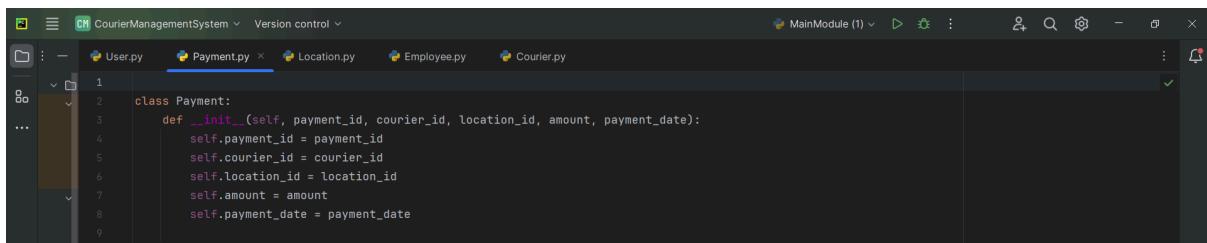
```
class location:
    def __init__(self, location_id, location_name, address):
        self.location_id = location_id
        self.location_name = location_name
        self.address = address
```

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



```
1 class CourierCompany:
2     def __init__(self, company_id, company_name, couriers, employees, locations):
3         self.company_id = company_id
4         self.company_name = company_name
5         self.couriers = couriers
6         self.employees = employees
7         self.locations = locations
```

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



```
1 class Payment:
2     def __init__(self, payment_id, courier_id, location_id, amount, payment_date):
3         self.payment_id = payment_id
4         self.courier_id = courier_id
5         self.location_id = location_id
6         self.amount = amount
7         self.payment_date = payment_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.*/

```
from abc import ABC, abstractmethod

class ICourierUserService(ABC):
    @abstractmethod
    def place_order(self, courier_obj):
        pass

    @abstractmethod
    def get_order_status(self, tracking_number):
        pass

    @abstractmethod
    def cancel_order(self, tracking_number):
        pass

    @abstractmethod
    def get_assigned_orders(self, courier_staff_id):
        pass

    2 usages (2 dynamic)
    @abstractmethod
    def add_courier_staff(self, name, contact_number):
        pass
```

// 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. */

```
from abc import ABC, abstractmethod

class ICourierAdminService(ABC):
    @abstractmethod
    def place_order(self, courier_obj):
        pass

    @abstractmethod
    def get_order_status(self, tracking_number):
        pass

    @abstractmethod
    def cancel_order(self, tracking_number):
        pass

    @abstractmethod
    def get_assigned_orders(self, courier_staff_id):
        pass

    2 usages (2 dynamic)
    @abstractmethod
    def add_courier_staff(self, name, contact_number):
        pass
```

Task 7: Exception Handling

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch finally, throw & throws keyword usage)

Define the following custom exceptions and throw them in methods whenever needed . Handle all the exceptions 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

```

1
2     class InvalidEmployeeIdException(Exception):
3         pass

```



```

10 usages
1
2     class TrackingNumberNotFoundException(Exception):
3         pass

```

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

```

1
2     3 usages
3     class CourierCompanyCollection:
4         def __init__(self):
5             self.courier_companies = []
6
7         def add_courier_company(self, courier_company):
8             self.courier_companies.append(courier_company)
9
10        def get_courier_companies(self):
11            return self.courier_companies
12
13        4 usages (4 dynamic)
14        def add_courier(self, courier):
15            for company in self.courier_companies:
16                if company.get_company_id() == courier.get_company_id():
17                    company.add_courier(courier)
18
19        3 usages (3 dynamic)
20        def add_courier_staff(self, courier_staff):
21            for company in self.courier_companies:
22                if company.get_company_id() == courier_staff.get_company_id():
23                    company.add_courier_staff(courier_staff)
24
25        9 usages (9 dynamic)
26        def get_couriers(self):
27            all_couriers = []
28            for company in self.courier_companies:
29                all_couriers.extend(company.get_couriers())

```

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

```

CourierManagementSystem Version control MainModule (1) ...
/CourierServiceCollectionImpl.py
1   from dao.CourierUserServiceImpl import CourierUserServiceImpl
2   from entity.Courier import Courier
3   from entity.Employee import Employee
4   from exception.TrackingNumberNotFoundException import TrackingNumberNotFoundException
5   import datetime
6
7   class CourierServiceCollectionImpl(CourierUserServiceImpl):
8       def __init__(self, company_obj):
9           super().__init__(company_obj)
10
11      def place_order(self, sender_name, receiver_name, weight):
12          courier_id = self.generate_courier_id()
13          status = "Processing"
14          tracking_number = super().place_order(courier_id, sender_name, receiver_name, weight, status)
15
16          courier = Courier(courier_id, sender_name, None, receiver_name, weight, status, tracking_number, None)
17          self.company_obj.add_courier(courier)
18
19          return tracking_number
20
21      1 usage
22      def generate_courier_id(self):
23          return len(self.company_obj.get_couriers()) + 1
24
25      2 usages
26      def cancel_order(self, tracking_number):
27          courier = self.find_courier_by_tracking_number(tracking_number)
28
29          if courier:
30              courier.status = "Cancelled"
31              return True
32          else:
33              return False
34
35      def get_assigned_orders(self, courier_staff_id):
36          assigned_orders = [courier for courier in self.company_obj.get_couriers() if courier.courier_id == courier_staff_id]
37          return assigned_orders
38
39      2 usages
40      def find_courier_by_tracking_number(self, tracking_number):
41          for courier in self.company_obj.get_couriers():
42              if courier.tracking_number == tracking_number:
43                  return courier
44          return None
45
46      3 usages (3 dynamic)
47      def add_courier_staff(self, name, contact_number):
48          courier_staff_id = self.generate_courier_staff_id()
49          employee = Employee(courier_staff_id, name, contact_number, contact_number, "Courier Driver", role=0)
50
51          self.company_obj.add_courier_staff(employee)
52
53          return courier_staff_id
54
55      1 usage
56      def generate_courier_staff_id(self):
57          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.

The image shows two screenshots of a code editor interface, likely PyCharm, displaying Python code for a Courier Management System.

CourierUserServiceImpl.py:

```
4 usages
class CourierUserServiceImpl(ICourierUserService):
    def __init__(self, company_obj):
        self.company_obj = company_obj

    2 usages
    def place_order(self, courier_id, sender_name, receiver_name, weight, status):
        # Implement logic to place a new courier order
        tracking_number = f'{datetime.datetime.now().strftime("%Y%m%d%H%M%S")}{courier_id}'
        delivery_date = datetime.date.today() + datetime.timedelta(days=3) # Placeholder for delivery date logic
        courier = Courier(courier_id, sender_name, sender_address=None, receiver_name, receiver_address=None, weight, status, tracking_number, delivery_date)
        self.company_obj.add_courier(courier)
        return tracking_number

    def get_order_status(self, tracking_number):
        # Implement logic to get the status of a courier order
        courier = self.find_courier_by_tracking_number(tracking_number)

        if courier:
            return courier.status
        else:
            raise TrackingNumberNotFoundException(f"Tracking number {tracking_number} not found.")

    def cancel_order(self, tracking_number):
        # Implement logic to cancel a courier order
        courier = self.find_courier_by_tracking_number(tracking_number)

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

    2 usages
    def find_courier_by_tracking_number(self, tracking_number):
        # Helper method to find a courier by tracking number
        for courier in self.company_obj.get_couriers():
            if courier.tracking_number == tracking_number:
                return courier
        return None
```

CourierAdminServiceImpl.py:

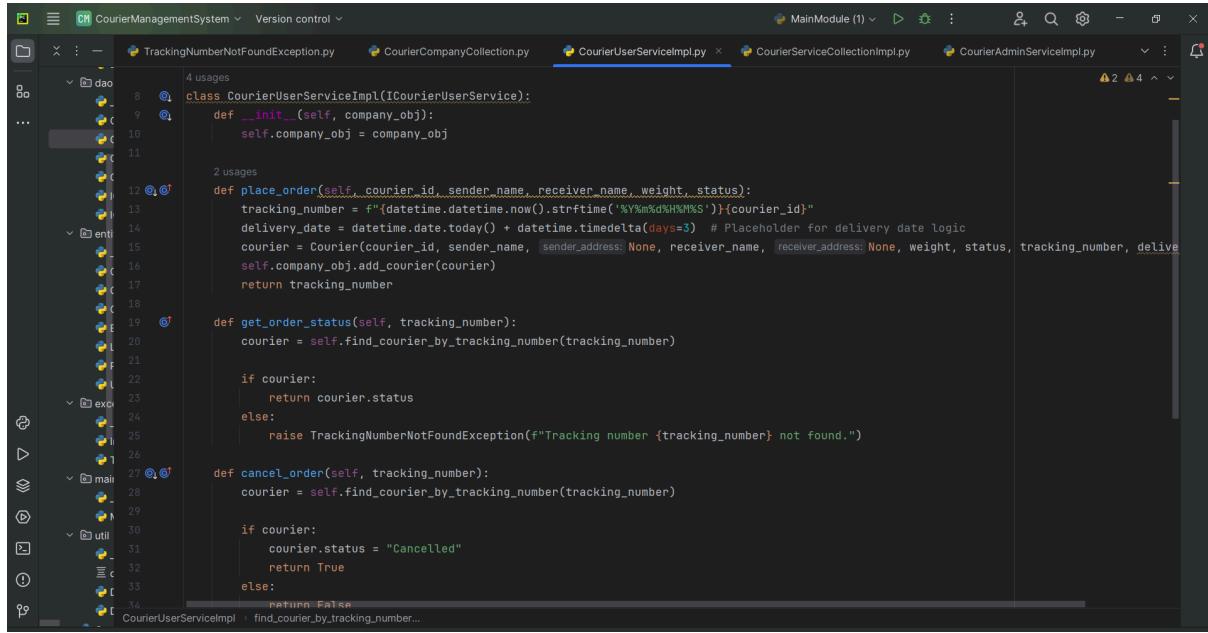
```
29 @ ③
def cancel_order(self, tracking_number):
    # Implement logic to cancel a courier order
    courier = self.find_courier_by_tracking_number(tracking_number)

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

39 @ ③
def get_assigned_orders(self, courier_staff_id):
    # Implement logic to get a list of orders assigned to a specific courier staff member
    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):
    # Helper method to find a courier by tracking number
    for courier in self.company_obj.get_couriers():
        if courier.tracking_number == tracking_number:
            return courier
    return None
```

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



```
class CourierUserServiceImpl(ICourierUserService):
    def __init__(self, company_obj):
        self.company_obj = company_obj

    def place_order(self, courier_id, sender_name, receiver_name, weight, status):
        tracking_number = f"{datetime.datetime.now().strftime('%Y%m%d%H%M%S')}{courier_id}"
        delivery_date = datetime.date.today() + datetime.timedelta(days=3) # Placeholder for delivery date logic
        courier = Courier(courier_id, sender_name, None, receiver_name, None, weight, status, tracking_number, delivery_date)
        self.company_obj.add_courier(courier)
        return tracking_number

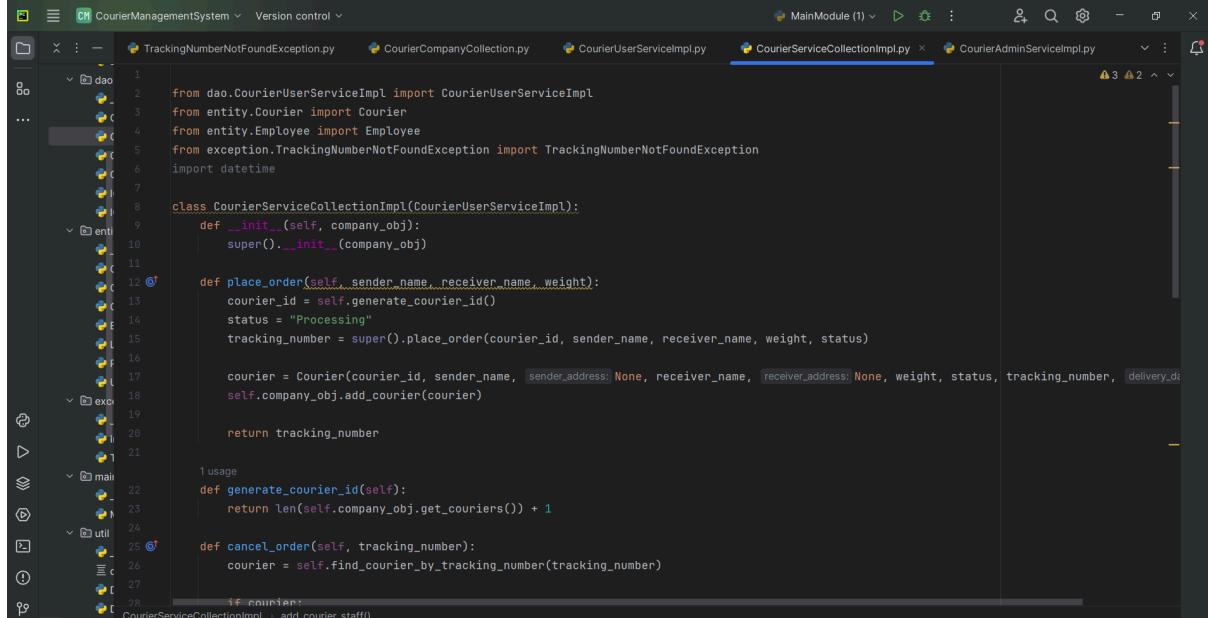
    def get_order_status(self, tracking_number):
        courier = self.find_courier_by_tracking_number(tracking_number)

        if courier:
            return courier.status
        else:
            raise TrackingNumberNotFoundException(f"Tracking number {tracking_number} not found.")

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

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

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



```
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, None, receiver_name, None, weight, status, tracking_number, None)
        self.company_obj.add_courier(courier)

        return tracking_number

    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:
            CourierServiceCollectionImpl._add_courier_staff()
```

```

24     def cancel_order(self, tracking_number):
25         courier = self.find_courier_by_tracking_number(tracking_number)
26
27         if courier:
28             courier.status = "Cancelled"
29             return True
30         else:
31             return False
32
33     def get_assigned_orders(self, courier_staff_id):
34         assigned_orders = [courier for courier in self.company_obj.get_couriers() if courier.courier_id == courier_staff_id]
35
36         return assigned_orders
37
38     2 usages
39     def find_courier_by_tracking_number(self, tracking_number):
40         for courier in self.company_obj.get_couriers():
41             if courier.tracking_number == tracking_number:
42                 return courier
43
44     3 usages (3 dynamic)
45     def add_courier_staff(self, name, contact_number):
46         courier_staff_id = self.generate_courier_staff_id()
47         employee = Employee(courier_staff_id, name, contact_number, contact_number="Courier Driver", role=0)
48         self.company_obj.add_courier_staff(employee)
49
50
51     1 usage
52     CourierServiceCollectionImpl.add_courier_staff()

```

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).

```

1
2     import mysql.connector
3
4     from util.DBPropertyUtil import DBPropertyUtil
5
6     class DBConnUtil:
7         @staticmethod
8         def get_connection():
9             try:
10                 db_properties = DBPropertyUtil.get_db_properties()
11                 connection = mysql.connector.connect(**db_properties)
12                 if connection.is_connected():
13                     print("Connected to the database")
14
15             return connection
16
17
18         except mysql.connector.Error as e:
19             print(f"Error: {e}")
20             raise e

```

DBPropertyUtil.py

```

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'),
        }

```

MainModule.py

```

# 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}")

    main() -> while True

```

MainModule.py (Continued)

```

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.")

main() -> while True

```