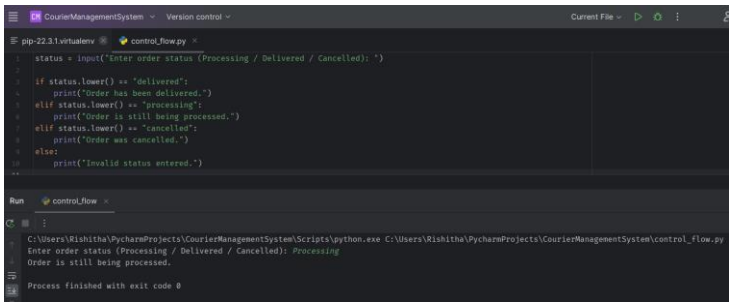
COURIER MANAGEMENT SYSTEM

ASSIGNMENT-02

Coding Task 1: Control Flow Statements

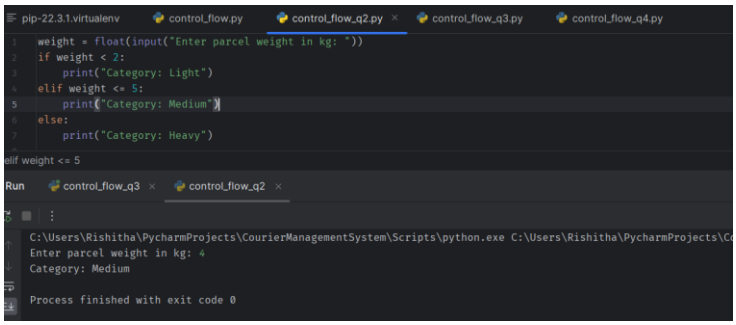
1.Write a program that checks whether a given order is delivered or not based on its status (e.g., "Processing," "Delivered," "Cancelled"). Use if-else statements for this.

```
status = input("Enter order status (Processing / Delivered / Cancelled): ")
if status.lower() == "delivered":
    print("Order has been delivered.")
elif status.lower() == "processing":
    print("Order is still being processed.")
elif status.lower() == "cancelled":
    print("Order was cancelled.")
else:
    print("Invalid status entered.")
```



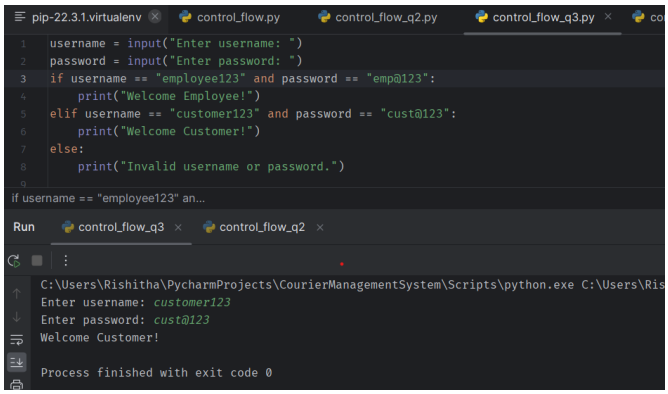
2. Implement a switch-case statement to categorize parcels based on their weight into "Light," "Medium," or "Heavy."

```
weight = float(input("Enter parcel weight in kg: "))
if weight < 2:
    print("Category: Light")
elif weight <= 5:
    print("Category: Medium")
else:
    print("Category: Heavy")
```



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

```
username = input("Enter username: ")
password = input("Enter password: ")
if username == "employee123" and password == "emp@123":
    print("Welcome Employee!")
elif username == "customer123" and password == "cust@123":
    print("Welcome Customer!")
else:
    print("Invalid username or password.")
```



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 = [
    {"name": "Courier A", "capacity": 3},
    {"name": "Courier B", "capacity": 5},
    {"name": "Courier C", "capacity": 2}
]
required_capacity = int(input("Enter required load capacity: "))
assigned = False
for courier in couriers:
    if courier["capacity"] >= required_capacity:
        print("Assigned to", courier["name"])
        assigned = True
        break
if not assigned:
    print("No courier available with enough capacity.")
```

```python
couriers = [
    {"name": "Courier A", "capacity": 3},
    {"name": "Courier B", "capacity": 5},
    {"name": "Courier C", "capacity": 2}
]
required_capacity = int(input("Enter required load capacity: "))
assigned = False
for courier in couriers:
    if courier["capacity"] >= required_capacity:
        print("Assigned to", courier["name"])
        assigned = True
        break
if not assigned:
    print("No courier available with enough capacity.")
```

```
Run   control_flow_q3   control_flow_q2
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users
Enter username: customer123
Enter password: cust@123
Welcome Customer!

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.

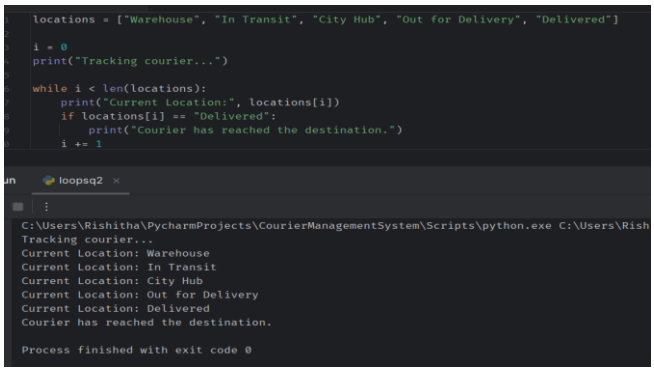
```python
orders = [
    {"order_id": 1, "user_id": 101, "tracking_no": "TRK001"},
    {"order_id": 2, "user_id": 102, "tracking_no": "TRK002"},
    {"order_id": 3, "user_id": 101, "tracking_no": "TRK003"}
]
user = int(input("Enter user ID to view orders: "))
print("Orders placed by user", user, ":")
found = False
for o in orders:
    if o["user_id"] == user:
        print("Order ID:", o["order_id"], "- Tracking:", o["tracking_no"])
        found = True
```

```python
orders = [
    {"order_id": 1, "user_id": 101, "tracking_no": "TRK001"},
    {"order_id": 2, "user_id": 102, "tracking_no": "TRK002"},
    {"order_id": 3, "user_id": 101, "tracking_no": "TRK003"}
]

user = int(input("Enter user ID to view orders: "))

print("Orders placed by user", user, ":")
found = False
for o in orders:
    if o["user_id"] == user:
        print("Order ID:", o["order_id"], "- Tracking:", o["tracking_no"])
        found = True
```

```
n   loopsq1   control_flow_q2
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects
Enter user ID to view orders: 101
Orders placed by user 101 :
Order ID: 1 - Tracking: TRK001
Order ID: 3 - Tracking: TRK003

Process finished with exit code 0
```

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

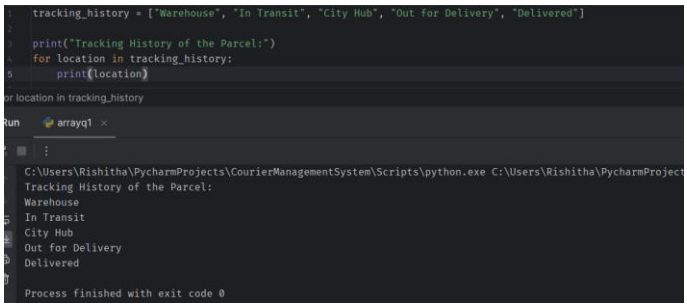
```
locations = ["Warehouse", "In Transit", "City Hub", "Out for Delivery", "Delivered"]
i = 0
print("Tracking courier...")
while i < len(locations):
    print("Current Location:", locations[i])
    if locations[i] == "Delivered":
        print("Courier has reached the destination.")
    i += 1
```



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.

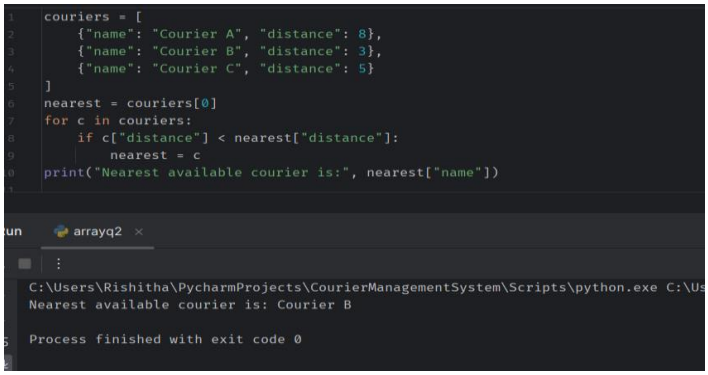
```
tracking_history = ["Warehouse", "In Transit", "City Hub", "Out for Delivery", "Delivered"]
print("Tracking History of the Parcel:")
for location in tracking_history:
    print(location)
```



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

```
couriers = [
    {"name": "Courier A", "distance": 8},
    {"name": "Courier B", "distance": 3},
    {"name": "Courier C", "distance": 5}
]
```

```
nearest = couriers[0]
for c in couriers:
    if c["distance"] < nearest["distance"]:
        nearest = c
print("Nearest available courier is:", nearest["name"])
```



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.

```
tracking_info = [
    ["TRK001", "In Transit"],
    ["TRK002", "Out for Delivery"],
    ["TRK003", "Delivered"]
]
track = input("Enter your parcel tracking number: ")
found = False
for item in tracking_info:
    if item[0] == track:
        status = item[1]
        if status == "In Transit":
            print("Parcel is in transit.")
        elif status == "Out for Delivery":
            print("Parcel is out for delivery.")
        elif status == "Delivered":
            print("Parcel has been delivered.")
        found = True
        break
if not found:
    print("Tracking number not found.")
```

```python
tracking_info = [
    ["TRK001", "In Transit"],
    ["TRK002", "Out for Delivery"],
    ["TRK003", "Delivered"]
]

track = input("Enter your parcel tracking number: ")
found = False

for item in tracking_info:
    if item[0] == track:
        status = item[1]
        if status == "In Transit":
            print("Parcel is in transit.")
        elif status == "Out for Delivery":
            print("Parcel is out for delivery.")
        elif status == "Delivered":
            print("Parcel has been delivered.")
        found = True
        break

if not found:
    print("Tracking number not found.")
```

```
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\
Enter your parcel tracking number: TRK003
Parcel has been delivered.

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

```python
import re
def validate(data, detail):
    if detail == "name":
        if data.isalpha() and data.istitle():
            print("Valid name.")
        else:
            print("Invalid name.")

    elif detail == "address":
        if re.match("^[a-zA-Z0-9\s,.-]+$", data):
            print("Valid address.")
        else:
            print("Invalid address.")
    elif detail == "phone":
        if re.match("^\d{3}-\d{3}-\d{4}$", data):
            print("Valid phone number.")
        else:
            print("Invalid phone number.")
    else:
        print("Unknown detail type.")
validate("Rishitha", "name")
validate("Plot 12, MG Road", "address")
validate("123-456-7890", "phone")
```

```python
import re
def validate(data, detail):
    if detail == "name":
        if data.isalpha() and data.istitle():
            print("Valid name.")
        else:
            print("Invalid name.")

    elif detail == "address":
        if re.match( pattern: "^[a-zA-Z0-9\s,.-]+$", data):
            print("Valid address.")
        else:
            print("Invalid address.")

    elif detail == "phone":
        if re.match( pattern: "^\d{3}-\d{3}-\d{4}$", data):
            print("Valid phone number.")
        else:
            print("Invalid phone number.")
    else:
        print("Unknown detail type.")
validate( data: "Rishitha",  detail: "name")
validate( data: "Plot 12, MG Road",  detail: "address")
validate( data: "123-456-7890",  detail: "phone")
```

```
10 ×
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users
Valid name.
Valid address.
Valid phone number.

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.

```python
def format_address(street, city, state, zip_code):
    if not zip_code.isdigit() or len(zip_code) != 6:
        print("Invalid zip code.")
        return
    street = street.title()
    city = city.title()
    state = state.upper()
    full_address = f"{street}, {city}, {state} - {zip_code}"
    print("Formatted Address:", full_address)
format_address("12 mg road", "hyderabad", "telangana", "500001")
```

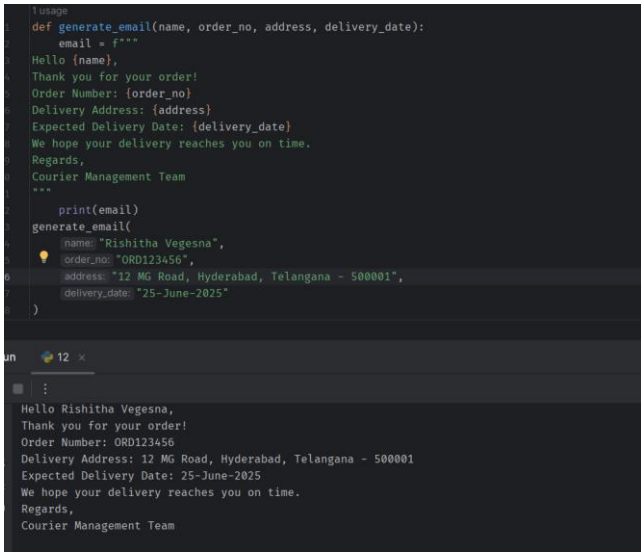
```python
def format_address(street, city, state, zip_code):
    if not zip_code.isdigit() or len(zip_code) != 6:
        print("Invalid zip code.")
        return

    street = street.title()
    city = city.title()
    state = state.upper()

    full_address = f"{street}, {city}, {state} - {zip_code}"
    print("Formatted Address:", full_address)
format_address( street: "12 mg road",  city: "hyderabad",  state: "telangana",  zip_code: "500001")
```

```
11 ×
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects
Formatted Address: 12 Mg Road, Hyderabad, TELANGANA - 500001

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.

```python
def generate_email(name, order_no, address, delivery_date):
    email = f"""
Hello {name},
Thank you for your order!
```

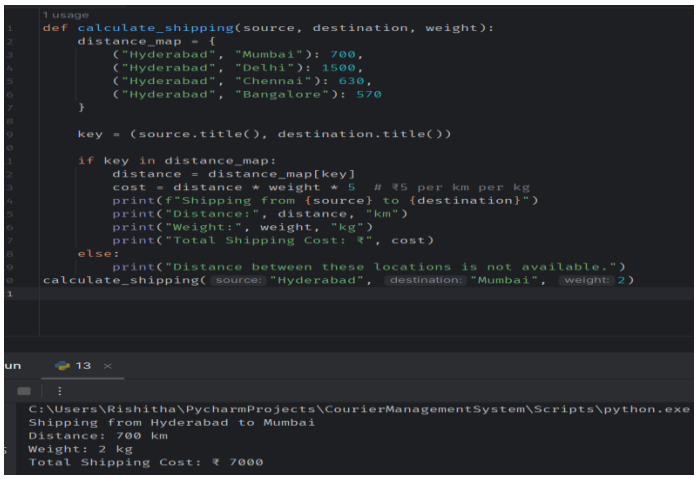
```
Order Number: {order_no}
Delivery Address: {address}
Expected Delivery Date: {delivery_date}
We hope your delivery reaches you on time.
Regards,
Courier Management Team
"""
    print(email)
generate_email(
    "Rishitha Vegesna",
    "ORD123456",
    "12 MG Road, Hyderabad, Telangana - 500001",
    "25-June-2025"
)
```



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.

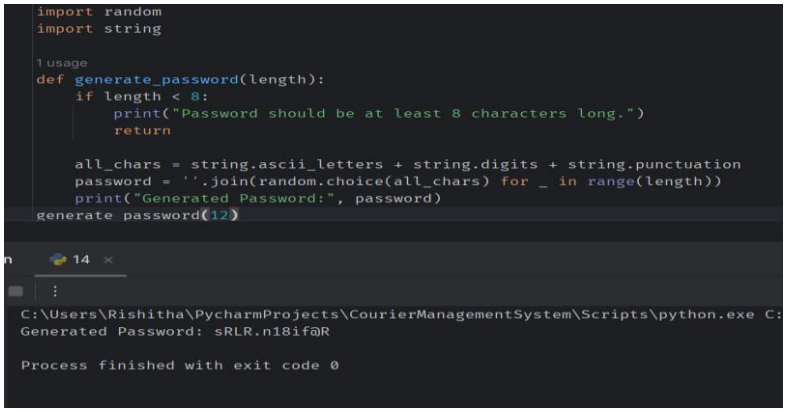
```
def calculate_shipping(source, destination, weight):
    distance_map = {
        ("Hyderabad", "Mumbai"): 700,
        ("Hyderabad", "Delhi"): 1500,
        ("Hyderabad", "Chennai"): 630,
        ("Hyderabad", "Bangalore"): 570
    }

    key = (source.title(), destination.title())

    if key in distance_map:
        distance = distance_map[key]
```

```
        cost = distance * weight * 5  # ₹5 per km per kg
        print(f"Shipping from {source} to {destination}")
        print("Distance:", distance, "km")
        print("Weight:", weight, "kg")
        print("Total Shipping Cost: ₹", cost)
    else:
        print("Distance between these locations is not available.")
calculate_shipping("Hyderabad", "Mumbai", 2)
```



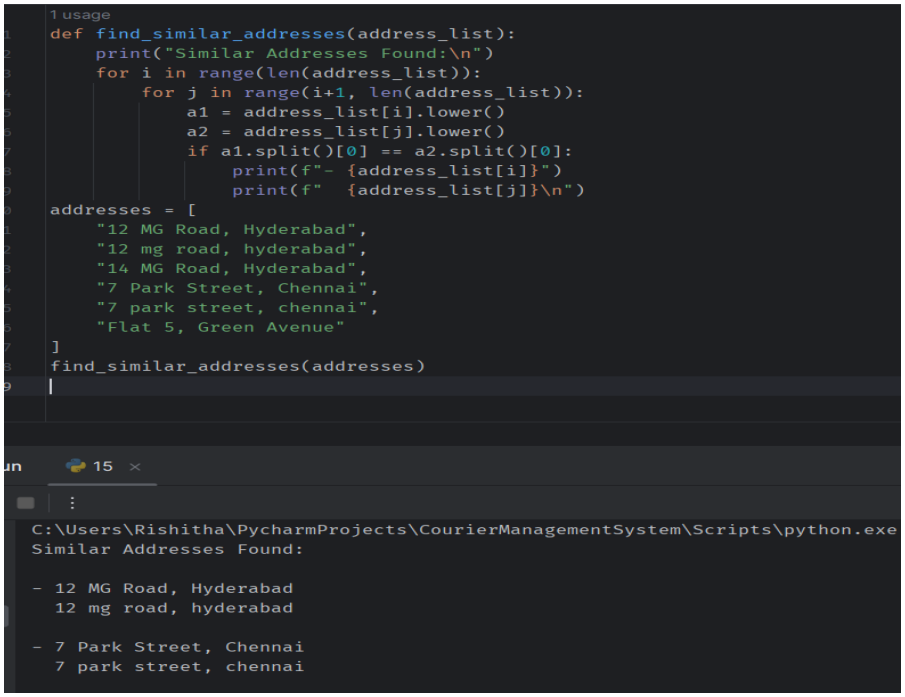
14. Password Generator: Create a function that generates secure passwords for courier system accounts. Ensure the passwords contain a mix of uppercase letters, lowercase letters, numbers, and special characters.

```
import random
import string
def generate_password(length):
    if length < 8:
        print("Password should be at least 8 characters long.")
        return
    all_chars = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(all_chars) for _ in range(length))
    print("Generated Password:", password)
generate_password(12)
```



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.

```
def find_similar_addresses(address_list):
    print("Similar Addresses Found:\n")
    for i in range(len(address_list)):
        for j in range(i+1, len(address_list)):
            a1 = address_list[i].lower()
            a2 = address_list[j].lower()
            if a1.split()[0] == a2.split()[0]:
                print(f"- {address_list[i]}")
                print(f"  {address_list[j]}\n")
addresses = [
    "12 MG Road, Hyderabad",
    "12 mg road, hyderabad",
    "14 MG Road, Hyderabad",
    "7 Park Street, Chennai",
    "7 park street, chennai",
    "Flat 5, Green Avenue"
]
find_similar_addresses(addresses)
```



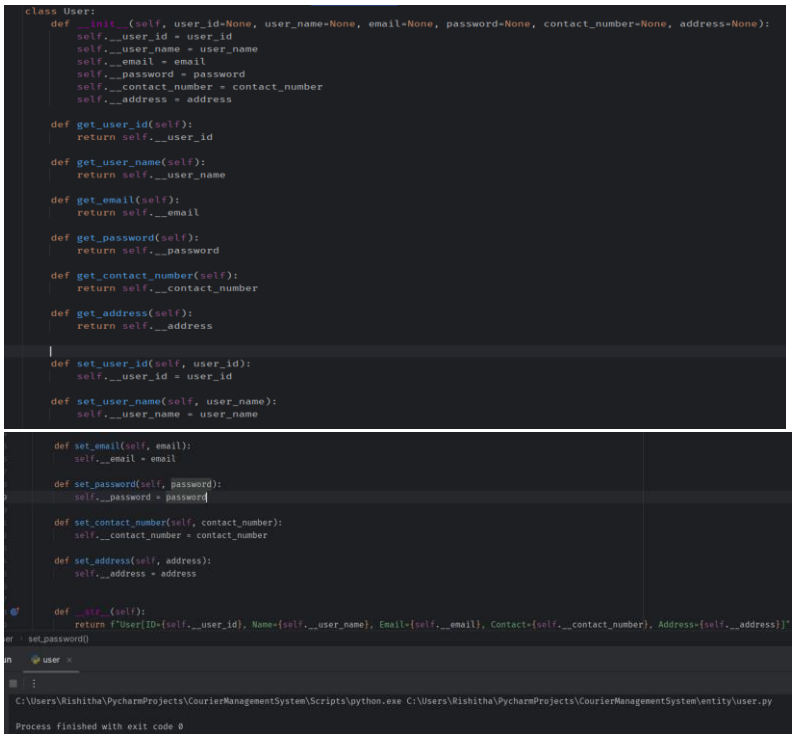
Task 5: Object Oriented Programming

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

```
class User:
    def __init__(self, user_id=None, user_name=None, email=None, password=None,
contact_number=None, address=None):
        self.__user_id = user_id
        self.__user_name = user_name
        self.__email = email
        self.__password = password
        self.__contact_number = contact_number
        self.__address = address

    def get_user_id(self):
        return self.__user_id

    def get_user_name(self):
        return self.__user_name

    def get_email(self):
        return self.__email

    def get_password(self):
        return self.__password

    def get_contact_number(self):
        return self.__contact_number

    def get_address(self):
        return self.__address

     def set_user_id(self, user_id):
        self.__user_id = user_id

    def set_user_name(self, user_name):
        self.__user_name = user_name

    def set_email(self, email):
        self.__email = email

    def set_password(self, password):
```

```
        self.__password = password

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number

    def set_address(self, address):
        self.__address = address


    def __str__(self):
        return f"User[ID={self.__user_id}, Name={self.__user_name}, Email={self.__email},
Contact={self.__contact_number}, Address={self.__address}]"
```



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

```
class Courier:
    def __init__(self, courier_id=None, sender_name=None, sender_address=None,
                 receiver_name=None, receiver_address=None, weight=None,
                 status=None, tracking_number=None, delivery_date=None, user_id=None):
        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
    self.__user_id = user_id


def get_courier_id(self):
    return self.__courier_id

def get_sender_name(self):
    return self.__sender_name

def get_sender_address(self):
    return self.__sender_address

def get_receiver_name(self):
    return self.__receiver_name

def get_receiver_address(self):
    return self.__receiver_address

def get_weight(self):
    return self.__weight

def get_status(self):
    return self.__status

def get_tracking_number(self):
    return self.__tracking_number

def get_delivery_date(self):
    return self.__delivery_date

def get_user_id(self):
    return self.__user_id

# Setters
def set_courier_id(self, courier_id):
    self.__courier_id = courier_id

def set_sender_name(self, sender_name):
    self.__sender_name = sender_name

def set_sender_address(self, sender_address):
    self.__sender_address = sender_address
```

```
def set_receiver_name(self, receiver_name):
    self.__receiver_name = receiver_name

def set_receiver_address(self, receiver_address):
    self.__receiver_address = receiver_address

def set_weight(self, weight):
    self.__weight = weight

def set_status(self, status):
    self.__status = status

def set_tracking_number(self, tracking_number):
    self.__tracking_number = tracking_number

def set_delivery_date(self, delivery_date):
    self.__delivery_date = delivery_date

def set_user_id(self, user_id):
    self.__user_id = user_id

def __str__(self):
    return (f"Courier[ID={self.__courier_id}, Sender={self.__sender_name}, "
            f"Receiver={self.__receiver_name}, Weight={self.__weight}kg, "
            f"Status={self.__status}, TrackingNo={self.__tracking_number}, "
            f"DeliveryDate={self.__delivery_date}, UserID={self.__user_id}]")
```

```
class Courier:
    def __init__(self, courier_id=None, sender_name=None, sender_address=None,
                 receiver_name=None, receiver_address=None, weight=None,
                 status=None, tracking_number=None, delivery_date=None, user_id=None):
        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
        self.__user_id = user_id


    def get_courier_id(self):
        return self.__courier_id

    1 usage
    def get_sender_name(self):
        return self.__sender_name

    1 usage
    def get_sender_address(self):
        return self.__sender_address

    1 usage
    def get_receiver_name(self):
        return self.__receiver_name

    1 usage
    def get_receiver_address(self):
        return self.__receiver_address
```

```python
    def get_weight(self):
        return self.__weight

    3 usages (2 dynamic)
    def get_status(self):
        return self.__status

    7 usages (4 dynamic)
    def get_tracking_number(self):
        return self.__tracking_number

    1 usage
    def get_delivery_date(self):
        return self.__delivery_date

    1 usage
    def get_user_id(self):
        return self.__user_id

    # Setters
    def set_courier_id(self, courier_id):
        self.__courier_id = courier_id

    def set_sender_name(self, sender_name):
        self.__sender_name = sender_name

    def set_sender_address(self, sender_address):
        self.__sender_address = sender_address

    def set_receiver_name(self, receiver_name):
        self.__receiver_name = receiver_name

    def set_receiver_address(self, receiver_address):
        self.__receiver_address = receiver_address
```

```python
    def set_receiver_address(self, receiver_address):
        self.__receiver_address = receiver_address

    def set_weight(self, weight):
        self.__weight = weight

    3 usages (3 dynamic)
    def set_status(self, status):
        self.__status = status

    3 usages
    def set_tracking_number(self, tracking_number):
        self.__tracking_number = tracking_number

    def set_delivery_date(self, delivery_date):
        self.__delivery_date = delivery_date

    def set_user_id(self, user_id):
        self.__user_id = user_id


    def __str__(self):
        return (f"Courier[ID={self.__courier_id}, Sender={self.__sender_name}, "
```

```
Run    courier
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects\

Process finished with exit code 0
```

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

```
class Employee:
    def __init__(self, employee_id=None, employee_name=None, email=None, contact_number=None, role=None, salary=None):
        self.__employee_id = employee_id
        self.__employee_name = employee_name
        self.__email = email
        self.__contact_number = contact_number
        self.__role = role
        self.__salary = salary
```

```
def get_employee_id(self):
    return self.__employee_id

def set_employee_id(self, employee_id):
    self.__employee_id = employee_id

def get_employee_name(self):
    return self.__employee_name

def set_employee_name(self, employee_name):
    self.__employee_name = employee_name

def get_email(self):
    return self.__email

def set_email(self, email):
    self.__email = email

def get_contact_number(self):
    return self.__contact_number

def set_contact_number(self, contact_number):
    self.__contact_number = contact_number

def get_role(self):
    return self.__role

def set_role(self, role):
    self.__role = role

def get_salary(self):
    return self.__salary

def set_salary(self, salary):
    self.__salary = salary

def __str__(self):
    return (f"Employee[ID={self.__employee_id}, Name={self.__employee_name}, "
            f"Email={self.__email}, Contact={self.__contact_number}, Role={self.__role}, "
            f"Salary={self.__salary}]")
```

```python
class Employee:
    def __init__(self, employee_id=None, employee_name=None, email=None, contact_number=None, role=None, salary=None):
        self.__employee_id = employee_id
        self.__employee_name = employee_name
        self.__email = email
        self.__contact_number = contact_number
        self.__role = role
        self.__salary = salary

    2 usages (2 dynamic)
    def get_employee_id(self):
        return self.__employee_id

    2 usages (2 dynamic)
    def set_employee_id(self, employee_id):
        self.__employee_id = employee_id

    def get_employee_name(self):
        return self.__employee_name

    def set_employee_name(self, employee_name):
        self.__employee_name = employee_name

    def get_email(self):
        return self.__email

    def set_email(self, email):
        self.__email = email

    def get_contact_number(self):
        return self.__contact_number

    def set_contact_number(self, contact_number):
        self.__contact_number = contact_number
```

```python
    def get_role(self):
        return self.__role

    def set_role(self, role):
        self.__role = role

    def get_salary(self):
        return self.__salary

    def set_salary(self, salary):
        self.__salary = salary

    def __str__(self):
        return (f"Employee[ID={self.__employee_id}, Name={self.__employee_name}, "
                f"Email={self.__email}, Contact={self.__contact_number}, Role={self.__role}, "
                f"Salary={self.__salary}]")
```

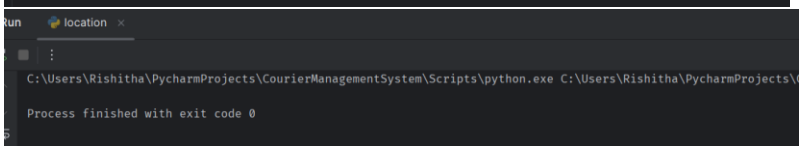
```
Employee
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects

Process finished with exit code 0
```

4. Location Class Variables LocationID , LocationName , Address

```python
class Location:
    def __init__(self, location_id=None, location_name=None, address=None):
        self.__location_id = location_id
        self.__location_name = location_name
        self.__address = address

    # Getters
    def get_location_id(self):
        return self.__location_id

    def get_location_name(self):
        return self.__location_name

    def get_address(self):
        return self.__address

    # Setters
    def set_location_id(self, location_id):
```

```
        self.__location_id = location_id

    def set_location_name(self, location_name):
        self.__location_name = location_name

    def set_address(self, address):
        self.__address = address


    def __str__(self):
        return f"Location[ID={self.__location_id}, Name={self.__location_name},
Address={self.__address}]"
```

```
class Location:
    def __init__(self, location_id=None, location_name=None, address=None):
        self.__location_id = location_id
        self.__location_name = location_name
        self.__address = address

    # Getters
    def get_location_id(self):
        return self.__location_id

    def get_location_name(self):
        return self.__location_name

    def get_address(self):
        return self.__address

    # Setters
    def set_location_id(self, location_id):
        self.__location_id = location_id

    def set_location_name(self, location_name):
        self.__location_name = location_name

    def set_address(self, address):
        self.__address = address


    def __str__(self):
        return f"Location[ID={self.__location_id}, Name={self.__location_name}, Address={self.__address}]"
```



```
Run    location
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects\

Process finished with exit code 0
```

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

```
from entity.courier import Courier
from entity.employee import Employee
from entity.location import Location

class CourierCompany:
    def __init__(self, company_name=None, courier_details=None, employee_details=None,
location_details=None):
        self.__company_name = company_name
        self.__courier_details = courier_details if courier_details is not None else []
        self.__employee_details = employee_details if employee_details is not None else []
        self.__location_details = location_details if location_details is not None else []
```

```
# Getters
def get_company_name(self):
    return self.__company_name

def get_courier_details(self):
    return self.__courier_details

def get_employee_details(self):
    return self.__employee_details

def get_location_details(self):
    return self.__location_details

# Setters
def set_company_name(self, company_name):
    self.__company_name = company_name

def set_courier_details(self, courier_details):
    self.__courier_details = courier_details

def set_employee_details(self, employee_details):
    self.__employee_details = employee_details

def set_location_details(self, location_details):
    self.__location_details = location_details

# toString
def __str__(self):
    return (f"CourierCompany[Name={self.__company_name}, "
            f"Couriers={len(self.__courier_details)}, "
            f"Employees={len(self.__employee_details)}, "
            f"Locations={len(self.__location_details)}]")
```

```python
from entity.courier import Courier
from entity.employee import Employee
from entity.location import Location

2 usages
class CourierCompany:
    def __init__(self, company_name=None, courier_details=None, employee_details=None, location_details=None):
        self.__company_name = company_name
        self.__courier_details = courier_details if courier_details is not None else []
        self.__employee_details = employee_details if employee_details is not None else []
        self.__location_details = location_details if location_details is not None else []

    # Getters
    def get_company_name(self):
        return self.__company_name

    3 usages
    def get_courier_details(self):
        return self.__courier_details

    2 usages (2 dynamic)
    def get_employee_details(self):
        return self.__employee_details

    def get_location_details(self):
        return self.__location_details

    # Setters
    def set_company_name(self, company_name):
        self.__company_name = company_name
```

```python
    def set_courier_details(self, courier_details):
        self.__courier_details = courier_details

    def set_employee_details(self, employee_details):
        self.__employee_details = employee_details

    def set_location_details(self, location_details):
        self.__location_details = location_details

    # toString
    def __str__(self):
        return (f"CourierCompany[Name={self.__company_name}, "
                f"Couriers={len(self.__courier_details)}, "
                f"Employees={len(self.__employee_details)}, "
                f"Locations={len(self.__location_details)}]")
```

```
ourierCompany
un    courier_company
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\

Process finished with exit code 0
```

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

```python
from datetime import date

class Payment:
    def __init__(self, payment_id=None, courier_id=None, amount=None, payment_date=None):
        self.__payment_id = payment_id
        self.__courier_id = courier_id
        self.__amount = amount
        self.__payment_date = payment_date if payment_date else date.today()

    # Getters
    def get_payment_id(self):
        return self.__payment_id

    def get_courier_id(self):
```

```
    return self.__courier_id

def get_amount(self):
    return self.__amount

def get_payment_date(self):
    return self.__payment_date

# Setters
def set_payment_id(self, payment_id):
    self.__payment_id = payment_id

def set_courier_id(self, courier_id):
    self.__courier_id = courier_id

def set_amount(self, amount):
    self.__amount = amount

def set_payment_date(self, payment_date):
    self.__payment_date = payment_date

# toString
def __str__(self):
    return (f"Payment[ID={self.__payment_id}, CourierID={self.__courier_id}, "
            f"Amount={self.__amount}, Date={self.__payment_date}]")
```

```
from datetime import date

class Payment:
    def __init__(self, payment_id=None, courier_id=None, amount=None, payment_date=None
        self.__payment_id = payment_id
        self.__courier_id = courier_id
        self.__amount = amount
        self.__payment_date = payment_date if payment_date else date.today()

    # Getters
    def get_payment_id(self):
        return self.__payment_id

    def get_courier_id(self):
        return self.__courier_id

    def get_amount(self):
        return self.__amount

    def get_payment_date(self):
        return self.__payment_date

    # Setters
    def set_payment_id(self, payment_id):
        self.__payment_id = payment_id

    def set_courier_id(self, courier_id):
        self.__courier_id = courier_id

    def set_amount(self, amount):
        self.__amount = amount
```

```python
    def set_amount(self, amount):
        self.__amount = amount

    def set_payment_date(self, payment_date):
        self.__payment_date = payment_date

    # toString
    def __str__(self):
        return (f"Payment[ID={self.__payment_id}, CourierID={self.__courier_id}, "
                f"Amount={self.__amount}, Date={self.__payment_date}]")
```

```
yment › set_amount()
payment
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha

Process finished with exit code 0
```

Task 6: Service Provider Interface /Abstract class

Create 2 Interface /Abstract class ICourierUserService and ICourierAdminService interface ICourierUserService { // Customer-related functions

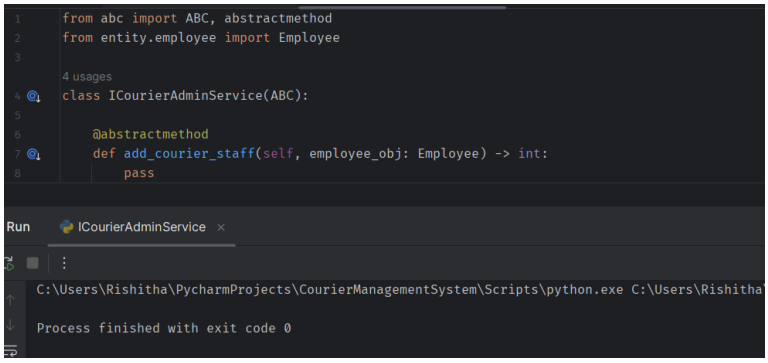
ICourierUserService.py

```python
from abc import ABC, abstractmethod
from entity.courier import Courier
class ICourierUserService(ABC):
    @abstractmethod
    def place_order(self, courier_obj: Courier) -> str:
        pass

    @abstractmethod
    def get_order_status(self, tracking_number: str) -> str:
        pass
    @abstractmethod
    def cancel_order(self, tracking_number: str) -> bool:
        pass
    @abstractmethod
    def get_assigned_order(self, courier_staff_id: int) -> list:
        pass
```

```python
main.py   ICourierUserService.py   courier.py   employee.py   location.py   courier_c
from abc import ABC, abstractmethod
from entity.courier import Courier

4 usages
class ICourierUserService(ABC):

    @abstractmethod
    def place_order(self, courier_obj: Courier) -> str:
        pass

    @abstractmethod
    def get_order_status(self, tracking_number: str) -> str:
        pass

    @abstractmethod
    def cancel_order(self, tracking_number: str) -> bool:
        pass

    @abstractmethod
    def get_assigned_order(self, courier_staff_id: int) -> list:
        pass
ICourierUserService › cancel_order()
Run   ICourierUserService
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha

Process finished with exit code 0
```

ICourierAdminService.py

```
from abc import ABC, abstractmethod
from entity.employee import Employee

class ICourierAdminService(ABC):

    @abstractmethod
    def add_courier_staff(self, employee_obj: Employee) -> int:
        pass
```

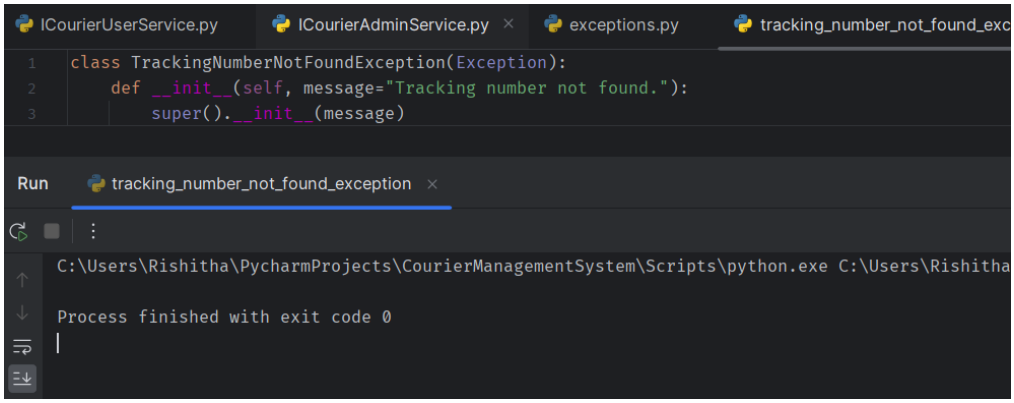


Task 7: Exception Handling

Define the following custom exceptions and throw them in methods whenever needed . Handle all the excpetionsin main method, 1. TrackingNumberNotFoundException :throw this exception when user try to withdraw amount or transfer amount to another acco 2. InvalidEmployeeIdException throw this exception when id entered for the employee not existing in the system

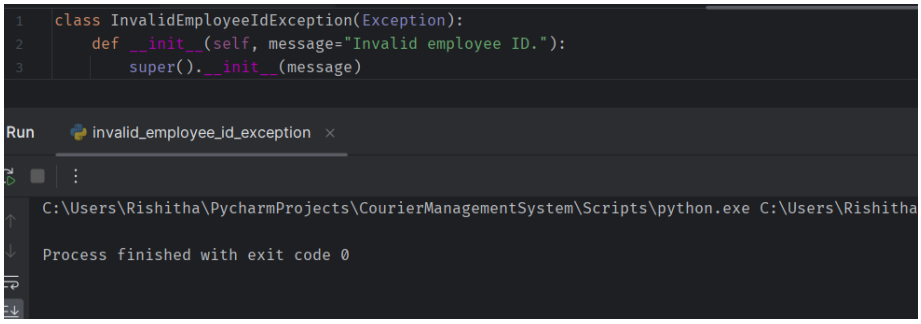
1)TrackingNumberNotFoundException

```
class TrackingNumberNotFoundException(Exception):
    def __init__(self, message="Tracking number not found."):
        super().__init__(message)
```



2)InvalidEmployeeIdException

```
class InvalidEmployeeIdException(Exception):
    def __init__(self, message="Invalid employee ID."):
        super().__init__(message)
```



Task 8: 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

CourierCompanyCollection

```
from entity.courier import Courier
from entity.employee import Employee
from entity.location import Location

class CourierCompanyCollection:
    def __init__(self, company_name=None):
        self.__company_name = company_name
        self.__courier_details = []
        self.__employee_details = []
        self.__location_details = []

    # Getters
    def get_company_name(self):
        return self.__company_name

    def get_courier_details(self):
        return self.__courier_details

    def get_employee_details(self):
        return self.__employee_details

    def get_location_details(self):
        return self.__location_details

    # Setters
```

```
def set_company_name(self, name):
    self.__company_name = name

def set_courier_details(self, courier_list):
    self.__courier_details = courier_list

def set_employee_details(self, employee_list):
    self.__employee_details = employee_list

def set_location_details(self, location_list):
    self.__location_details = location_list

# toString
def __str__(self):
    return (f"CourierCompanyCollection[Name={self.__company_name}, "
            f"Couriers={len(self.__courier_details)}, "
            f"Employees={len(self.__employee_details)}, "
            f"Locations={len(self.__location_details)}]")
```

```
from entity.courier import Courier
from entity.employee import Employee
from entity.location import Location

2 usages
class CourierCompanyCollection:
    def __init__(self, company_name=None):
        self.__company_name = company_name
        self.__courier_details = []
        self.__employee_details = []
        self.__location_details = []

    # Getters
    def get_company_name(self):
        return self.__company_name

    3 usages
    def get_courier_details(self):
        return self.__courier_details

    2 usages (2 dynamic)
    def get_employee_details(self):
        return self.__employee_details

    def get_location_details(self):
        return self.__location_details

    # Setters
    def set_company_name(self, name):
        self.__company_name = name
```

```python
    # Setters
    def set_company_name(self, name):
        self.__company_name = name

    def set_courier_details(self, courier_list):
        self.__courier_details = courier_list

    def set_employee_details(self, employee_list):
        self.__employee_details = employee_list

    def set_location_details(self, location_list):
        self.__location_details = location_list

    # toString
    def __str__(self):
        return (f"CourierCompanyCollection[Name={self.__company_name}, "
                f"Couriers={len(self.__courier_details)}, "
                f"Employees={len(self.__employee_details)}, "
                f"Locations={len(self.__location_details)}]")
```

```
erCompanyCollection › set_location_details()
CourierCompanyCollection
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\

Process finished with exit code 0
```

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

CourierAdminServiceCollectionImpl

```python
from service.ICourierUserService import ICourierUserService
from entity.CourierCompanyCollection import CourierCompanyCollection
from entity.courier import Courier

class CourierUserServiceCollectionImpl(ICourierUserService):

    tracking_counter = 1000

    def __init__(self):
        self.company_obj = CourierCompanyCollection("SpeedyX Couriers")

    def place_order(self, courier_obj: Courier) -> str:
        CourierUserServiceCollectionImpl.tracking_counter += 1
        tracking_number = f"TRK{CourierUserServiceCollectionImpl.tracking_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company_obj.get_courier_details().append(courier_obj)
        return tracking_number

    def get_order_status(self, tracking_number: str) -> str:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                return c.get_status()
        return "Tracking number not found."

    def cancel_order(self, tracking_number: str) -> bool:
```

```
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                c.set_status("Cancelled")
                return True
        return False

    def get_assigned_order(self, courier_staff_id: int) -> list:
        return []
```

```python
from service.ICourierUserService import ICourierUserService
from entity.CourierCompanyCollection import CourierCompanyCollection
from entity.courier import Courier

4 usages
class CourierUserServiceCollectionImpl(ICourierUserService):

    tracking_counter = 1000

    def __init__(self):
        self.company_obj = CourierCompanyCollection("SpeedyX Couriers")

    def place_order(self, courier_obj: Courier) -> str:
        CourierUserServiceCollectionImpl.tracking_counter += 1
        tracking_number = f"TRK{CourierUserServiceCollectionImpl.tracking_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company_obj.get_courier_details().append(courier_obj)
        return tracking_number

    def get_order_status(self, tracking_number: str) -> str:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                return c.get_status()
        return "Tracking number not found."

    def cancel_order(self, tracking_number: str) -> bool:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                c.set_status("Cancelled")
                return True
        return False

    def get_assigned_order(self, courier_staff_id: int) -> list:
        return []
```

CourierAdminServiceCollectionImpl

```
from service.CourierUserServiceCollectionImpl import CourierUserServiceCollectionImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

class CourierAdminServiceCollectionImpl(CourierUserServiceCollectionImpl,
ICourierAdminService):

    employee_id_counter = 100

    def add_courier_staff(self, employee_obj: Employee) -> int:
        CourierAdminServiceCollectionImpl.employee_id_counter += 1

employee_obj.set_employee_id(CourierAdminServiceCollectionImpl.employee_id_counter)
        self.company_obj.get_employee_details().append(employee_obj)
        return employee_obj.get_employee_id()
```

```python
from service.CourierUserServiceCollectionImpl import CourierUserServiceCollectionImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

2 usages
class CourierAdminServiceCollectionImpl(CourierUserServiceCollectionImpl, ICourierAdminService):

    employee_id_counter = 100

    def add_courier_staff(self, employee_obj: Employee) -> int:
        CourierAdminServiceCollectionImpl.employee_id_counter += 1
        employee_obj.set_employee_id(CourierAdminServiceCollectionImpl.employee_id_counter)
        self.company_obj.get_employee_details().append(employee_obj)
        return employee_obj.get_employee_id()
```

```
un    CourierUserServiceCollectionImpl
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\Pycha

Process finished with exit code 0
```

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.

CourierUserServiceImpl.py

```python
from service.ICourierUserService import ICourierUserService
from entity.courier_company import CourierCompany
from entity.courier import Courier

class CourierUserServiceImpl(ICourierUserService):

    tracking_counter = 1000

    def __init__(self):
        self.company_obj = CourierCompany("SpeedyX Couriers", [], [], [])

    def place_order(self, courier_obj: Courier) -> str:
        CourierUserServiceImpl.tracking_counter += 1
        tracking_number = f"TRK{CourierUserServiceImpl.tracking_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company_obj.get_courier_details().append(courier_obj)
        return tracking_number

    def get_order_status(self, tracking_number: str) -> str:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                return c.get_status()
        return "Tracking number not found."

    def cancel_order(self, tracking_number: str) -> bool:
        for c in self.company_obj.get_courier_details():
```

```
        if c.get_tracking_number() == tracking_number:
            c.set_status("Cancelled")
            return True
    return False

def get_assigned_order(self, courier_staff_id: int) -> list:
    return []
```

```python
from service.ICourierUserService import ICourierUserService
from entity.courier_company import CourierCompany
from entity.courier import Courier

4 usages
class CourierUserServiceImpl(ICourierUserService):

    tracking_counter = 1000

    def __init__(self):
        self.company_obj = CourierCompany( company_name: "SpeedyX Couriers",  courier_details: [],  employee_details: [],  location_details: [])

    def place_order(self, courier_obj: Courier) -> str:
        CourierUserServiceImpl.tracking_counter += 1
        tracking_number = f"TRK{CourierUserServiceImpl.tracking_counter}"
        courier_obj.set_tracking_number(tracking_number)
        self.company_obj.get_courier_details().append(courier_obj)
        return tracking_number

    def get_order_status(self, tracking_number: str) -> str:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                return c.get_status()
        return "Tracking number not found."

    def cancel_order(self, tracking_number: str) -> bool:
        for c in self.company_obj.get_courier_details():
            if c.get_tracking_number() == tracking_number:
                c.set_status("Cancelled")
                return True
```

```python
                c.set_status("Cancelled")
                return True
        return False

    def get_assigned_order(self, courier_staff_id: int) -> list:
        return []
```

```
CourierUserServiceImpl
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\

Process finished with exit code 0
```

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

```
from service.CourierUserServiceImpl import CourierUserServiceImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

class CourierAdminServiceImpl(CourierUserServiceImpl, ICourierAdminService):

    employee_id_counter = 100

    def add_courier_staff(self, employee_obj: Employee) -> int:
        CourierAdminServiceImpl.employee_id_counter += 1
        employee_obj.set_employee_id(CourierAdminServiceImpl.employee_id_counter)
```

self.company_obj.get_employee_details().append(employee_obj)
return employee_obj.get_employee_id()

```python
from service.CourierUserServiceImpl import CourierUserServiceImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

2 usages
class CourierAdminServiceImpl(CourierUserServiceImpl, ICourierAdminService):

    employee_id_counter = 100

    def add_courier_staff(self, employee_obj: Employee) -> int:
        CourierAdminServiceImpl.employee_id_counter += 1
        employee_obj.set_employee_id(CourierAdminServiceImpl.employee_id_counter)
        self.company_obj.get_employee_details().append(employee_obj)
        return employee_obj.get_employee_id()
```

```
CourierAdminServiceImpl
Run    CourierAdminServiceImpl
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\

Process finished with exit code 0
```

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

from service.CourierUserServiceCollectionImpl import CourierUserServiceCollectionImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

class CourierAdminServiceCollectionImpl(CourierUserServiceCollectionImpl, ICourierAdminService):

employee_id_counter = 100

def add_courier_staff(self, employee_obj: Employee) -> int:
CourierAdminServiceCollectionImpl.employee_id_counter += 1

employee_obj.set_employee_id(CourierAdminServiceCollectionImpl.employee_id_counter)
self.company_obj.get_employee_details().append(employee_obj)
return employee_obj.get_employee_id()

```python
from service.CourierUserServiceCollectionImpl import CourierUserServiceCollectionImpl
from service.ICourierAdminService import ICourierAdminService
from entity.employee import Employee

2 usages
class CourierAdminServiceCollectionImpl(CourierUserServiceCollectionImpl, ICourierAdminService):

    employee_id_counter = 100

    def add_courier_staff(self, employee_obj: Employee) -> int:
        CourierAdminServiceCollectionImpl.employee_id_counter += 1
        employee_obj.set_employee_id(CourierAdminServiceCollectionImpl.employee_id_counter)
        self.company_obj.get_employee_details().append(employee_obj)
        return employee_obj.get_employee_id()
```

```
erAdminServiceCollectionIm...
CourierAdminServiceCollectionImpl
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects

Process finished with exit code 0
```

Task 9: Database Interaction

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.

Database Connection using db.properties

```
[mysql]
host = localhost
user = root
password = bablu345.
database = CourierDB
port = 3306
```

```
[mysql]
host = localhost
user = root
password = bablu345.
database = CourierDB
port = 3306
```

DBConnection

```
import mysql.connector
import configparser

class DBConnection:
    __connection = None

    @staticmethod
    def get_connection():
        if DBConnection.__connection is None:
            config = configparser.ConfigParser()
            config.read('db.properties')
            DBConnection.__connection = mysql.connector.connect(
                host=config['mysql']['host'],
                user=config['mysql']['user'],
                password=config['mysql']['password'],
                database=config['mysql']['database'],
                port=int(config['mysql']['port'])
            )
        return DBConnection.__connection
```

```python
import mysql.connector
import configparser

5 usages
class DBConnection:
    __connection = None

    1 usage
    @staticmethod
    def get_connection():
        if DBConnection.__connection is None:
            config = configparser.ConfigParser()
            config.read('db.properties')
            DBConnection.__connection = mysql.connector.connect(
                host=config['mysql']['host'],
                user=config['mysql']['user'],
                password=config['mysql']['password'],
                database=config['mysql']['database'],
                port=int(config['mysql']['port'])
            )
        return DBConnection.__connection
```

```
Run    DBConnection
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha

Process finished with exit code 0
```

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.

CourierServiceDB

```
from connectionutil.DBConnection import DBConnection
from entity.courier import Courier

class CourierServiceDb:

    def __init__(self):
        self.connection = DBConnection.get_connection()
        self.cursor = self.connection.cursor()

    def _generate_tracking_number(self):
        self.cursor.execute("SELECT TrackingNumber FROM COURIER ORDER BY CourierID DESC
LIMIT 1")
        last = self.cursor.fetchone()
        if last and last[0] and last[0].startswith("TRK") and last[0][3:].isdigit():
            next_number = int(last[0][3:]) + 1
        else:
            next_number = 10001
        return f"TRK{next_number}"

    def insert_courier(self, courier: Courier):
        tracking_number = self._generate_tracking_number()
        courier.set_tracking_number(tracking_number)

        sql = """INSERT INTO COURIER
        (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status,
TrackingNumber, DeliveryDate, UserID)
```

```
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"""

        values = (
            courier.get_sender_name(),
            courier.get_sender_address(),
            courier.get_receiver_name(),
            courier.get_receiver_address(),
            courier.get_weight(),
            courier.get_status(),
            courier.get_tracking_number(),
            courier.get_delivery_date(),
            courier.get_user_id()
        )

        self.cursor.execute(sql, values)
        self.connection.commit()
        print(f"Courier inserted successfully with TrackingNumber:
{courier.get_tracking_number()}")

    def get_courier_status(self, tracking_number: str):
        sql = "SELECT Status FROM COURIER WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (tracking_number,))
        result = self.cursor.fetchone()
        return result[0] if result else "Tracking number not found."

    def update_status(self, tracking_number: str, new_status: str):
        sql = "UPDATE COURIER SET Status = %s WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (new_status, tracking_number))
        self.connection.commit()
        return self.cursor.rowcount

    def get_delivery_history(self, user_id: int):
        sql = "SELECT * FROM COURIER WHERE UserID = %s"
        self.cursor.execute(sql, (user_id,))
        result = self.cursor.fetchall()
        cols = [desc[0] for desc in self.cursor.description]
        return [dict(zip(cols, row)) for row in result]
```

```python
from connectionutil.DBConnection import DBConnection
from entity.courier import Courier

2 usages
class CourierServiceDb:

    def __init__(self):
        self.connection = DBConnection.get_connection()
        self.cursor = self.connection.cursor()

    1 usage
    def _generate_tracking_number(self):
        self.cursor.execute("SELECT TrackingNumber FROM COURIER ORDER BY CourierID DESC LIMIT 1")
        last = self.cursor.fetchone()
        if last and last[0] and last[0].startswith("TRK") and last[0][3:].isdigit():
            next_number = int(last[0][3:]) + 1
        else:
            next_number = 10001
        return f"TRK{next_number}"

    1 usage
    def insert_courier(self, courier: Courier):
        tracking_number = self._generate_tracking_number()
        courier.set_tracking_number(tracking_number)

        sql = """INSERT INTO COURIER
        (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, UserID)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"""

        values = (
            courier.get_sender_name(),
            courier.get_sender_address(),
```

```python
            courier.get_tracking_number(),
            courier.get_delivery_date(),
            courier.get_user_id()
        )

        self.cursor.execute(sql, values)
        self.connection.commit()
        print(f"Courier inserted successfully with TrackingNumber: {courier.get_tracking_number()}")

    2 usages
    def get_courier_status(self, tracking_number: str):
        sql = "SELECT Status FROM COURIER WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (tracking_number,))
        result = self.cursor.fetchone()
        return result[0] if result else "Tracking number not found."

    1 usage
    def update_status(self, tracking_number: str, new_status: str):
        sql = "UPDATE COURIER SET Status = %s WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (new_status, tracking_number))
        self.connection.commit()
        return self.cursor.rowcount

    1 usage
    def get_delivery_history(self, user_id: int):
        sql = "SELECT * FROM COURIER WHERE UserID = %s"
        self.cursor.execute(sql, (user_id,))
        result = self.cursor.fetchall()
        cols = [desc[0] for desc in self.cursor.description]
        return [dict(zip(cols, row)) for row in result]
```

```
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\main.py
Courier inserted successfully with TrackingNumber: TRK94132
Current Status: Processing
Updated Status: Delivered

Delivery History for User ID 1:
{'CourierID': 103, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 107, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 109, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 110, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 111, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 112, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 113, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 116, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processin
{'CourierID': 119, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 120, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 121, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 122, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 123, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 124, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 125, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 126, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered

Process finished with exit code 0
```

3. Include methods to insert, update, and retrieve data from the database (e.g., inserting a new order, updating courier status).

CourierServiceDB

from connectionutil.DBConnection import DBConnection
from entity.courier import Courier

```
class CourierServiceDb:

    def __init__(self):
        self.connection = DBConnection.get_connection()
        self.cursor = self.connection.cursor()

    def _generate_tracking_number(self):
        self.cursor.execute("SELECT TrackingNumber FROM COURIER ORDER BY CourierID DESC
LIMIT 1")
        last = self.cursor.fetchone()
        if last and last[0] and last[0].startswith("TRK") and last[0][3:].isdigit():
            next_number = int(last[0][3:]) + 1
        else:
            next_number = 10001
        return f"TRK{next_number}"

    def insert_courier(self, courier: Courier):
        tracking_number = self._generate_tracking_number()
        courier.set_tracking_number(tracking_number)

        sql = """INSERT INTO COURIER
        (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status,
TrackingNumber, DeliveryDate, UserID)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"""

        values = (
            courier.get_sender_name(),
            courier.get_sender_address(),
            courier.get_receiver_name(),
            courier.get_receiver_address(),
            courier.get_weight(),
            courier.get_status(),
            courier.get_tracking_number(),
            courier.get_delivery_date(),
            courier.get_user_id()
        )

        self.cursor.execute(sql, values)
        self.connection.commit()
        print(f"Courier inserted successfully with TrackingNumber:
{courier.get_tracking_number()}")

    def get_courier_status(self, tracking_number: str):
```

```
    sql = "SELECT Status FROM COURIER WHERE TrackingNumber = %s"
    self.cursor.execute(sql, (tracking_number,))
    result = self.cursor.fetchone()
    return result[0] if result else "Tracking number not found."

def update_status(self, tracking_number: str, new_status: str):
    sql = "UPDATE COURIER SET Status = %s WHERE TrackingNumber = %s"
    self.cursor.execute(sql, (new_status, tracking_number))
    self.connection.commit()
    return self.cursor.rowcount

def get_delivery_history(self, user_id: int):
    sql = "SELECT * FROM COURIER WHERE UserID = %s"
    self.cursor.execute(sql, (user_id,))
    result = self.cursor.fetchall()
    cols = [desc[0] for desc in self.cursor.description]
    return [dict(zip(cols, row)) for row in result]
```

```python
from connectionutil.DBConnection import DBConnection
from entity.courier import Courier

2 usages
class CourierServiceDb:

    def __init__(self):
        self.connection = DBConnection.get_connection()
        self.cursor = self.connection.cursor()

    1 usage
    def _generate_tracking_number(self):
        self.cursor.execute("SELECT TrackingNumber FROM COURIER ORDER BY CourierID DESC LIMIT 1")
        last = self.cursor.fetchone()
        if last and last[0] and last[0].startswith("TRK") and last[0][3:].isdigit():
            next_number = int(last[0][3:]) + 1
        else:
            next_number = 10001
        return f"TRK{next_number}"

    1 usage
    def insert_courier(self, courier: Courier):
        tracking_number = self._generate_tracking_number()
        courier.set_tracking_number(tracking_number)

        sql = """INSERT INTO COURIER
        (SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate, UserID)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"""

        values = (
            courier.get_sender_name(),
            courier.get_sender_address(),
```

```python
            courier.get_tracking_number(),
            courier.get_delivery_date(),
            courier.get_user_id()
        )

        self.cursor.execute(sql, values)
        self.connection.commit()
        print(f"Courier inserted successfully with TrackingNumber: {courier.get_tracking_number()}")

    2 usages
    def get_courier_status(self, tracking_number: str):
        sql = "SELECT Status FROM COURIER WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (tracking_number,))
        result = self.cursor.fetchone()
        return result[0] if result else "Tracking number not found."

    1 usage
    def update_status(self, tracking_number: str, new_status: str):
        sql = "UPDATE COURIER SET Status = %s WHERE TrackingNumber = %s"
        self.cursor.execute(sql, (new_status, tracking_number))
        self.connection.commit()
        return self.cursor.rowcount

    1 usage
    def get_delivery_history(self, user_id: int):
        sql = "SELECT * FROM COURIER WHERE UserID = %s"
        self.cursor.execute(sql, (user_id,))
        result = self.cursor.fetchall()
        cols = [desc[0] for desc in self.cursor.description]
        return [dict(zip(cols, row)) for row in result]
```

```
C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\Scripts\python.exe C:\Users\Rishitha\PycharmProjects\CourierManagementSystem\main.py
Courier inserted successfully with TrackingNumber: TRK94132
Current Status: Processing
Updated Status: Delivered

Delivery History for User ID 1:
{'CourierID': 103, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 107, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 109, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 110, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 111, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 112, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 113, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 116, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Processing
{'CourierID': 119, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 120, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 121, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 122, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 123, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 124, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 125, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered
{'CourierID': 126, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Delivered

Process finished with exit code 0
```

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).

```
from entity.courier import Courier
from service.CourierServiceDb import CourierServiceDb
from datetime import date

if __name__ == "__main__":
    # Initialize the service
    courier_service = CourierServiceDb()

    # Create a new courier
    courier = Courier(
        sender_name="Rishitha",
        sender_address="Hyderabad",
        receiver_name="Siva",
        receiver_address="Chennai",
        weight=3.5,
        status="Processing",
        delivery_date=str(date.today()),
        user_id=1
    )

    # Insert the courier into the database
    courier_service.insert_courier(courier)

    # Get the status of the inserted courier
    tracking_number = courier.get_tracking_number()
    status = courier_service.get_courier_status(tracking_number)
    print("Current Status:", status)

    # Update the courier status to 'Delivered'
    courier_service.update_status(tracking_number, "Delivered")
```

```
updated_status = courier_service.get_courier_status(tracking_number)
print("Updated Status:", updated_status)

# Delivery History for a specific user
history = courier_service.get_delivery_history(1)
print("\nDelivery History for User ID 1:")
for record in history:
    print(record)

# Shipment Status Report
print("\nShipment Status Report:")
status_report = courier_service.get_status_report()
for status, total in status_report:
    print(f"{status}: {total} parcels")
```

```python
from entity.courier import Courier
from service.CourierServiceDb import CourierServiceDb
from datetime import date

if __name__ == "__main__":
    # Initialize the service
    courier_service = CourierServiceDb()

    # Create a new courier
    courier = Courier(
        sender_name="Rishitha",
        sender_address="Hyderabad",
        receiver_name="Siva",
        receiver_address="Chennai",
        weight=3.5,
        status="Processing",
        delivery_date=str(date.today()),
        user_id=1
    )

    # Insert the courier into the database
    courier_service.insert_courier(courier)

    # Get the status of the inserted courier
    tracking_number = courier.get_tracking_number()
    status = courier_service.get_courier_status(tracking_number)
    print("Current Status:", status)

    # Update the courier status to 'Delivered'
    courier_service.update_status(tracking_number, new_status: "Delivered")
    updated_status = courier_service.get_courier_status(tracking_number)
    print("Updated Status:", updated_status)
```

```python
# Delivery History for a specific user
history = courier_service.get_delivery_history(1)
print("\nDelivery History for User ID 1:")
for record in history:
    print(record)

# Shipment Status Report
print("\nShipment Status Report:")
status_report = courier_service.get_status_report()
for status, total in status_report:
    print(f"{status}: {total} parcels")
```

Delivery history and Shipment Status Report:

```
Courier inserted successfully with TrackingNumber: TRK94133
Current Status: Processing
Updated Status: Delivered

Delivery History for User ID 1:
{'CourierID': 103, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 107, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 109, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 110, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 111, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 112, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 113, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 116, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Process
{'CourierID': 119, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 120, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 121, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 122, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 123, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 124, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 125, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 126, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver
{'CourierID': 127, 'SenderName': 'Rishitha', 'SenderAddress': 'Hyderabad', 'ReceiverName': 'Siva', 'ReceiverAddress': 'Chennai', 'Weight': Decimal('3.50'), 'Status': 'Deliver

Shipment Status Report:
Pending: 1 parcels
In-Transit: 1 parcels
Processing: 8 parcels
Delivered: 9 parcels

Process finished with exit code 0
```