Case Study – Car Rental System

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

   Table: Vehicle
   create table vehicle (vehicleid int primary key, make varchar(50), model varchar(50), year int, dailyrate double, status varchar(20), passengercapacity int, enginecapacity double);

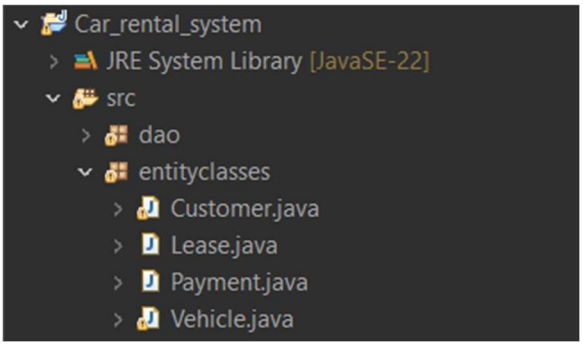
   Table: Lease
   create table lease (leaseid int primary key auto_increment, vehicleid int, customerid int, startdate date, enddate date, type varchar(20), foreign key (vehicleid) references vehicle(vehicleid), foreign key (customerid) references customer(customerid));

   Table: Customer
   create table customer (customerid int primary key, firstname varchar(50), lastname varchar(50), email varchar(100), phonenumber int);

   Table: Payment
   create table payment (paymentid int primary key auto_increment, leaseid int, paymentdate date, amount double, foreign key (leaseid) references lease(leaseid));

```
mysql> create table vehicle (vehicleid int primary key, make varchar(50), model varchar(50), year int, dailyrate double, status varchar(20), passengercapacity int, enginecapacity double);
Query OK, 0 rows affected (0.06 sec)

mysql> create table customer (customerid int primary key, firstname varchar(50), lastname varchar(50), email varchar(100), phonenumber int);
Query OK, 0 rows affected (0.03 sec)

mysql> create table lease (leaseid int primary key auto_increment, vehicleid int, customerid int, startdate date, enddate date, type varchar(20), foreign key (vehicleid) references vehicle(vehicleid), foreign
 key (customerid) references customer(customerid));
Query OK, 0 rows affected (0.08 sec)

mysql> create table payment (paymentid int primary key auto_increment, leaseid int, paymentdate date, amount double, foreign key (leaseid) references lease(leaseid));
Query OK, 0 rows affected (0.06 sec)
```

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



Vehicle.java

```
package entityclasses;
import java.util.*;

public class Vehicle {
	private int vehicleID;
	private String make;
	private String model;
	private int year;
	private double dailyrate;
	private String status;
	private int passengercapacity;
	private double enginecapacity;

	public Vehicle() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Vehicle(int vehicleID, String make, String model, int year, double dailyrate,
String status,
			int passengercapacity, double enginecapacity) {
		super();
		this.vehicleID = vehicleID;
		this.make = make;
		this.model = model;
		this.year = year;
		this.dailyrate = dailyrate;
		this.status = status;
		this.passengercapacity = passengercapacity;
		this.enginecapacity = enginecapacity;
	}

	public int getVehicleID() {
		return vehicleID;
	}

	public void setVehicleID(int vehicleID) {
		this.vehicleID = vehicleID;
	}

	public String getMake() {
		return make;
	}

	public void setMake(String make) {
		this.make = make;
	}

	public String getModel() {
		return model;
	}
```

```
	public void setModel(String model) {
		this.model = model;
	}

	public int getYear() {
		return year;
	}

	public void setYear(int year) {
		this.year = year;
	}

	public double getDailyrate() {
		return dailyrate;
	}

	public void setDailyrate(double dailyrate) {
		this.dailyrate = dailyrate;
	}

	public String getStatus() {
		return status;
	}

	public void setStatus(String status) {
		this.status = status;
	}

	public int getPassengercapacity() {
		return passengercapacity;
	}

	public void setPassengercapacity(int passengercapacity) {
		this.passengercapacity = passengercapacity;
	}

	public double getEnginecapacity() {
		return enginecapacity;
	}

	public void setEnginecapacity(double enginecapacity) {
		this.enginecapacity = enginecapacity;
	}

	@Override
	public String toString() {
		return "Vehicle [vehicleID=" + vehicleID + ", make=" + make + ", model="
+ model + ", year=" + year
				+ ", dailyrate=" + dailyrate + ", status=" + status + ",
passengercapacity=" + passengercapacity
				+ ", enginecapacity=" + enginecapacity + "]";
	}
```

```
}
```

Customer.java

```
package entityclasses;
import java.util.*;

public class Customer {
	private int customerID;
   private String firstName;
   private String lastName;
   private String email;
   private String phoneNumber;

	public Customer() {
		super();
		// TODO Auto-generated constructor stub
	}

	public Customer(int customerID, String firstName, String lastName, String email,
String phonenumber) {
		super();
		this.customerID = customerID;
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
		this.phoneNumber = phonenumber;
	}

	public int getCustomerID() {
		return customerID;
	}

	public void setCustomerID(int customerID) {
		this.customerID = customerID;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
```

```
	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	public String getPhoneNumber() {
		return phoneNumber;
	}

	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}

	@Override
	public String toString() {
		return "Customer [customerID=" + customerID + ", firstName=" +
firstName + ", lastName=" + lastName + ", email="
				+ email + ", phoneNumber=" + phoneNumber + "]";
	}



}
```

Lease.java

```
package entityclasses;
import java.util.*;

public class Lease{
   private int leaseID;
   private int vehicleID;
   private int customerID;
   private Date startDate;
   private Date endDate;
   private String Type;
	public Lease() {
		super();
		// TODO Auto-generated constructor stub
	}
	public Lease(int vehicleID, int customerID, Date startDate, Date endDate) {
		super();
		this.vehicleID = vehicleID;
		this.customerID = customerID;
		this.startDate = startDate;
		this.endDate = endDate;
	}
	public int getVehicleID() {
		return vehicleID;
```

```
	}
	public void setVehicleID(int vehicleID) {
		this.vehicleID = vehicleID;
	}
	public int getCustomerID() {
		return customerID;
	}
	public void setCustomerID(int customerID) {
		this.customerID = customerID;
	}
	public Date getStartDate() {
		return startDate;
	}
	public void setStartDate(Date startDate) {
		this.startDate = startDate;
	}
	public Date getEndDate() {
		return endDate;
	}
	public void setEndDate(Date endDate) {
		this.endDate = endDate;
	}



	@Override
	public String toString() {
		return "Lease [leaseID=" + leaseID + ", vehicleID=" + vehicleID + ",
customerID=" + customerID + ", startDate="
				+ startDate + ", endDate=" + endDate + ", Type=" + Type
+ "]";
	}
	public int getLeaseID() {
		return leaseID;
	}
	public void setLeaseID(int leaseID) {
		this.leaseID = leaseID;
	}
	public String getType() {
		return Type;
	}
	public void setType(String type) {
		Type = type;
	}


}
```

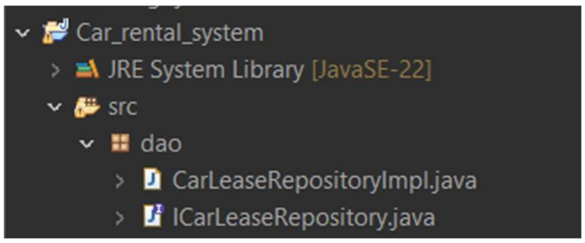
Payment.java

```
package entityclasses;
import java.util.*;
```

```
public class Payment{
    private int paymentID;
    private int leaseID;
    private Date paymentDate;
    private double amount;

        public Payment() {
            super();
            // TODO Auto-generated constructor stub
        }

        public Payment(int paymentID, int leaseID, Date paymentDate, double amount) {
            super();
            this.paymentID = paymentID;
            this.leaseID = leaseID;
            this.paymentDate = paymentDate;
            this.amount = amount;
        }

        public int getPaymentID() {
            return paymentID;
        }

        public void setPaymentID(int paymentID) {
            this.paymentID = paymentID;
        }

        public int getLeaseID() {
            return leaseID;
        }

        public void setLeaseID(int leaseID) {
            this.leaseID = leaseID;
        }

        public Date getPaymentDate() {
            return paymentDate;
        }

        public void setPaymentDate(Date paymentDate) {
            this.paymentDate = paymentDate;
        }

        public double getAmount() {
            return amount;
        }

        public void setAmount(double amount) {
            this.amount = amount;
        }

        @Override
        public String toString() {
```

```
                return "Payment [paymentID=" + paymentID + ", leaseID=" + leaseID + ",
paymentDate=" + paymentDate + ", amount="
                                + amount + "]";
        }



}
```

3. **Create Interface for ICarLeaseRepository and add following methods which interact with database. Implement the above interface in a class called ICarLeaseRepositoryImpl in package dao.**



ICarLeaseRepository.java

```
package dao;
import entityclasses.*;
import myexceptions.*;
import java.util.*;

public interface ICarLeaseRepository {

    boolean addCar(Vehicle car);
    boolean removeCar(int carID);
    HashMap<Integer, Vehicle> listAvailableCars();
    HashMap<Integer, Vehicle> listRentedCars();
    boolean findCarById(int carID) throws CarNotFoundException;

    boolean addCustomer(Customer customer);
    boolean removeCustomer(int customerID) throws CustomerNotFoundException;;
    HashMap<Integer, Customer> listCustomers();
    Customer findCustomerById(int customerID) throws Exception;

    Lease createLease(int customerID, int carID, Date startDate, Date endDate, String type)
throws CarNotFoundException;
    boolean returnCar(int leaseID) throws LeaseNotFoundException;
    HashMap<Integer, Lease> listActiveLeases();
    HashMap<Integer, Lease> listLeaseHistory();

    boolean recordPayment(Lease lease, double amount);
 }
```

CarLeaseRepositoryImpl.java

```
package dao;
import myexceptions.*;
import entityclasses.*;
import util.DBConnUtil;
import java.sql.*;
import java.util.*;
import java.util.Date;

public class CarLeaseRepositoryImpl implements ICarLeaseRepository {
    private Connection con;

    public CarLeaseRepositoryImpl() throws SQLException {
        con = DBConnUtil.getconnectiondb();
    }

    @Override
    public boolean addCar(Vehicle car) {
        boolean addc = false;
        try {
            PreparedStatement ps = con.prepareStatement("insert into vehicle values (?, ?, ?, ?,
?, ?, ?, ?)");
            ps.setInt(1, car.getVehicleID());
            ps.setString(2, car.getMake());
            ps.setString(3, car.getModel());
            ps.setInt(4, car.getYear());
            ps.setDouble(5, car.getDailyrate());
            ps.setString(6, car.getStatus());
            ps.setInt(7, car.getPassengercapacity());
            ps.setDouble(8, car.getEnginecapacity());
            ps.executeUpdate();
            addc = true;
        } catch (SQLException e) {
            System.out.println("Could not add the car.");
            e.printStackTrace();
        }
        return addc;
    }

    @Override
    public boolean removeCar(int carID) {
        boolean removec = false;
        try {
            PreparedStatement ps = con.prepareStatement("delete from vehicle where vehicleid
= ?");
            ps.setInt(1, carID);
            int r = ps.executeUpdate();
            if (r > 0) {
                removec = true;
            }
        } catch (SQLException e) {
            System.out.println("Could not remove the car.");
```

```
            e.printStackTrace();
        }
        return removec;
    }

    @Override
    public HashMap<Integer, Vehicle> listAvailableCars() {
        HashMap<Integer, Vehicle> availableCars = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from vehicle where status
= 'available'");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Vehicle v = new Vehicle(
                    rs.getInt("vehicleid"),
                    rs.getString("make"),
                    rs.getString("model"),
                    rs.getInt("year"),
                    rs.getDouble("dailyrate"),
                    rs.getString("status"),
                    rs.getInt("passengercapacity"),
                    rs.getDouble("enginecapacity")
                );
                availableCars.put(v.getVehicleID(), v);
            }
        } catch (SQLException e) {
            System.out.println("Could not list available cars.");
            e.printStackTrace();
        }
        return availableCars;
    }

    @Override
    public HashMap<Integer, Vehicle> listRentedCars() {
        HashMap<Integer, Vehicle> rentedCars = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from vehicle where status
= 'rented'");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Vehicle v = new Vehicle(
                    rs.getInt("vehicleid"),
                    rs.getString("make"),
                    rs.getString("model"),
                    rs.getInt("year"),
                    rs.getDouble("dailyrate"),
                    rs.getString("status"),
                    rs.getInt("passengercapacity"),
                    rs.getDouble("enginecapacity")
                );
                rentedCars.put(v.getVehicleID(), v);
            }
        } catch (SQLException e) {
```

```
            System.out.println("Could not list rented cars.");
            e.printStackTrace();
        }
        return rentedCars;
    }

    @Override
    public boolean findCarById(int carID) throws CarNotFoundException {
        boolean found = false;
        try {
            PreparedStatement ps = con.prepareStatement("select * from vehicle where
vehicleid = ?");
            ps.setInt(1, carID);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                return true;
            }
            else {
                    throw new CarNotFoundException("No vehicle found with this id");
            }
        } catch (SQLException e) {
            System.out.println("Problem finding car by ID: ");

        }
        return found;
    }

    @Override
    public boolean addCustomer(Customer customer) {
        boolean added = false;
        try {
            PreparedStatement ps = con.prepareStatement("insert into customer values (?, ?, ?,
?, ?)");
            ps.setInt(1, customer.getCustomerID());
            ps.setString(2, customer.getFirstName());
            ps.setString(3, customer.getLastName());
            ps.setString(4, customer.getEmail());
            ps.setString(5, customer.getPhoneNumber());
            ps.executeUpdate();
            added = true;
        } catch (SQLException e) {
            System.out.println("Could not add customer.");
            e.printStackTrace();
        }
        return added;
    }

    @Override
    public boolean removeCustomer(int customerID) {
        boolean removed = false;
        try {
            PreparedStatement ps = con.prepareStatement("delete from customer where
customerid = ?");
```

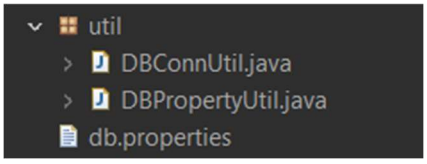
```
            ps.setInt(1, customerID);
            int r = ps.executeUpdate();
            if (r > 0) {
                removed = true;
            }
        } catch (SQLException e) {
            System.out.println("Unable to remove customer");
            e.printStackTrace();
        }
        return removed;
    }

    @Override
    public HashMap<Integer, Customer> listCustomers() {
        HashMap<Integer, Customer> customers = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from customer");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Customer c = new Customer(
                    rs.getInt("customerid"),
                    rs.getString("firstname"),
                    rs.getString("lastname"),
                    rs.getString("email"),
                    rs.getString("phonenumber")
                );
                customers.put(c.getCustomerID(), c);
            }
        } catch (SQLException e) {
            System.out.println("Problem listing customers: ");
            e.printStackTrace();
        }
        return customers;
    }

    @Override
    public Customer findCustomerById(int customerID) throws CustomerNotFoundException{
        Customer customer = null;
        try {
            PreparedStatement ps = con.prepareStatement("select * from customer where
customerid = ?");
            ps.setInt(1, customerID);
            ResultSet rs = ps.executeQuery();
            if (rs.next()) {
                customer = new Customer(
                    rs.getInt("customerid"),
                    rs.getString("firstname"),
                    rs.getString("lastname"),
                    rs.getString("email"),
                    rs.getString("phonenumber")
                );
            }
            else {
```

```
                throw new CustomerNotFoundException("No customer found with this id");
            }
        } catch (SQLException e) {
            System.out.println("Unable to find customer by ID");

        }
        return customer;
    }


    @Override
    public Lease createLease(int customerID, int carID, Date startDate, Date endDate, String
type) throws CarNotFoundException {
        Lease lease = null;
        try {
            PreparedStatement ps = con.prepareStatement(
                "insert into lease(customerid, vehicleid, startdate, enddate,type) value (?,?,?,?,?)",
                Statement.RETURN_GENERATED_KEYS
            );
            ps.setInt(1, customerID);
            ps.setInt(2, carID);
            ps.setDate(3, new java.sql.Date(startDate.getTime()));
            ps.setDate(4, new java.sql.Date(endDate.getTime()));
            ps.setString(5, type);
            ps.executeUpdate();

            ResultSet rs = ps.getGeneratedKeys();
            if (rs.next()) {
                int leaseID = rs.getInt(1);
                lease = new Lease(carID, customerID, startDate, endDate);
                lease.setLeaseID(leaseID);
                PreparedStatement updatePs = con.prepareStatement(
                    "update vehicle set status = 'rented' where vehicleid = ?"
                );
                updatePs.setInt(1, carID);
                updatePs.executeUpdate();
            }
        }
          catch (SQLIntegrityConstraintViolationException e) {
                throw new CarNotFoundException("Car or customer not found with given ID");
            }

        catch (SQLException e) {
            System.out.println("Unable to creating lease");
            e.printStackTrace();
        }
        return lease;
    }

    @Override
    public boolean returnCar(int leaseID) throws LeaseNotFoundException{
        boolean returned = false;
        try {
```

```
            PreparedStatement getPs = con.prepareStatement(
                "select vehicleid from lease where leaseid = ?"
            );
            getPs.setInt(1, leaseID);
            ResultSet rs = getPs.executeQuery();

            if (rs.next()) {
                int vehicleID = rs.getInt("vehicleid");
                PreparedStatement deletePs = con.prepareStatement(
                    "delete from lease where leaseid = ?"
                );
                deletePs.setInt(1, leaseID);
                deletePs.executeUpdate();
                PreparedStatement updatePs = con.prepareStatement(
                    "update vehicle set status = 'available' WHERE vehicleid = ?"
                );
                updatePs.setInt(1, vehicleID);
                updatePs.executeUpdate();

                returned = true;
            }
            else {
                    throw new LeaseNotFoundException("No lease found with this id");
            }
        } catch (SQLException e) {
            System.out.println("Error while returning car");

        }
        return returned;
    }

    @Override
    public HashMap<Integer, Lease> listActiveLeases() {
        HashMap<Integer, Lease> activeLeases = new HashMap<>();
        try {
            PreparedStatement ps = con.prepareStatement("select * from lease");
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                Lease lease = new Lease(
                    rs.getInt("vehicleid"),
                    rs.getInt("customerid"),
                    rs.getDate("startdate"),
                    rs.getDate("enddate")
                );
                lease.setLeaseID(rs.getInt("leaseid"));
                activeLeases.put(lease.getLeaseID(), lease);
            }
        } catch (SQLException e) {
            System.out.println("unable to list active leases");
            e.printStackTrace();
        }
        return activeLeases;
    }
```

```
    @Override
    public HashMap<Integer, Lease> listLeaseHistory() {
        return listActiveLeases();
    }

    @Override
    public boolean recordPayment(Lease lease, double amount) {
        boolean recorded = false;
        try {
            PreparedStatement ps = con.prepareStatement(
                "insert into payment(leaseid, paymentdate, amount) values (?, ?, ?)"
            );
            ps.setInt(1, lease.getLeaseID());
            ps.setDate(2, new java.sql.Date(System.currentTimeMillis()));
            ps.setDouble(3, amount);
            ps.executeUpdate();
            recorded = true;
        } catch (SQLException e) {
            System.out.println("Unable to record payment");
            e.printStackTrace();
        }
        return recorded;
    }

    public void close() {
        try {
            if (con != null && !con.isClosed()) {
                con.close();
            }
        } catch (SQLException e) {
            System.out.println("Problem closing connection: " + e.getMessage());
            e.printStackTrace();
        }
    }

}
```

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



db.properties

```
user=root
password=b14163709
port=3306
system=localhost
protocol=jdbc:mysql:
database=car_rental_system
```

DBPropertyUtil.java

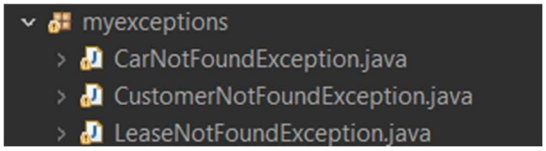
```
package util;
import java.util.Properties;
import java.io.FileInputStream;
import java.io.IOException;
public class DBPropertyUtil {
public static String getConstr(String filename) throws IOException{
        String constr=null;
        Properties prop=new Properties();
        FileInputStream fis=new FileInputStream(filename);
        prop.load(fis);
        String user=prop.getProperty("user");
        String password=prop.getProperty("password");
        String port=prop.getProperty("port");
        String protocol=prop.getProperty("protocol");
        String system=prop.getProperty("system");
        String database=prop.getProperty("database");

        constr=protocol+"//"+system+":"+port+"/"+database+"?"+"user="+user+"&passw
ord="+password;
        return constr;

    }

}
```

DBConnUtil.java

```
package util;
import java.sql.*;
import java.io.*;
public class DBConnUtil {
        private static final String filename="db.properties";
        public static Connection getconnectiondb() {
                Connection con=null;
                String constr=null;
                try {
                        constr=DBPropertyUtil.getConstr(filename);
                }
                catch(IOException e){
                        System.out.println("connection string creation failed");

                }
                if(constr!=null) {
                    try {
                     con=DriverManager.getConnection(constr);
                    }catch (SQLException e) {
                     System.out.println("Error While Establishing DBConnection.....");
                     e.printStackTrace();
                    }
                   }
```

```
                return con;
        }

}
```

5. **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.java

```
package myexceptions;

public class CarNotFoundException extends Exception {

        public CarNotFoundException(String m) {

                super(m);

                // TODO Auto-generated constructor stub

        }

}
```

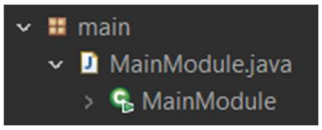
CustomerNotFoundException.java

```
package myexceptions;

public class CustomerNotFoundException extends Exception {

        public CustomerNotFoundException(String m) {

                super(m);

                // TODO Auto-generated constructor stub

        }

}
```

LeaseNotFoundException.java

```
package myexceptions;

public class LeaseNotFoundException extends Exception {

        public LeaseNotFoundException(String m) {

                super(m);

                // TODO Auto-generated constructor stub

        }

}
```

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



MainModule.java

```
package main;

import java.sql.Date;
import java.util.*;

import dao.CarLeaseRepositoryImpl;
import dao.ICarLeaseRepository;
import entityclasses.Customer;
import entityclasses.Lease;
import entityclasses.Vehicle;
import myexceptions.*;

public class MainModule {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ICarLeaseRepository repo = null;

        try {
            try {
                repo = new CarLeaseRepositoryImpl();
            } catch (Exception e) {
                System.out.println("Failed to connect to database!");
                return;
            }

            while (true) {
                System.out.println("Welcome to Car Rental System!");
                System.out.println("1. Add Car");
                System.out.println("2. Remove Car");
                System.out.println("3. Find Car by ID");
                System.out.println("4. List Available Cars");
                System.out.println("5. List Rented Cars");
                System.out.println("6. Add Customer");
                System.out.println("7. Remove Customer");
                System.out.println("8. Find Customer by ID");
                System.out.println("9. List Customers");
                System.out.println("10. Create Lease");
                System.out.println("11. Return Car");
                System.out.println("12. List Active Leases");
                System.out.println("13. List Lease History");
                System.out.println("14. Record Payment");
                System.out.println("15. Exit");
                System.out.print("Enter your choice: ");
```

```
                int choice = sc.nextInt();
                sc.nextLine();

                try {
                    switch (choice) {
                        case 1:
                            System.out.print("Vehicle ID: ");
                            int id = sc.nextInt();
                            sc.nextLine();
                            System.out.print("Make: ");
                            String make = sc.nextLine();
                            System.out.print("Model: ");
                            String model = sc.nextLine();
                            System.out.print("Year: ");
                            int year = sc.nextInt();
                            System.out.print("Daily Rate: ");
                            double rate = sc.nextDouble();
                            sc.nextLine();
                            System.out.print("Status (available/rented): ");
                            String status = sc.nextLine();
                            System.out.print("Passenger Capacity: ");
                            int capacity = sc.nextInt();
                            System.out.print("Engine Capacity: ");
                            double engine = sc.nextDouble();

                            Vehicle car = new Vehicle(id, make, model, year, rate, status, capacity,
engine);
                            if (repo.addCar(car)) {
                                System.out.println("Car added successfully.");
                            } else {
                                System.out.println("Failed to add car.");
                            }
                            break;

                        case 2:
                            System.out.print("Enter Vehicle ID to remove: ");
                            int carIdToRemove = sc.nextInt();
                            if (repo.removeCar(carIdToRemove)) {
                                System.out.println("Car removed successfully.");
                            } else {
                                System.out.println("Failed to remove car or car not found.");
                            }
                            break;

                        case 3:
                            System.out.print("Enter Vehicle ID to find: ");
                            int carIdToFind = sc.nextInt();
                            sc.nextLine();
                            try {
                                if (repo.findCarById(carIdToFind)) {
                                    System.out.println("Car found in the system.");
                                } else {
```

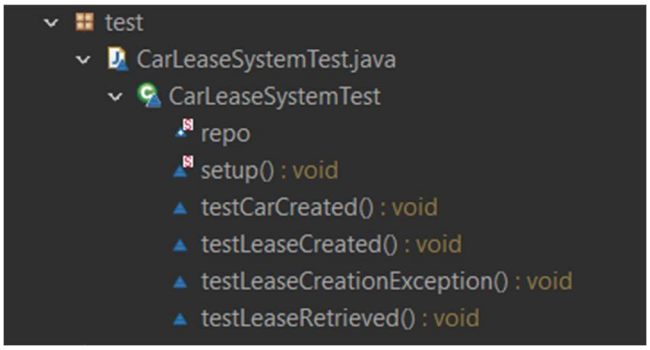
```
                    System.out.println("Car not found.");
                }
            } catch (CarNotFoundException e) {
                System.out.println(" ❗ Error finding car: ");
                e.printStackTrace();
            }
            break;
        case 4:
            HashMap<Integer, Vehicle> availableCars = repo.listAvailableCars();
            System.out.println("Available Cars (" + availableCars.size() + "):");
            for (Vehicle v : availableCars.values()) {
                System.out.println(v);
            }
            break;

        case 5:
            HashMap<Integer, Vehicle> rentedCars = repo.listRentedCars();
            System.out.println("Rented Cars (" + rentedCars.size() + "):");
            for (Vehicle v : rentedCars.values()) {
                System.out.println(v);
            }
            break;

        case 6:
            System.out.print("Customer ID: ");
            int cid = sc.nextInt();
            sc.nextLine();
            System.out.print("First Name: ");
            String fname = sc.nextLine();
            System.out.print("Last Name: ");
            String lname = sc.nextLine();
            System.out.print("Email: ");
            String email = sc.nextLine();
            System.out.print("Phone Number: ");
            String phone = sc.nextLine();
            sc.nextLine();

            Customer cust = new Customer(cid, fname, lname, email, phone);
            if (repo.addCustomer(cust)) {
                System.out.println("Customer added successfully.");
            } else {
                System.out.println("Failed to add customer.");
            }
            break;

        case 7 :
            System.out.print("Enter Customer ID to remove: ");
            int customerIdToRemove = sc.nextInt();
            if (repo.removeCustomer(customerIdToRemove)) {
                System.out.println("Customer removed successfully.");
            } else {
                System.out.println("Failed to remove customer or customer not
found.");
```

```
                    }
                    break;

                case 8:
                    System.out.print("Enter Customer ID: ");
                    int searchCustId = sc.nextInt();
                    sc.nextLine();
                    try {
                        Customer foundCustomer = repo.findCustomerById(searchCustId);
                        if (foundCustomer != null) {
                            System.out.println("Customer Found!");
                            System.out.println(foundCustomer);
                        } else {
                            System.out.println("Customer not found.");
                        }
                    } catch (CustomerNotFoundException e) {
                        System.out.println("Error finding customer!");
                        e.printStackTrace();
                    }
                    break;


                case 9:
                    HashMap<Integer, Customer> customers = repo.listCustomers();
                    System.out.println("Customers (" + customers.size() + "):");
                    for (Customer c : customers.values()) {
                        System.out.println(c);
                    }
                    break;

                case 10:
                    System.out.print("Customer ID: ");
                    int leaseCustId = sc.nextInt();
                    System.out.print("Vehicle ID: ");
                    int vehicleId = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Start Date (yyyy-mm-dd): ");
                    Date start = Date.valueOf(sc.nextLine());
                    System.out.print("End Date (yyyy-mm-dd): ");
                    Date end = Date.valueOf(sc.nextLine());
                    System.out.print("Type: (Dailylease/monthlylease) ");
                    String type = sc.nextLine();

                    Lease lease = repo.createLease(leaseCustId, vehicleId, start, end, type);
                    if (lease != null) {
                        System.out.println("Lease created successfully. Lease ID: " +
lease.getLeaseID());
                    } else {
                        System.out.println("Failed to create lease.");
                    }
                    break;

                case 11:
```

```
            System.out.print("Enter Lease ID to return: ");
            int returnLeaseId = sc.nextInt();
            sc.nextLine();
            if (repo.returnCar(returnLeaseId)) {
                System.out.println("Car returned successfully.");
            } else {
                System.out.println("Failed to return car.");
            }
            break;

        case 12:
            HashMap<Integer, Lease> activeLeases = repo.listActiveLeases();
            System.out.println("\n Active Leases (" + activeLeases.size() + "):");
            for (Lease l : activeLeases.values()) {
                System.out.println(l);
            }
            break;

        case 13:
            HashMap<Integer, Lease> leaseHistory = repo.listLeaseHistory();
            System.out.println("Lease History (" + leaseHistory.size() + "):");
            for (Lease l : leaseHistory.values()) {
                System.out.println(l);
            }
            break;

        case 14:
            System.out.print("Lease ID: ");
            int paymentLeaseId = sc.nextInt();
            System.out.print("Amount: ");
            double amount = sc.nextDouble();
            sc.nextLine();

            HashMap<Integer, Lease> leases = repo.listActiveLeases();
            Lease leaseToPay = leases.get(paymentLeaseId);

            if (leaseToPay != null) {
                if (repo.recordPayment(leaseToPay, amount)) {
                    System.out.println("Payment recorded successfully.");
                } else {
                    System.out.println("Failed to record payment.");
                }
            } else {
                System.out.println("Lease not found.");
            }
            break;

        case 15:
            System.out.println("Thank you. Visit again!");


        default:
            System.out.println("Invalid choice. Try again with a valid choice");
```

```
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input. Please enter the correct data.");
                sc.nextLine();
            } catch (Exception e) {
                System.out.println("Error: " + e.getMessage());
                e.printStackTrace();
            }
        }
    } finally {
        if (sc != null) {
            sc.close();
        }
    }
  }
}
```

7. **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:**



CarLeaseSystemTest.java

```
package test;

import static org.junit.jupiter.api.Assertions.*;

import java.sql.Date;
import java.sql.SQLException;
import java.util.HashMap;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import myexceptions.CarNotFoundException;
import dao.CarLeaseRepositoryImpl;
import entityclasses.Lease;
import entityclasses.Vehicle;

class CarLeaseSystemTest {

    static CarLeaseRepositoryImpl r;
```

```
    @BeforeAll
    static void setup() throws SQLException {
        r = new CarLeaseRepositoryImpl();
    }

    @Test
    void testCarCreated() {
        Vehicle car = new Vehicle(139, "Honda", "Civic", 2022, 1500.0, "available", 5, 1.5);
        boolean result = r.addCar(car);
        assertTrue(result, "Car should be created successfully");
    }

    @Test
    void testLeaseCreated() throws CarNotFoundException {
        Lease lease = r.createLease(12, 101, Date.valueOf("2024-04-01"), Date.valueOf("2024-
04-10"), "dailylease");
        assertNotNull(lease, "Lease should be created successfully");
    }

    @Test
    void testLeaseRetrieved() {
        HashMap<Integer, Lease> leaseMap = r.listActiveLeases();
        assertFalse(leaseMap.isEmpty(), "Active lease list should not be empty");
    }

    @Test
    void testLeaseCreationException() {
        assertThrows(CarNotFoundException.class, () -> {
            r.createLease(1359, 499, Date.valueOf("2024-04-01"), Date.valueOf("2024-04-10"),
"dailylease");
        }, "Exception should be thrown for invalid customer or car ID");
    }
}
```

OUTPUT OF CAR RENTAL SYSTEM

1. Add Car

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Find Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 1
Vehicle ID: 5
Make: Tata
Model: sumo
Year: 2023
Daily Rate: 50
Status (available/rented): rented
Passenger Capacity: 5
Engine Capacity: 900
Car added successfully.
```

2. Remove Car

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Find Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 2
Enter Vehicle ID to remove: 5
Car removed successfully.
```

3. Check Car by ID

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 3
Enter Vehicle ID to find: 3
CarID found in the system.
```

4. List Available Cars

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 4
Available Cars (3):
Vehicle [vehicleID=1, make=Maruti, model=suzuki, year=2024, dailyrate=50.0, status=available, passengercapacity=4, enginecapacity=800.0]
Vehicle [vehicleID=2, make=Hyundai, model=i20, year=2023, dailyrate=60.0, status=available, passengercapacity=5, enginecapacity=1200.0]
Vehicle [vehicleID=4, make=Mahindra, model=Thar, year=2024, dailyrate=95.0, status=available, passengercapacity=4, enginecapacity=2000.0]
```

5. List Rented Cars

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 5
Rented Cars (1):
Vehicle [vehicleID=3, make=Honda, model=City, year=2022, dailyrate=70.0, status=rented, passengercapacity=5, enginecapacity=1200.0]
```

6. Add Customer

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 6
Customer ID: 16
First Name: narendra
Last Name: modi
Email: namo@gmail.com
Phone Number: 9786789876
```

7. Remove Customer

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 7
Enter Customer ID to remove: 16
Customer removed successfully.
```

8. Find Customer by ID

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 8
Enter Customer ID: 14
Customer Found!
Customer [customerID=14, firstName=Madhu, lastName=K, email=kallamadhu@gmail.com, phoneNumber=9842367643]
```

9. List Customers

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 9
Customers (4):
Customer [customerID=12, firstName=sai, lastName=vignesh, email=sai@gmail.com, phoneNumber=9908287108]
Customer [customerID=13, firstName=Adarsh, lastName=K, email=appu@gmail.com, phoneNumber=9876787698]
Customer [customerID=14, firstName=Madhu, lastName=K, email=kallamadhu@gmail.com, phoneNumber=9842367643]
Customer [customerID=15, firstName=Shiva, lastName=Kumar, email=shiva@gmail.com, phoneNumber=8247821461]
```

10. Create Lease

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 10
Customer ID: 13
Vehicle ID: 3
Start Date (yyyy-mm-dd): 2025-04-08
End Date (yyyy-mm-dd): 2025-04-10
Type: (Dailylease/monthlylease) dailylease
Lease created successfully. Lease ID: 30
```

11.Return Car

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 11
Enter Lease ID to return: 30
Car returned successfully.
```

12.List Lease History

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 13
Lease History (3):
Lease [leaseID=1, vehicleID=2, customerID=12, startDate=2025-03-01, endDate=2025-06-01]
Lease [leaseID=2, vehicleID=3, customerID=13, startDate=2025-02-10, endDate=2025-02-15]
Lease [leaseID=31, vehicleID=4, customerID=12, startDate=2025-05-05, endDate=2025-06-05]
```

13.Record Payment

```
Welcome to Car Rental System!
1. Add Car
2. Remove Car
3. Check Car by ID
4. List Available Cars
5. List Rented Cars
6. Add Customer
7. Remove Customer
8. Find Customer by ID
9. List Customers
10. Create Lease
11. Return Car
12. List Active Leases
13. List Lease History
14. Record Payment
15. Exit
Enter your choice: 14
Lease ID: 31
Amount: 2400
Payment recorded successfully.
```

14.Exit

```
Enter your choice: 15
Thank you. Visit again!
```

JUNIT TESTCASE RESULT