Hexaware Technologies

Case Study : Car Rental System
Name - Astha Raj

The main focus of this case study is to implement core functionalities like Car Management, Customer Management, Lease Management and Payment Handling.

Create following tables in SQL Schema with appropriate class and write the unit test case for the Car Rental application.
Schema Design:

-> Created a database named carrentalsystem.

MySQL 8.0 Command Line Client

```
mysql> create database carrentalsystem;
Query OK, 1 row affected (0.00 sec)

mysql> use carrentalsystem;
Database changed
```

1. Vehicle Table:

- vehicleID (Primary Key)
- make
- model
- year
- dailyRate
- status (Available, NotAvailable)
- passengerCapacity
- engineCapacity

```
mysql> CREATE TABLE Vehicle (
    ->     vehicleID INT PRIMARY KEY AUTO_INCREMENT,
    ->     make VARCHAR(100),
    ->     model VARCHAR(100),
    ->     year INT,
    ->     dailyRate DECIMAL(10, 2),
    ->     status ENUM('Available','NotAvailable'),
    ->     passengerCapacity INT,
    ->     engineCapacity INT
    -> );
Query OK, 0 rows affected (0.02 sec)
```

-> Inserted 10 Records also:

```
mysql> select * from vehicle;
+-----------+----------+----------+------+-----------+--------------+-------------------+----------------+
| vehicleID | make     | model    | year | dailyRate | status       | passengerCapacity | engineCapacity |
+-----------+----------+----------+------+-----------+--------------+-------------------+----------------+
|         1 | Toyota   | Camry    | 2022 |     50.00 | Available    |                 4 |           1450 |
|         2 | Honda    | Civic    | 2023 |     45.00 | Available    |                 7 |           1500 |
|         3 | Ford     | Focus    | 2022 |     48.00 | NotAvailable |                 4 |           1400 |
|         4 | Nissan   | Altima   | 2023 |     52.00 | Available    |                 7 |           1200 |
|         5 | Chevrolet | Malibu  | 2022 |     47.00 | Available    |                 4 |           1800 |
|         6 | Hyundai  | Sonata   | 2023 |     49.00 | NotAvailable |                 7 |           1400 |
|         7 | BMW      | 3 Series | 2023 |     60.00 | Available    |                 7 |           2499 |
|         8 | Mercedes | C-Class  | 2022 |     58.00 | Available    |                 8 |           2599 |
|         9 | Audi     | A4       | 2022 |     55.00 | NotAvailable |                 4 |           2500 |
|        10 | Lexus    | ES       | 2023 |     54.00 | Available    |                 4 |           2500 |
+-----------+----------+----------+------+-----------+--------------+-------------------+----------------+
10 rows in set (0.00 sec)
```

2. Customer Table:

- customerID (Primary Key)
- firstName
- lastName
- email
- phoneNumber

```
mysql> CREATE TABLE Customer (
    ->     customerID INT PRIMARY KEY AUTO_INCREMENT,
    ->     firstName VARCHAR(100),
    ->     lastName VARCHAR(100),
    ->     email VARCHAR(100),
    ->     phoneNumber VARCHAR(20)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

-> Inserted 10 Records also:

MySQL 8.0 Command Line Client

```
mysql> select * from customer;
+------------+-----------+----------+----------------------+--------------+
| customerID | firstName | lastName | email                | phoneNumber  |
+------------+-----------+----------+----------------------+--------------+
|          1 | John      | Doe      | johndoe@example.com  | 555-555-5555 |
|          2 | Jane      | Smith    | janesmith@example.com | 555-123-4567 |
|          3 | Robert    | Johnson  | robert@example.com   | 555-789-1234 |
|          4 | Sarah     | Brown    | sarah@example.com    | 555-456-7890 |
|          5 | David     | Lee      | david@example.com    | 555-987-6543 |
|          6 | Laura     | Hall     | laura@example.com    | 555-234-5678 |
|          7 | Michael   | Davis    | michael@example.com  | 555-876-5432 |
|          8 | Emma      | Wilson   | emma@example.com     | 555-432-1098 |
|          9 | William   | Taylor   | william@example.com  | 555-321-6547 |
|         10 | Olivia    | Adams    | olivia@example.com   | 555-765-4321 |
+------------+-----------+----------+----------------------+--------------+
10 rows in set (0.00 sec)
```

3. Lease Table:

- leaseID (Primary Key)
- vehicleID (Foreign Key referencing Vehicle Table)
- customerID (Foreign Key referencing Customer Table)
- startDate
- endDate
- type (to distinguish between DailyLease and MonthlyLease)

```
mysql> CREATE TABLE Lease (
    ->     leaseID INT PRIMARY KEY AUTO_INCREMENT,
    ->     vehicleID INT,
    ->     customerID INT,
    ->     startDate DATE,
    ->     endDate DATE,
    ->     type ENUM('DailyLease', 'MonthlyLease'),
    ->     FOREIGN KEY (vehicleID) REFERENCES Vehicle(vehicleID),
    ->     FOREIGN KEY (customerID) REFERENCES Customer(customerID)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

-> Inserted 10 Records

```
mysql> select * from lease;
+---------+-----------+------------+------------+------------+--------------+
| leaseID | vehicleID | customerID | startDate  | endDate    | type         |
+---------+-----------+------------+------------+------------+--------------+
|       1 |         1 |          1 | 2023-01-01 | 2023-01-05 | DailyLease   |
|       2 |         2 |          2 | 2023-02-15 | 2023-02-28 | MonthlyLease |
|       3 |         3 |          3 | 2023-03-10 | 2023-03-15 | DailyLease   |
|       4 |         4 |          4 | 2023-04-20 | 2023-04-30 | MonthlyLease |
|       5 |         5 |          5 | 2023-05-05 | 2023-05-10 | DailyLease   |
|       6 |         4 |          3 | 2023-06-15 | 2023-06-30 | MonthlyLease |
|       7 |         7 |          7 | 2023-07-01 | 2023-07-10 | DailyLease   |
|       8 |         8 |          8 | 2023-08-12 | 2023-08-15 | MonthlyLease |
|       9 |         3 |          3 | 2023-09-07 | 2023-09-10 | DailyLease   |
|      10 |        10 |         10 | 2023-10-10 | 2023-10-31 | MonthlyLease |
+---------+-----------+------------+------------+------------+--------------+
10 rows in set (0.00 sec)
```

4. Payment Table:

- paymentID (Primary Key)
- leaseID (Foreign Key referencing Lease Table)
- paymentDate
- amount

```
mysql> CREATE TABLE Payment (
    ->     paymentID INT PRIMARY KEY AUTO_INCREMENT,
    ->     leaseID INT,
    ->     paymentDate DATE,
    ->     amount DECIMAL(10, 2),
    ->     FOREIGN KEY (leaseID) REFERENCES Lease(leaseID)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> _
```

-> Inserted 10 records also

MySQL 8.0 Command Line Client

```
mysql> select * from payment
    -> ;
+-----------+---------+-------------+---------+
| paymentID | leaseID | paymentDate | amount  |
+-----------+---------+-------------+---------+
|         1 |       1 | 2023-01-03  |  200.00 |
|         2 |       2 | 2023-02-20  | 1000.00 |
|         3 |       3 | 2023-03-12  |   75.00 |
|         4 |       4 | 2023-04-12  |  900.00 |
|         5 |       5 | 2023-05-12  |   60.00 |
|         6 |       6 | 2023-06-12  | 1200.00 |
|         7 |       7 | 2023-07-12  |   40.00 |
|         8 |       8 | 2023-08-14  | 1100.00 |
|         9 |       9 | 2023-09-09  |   80.00 |
|        10 |      10 | 2023-10-25  | 1500.00 |
+-----------+---------+-------------+---------+
10 rows in set (0.00 sec)
```

The following **Directory structure** is to be followed in the application.

• **entity/model**

• Create entity classes in this package. All entity class should not have any business logic.

• **dao**

• Create Service Provider interface to showcase functionalities. • Create the implementation class for the above interface with db interaction.

• **exception**

• Create user defined exceptions in this package and handle exceptions whenever needed.

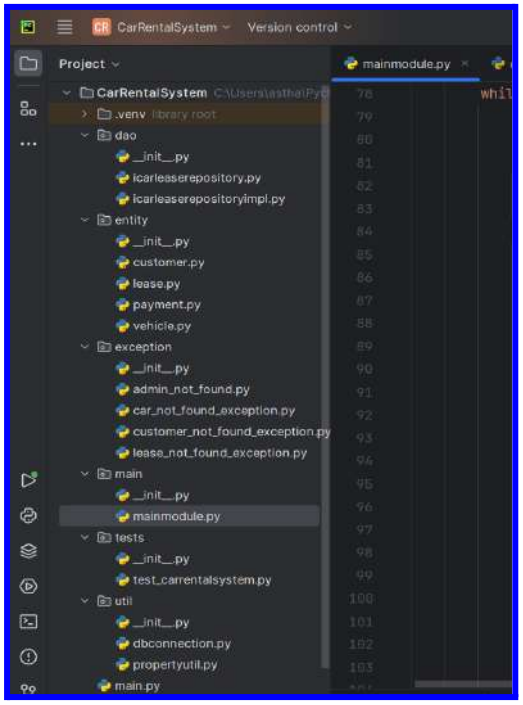
• **util**

• Create a DBPropertyUtil class with a static function which takes property file name as parameter and returns connection string.

• Create a DBConnUtil class which holds static method which takes connection string as parameter file and returns connection object(Use method defined in DBPropertyUtil class to get the connection String).

• **main**

• Create a class MainModule and demonstrate the functionalities in a menu driven application.



5. Create the model/entity classes corresponding to the schema within package entity with variables declared private, constructors(default and parametrized) and getters,setters)

-> Customer.

customer.py × lease.py payment.py vehicle.py

```python
12 usages
class Customer:
    def __init__(self, customerID, firstName, lastName, email, phoneNumber):
        self.__customerID = customerID
        self.__firstName = firstName
        self.__lastName = lastName
        self.__email = email
        self.__phoneNumber = phoneNumber

    1 usage
    @property
    def customerID(self):
        return self.__customerID

    4 usages
    @property
    def firstName(self):
        return self.__firstName

    @firstName.setter
    def firstName(self, first_name):
        self.__firstName = first_name
```

customer.py × lease.py payment.py vehicle.py

```python
    4 usages
    @property
    def lastName(self):
        return self.__lastName

    @lastName.setter
    def lastName(self, last_name):
        self.__lastName = last_name

    4 usages
    @property
    def email(self):
        return self.__email

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

    3 usages
    @property
    def phoneNumber(self):
        return self.__phoneNumber

    @phoneNumber.setter
    def phoneNumber(self, phone_number):
        self.__phoneNumber = phone_number
```

customer

-> Lease

```python
18 usages
class Lease:
    def __init__(self, leaseID, vehicleID, customerID, startDate, endDate, type):
        self.__leaseID = leaseID
        self.__vehicleID = vehicleID
        self.__customerID = customerID
        self.__startDate = startDate
        self.__endDate = endDate
        self.__type = type


    8 usages
    @property
    def leaseID(self):
        return self.__leaseID

    1 usage
    @property
    def vehicleID(self):
        return self.__vehicleID

    1 usage
    @property
    def customerID(self):
        return self.__customerID
```

```python
    @property
    def startDate(self):
        return self.__startDate

    @startDate.setter
    def startDate(self, set_startDate):
        self.__startDate = set_startDate

    3 usages
    @property
    def endDate(self):
        return self.__endDate

    @endDate.setter
    def endDate(self, set_endDate):
        self.__endDate = set_endDate

    2 usages
    @property
    def type(self):
        return self.__type

    @type.setter
    def type(self, set_type):
        self.__type = set_type
```

->Payment

customer.py lease.py payment.py vehicle.py

```python
5 usages
class Payment:
    def __init__(self, paymentID, leaseID, paymentDate, amount):
        self.__paymentID = paymentID
        self.__leaseID = leaseID
        self.__paymentDate = paymentDate
        self.__amount = amount

    1 usage
    @property
    def paymentID(self):
        return self.__paymentID

    @property
    def leaseID(self):
        return self.__leaseID

    2 usages
    @property
    def paymentDate(self):
        return self.__paymentDate

    @paymentDate.setter
    def paymentDate(self, payment_date):
        self.__paymentDate = payment_date
```

customer.py lease.py payment.py vehicle.py

```python
    @property
    def leaseID(self):
        return self.__leaseID

    2 usages
    @property
    def paymentDate(self):
        return self.__paymentDate

    @paymentDate.setter
    def paymentDate(self, payment_date):
        self.__paymentDate = payment_date

    2 usages
    @property
    def amount(self):
        return self.__amount

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

->Vehicle

```python
17 usages
class Vehicle:
    def __init__(self, vehicleID, make, model, year, dailyRate, status, passengerCapacity, engineCapacity):
        self.__vehicleID = vehicleID
        self.__make = make
        self.__model = model
        self.__year = year
        self.__dailyRate = dailyRate
        self.__status = status
        self.__passengerCapacity = passengerCapacity
        self.__engineCapacity = engineCapacity

    4 usages
    @property
    def vehicleID(self):
        return self.__vehicleID

    5 usages
    @property
    def make(self):
        return self.__make

    @make.setter
    def make(self, set_make):
        self.__make = set_make
```

```python
    @property
    def model(self):
        return self.__model

    @model.setter
    def model(self, set_model):
        self.__model = set_model

    2 usages
    @property
    def year(self):
        return self.__year

    @year.setter
    def year(self, set_year):
        self.__year = set_year

    2 usages
    @property
    def dailyRate(self):
        return self.__dailyRate

    @dailyRate.setter
    def dailyRate(self, set_dailyRate):
        self.__dailyRate = set_dailyRate
```

customer.py lease.py payment.py vehicle.py

```python
    @property
    def status(self):
        return self.__status

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

    2 usages
    @property
    def passengerCapacity(self):
        return self.__passengerCapacity

    @passengerCapacity.setter
    def passengerCapacity(self, passenger_capacity):
        self.__passengerCapacity = passenger_capacity

    2 usages
    @property
    def engineCapacity(self):
        return self.__engineCapacity

    @engineCapacity.setter
    def engineCapacity(self, engine_capacity):
        self.__engineCapacity = engine_capacity
```

6. Service Provider Interface/Abstract class:

Keep the interfaces and implementation classes in package dao

• Create Interface for ICarLeaseRepository and add following methods which interact with database.

• Car Management

1. **addCar(Car car)**
   parameter : Car
   return type : void
2. **removeCar()**
   parameter : carID
   return type : void
3. **listAvailableCars() -**
   parameter: NIL
   return type: return List of Car
4. **listRentedCars() – return List of Car**
   parameter: NIL
   return type: return List of Car

5. findCarById(int carID) - return Car if found or throw exception

parameter: NIL
return type: return List of Car

- **Customer Management**

1. addCustomer(Customer customer)

parameter : Customer
return type : void

2. void removeCustomer(int customerID)

parameter : CustomerID
return type : void

3. listCustomers()

parameter : NIL
return type : list of customer

4. findCustomerById(int customerID)

parameter : CustomerID
return type : Customer

- **Lease Management**

1. createLease()

parameter : int customerID, int carID, Date startDate, Date endDate
return type : Lease

2. void returnCar();

parameter : int leaseID
return type : Lease info

3. List listActiveLeases();

parameter : NIL
return type : Lease list

4. listLeaseHistory();

parameter : NIL
return type : Lease list

- **Payment Handling**

  1. void recordPayment();

  parameter : Lease lease, double amount

  return type : void

```python
from abc import ABC, abstractmethod
from typing import List
from entity.payment import Payment
from entity.vehicle import Vehicle
from entity.customer import Customer
from entity.lease import Lease


2 usages
class ICarLeaseRepository(ABC):
    @abstractmethod
    def addCar(self, car: Vehicle):
        pass

    @abstractmethod
    def removeCar(self, carID: int):
        pass

    @abstractmethod
    def listAvailableCars(self) -> List[Vehicle]:
        pass

    @abstractmethod
    def listRentedCars(self) -> List[Vehicle]:
        pass
```

customer.py lease.py payment.py vehicle.py icarleaserepository.py

```python
        pass

    @abstractmethod
    def findCarById(self, carID: int) -> Vehicle:
        pass

    @abstractmethod
    def addCustomer(self, customer: Customer):
        pass

    @abstractmethod
    def removeCustomer(self, customerID: int):
        pass

    @abstractmethod
    def listCustomers(self) -> List[Customer]:
        pass

    @abstractmethod
    def findCustomerById(self, customerID: int) -> Customer:
        pass

    @abstractmethod
    def createLease(self, customerID: int, carID: int, startDate, endDate, type) -> Lease:
        pass
```

ntrol mainmodule

customer.py lease.py payment.py vehicle.py icarleaserepository.py

```python
        pass

    @abstractmethod
    def returnCar(self, leaseID: int) -> Lease:
        pass

    @abstractmethod
    def findLeaseById(self, leaseID: int) -> Lease:
        pass

    @abstractmethod
    def listActiveLeases(self) -> List[Lease]:
        pass

    @abstractmethod
    def listLeaseHistory(self) -> List[Lease]:
        pass

    @abstractmethod
    def recordPayment(self, lease: Lease, amount: float):
        pass

    @abstractmethod
    def retrievePaymentHistory(self, leaseID: int) -> List[Payment]:
        pass
```

```python
        def findLeaseById(self, leaseID: int) -> Lease:
            pass

    @abstractmethod
    def listActiveLeases(self) -> List[Lease]:
        pass

    @abstractmethod
    def listLeaseHistory(self) -> List[Lease]:
        pass

    @abstractmethod
    def recordPayment(self, lease: Lease, amount: float):
        pass

    @abstractmethod
    def retrievePaymentHistory(self, leaseID: int) -> List[Payment]:
        pass

    @abstractmethod
    def calculateTotalRevenue(self) -> float:
        pass
```

7. Implement the above interface in a class called ICarLeaseRepositoryImpl in package dao.

```python
4 usages
class ICarLeaseRepositoryImpl(ICarLeaseRepository):
    def __init__(self, connection=None):
        self.connection = connection

    2 usages
    def addCar(self, car: Vehicle):
        cursor = self.connection.cursor()
        try:
            cursor.execute(
                "INSERT INTO vehicle (make, model, year, dailyRate, status, passengerCapacity, engineCapacity) "
                "VALUES (%s, %s, %s, %s, %s, %s, %s)",
                (car.make, car.model, car.year, car.dailyRate, car.status,
                 car.passengerCapacity, car.engineCapacity))
            self.connection.commit()
        finally:
            cursor.close()

    1 usage
    def removeCar(self, carID: int):
        cursor = self.connection.cursor()
        try:
            # Check if the car has active leases
            active_leases = self.listActiveLeases()
            active_leases_for_car = [lease for lease in active_leases if lease.vehicleID == carID]
```

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

```python
            active_leases_for_car = [lease for lease in active_leases if lease.vehicleID == carID]

            if active_leases_for_car:
                raise ActiveLeasesExistException("Cannot remove car with active leases.")

            cursor.execute("DELETE FROM vehicle WHERE vehicleID=%s", (carID,))
            if cursor.rowcount == 0:
                raise CarNotFoundException(f"Car with ID {carID} not found.")
            self.connection.commit()
        finally:
            cursor.close()

    1 usage
    def listAvailableCars(self) -> List[Vehicle]:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM vehicle WHERE status='available'")
            rows = cursor.fetchall()

            cars = []
            for row in rows:
                car = Vehicle(row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7])
                cars.append(car)

            return cars
        finally:
```

```python
    1 usage
    def listRentedCars(self) -> List[Vehicle]:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM vehicle WHERE status='notAvailable'")
            rows = cursor.fetchall()

            cars = []
            for row in rows:
                car = Vehicle(row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7])
                cars.append(car)

            return cars
        finally:
            cursor.close()
```

```python
    def findCarById(self, carID: int) -> Vehicle:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM vehicle WHERE vehicleID=%s", (carID,))
            row = cursor.fetchone()

            if row:
                return Vehicle(row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7])
            else:
                raise CarNotFoundException(f"Car with ID {carID} not found.")
        finally:
            cursor.close()

    1 usage
    def addCustomer(self, customer: Customer):
        cursor = self.connection.cursor()
        try:
            cursor.execute("INSERT INTO customer (firstName, lastName, email, phoneNumber) VALUES (%s, %s, %s, %s)",
                           (customer.firstName, customer.lastName, customer.email, customer.phoneNumber))
            self.connection.commit()
        finally:
            cursor.close()
```

```python
    1 usage
    def removeCustomer(self, customerID: int):
        cursor = self.connection.cursor()
        try:
            # Check if the customer has active leases
            active_leases = self.listActiveLeases()
            active_leases_for_customer = [lease for lease in active_leases if lease.customerID == customerID]

            if active_leases_for_customer:
                raise ActiveLeasesExistException("Cannot remove customer with active leases.")

            cursor.execute("DELETE FROM customer WHERE customerID=%s", (customerID,))
            self.connection.commit()
        finally:
            cursor.close()

```

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

```python
    def listCustomers(self) -> List[Customer]:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM customer")
            rows = cursor.fetchall()

            customers = []
            for row in rows:
                customer = Customer(row[0], row[1], row[2], row[3], row[4])
                customers.append(customer)

            return customers
        finally:
            cursor.close()

    3 usages
    def findCustomerById(self, customerID: int) -> Customer:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM customer WHERE customerID=%s", (customerID,))
            row = cursor.fetchone()

            if row:
                return Customer(row[0], row[1], row[2], row[3], row[4])
            else:
                raise CustomerNotFoundException(f"Customer with ID {customerID} not found.")
```

ICarLeaseRepositoryImpl › update_customer_info() › try › if email is not None

mainmodule

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

```python
            cursor.close()

    2 usages
    def createLease(self, customerID: int, carID: int, startDate, endDate, lease_type) -> Lease:
        cursor = self.connection.cursor()
        try:
            cursor.execute(
                "INSERT INTO lease (vehicleID, customerID, startDate, endDate, type ) VALUES (%s, %s, %s, %s, %s)",
                (carID, customerID, startDate, endDate, lease_type))
            self.connection.commit()
            cursor.execute("SELECT LAST_INSERT_ID()")
            leaseID = cursor.fetchone()[0]

            return Lease(leaseID, carID, customerID, startDate, endDate,
                         lease_type)
        finally:
            cursor.close()
```

mainmodule

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

```python
    1 usage
    def returnCar(self, leaseID: int) -> Lease:
        cursor = self.connection.cursor()
        try:
            cursor.execute("UPDATE lease SET endDate = %s WHERE leaseID = %s",
                           (datetime.now().date(), leaseID))
            self.connection.commit()
            return self.findLeaseById(leaseID)
        finally:
            cursor.close()

    6 usages
    def findLeaseById(self, leaseID: int) -> Lease:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM lease WHERE leaseID = %s", (leaseID,))
            result = cursor.fetchone()

            if result:
                lease_id, car_id, customer_id, start_date, end_date, type = result
                return Lease(lease_id, car_id, customer_id, start_date, end_date, type)
            else:
                raise LeaseNotFoundException(f"Lease with ID {leaseID} not found.")
        finally:
            cursor.close()
```

mainmodule

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

```python
        try:
            cursor.execute("SELECT * FROM customer")
            rows = cursor.fetchall()

            customers = []
            for row in rows:
                customer = Customer(row[0], row[1], row[2], row[3], row[4])
                customers.append(customer)

            return customers
        finally:
            cursor.close()

    3 usages
    def findCustomerById(self, customerID: int) -> Customer:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM customer WHERE customerID=%s", (customerID,))
            row = cursor.fetchone()

            if row:
                return Customer(row[0], row[1], row[2], row[3], row[4])
            else:
                raise CustomerNotFoundException(f"Customer with ID {customerID} not found.")
        finally:
            cursor.close()
```

mainmodule

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl.py

2 usages

```python
def createLease(self, customerID: int, carID: int, startDate, endDate, lease_type) -> Lease:
    cursor = self.connection.cursor()
    try:
        cursor.execute(
            "INSERT INTO lease (vehicleID, customerID, startDate, endDate, type ) VALUES (%s, %s, %s, %s, %s)",
            (carID, customerID, startDate, endDate, lease_type))
        self.connection.commit()
        cursor.execute("SELECT LAST_INSERT_ID()")
        leaseID = cursor.fetchone()[0]

        return Lease(leaseID, carID, customerID, startDate, endDate,
                     lease_type)
    finally:
        cursor.close()

1 usage
def returnCar(self, leaseID: int) -> Lease:
    cursor = self.connection.cursor()
    try:
        cursor.execute("UPDATE lease SET endDate = %s WHERE leaseID = %s",
                       (datetime.now().date(), leaseID))
        self.connection.commit()
        return self.findLeaseById(leaseID)
    finally:
        cursor.close()
```

mainmodule

customer.py | lease.py | payment.py | vehicle.py | icarleaserepository.py | icarleaserepositoryimpl

```python
        cursor.close()

6 usages
def findLeaseById(self, leaseID: int) -> Lease:
    cursor = self.connection.cursor()
    try:
        cursor.execute("SELECT * FROM lease WHERE leaseID = %s", (leaseID,))
        result = cursor.fetchone()

        if result:
            lease_id, car_id, customer_id, start_date, end_date, type = result
            return Lease(lease_id, car_id, customer_id, start_date, end_date, type)
        else:
            raise LeaseNotFoundException(f"Lease with ID {leaseID} not found.")
    finally:
        cursor.close()
```

```python
    def listActiveLeases(self) -> List[Lease]:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM lease WHERE startDate <= CURDATE() AND endDate >= CURDATE()")
            rows = cursor.fetchall()
            leases = []
            for row in rows:
                lease = Lease(row[0], row[1], row[2], row[3], row[4], row[5])
                leases.append(lease)

            return leases
        finally:
            cursor.close()

    1 usage
    def listLeaseHistory(self) -> List[Lease]:
        if not self.connection:
            raise ValueError("Connection is not provided.")
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM lease")
            rows = cursor.fetchall()

            leases = []
            for row in rows:
                lease = Lease(row[0], row[1], row[2], row[3], row[4], row[5])
```

ICarLeaseRepositoryImpl

```python
    1 usage
    def recordPayment(self, lease: Lease, amount: float):
        cursor = self.connection.cursor()
        try:
            print("Lease id", lease.leaseID)
            cursor.execute("INSERT INTO payment (leaseID, paymentDate, amount) VALUES (%s, %s, %s)",
                           (lease.leaseID, datetime.now().strftime("%Y-%m-%d"), amount))
            self.connection.commit()
        finally:
            cursor.close()

    1 usage
    def retrievePaymentHistory(self, leaseID: int) -> List[Payment]:
        cursor = self.connection.cursor()
        try:
            cursor.execute("SELECT * FROM Payment WHERE leaseID = %s", (leaseID,))
            rows = cursor.fetchall()
            payment_history = []
            for row in rows:
                payment = Payment(row[0], row[1], row[2], row[3])
                payment_history.append(payment)
            self.connection.commit()
            return payment_history
        except Exception as e:
            raise LeaseNotFoundException(f"Lease with ID {leaseID} not found.") from e
```

rLeaseRepositoryImpl

```python
1 usage
def calculateTotalRevenue(self) -> float:
    cursor = self.connection.cursor()
    try:
        cursor.execute("SELECT SUM(amount) FROM Payment")
        total_revenue = cursor.fetchone()[0]
        return total_revenue
    finally:
        cursor.close()
```

ICarLeaseRepositoryImpl

8. Connect your application to the SQL database and write code to establish a connection to your SQL database.

• Create a utility class DBConnection in a package util 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 f ile. • Create a utility class PropertyUtil which contains a static method named getPropertyString() which reads a property fie containing connection details like hostname, dbname, username, password, port number and returns a connection string.

hicle.py | icarleaserepository.py | test_carrentalsystem.py | customer.py | lease.py | dbconnection.py

```python
import mysql.connector
from mysql.connector import Error
from util.propertyutil import propertyUtil


9 usages
class DBConnection:

    connection = None

    2 usages
    @staticmethod
    def getConnection():
        if DBConnection.connection is None:
            try:
                connection_string = propertyUtil.getPropertyString()
                DBConnection.connection = mysql.connector.connect(**connection_string)

                if DBConnection.connection.is_connected():
                    print("Database connected successfully")

            except Error as e:
                print(f'Error : {e}')

        return DBConnection.connection
```

```python
2 usages
class propertyUtil:

    1 usage
    @staticmethod
    def getPropertyString():
        connection_string = {
            'host': "localhost",
            'database': "carrentalsystem",
            'user': "root",
            'password': "ABcd300#$"
        }

        return connection_string
```

9. Create the exceptions in package myexceptions and create the following custom exceptions and throw them in methods whenever needed. Handle all the exceptions in main method,

• CarNotFoundException: throw this exception when user enters an invalid car id which doesn't exist in db.

• LeaseNotFoundException: throw this exception when user enters an invalid lease id which doesn't exist in db.

• CustomerrNotFoundException: throw this exception when user enters an invalid customer id which doesn't exist in db.

```python
9 usages
class CarNotFoundException(Exception):
    pass
```

```python
9 usages
class CustomerNotFoundException(Exception):
    pass
```

```python
9 usages
class LeaseNotFoundException(Exception):
    pass


6 usages
class ActiveLeasesExistException(Exception):
    pass
```

• Create a class MainModule and demonstrate the functionalities in a menu driven application.

-> OUTPUTS OF CAR RENTAL SYSTEM

```
C:\Users\astha\PycharmProjects\CarRentalSystem\.venv\Scripts\python.exe C:\Users\astha\Pychar
Database connected successfully

You are welcome in Car Rental System
1. Admin Menu
2. User Menu
0. Exiting
Enter choice from (0-2)1
Enter the password to get admin access: astha
Error: Admin Not Found

You are welcome in Car Rental System
1. Admin Menu
2. User Menu
0. Exiting
Enter choice from (0-2)1
Enter the password to get admin access: admin@123
Correct Password: Welcome!
```

```
**************************
1. Add Customer
2. Remove Customer
3. Add a car
4. Remove a car
5. Find Customer By ID
6. Find Car By ID
7. Find lease by ID
8. List all Customers
9. Create Lease
10. List Active Leases
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)1
Add a Customer:
Enter First Name: Astha
Enter Last Name: Raj
Enter Email: astha@gmail.com
Enter Phone Number: 125-896-7469
Customer added successfully.
```

```
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)2
Remove a Customer:
Enter Customer ID to remove: 11
Customer removed successfully.
```

```
3. Add a car
4. Remove a car
5. Find Customer By ID
6. Find Car By ID
7. Find lease by ID
8. List all Customers
9. Create Lease
10. List Active Leases
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)3
Adding a new Car:
Enter Make: Audi
Enter Model: A5
Enter Year: 2023
Enter Daily Rate: 55.00
Enter Status (Available/NotAvailable): NotAvailable
Enter Passenger Capacity: 4
Enter Engine Capacity: 3500
Great! Car added successfully. None
```

```
13. Update Customer
0. exit
Enter choice from (0-13)4
Remove a Car:
Enter Car ID to remove: 11
Car removed successfully.
```

```
Project    mainmodule.py
Run    mainmodule    propertyutil
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)4
Remove a Car:
Enter Car ID to remove: 16
Error: Car with ID 16 not found.

 Admin Menu
```

```
CR CarRentalSystem    Version control
Project    mainmodule.py
Run    mainmodule    propertyutil
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)5
Find Customer by ID:
Enter Customer ID to find: 10
Customer Found: Olivia Adams - Email: olivia@example.com

 Admin Menu
**************************
```

```
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)6
Find Car by ID:
Enter Car ID to find: 5
Car Found: Chevrolet Malibu - Status: Available

 Admin Menu
**************************
1. Add Customer
2. Remove Customer
```

```
13. Update Customer
0. exit
Enter choice from (0-13)7
Finding lease by id
Enter lease ID: 10
Lease found:
Lease ID:,10, leaese-type: MonthlyLease

 Admin Menu
**************************
```

```
0. exit
Enter choice from (0-13)8
List Customers:
Customers:
1. John Doe - Email: johndoe@example.com,  PhoneNumber: 555-555-5555
2. Jane Smith - Email: janesmith@example.com,  PhoneNumber: 555-123-4567
3. Robert Johnson - Email: robert@example.com,  PhoneNumber: 555-789-1234
4. Sarah Brown - Email: sarah@example.com,  PhoneNumber: 555-456-7890
5. David Lee - Email: david@example.com,  PhoneNumber: 555-987-6543
6. Laura Hall - Email: laura@example.com,  PhoneNumber: 555-234-5678
7. Michael Davis - Email: michael@example.com,  PhoneNumber: 555-876-5432
8. Emma Wilson - Email: emma@example.com,  PhoneNumber: 555-432-1098
9. William Taylor - Email: william@example.com,  PhoneNumber: 555-321-6547
10. Olivia Adams - Email: olivia@example.com,  PhoneNumber: 123-896-7458
12. Astha Raj - Email: astha@gmail.com,  PhoneNumber: 125-896-7469

 Admin Menu
```

```
13. Update Customer
0. exit
Enter choice from (0-13)9
Creating Lease:
Enter Customer ID: 3
Enter Vehicle ID: 3
Enter Start Date (YYYY-MM-DD): 2024-02-02
Enter End Date (YYYY-MM-DD): 2024-03-02
Input type as DailyLease or MonthlyLease: MonthlyLease
Lease created successfully. Lease ID: 11

 Admin Menu
```

```
10. List Active Leases
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)10
List Active Leases:
Active Leases:
Lease ID: 11 - Start Date: 2024-02-02 - End Date: 2024-03-02

 Admin Menu
*************************
1. Add Customer
2. Remove Customer
3. Add a car
4. Remove a car
5. Find Customer By ID
```

```
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)11
List Lease History:
Lease History:
Lease ID: 1 - Start Date: 2023-01-01 - End Date: 2023-01-05
Lease ID: 2 - Start Date: 2023-02-15 - End Date: 2023-02-28
Lease ID: 3 - Start Date: 2023-03-10 - End Date: 2023-03-15
Lease ID: 4 - Start Date: 2023-04-20 - End Date: 2023-04-30
Lease ID: 5 - Start Date: 2023-05-05 - End Date: 2023-05-10
Lease ID: 6 - Start Date: 2023-06-15 - End Date: 2023-06-30
Lease ID: 7 - Start Date: 2023-07-01 - End Date: 2023-07-10
Lease ID: 8 - Start Date: 2023-08-12 - End Date: 2023-08-15
Lease ID: 9 - Start Date: 2023-09-07 - End Date: 2023-09-10
Lease ID: 10 - Start Date: 2023-10-10 - End Date: 2023-10-31
Lease ID: 11 - Start Date: 2024-02-02 - End Date: 2024-03-02

 Admin Menu
```

```
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)12
Total Revenue: 6155.00

 Admin Menu
```

```
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)13
Enter Customer ID: 10
Enter new Email (leave blank to keep unchanged):
Enter new Phone (leave blank to keep unchanged): 125-859-7469
Customer information updated successfully.
```

```
 Admin Menu
**************************
1. Add Customer
2. Remove Customer
3. Add a car
4. Remove a car
5. Find Customer By ID
6. Find Car By ID
7. Find lease by ID
8. List all Customers
9. Create Lease
10. List Active Leases
11. List Lease History
12. Calculate total Revenue
13. Update Customer
0. exit
Enter choice from (0-13)0
Exiting Car Rental System. Goodbye!

You are welcome in Car Rental System
1. Admin Menu
2. User Menu
0. Exiting
Enter choice from (0-2)
```

```
You are welcome in Car Rental System
1. Admin Menu
2. User Menu
0. Exiting
Enter choice from (0-2)2
Enter username: s
Enter password:
Invalid username or password. Please try again.
Enter username: astha
Enter password: raj
User login successful!

 User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)
```

```
 User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)1
List Rented Cars:
Rented Cars:
3. Ford Focus - Status: NotAvailable
6. Hyundai Sonata - Status: NotAvailable
9. Audi A4 - Status: NotAvailable

 User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
```

```
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)2
List Available Cars:
Available Cars:
1. Toyota Camry - Status: Available
2. Honda Civic - Status: Available
4. Nissan Altima - Status: Available
5. Chevrolet Malibu - Status: Available
7. BMW 3 Series - Status: Available
8. Mercedes C-Class - Status: Available
10. Lexus ES - Status: Available
```

```
User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)3
Return Car:
Enter Lease ID to return the car: 9
Car returned successfully. Lease ID: 9
```

```
User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)4
Record Payment:
Enter Lease ID for payment: 6
Enter Payment Amount: 100
Lease id 6
Payment recorded successfully.
```

```
User Menu
1. List Rented Cars
2. List Available Cars
3. Return Car
4. Record Payment
5. Retrieve Payment History for Customer
0. Exit
Enter choice from (0-4)5
Enter Lease ID: 6
Payment History:
Payment ID: 6, Amount: 1200.00, Date: 2023-06-12
Payment ID: 11, Amount: 100.00, Date: 2024-02-10
```

Unit Testing: 10.

Create Unit test cases for Ecommerce System are essential to ensure the correctness and reliability of your system. Following questions to guide the creation of Unit test cases:

- Write test case to test car created successfully or not.
- Write test case to test lease is created successfully or not.
- Write test case to test lease is retrieved successfully or not.
- write test case to test exception is thrown correctly or not when customer id or car id or lease id not found in database.

```python
15 usages
@pytest.fixture
def car_repository():
    connection = DBConnection.getConnection()
    return ICarLeaseRepositoryImpl(connection)
```

```
Tests passed: 6 of 6 tests – 11 ms

C:\Users\astha\PycharmProjects\CarRentalSystem\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition
Testing started at 23:41 ...
Launching pytest with arguments C:\Users\astha\PycharmProjects\CarRentalSystem\tests\test_carrentalsystem.py --no-header --no-

============================= test session starts ==============================
collecting ... collected 6 items

test_carrentalsystem.py::test_add_car Database connected successfully
PASSED                                                                   [ 16%]
test_carrentalsystem.py::test_create_lease PASSED                        [ 33%]
test_carrentalsystem.py::test_retrieve_lease PASSED                      [ 50%]
test_carrentalsystem.py::test_exception_for_customer_not_found PASSED    [ 66%]
test_carrentalsystem.py::test_exception_for_car_not_found PASSED         [ 83%]
test_carrentalsystem.py::test_exception_for_lease_not_found PASSED       [100%]

============================== 6 passed in 0.18s ===============================
```

```python
import pytest
from main.mainmodule import MainModule
from dao.icarleaserepositoryimpl import ICarLeaseRepositoryImpl
from entity.vehicle import Vehicle
from exception.car_not_found_exception import CarNotFoundException
from exception.customer_not_found_exception import CustomerNotFoundException
from exception.lease_not_found_exception import LeaseNotFoundException
import datetime
from util.dbconnection import DBConnection


15 usages
@pytest.fixture
def car_repository():
    connection = DBConnection.getConnection()
    return ICarLeaseRepositoryImpl(connection)


6 usages
@pytest.fixture
def main_module(car_repository):
    return MainModule()


def test_add_car(main_module, car_repository):
    car_id = 1
    make = "Toyota"
    model = "Camry"
    year = 2022
```

```python
def test_add_car(main_module, car_repository):
    car_id = 1
    make = "Toyota"
    model = "Camry"
    year = 2022
    daily_rate = 30
    status = "available"
    passenger_capacity = 5
    engine_capacity = 3

    new_car = Vehicle(car_id, make, model, year, daily_rate, status, passenger_capacity, engine_capacity)
    car_repository.addCar(new_car)

    assert car_repository.findCarById(new_car.vehicleID)


def test_create_lease(main_module, car_repository):
    customer_id = 1
    car_id = 1
    start_date = datetime.datetime.now().strftime("%Y-%m-%d")
    end_date = "2024-02-07"
    lease_type = "Monthly"

    created_lease = car_repository.createLease(customer_id, car_id, start_date, end_date, lease_type)

    assert car_repository.findLeaseById(created_lease.leaseID)


def test_retrieve_lease(main_module, car_repository):
```

```python
    lease_type = "Monthly"

    created_lease = car_repository.createLease(customer_id, car_id, start_date, end_date, lease_type)

    assert car_repository.findLeaseById(created_lease.leaseID)


def test_retrieve_lease(main_module, car_repository):
    lease_id = 1
    retrieved_lease = car_repository.findLeaseById(lease_id)
    assert retrieved_lease is not None


def test_exception_for_customer_not_found(main_module, car_repository):
    with pytest.raises(CustomerNotFoundException):
        car_repository.findCustomerById(9999)


def test_exception_for_car_not_found(main_module, car_repository):
    with pytest.raises(CarNotFoundException):
        car_repository.findCarById(9999)


def test_exception_for_lease_not_found(main_module, car_repository):
    with pytest.raises(LeaseNotFoundException):
        car_repository.findLeaseById(9999)
```

```
(.venv) PS C:\Users\astha\PycharmProjects\CarRentalSystem> pytest -k test_create_lease
========================================================================= test session starts =========================================================================
platform win32 -- Python 3.12.1, pytest-8.0.0, pluggy-1.4.0
rootdir: C:\Users\astha\PycharmProjects\CarRentalSystem
collected 6 items / 5 deselected / 1 selected

tests\test_carrentalsystem.py .                                                                                                                                      [100%]

============================================================================= 1 passed, 5 deselected in 0.14s =============================================================
(.venv) PS C:\Users\astha\PycharmProjects\CarRentalSystem> pytest -k test_add_car
========================================================================= test session starts =========================================================================
platform win32 -- Python 3.12.1, pytest-8.0.0, pluggy-1.4.0
rootdir: C:\Users\astha\PycharmProjects\CarRentalSystem
collected 6 items / 5 deselected / 1 selected

tests\test_carrentalsystem.py .                                                                                                                                      [100%]

============================================================================= 1 passed, 5 deselected in 0.13s =============================================================
(.venv) PS C:\Users\astha\PycharmProjects\CarRentalSystem> pytest -k test_retrieve_lease
========================================================================= test session starts =========================================================================
platform win32 -- Python 3.12.1, pytest-8.0.0, pluggy-1.4.0
rootdir: C:\Users\astha\PycharmProjects\CarRentalSystem
collected 6 items / 5 deselected / 1 selected

tests\test_carrentalsystem.py .                                                                                                                                      [100%]

============================================================================= 1 passed, 5 deselected in 0.42s =============================================================
(.venv) PS C:\Users\astha\PycharmProjects\CarRentalSystem>
```

THANK YOU