**Task 5: Object Oriented Programming**

Scope : Entity classes/Models/POJO, Abstraction/Encapsulation

Create the following model/entity classes within package entities with variables declared

private, constructors(default and parametrized,getters,setters and toString())

1. User Class:

Variables:

userID , userName , email , password , contactNumber , address

2. Courier Class

Variables: courierID , senderName , senderAddress , receiverName , receiverAddress , weight ,

status, trackingNumber , deliveryDate ,userId

3. Employee Class:

Variables employeeID , employeeName , email , contactNumber , role String, salary

4. Location Class

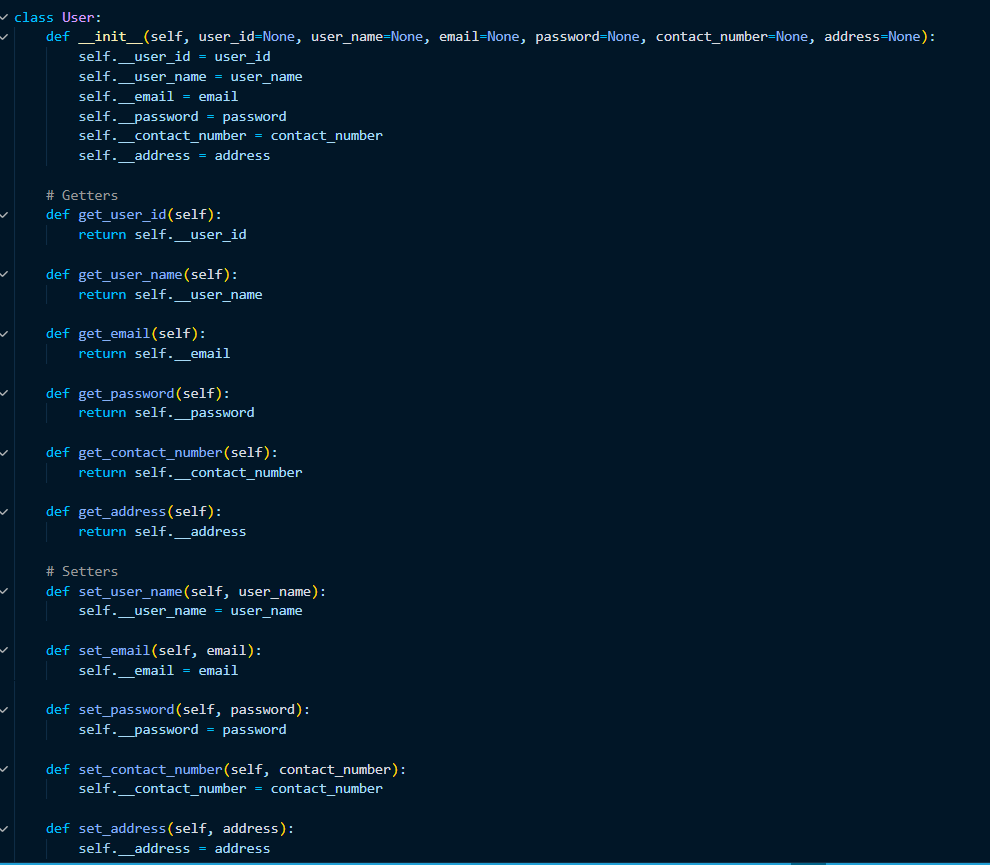
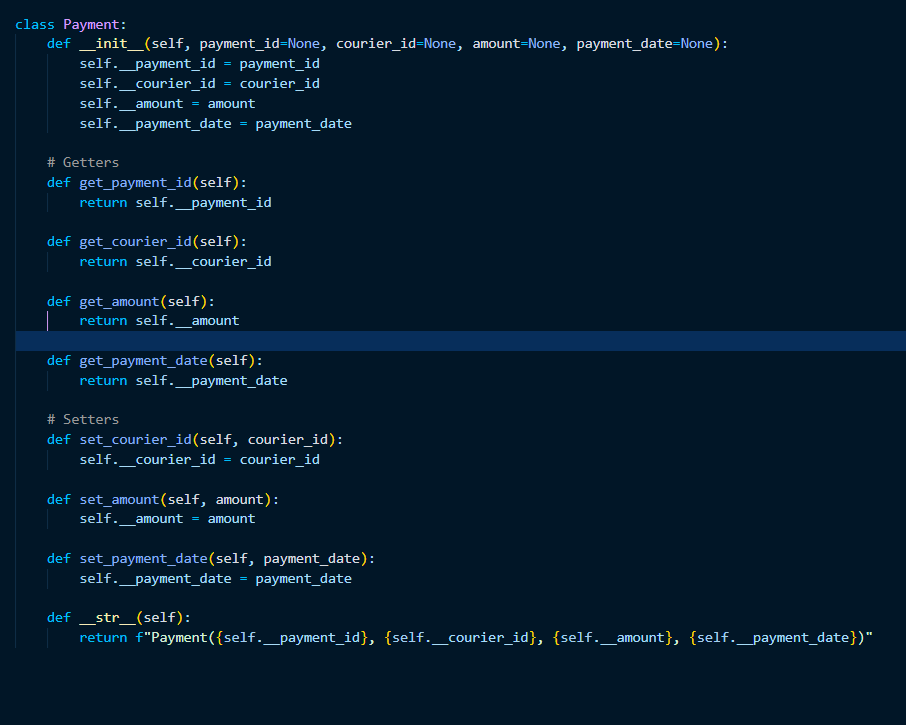
Variables LocationID , LocationName , Address

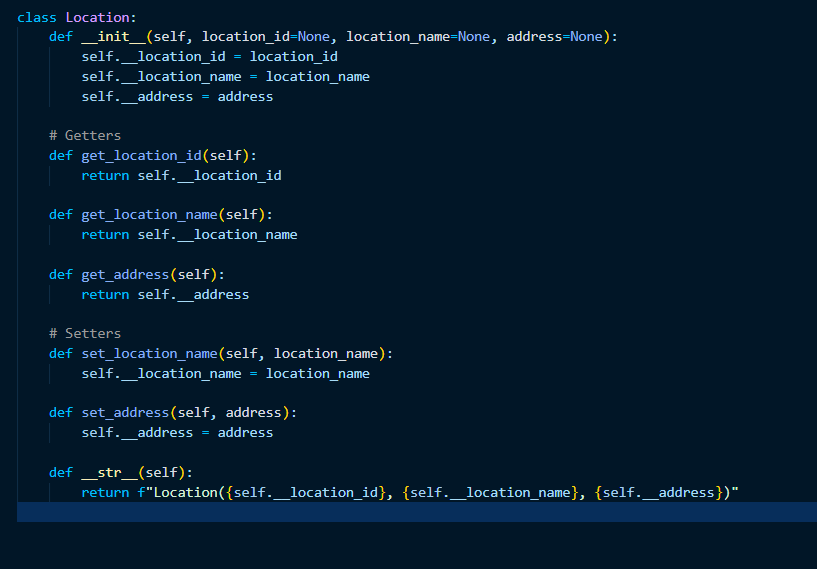
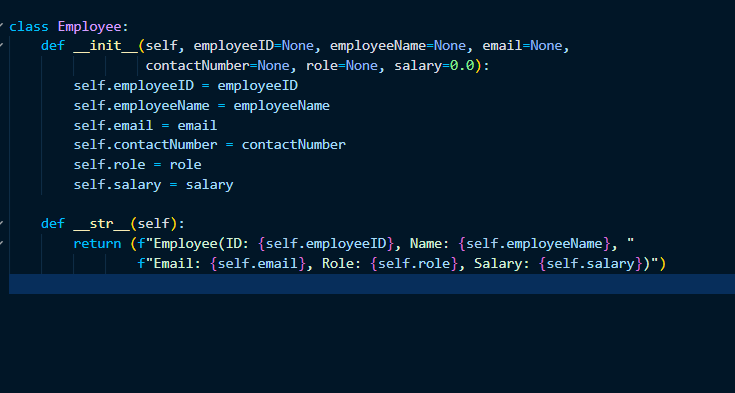
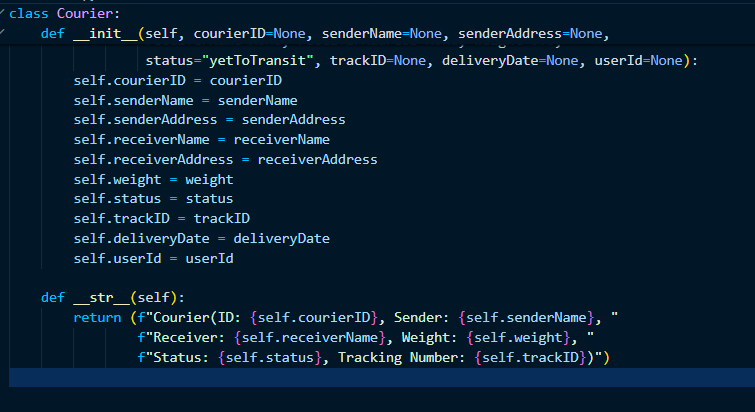
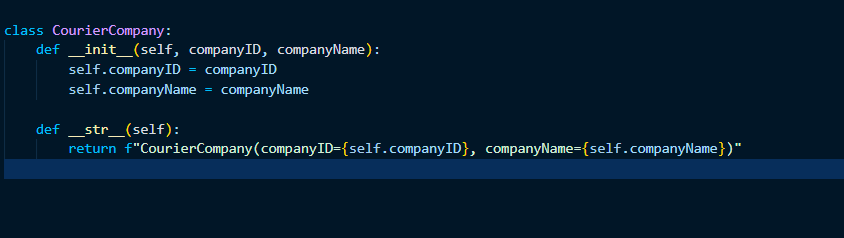
5. CourierCompany Class

Variables companyName , courierDetails -collection of Courier Objects, employeeDetailscollection of Employee Objects, locationDetails - collection of Location Objects.

6. Payment Class:

Variables PaymentID long, CourierID long, Amount double, PaymentDate Date

**Task 6: Service Provider Interface /Abstract class**

Create 2 Interface /Abstract class ICourierUserService and ICourierAdminService interface

ICourierUserService {

// Customer-related functions

placeOrder()

/\*\* Place a new courier order.

\* @param courierObj Courier object created using values entered by users

\* @return The unique tracking number for the courier order .

Use a static variable to generate unique tracking number. Initialize the static variable in Courier

class with some random value. Increment the static variable each time in the constructor to

generate next values.

getOrderStatus();

/\*\*Get the status of a courier order.

\*@param trackingNumber The tracking number of the courier order.

\* @return The status of the courier order (e.g., yetToTransit, In Transit, Delivered).

\*/

cancelOrder()

/\*\* Cancel a courier order.

\* @param trackingNumber The tracking number of the courier order to be canceled.

\* @return True if the order was successfully canceled, false otherwise.\*/

getAssignedOrder();

/\*\* Get a list of orders assigned to a specific courier staff member

\* @param courierStaffId The ID of the courier staff member.

\* @return A list of courier orders assigned to the staff member.\*/

// Admin functions

ICourierAdminService

int addCourierStaff(Employee obj);

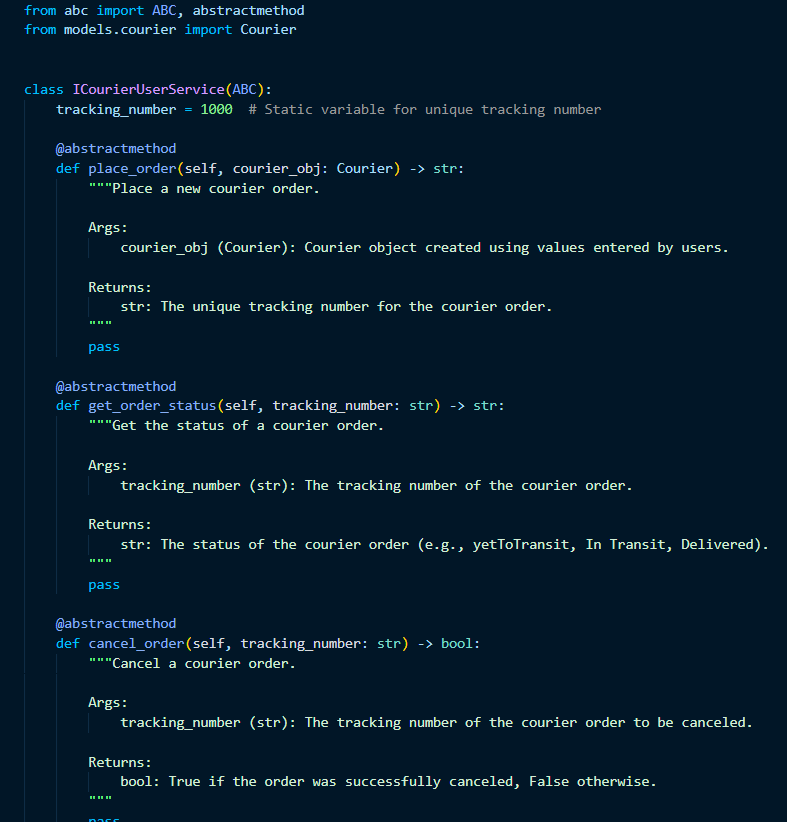
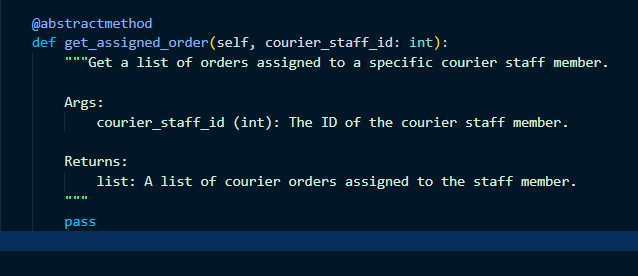
/\*\* Add a new courier staff member to the system.

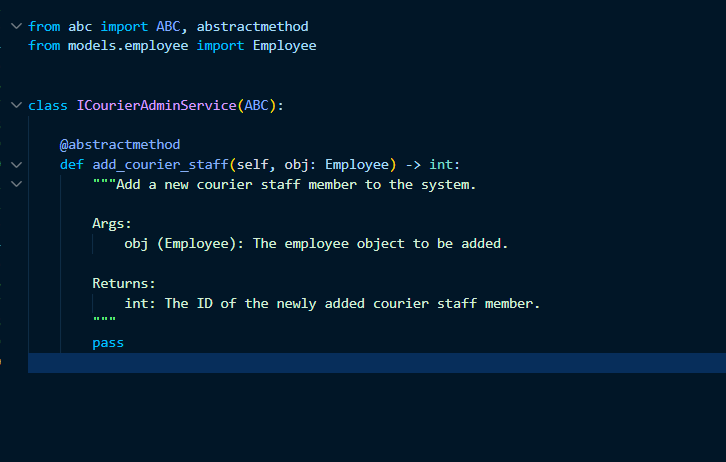
\* @param name The name of the courier staff member.

\* @param contactNumber The contact number of the courier staff member.

\* @return The ID of the newly added courier staff member.

\*/



**Task 7: Exception Handling**

(Scope: User Defined Exception/Checked /Unchecked Exception/Exception handling using try..catch

finally,thow & throws keyword usage)

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

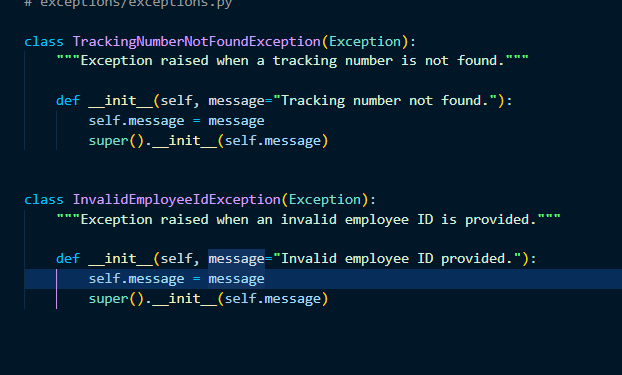
excpetionsin main method,

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



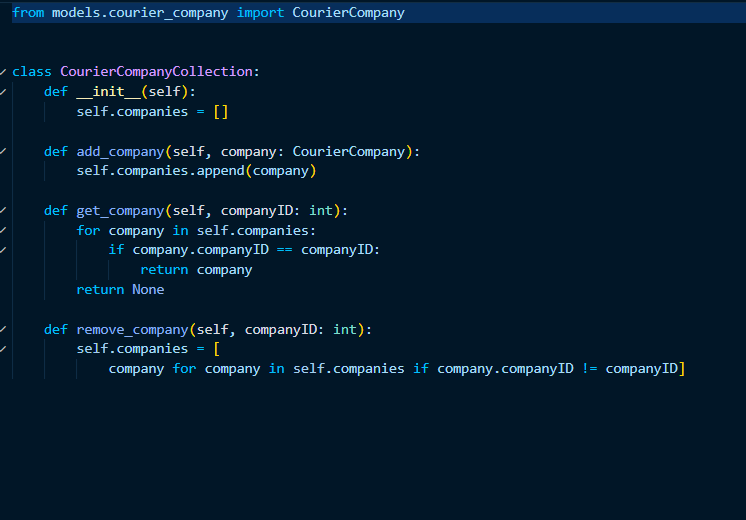
**Task 8: Collections**

Scope: ArrayList/Hashmap

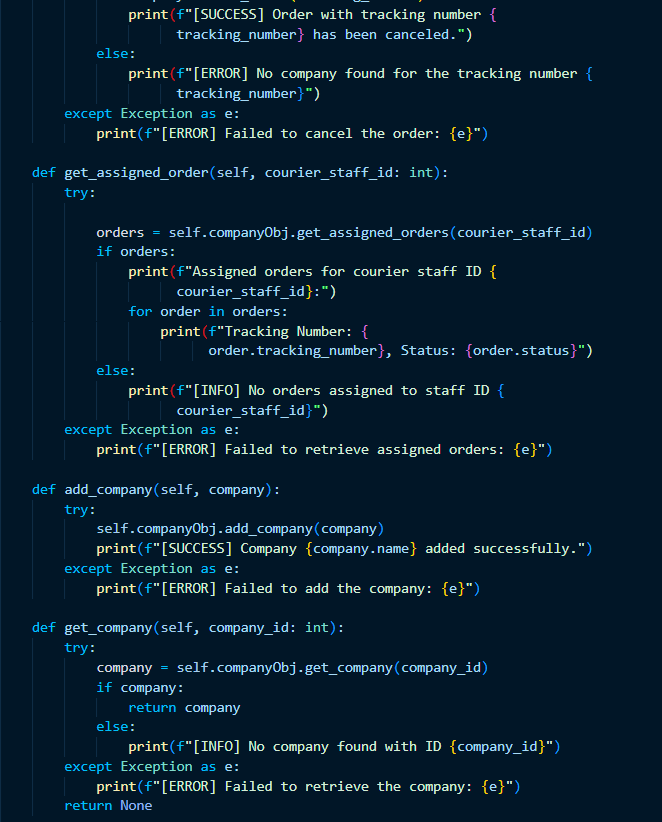
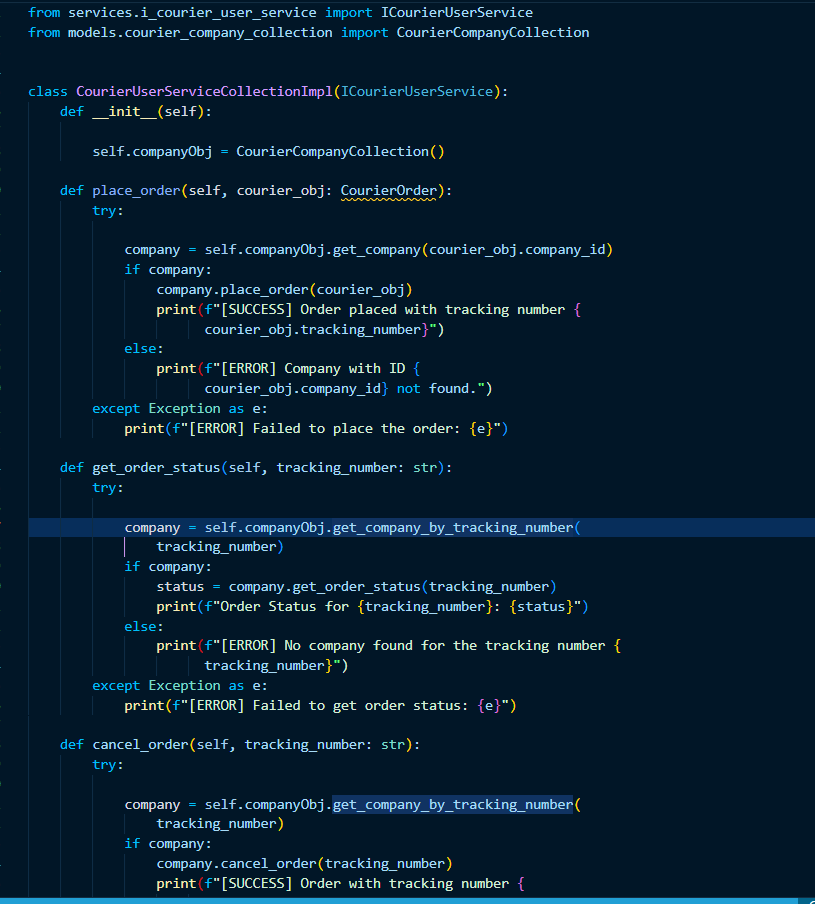
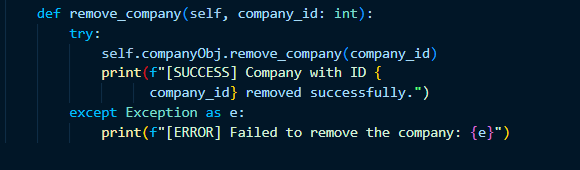
Task: Improve the Courier Management System by using Java collections:

1. Create a new model named CourierCompanyCollection in entity package replacing the Array of

Objects with List to accommodate dynamic updates in the CourierCompany class



2. Create a new implementation class CourierUserServiceCollectionImpl class in package dao which

implements ICourierUserService interface which holds a variable named companyObj of type CourierCompanyCollection  

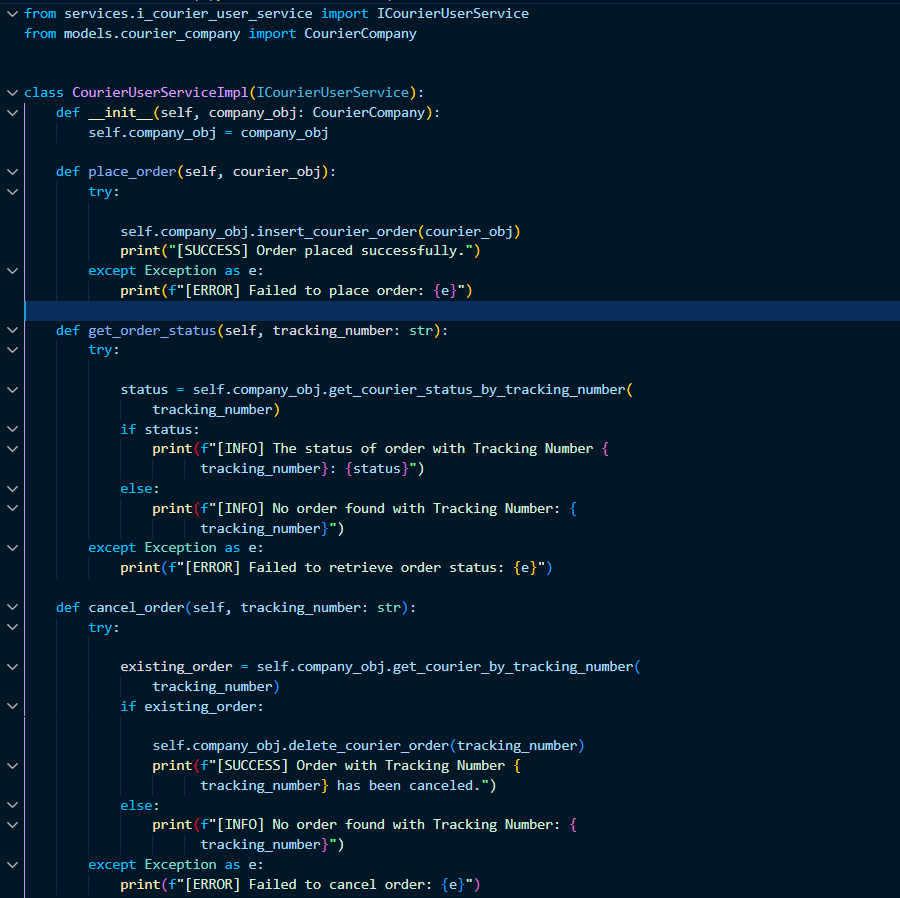
**Task 9: Service implementation**

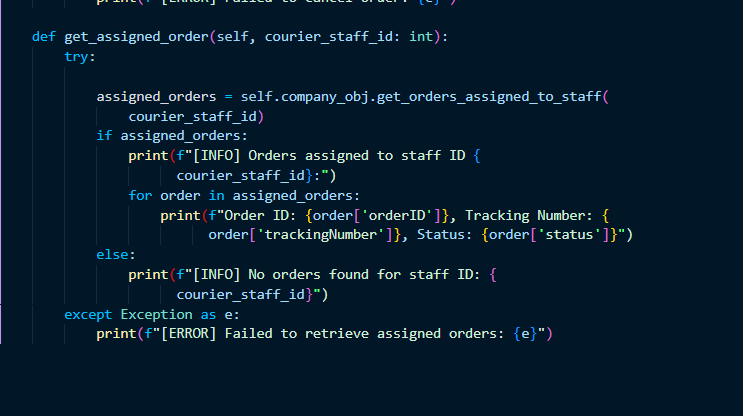
1.Create CourierUserServiceImpl class which implements ICourierUserService interface which

holds a variable named companyObj of type CourierCompany.

This variable can be used to access the Object Arrays to access data relevant in method

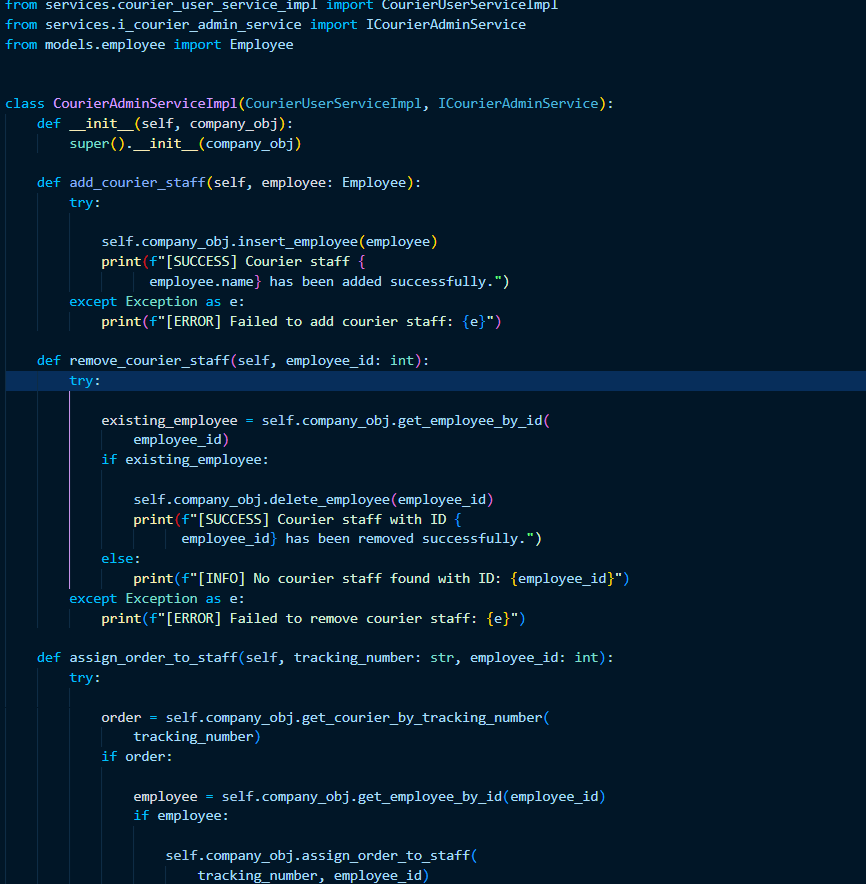
implementations.

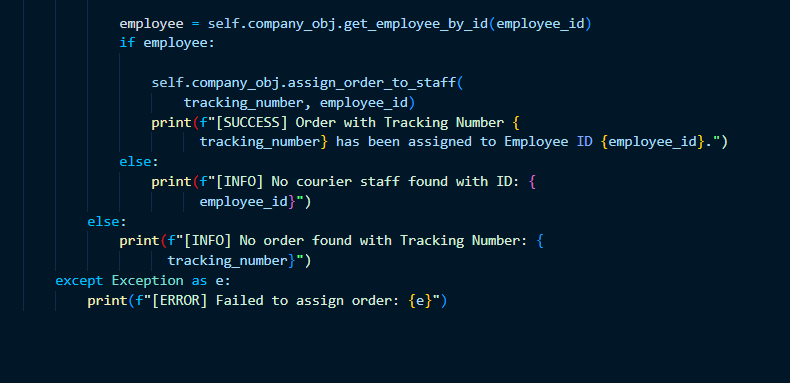




2. Create CourierAdminService Impl class which inherits from CourierUserServiceImpl and

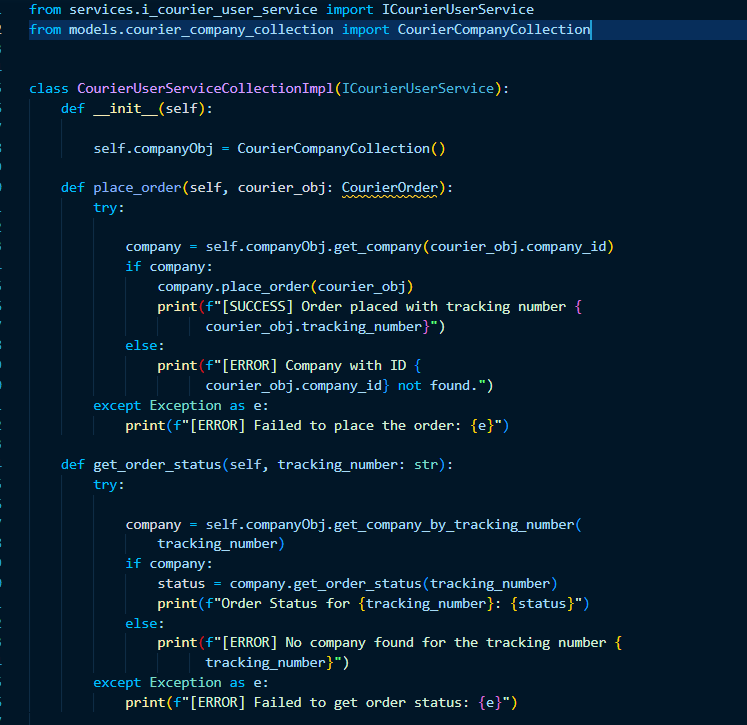
implements ICourierAdminService interface.

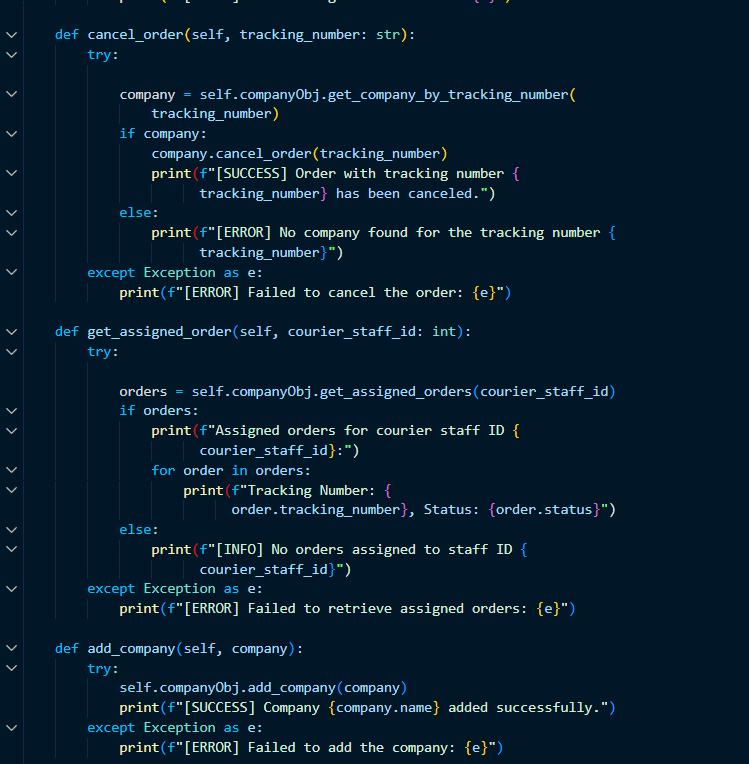


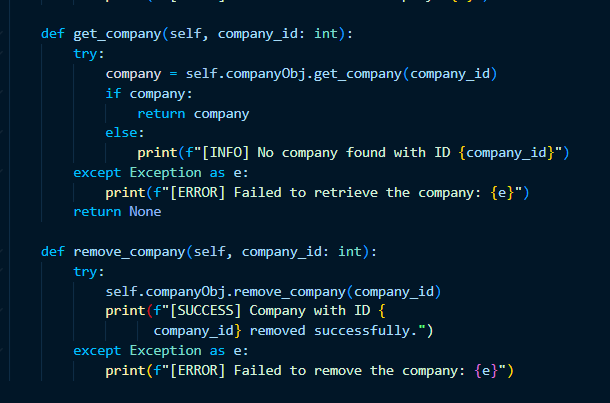


3. Create CourierAdminServiceCollectionImpl class which inherits from

CourierUserServiceColectionImpl and implements ICourierAdminService interface.







**Task 10: Database Interaction**

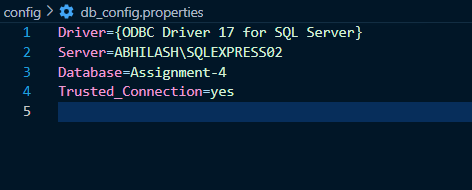
Connect your application to the SQL database for the Courier Management System

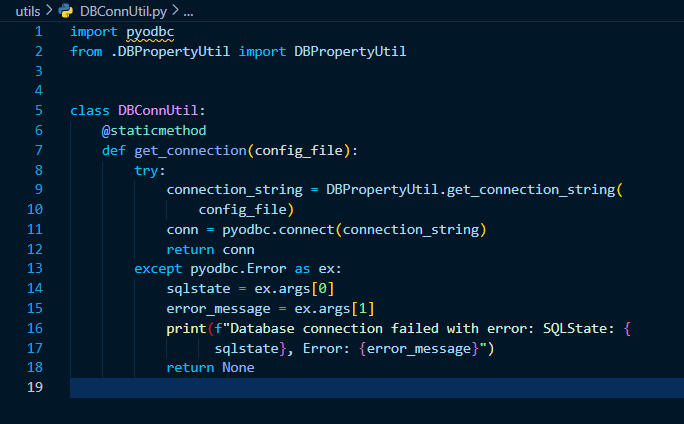
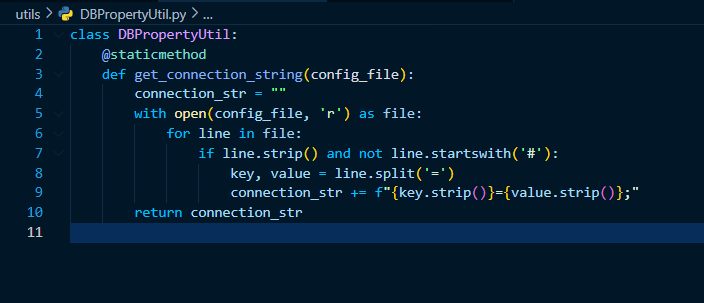
1. Write code to establish a connection to your SQL database.

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.

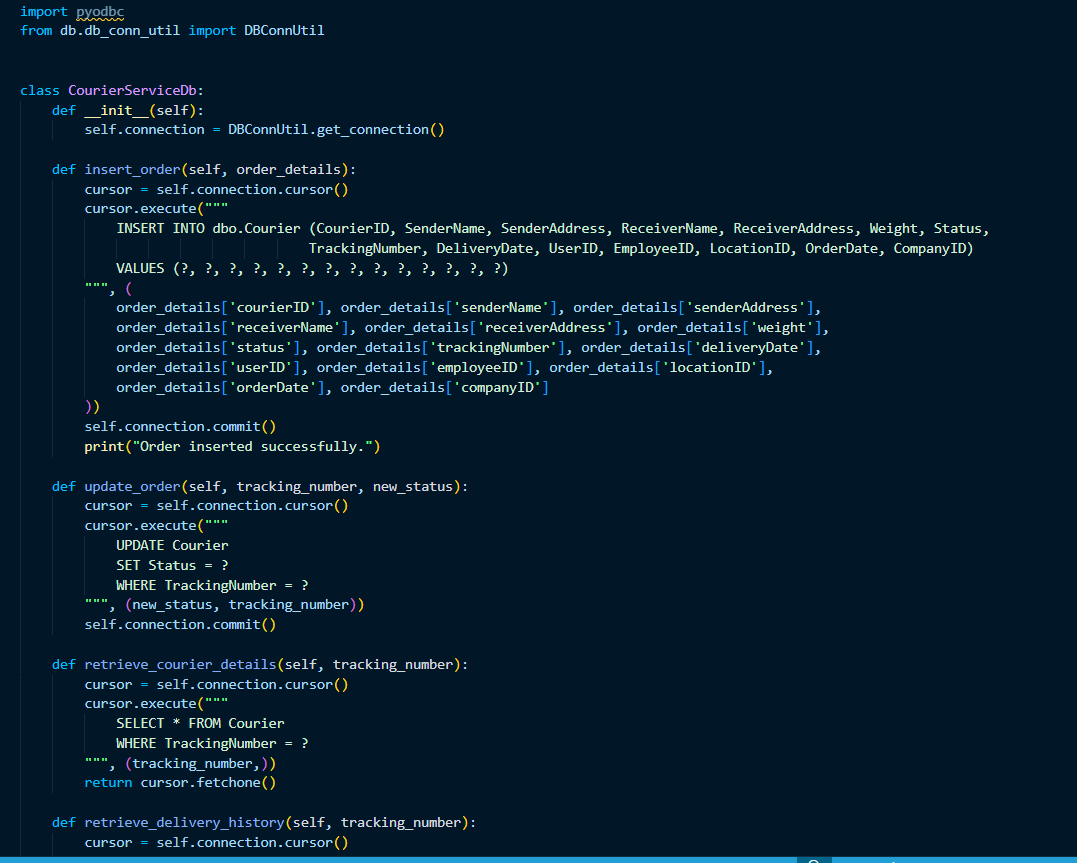
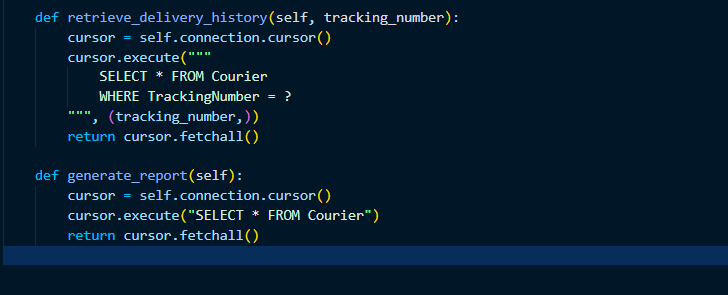


2. Create a Service class CourierServiceDb in dao with a static variable named connection of

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

