

Case Study - Car Rental System

by

Dinesh S

Project Overview :

The Car Rental System project is a menu-driven application with key functionalities like Customer Management, Car Management, Lease Management, and Payment Handling. It involves maintaining SQL-based tables for vehicles, customers, leases, and payments, with robust data handling using object-oriented principles. Custom exceptions manage errors, ensuring user-friendly interactions. Database utilities streamline connections, while comprehensive unit tests validate system reliability.

Schema Designing :

Created tables for vehicle, customer, payment, lease and inserted sample values.

```
mysql> use carrentdb;
Database changed
mysql> show tables;
+---------------------+
| Tables_in_carrentdb |
+---------------------+
| customer            |
| lease                |
| payment              |
| vehicle              |
+---------------------+
4 rows in set (0.03 sec)
```

```

mysql> select*from customer;
+-----+-----+-----+-----+
| customerID | firstName | lastName | email           | phoneNumber |
+-----+-----+-----+-----+
|      1 | John     | Doe      | john.doe@example.com | 1234567890   |
|      2 | Jane     | Smith    | jane.smith@example.com | 0987654321   |
+-----+-----+-----+-----+
2 rows in set (0.03 sec)

mysql> select*from lease;
+-----+-----+-----+-----+-----+-----+
| leaseID | vehicleID | customerID | startDate | endDate   | type   |
+-----+-----+-----+-----+-----+-----+
|      51 |        2 |          2 | 2024-12-01 | 2024-12-10 | Daily   |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> select*from vehicle;
+-----+-----+-----+-----+-----+-----+-----+
| vehicleID | make   | model  | year  | dailyRate | status  | passengerCapacity | engineCapacity |
+-----+-----+-----+-----+-----+-----+-----+
|      1 | Toyota | Camry  | 2020  |    100.00 | available | 5             | 2.50          |
|      2 | Honda  | Civic   | 2021  |     80.00 | available | 5             | 1.80          |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql> select*from payment;
+-----+-----+-----+-----+
| paymentID | leaseID | paymentDate | amount   |
+-----+-----+-----+-----+
|      52 |      51 | 2024-11-23 | 10000.00 |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

Project Directory Structure :

1) entity:

Created entity package to these classes and added getter, setter methods

- Customer Class
- Lease Class
- Payment Class
- Vehicle Class

2) dao:

Created Interface called ICarLeaseRepository and Implementation file called CarLeaseRepositoryImpl to implement the interface methods.

- ICarLeaseRepository class
- CarLeaseRepositoryImpl class

3) Exception:

Created exceptions for,

- CarNotFoundException Class
- CustomerNotFoundException Class
- LeaseNotFoundException Class

4) Util:

Created Util package to Connect to the database.

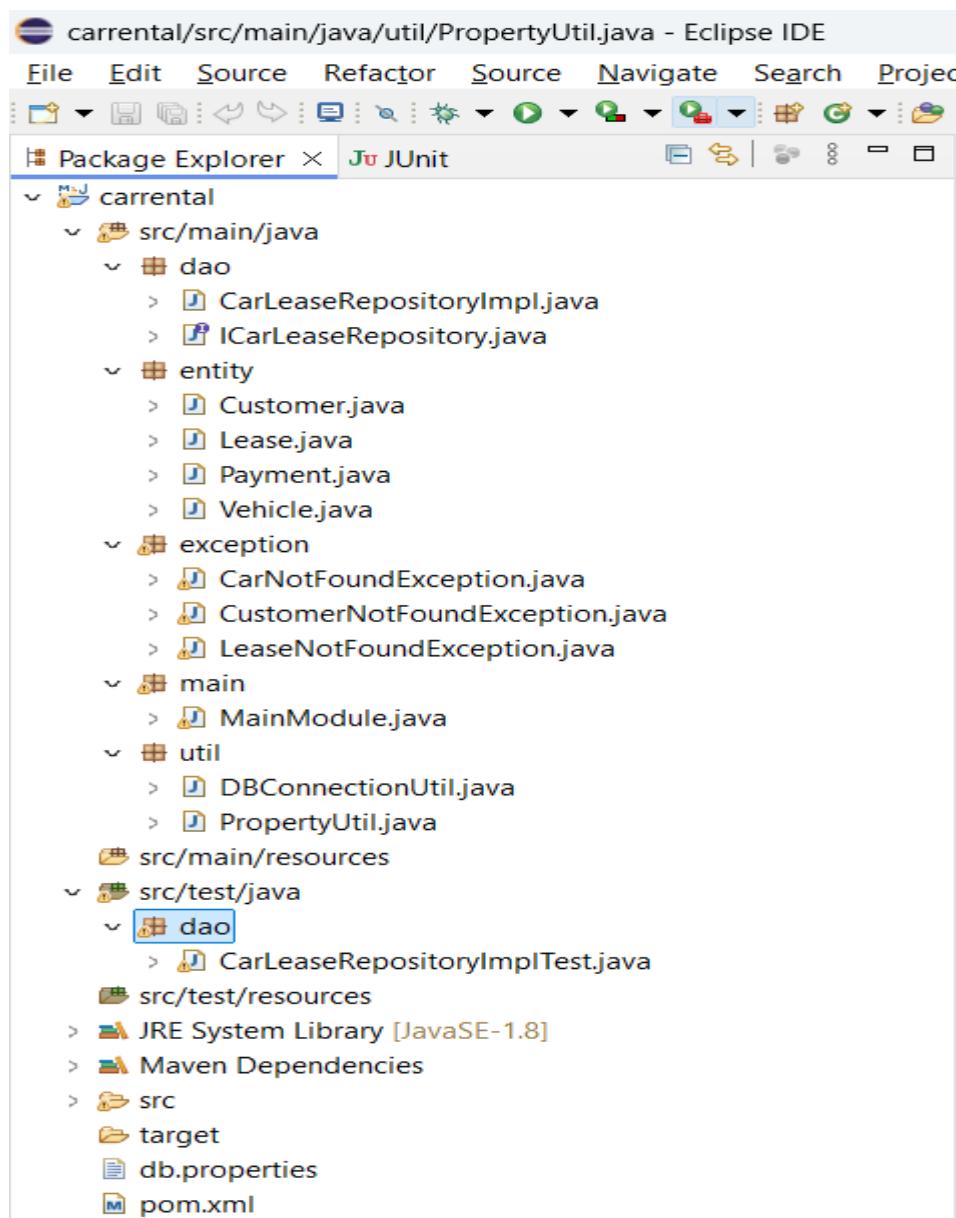
→ DBConnectionUtil class

→ PropertyUtil class

5) Main:

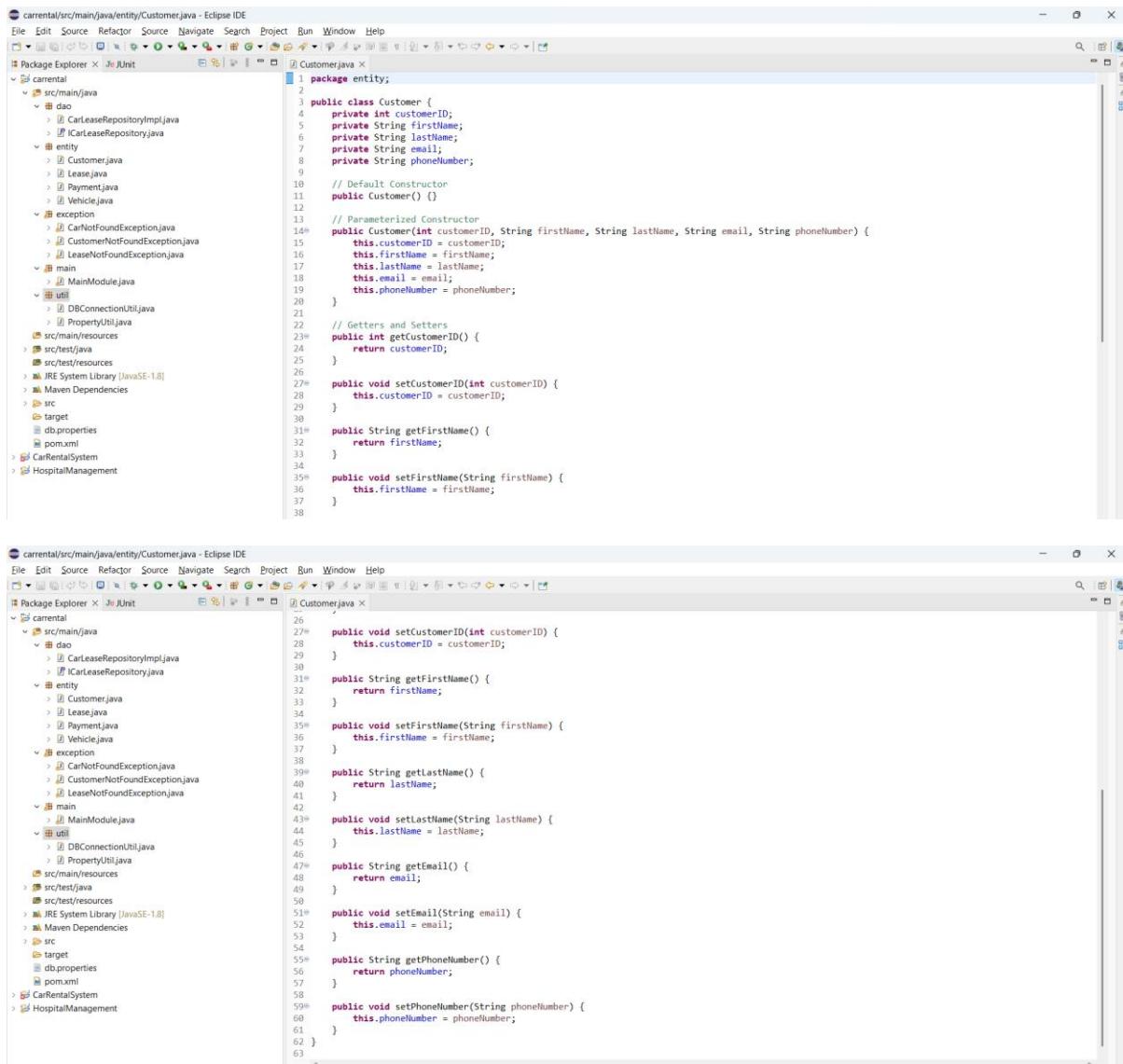
Main class to trigger all the methods and get input from user.

→ Main class



entity Package:

Customer.java :



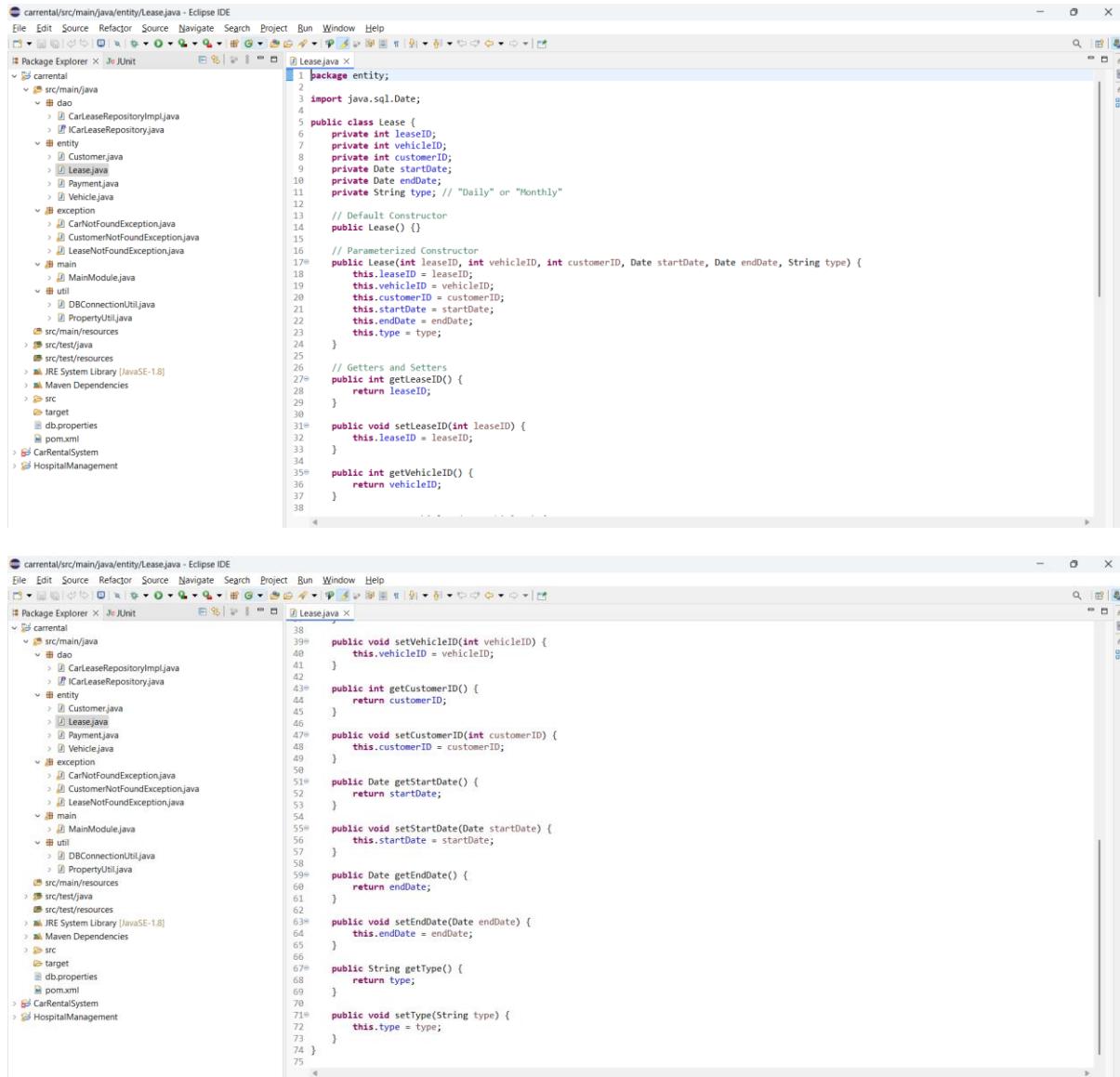
The screenshot shows the Eclipse IDE interface with the Customer.java file open in the central editor window. The file is located in the 'Carental' package under 'src/main/java/entity'. The code defines a Customer class with fields for customerID, firstName, lastName, email, and phoneNumber, along with constructors, getters, and setters for each.

```
1 package entity;
2
3 public class Customer {
4     private int customerID;
5     private String firstName;
6     private String lastName;
7     private String email;
8     private String phoneNumber;
9
10    // Default Constructor
11    public Customer() {}
12
13    // Parameterized Constructor
14    public Customer(int customerID, String firstName, String lastName, String email, String phoneNumber) {
15        this.customerID = customerID;
16        this.firstName = firstName;
17        this.lastName = lastName;
18        this.email = email;
19        this.phoneNumber = phoneNumber;
20    }
21
22    // Getters and Setters
23    public int getCustomerID() {
24        return customerID;
25    }
26
27    public void setCustomerID(int customerID) {
28        this.customerID = customerID;
29    }
30
31    public String getFirstName() {
32        return firstName;
33    }
34
35    public void setFirstName(String firstName) {
36        this.firstName = firstName;
37    }
38}
```

The second screenshot shows the same Customer.java file after some modifications, specifically adding getters and setters for the lastName field. The code now includes methods to get and set the lastName value.

```
26
27    public void setCustomerID(int customerID) {
28        this.customerID = customerID;
29    }
30
31    public String getFirstName() {
32        return firstName;
33    }
34
35    public void setFirstName(String firstName) {
36        this.firstName = firstName;
37    }
38
39    public String getLastName() {
40        return lastName;
41    }
42
43    public void setLastName(String lastName) {
44        this.lastName = lastName;
45    }
46
47    public String getEmail() {
48        return email;
49    }
50
51    public void setEmail(String email) {
52        this.email = email;
53    }
54
55    public String getPhoneNumber() {
56        return phoneNumber;
57    }
58
59    public void setPhoneNumber(String phoneNumber) {
60        this.phoneNumber = phoneNumber;
61    }
62}
```

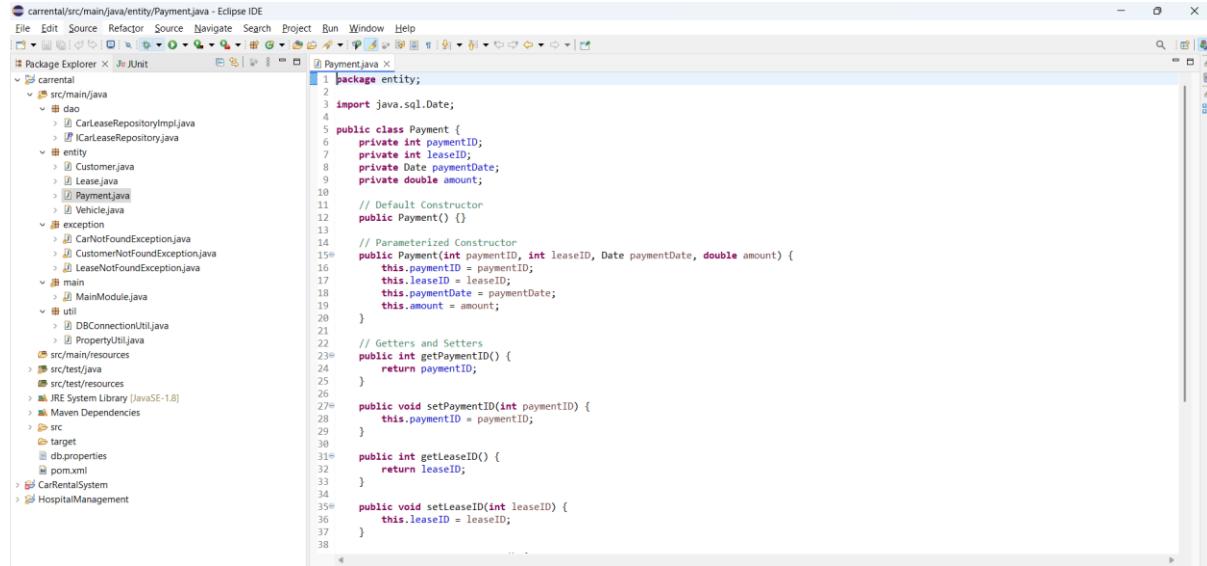
Lease.java:



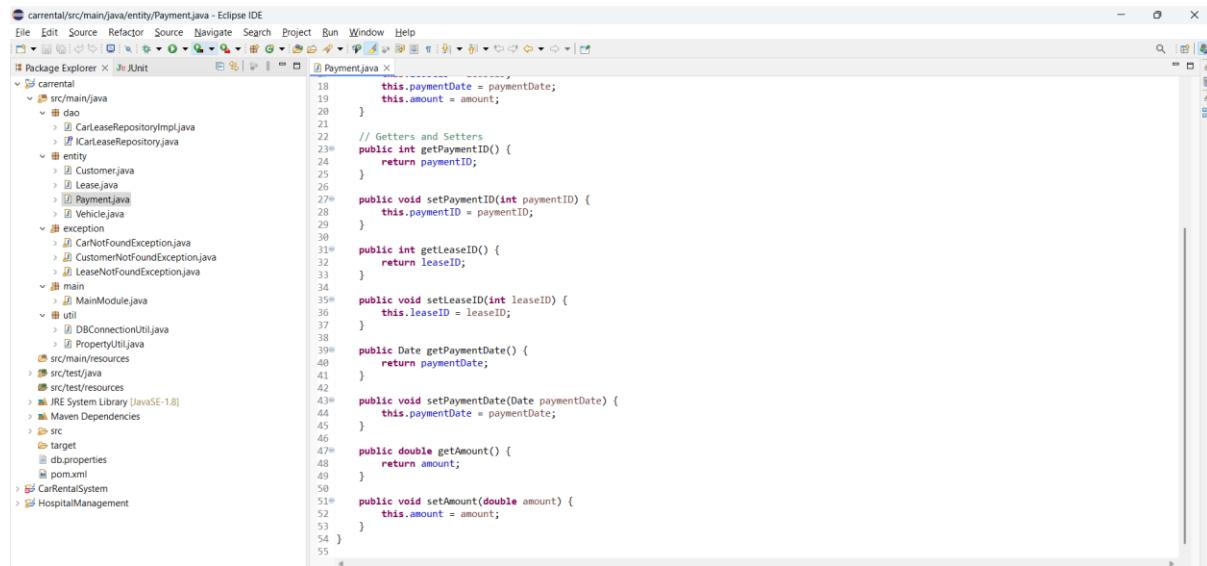
The image shows two side-by-side panes of the Eclipse IDE interface, both displaying the same Java code for the `Lease` class. The code defines a class `Lease` with various fields and methods, including constructors, getters, and setters. The top pane shows lines 1 through 38, and the bottom pane shows lines 38 through 75.

```
1 package entity;
2
3 import java.sql.Date;
4
5 public class Lease {
6     private int leaseID;
7     private int vehicleID;
8     private int customerID;
9     private Date startDate;
10    private Date endDate;
11    private String type; // "Daily" or "Monthly"
12
13    // Default Constructor
14    public Lease() {}
15
16    // Parameterized Constructor
17    public Lease(int leaseID, int vehicleID, Date startDate, Date endDate, String type) {
18        this.leaseID = leaseID;
19        this.vehicleID = vehicleID;
20        this.customerID = customerID;
21        this.startDate = startDate;
22        this.endDate = endDate;
23        this.type = type;
24    }
25
26    // Getters and Setters
27    public int getLeaseID() {
28        return leaseID;
29    }
30
31    public void setLeaseID(int leaseID) {
32        this.leaseID = leaseID;
33    }
34
35    public int getVehicleID() {
36        return vehicleID;
37    }
38
39    public void setVehicleID(int vehicleID) {
40        this.vehicleID = vehicleID;
41    }
42
43    public int getCustomerID() {
44        return customerID;
45    }
46
47    public void setCustomerID(int customerID) {
48        this.customerID = customerID;
49    }
50
51    public Date getStartDate() {
52        return startDate;
53    }
54
55    public void setStartDate(Date startDate) {
56        this.startDate = startDate;
57    }
58
59    public Date getEndDate() {
60        return endDate;
61    }
62
63    public void setEndDate(Date endDate) {
64        this.endDate = endDate;
65    }
66
67    public String getType() {
68        return type;
69    }
70
71    public void setType(String type) {
72        this.type = type;
73    }
74}
75
```

Payment.java:



```
currental/src/main/java/entity/Payment.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
currental
src/main/java
  dao
    CarLeaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundException.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
CarRentalSystem
HospitalManagement
Payment.java X
1 package entity;
2
3 import java.sql.Date;
4
5 public class Payment {
6   private int paymentID;
7   private int leaseID;
8   private Date paymentDate;
9   private double amount;
10
11  // Default Constructor
12  public Payment() {
13
14  // Parameterized Constructor
15  public Payment(int paymentID, int leaseID, Date paymentDate, double amount) {
16    this.paymentID = paymentID;
17    this.leaseID = leaseID;
18    this.paymentDate = paymentDate;
19    this.amount = amount;
20  }
21
22  // Getters and Setters
23  public int getPaymentID() {
24    return paymentID;
25  }
26
27  public void setPaymentID(int paymentID) {
28    this.paymentID = paymentID;
29  }
30
31  public int getLeaseID() {
32    return leaseID;
33  }
34
35  public void setLeaseID(int leaseID) {
36    this.leaseID = leaseID;
37  }
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
```



```
currental/src/main/java/entity/Payment.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
currental
src/main/java
  dao
    CarLeaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundException.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
CarRentalSystem
HospitalManagement
Payment.java X
1 package entity;
2
3 import java.sql.Date;
4
5 public class Payment {
6   private int paymentID;
7   private int leaseID;
8   private Date paymentDate;
9   private double amount;
10
11  // Default Constructor
12  public Payment() {
13
14  // Parameterized Constructor
15  public Payment(int paymentID, int leaseID, Date paymentDate, double amount) {
16    this.paymentID = paymentID;
17    this.leaseID = leaseID;
18    this.paymentDate = paymentDate;
19    this.amount = amount;
20  }
21
22  // Getters and Setters
23  public int getPaymentID() {
24    return paymentID;
25  }
26
27  public void setPaymentID(int paymentID) {
28    this.paymentID = paymentID;
29  }
30
31  public int getLeaseID() {
32    return leaseID;
33  }
34
35  public void setLeaseID(int leaseID) {
36    this.leaseID = leaseID;
37  }
38
39  public Date getPaymentDate() {
40    return paymentDate;
41  }
42
43  public void setPaymentDate(Date paymentDate) {
44    this.paymentDate = paymentDate;
45  }
46
47  public double getAmount() {
48    return amount;
49  }
50
51  public void setAmount(double amount) {
52    this.amount = amount;
53  }
54
55
```

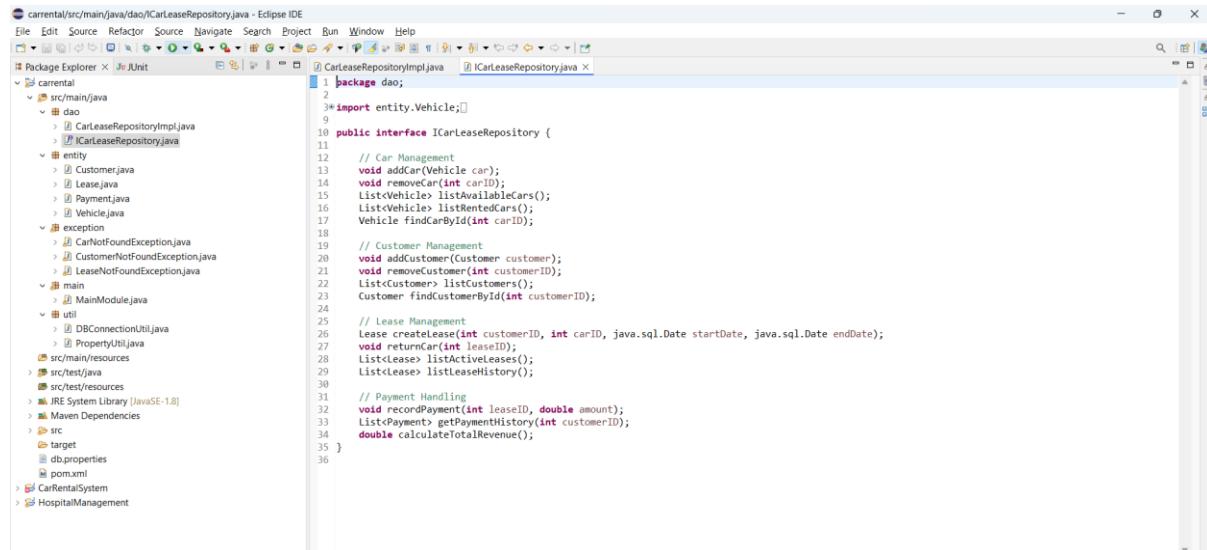
Vehicle.java:

The screenshots show the same Java code for the Vehicle class across three different stages of development. The code defines a vehicle with attributes like ID, make, model, year, daily rate, passenger capacity, and engine capacity, along with their corresponding getters and setters.

```
1 package entity;
2
3 public class Vehicle {
4     private int vehicleID;
5     private String make;
6     private String model;
7     private int year;
8     private double dailyRate;
9     private String status;
10    private int passengerCapacity;
11    private double engineCapacity;
12
13    // Constructor
14    public Vehicle(int vehicleID, String make, String model, int year, double dailyRate, String status, int passengerCapacity, double engineCapacity) {
15        this.vehicleID = vehicleID;
16        this.make = make;
17        this.model = model;
18        this.year = year;
19        this.dailyRate = dailyRate;
20        this.status = status;
21        this.passengerCapacity = passengerCapacity;
22        this.engineCapacity = engineCapacity;
23    }
24
25    // Getters and setters (if needed)
26    public int getVehicleID() {
27        return vehicleID;
28    }
29
30    public void setVehicleID(int vehicleID) {
31        this.vehicleID = vehicleID;
32    }
33
34    public String getMake() {
35        return make;
36    }
37
38    public void setMake(String make) {
39
40    }
41
42    public String getModel() {
43        return model;
44    }
45
46    public void setModel(String model) {
47        this.model = model;
48    }
49
50    public int getYear() {
51        return year;
52    }
53
54    public void setYear(int year) {
55        this.year = year;
56    }
57
58    public double getDailyRate() {
59        return dailyRate;
60    }
61
62    public void setDailyRate(double dailyRate) {
63        this.dailyRate = dailyRate;
64    }
65
66    public String getStatus() {
67        return status;
68    }
69
70    public void setStatus(String status) {
71        this.status = status;
72    }
73
74    public int getPassengerCapacity() {
75        return passengerCapacity;
76    }
77
78    public void setPassengerCapacity(int passengerCapacity) {
79        this.passengerCapacity = passengerCapacity;
80    }
81
82    public double getEngineCapacity() {
83        return engineCapacity;
84    }
85
86    public void setEngineCapacity(double engineCapacity) {
87        this.engineCapacity = engineCapacity;
88    }
89
90    // Override toString() for meaningful output
91    @Override
92    public String toString() {
93        return String.format(
94            "Vehicle ID: %d | Make: %s | Model: %s | Year: %d | Daily Rate: %.2f | Status: %s | Passenger Capacity: %d | Engine Capacity: %.1f",
95            vehicleID, make, model, year, dailyRate, status, passengerCapacity, engineCapacity
96        );
97    }
98 }
```

dao Package:

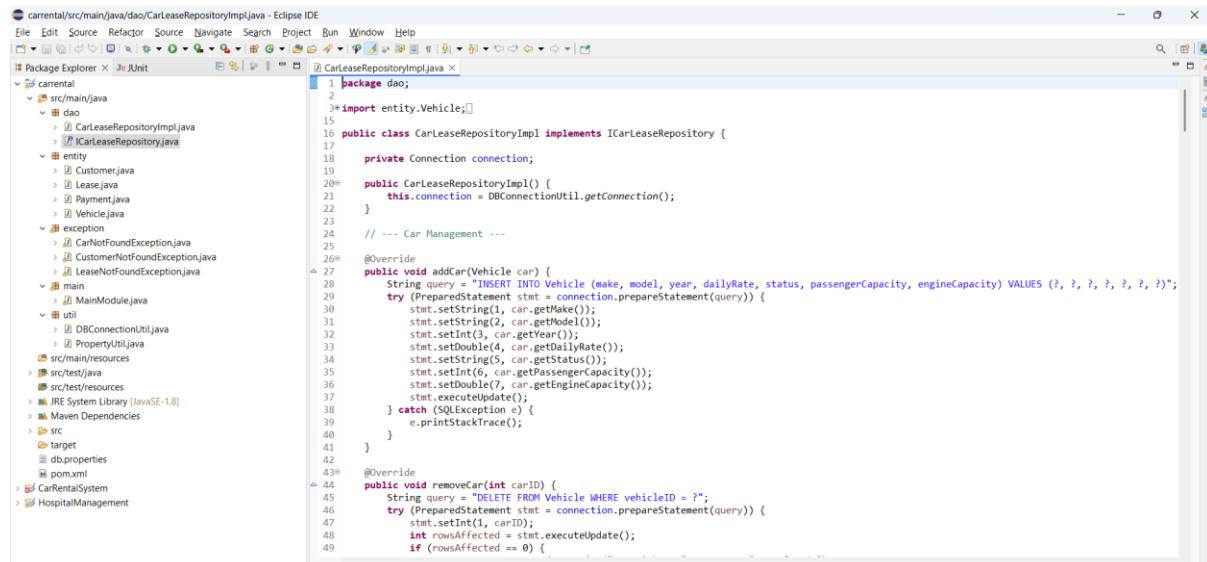
ICarLeaseRepository.java



```
carental/src/main/java/dao/ICarLeaseRepository.java - Eclipse IDE
File Edit Source Refactor Source Navigate Project Run Window Help
Package Explorer JUnit
src/main/java
  dao
    CarleaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundexception.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
CarRentalSystem
HospitalManagement

1 package dao;
2
3 import entity.Vehicle;
4
5 public interface ICarLeaseRepository {
6
7     // Car Management
8     void addCar(Vehicle car);
9     void removeCar(int carID);
10    List<Vehicle> listAvailableCars();
11    List<Vehicle> listRentedCars();
12    Vehicle findCarById(int carID);
13
14    // Customer Management
15    void addCustomer(Customer customer);
16    void removeCustomer(int customerID);
17    List<Customer> listCustomers();
18    Customer findCustomerById(int customerID);
19
20    // Lease Management
21    Lease createLease(int customerID, int carID, java.sql.Date startDate, java.sql.Date endDate);
22    void returnCar(int leaseID);
23    List<Lease> listActiveLeases();
24    List<Lease> listLeaseHistory();
25
26    // Payment Management
27    void recordPayment(int leaseID, double amount);
28    List<Payment> getPaymentHistory(int customerID);
29    double calculateTotalRevenue();
30
31 }
32
33 }
```

CarLeaseRepositoryImpl.java



```
carental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE
File Edit Source Refactor Source Navigate Project Run Window Help
Package Explorer JUnit
src/main/java
  dao
    CarleaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundexception.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
CarRentalSystem
HospitalManagement

1 package dao;
2
3 import entity.Vehicle;
4
5 public class CarLeaseRepositoryImpl implements ICarLeaseRepository {
6
7     private Connection connection;
8
9     public CarLeaseRepositoryImpl() {
10        this.connection = DBConnectionUtil.getConnection();
11    }
12
13    // --- Car Management ---
14
15    @Override
16    public void addCar(Vehicle car) {
17        String query = "INSERT INTO Vehicle (make, model, year, dailyRate, status, passengerCapacity, engineCapacity) VALUES (?, ?, ?, ?, ?, ?, ?)";
18        try (PreparedStatement stat = connection.prepareStatement(query)) {
19            stat.setString(1, car.getMake());
20            stat.setInt(2, car.getModel());
21            stat.setInt(3, car.getYear());
22            stat.setDouble(4, car.getDailyRate());
23            stat.setString(5, car.getStatus());
24            stat.setInt(6, car.getPassengerCapacity());
25            stat.setDouble(7, car.getEngineCapacity());
26            stat.executeUpdate();
27        } catch (SQLException e) {
28            e.printStackTrace();
29        }
30    }
31
32    @Override
33    public void removeCar(int carID) {
34        String query = "DELETE FROM Vehicle WHERE vehicleID = ?";
35        try (PreparedStatement stat = connection.prepareStatement(query)) {
36            stat.setInt(1, carID);
37            int rowsAffected = stat.executeUpdate();
38            if (rowsAffected == 0) {
39                System.out.println("Car not found or already deleted");
40            }
41        }
42    }
43}
```

currental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE

```

48     int rowsAffected = stmt.executeUpdate();
49     if (rowsAffected == 0) {
50         throw new CarNotFoundException("Car with ID " + carID + " not found.");
51     }
52 } catch (SQLException e) {
53     e.printStackTrace();
54 }
55 }

56 /**
57 * @Override
58 * public List<Vehicle> listAvailableCars() {
59     List<Vehicle> cars = new ArrayList<>();
60     String query = "SELECT * FROM Vehicle WHERE status = 'available'";
61     try (PreparedStatement stat = connection.prepareStatement(query)) {
62         ResultSet rs = stat.executeQuery();
63         while (rs.next()) {
64             cars.add(new Vehicle(rs));
65         }
66     } catch (SQLException e) {
67         e.printStackTrace();
68     }
69     return cars;
70 }
71 */

72 /**
73 * @Override
74 * public List<Vehicle> listRentedCars() {
75     List<Vehicle> rentedCars = new ArrayList<>();
76     String query = "SELECT * FROM Vehicle WHERE status = 'notAvailable'";
77     try (PreparedStatement stat = connection.prepareStatement(query)) {
78         ResultSet rs = stat.executeQuery();
79         while (rs.next()) {
80             rentedCars.add(new Vehicle(
81                 rs.getInt("vehicleID"),
82                 rs.getString("make"),
83                 rs.getInt("model"),
84                 rs.getDouble("year"),
85                 rs.getDouble("dailyRate"),
86                 rs.getString("status"),
87                 rs.getInt("passengerCapacity"),
88                 rs.getDouble("engineCapacity")));
89         }
90     } // Check if no cars are rented
91     if (rentedCars.isEmpty()) {
92         System.out.println("No cars are currently rented.");
93     }
94 } catch (SQLException e) {
95     System.err.println("Error while fetching rented cars: " + e.getMessage());
96     e.printStackTrace();
97 }
98 return rentedCars;
99 }

100 /**
101 * @Override
102 * public Vehicle findCarById(int carID) {
103     String query = "SELECT * FROM Vehicle WHERE vehicleID = ?";
104     try (PreparedStatement stat = connection.prepareStatement(query)) {
105         stat.setInt(1, carID);
106         ResultSet rs = stat.executeQuery();
107         if (rs.next()) {
108             return new Vehicle(
109                 rs.getInt("vehicleID"),
110                 rs.getString("make"),
111                 rs.getInt("model"),
112                 rs.getInt("year"),
113                 rs.getDouble("dailyRate"),
114                 rs.getString("status"), // Ensure the latest status is fetched
115                 rs.getInt("passengerCapacity"),
116                 rs.getDouble("engineCapacity"));
117         }
118     } else {
119         throw new CarNotFoundException("Car with ID " + carID + " not found.");
120     }
121 } catch (SQLException e) {
122     e.printStackTrace();
123     return null;
124 }
125 }

126 /**
127 * --- Customer Management ---
128 */

129 /**
130 * @Override
131 * public void addCustomer(Customer customer) {
132     String query = "INSERT INTO Customer (firstName, lastName, email, phoneNumber) VALUES (?, ?, ?, ?)";
133     try (PreparedStatement stat = connection.prepareStatement(query)) {
134         stat.setString(1, customer.getFirstName());
135         stat.setString(2, customer.getLastName());
136         stat.setString(3, customer.getEmail());
137         stat.setString(4, customer.getPhoneNumber());
138         stat.executeUpdate();
139     } catch (SQLException e) {
140         e.printStackTrace();
141     }
142 }
143 */

144 /**
145 * @Override
146 * public void removeCustomer(int customerID) {
147     // Step 1: Delete the payments associated with the customer
148     String deletePaymentsQuery = "DELETE FROM Payment WHERE leaseID IN (SELECT leaseID FROM Lease WHERE customerID = ?)";
149     try (PreparedStatement stat = connection.prepareStatement(deletePaymentsQuery)) {
150         stat.setInt(1, customerID);
151         stat.executeUpdate();
152     } catch (SQLException e) {
153         System.out.println("Error while deleting payments: " + e.getMessage());
154         e.printStackTrace();
155     }
156     // Step 2: Delete the leases associated with the customer
157     String deleteLeasesQuery = "DELETE FROM Lease WHERE customerID = ?";
158     try (PreparedStatement stat = connection.prepareStatement(deleteLeasesQuery)) {
159         stat.setInt(1, customerID);
160         stat.executeUpdate();
161     } catch (SQLException e) {
162         System.out.println("Error while deleting leases: " + e.getMessage());
163         e.printStackTrace();
164     }
165 }
166 */

```

currental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE

```

86         rs.getDouble("engineCapacity");
87     });
88 }
89 // Check if no cars are rented
90 if (rentedCars.isEmpty()) {
91     System.out.println("No cars are currently rented.");
92 }
93 } catch (SQLException e) {
94     System.err.println("Error while fetching rented cars: " + e.getMessage());
95     e.printStackTrace();
96 }
97 return rentedCars;
98 }

100 /**
101 * @Override
102 * public Vehicle findCarById(int carID) {
103     String query = "SELECT * FROM Vehicle WHERE vehicleID = ?";
104     try (PreparedStatement stat = connection.prepareStatement(query)) {
105         stat.setInt(1, carID);
106         ResultSet rs = stat.executeQuery();
107         if (rs.next()) {
108             return new Vehicle(
109                 rs.getInt("vehicleID"),
110                 rs.getString("make"),
111                 rs.getInt("model"),
112                 rs.getInt("year"),
113                 rs.getDouble("dailyRate"),
114                 rs.getString("status"), // Ensure the latest status is fetched
115                 rs.getInt("passengerCapacity"),
116                 rs.getDouble("engineCapacity"));
117         }
118     } else {
119         throw new CarNotFoundException("Car with ID " + carID + " not found.");
120     }
121 } catch (SQLException e) {
122     e.printStackTrace();
123     return null;
124 }
125 }

126 /**
127 * --- Customer Management ---
128 */

129 /**
130 * @Override
131 * public void addCustomer(Customer customer) {
132     String query = "INSERT INTO Customer (firstName, lastName, email, phoneNumber) VALUES (?, ?, ?, ?)";
133     try (PreparedStatement stat = connection.prepareStatement(query)) {
134         stat.setString(1, customer.getFirstName());
135         stat.setString(2, customer.getLastName());
136         stat.setString(3, customer.getEmail());
137         stat.setString(4, customer.getPhoneNumber());
138         stat.executeUpdate();
139     } catch (SQLException e) {
140         e.printStackTrace();
141     }
142 }
143 */

144 /**
145 * @Override
146 * public void removeCustomer(int customerID) {
147     // Step 1: Delete the payments associated with the customer
148     String deletePaymentsQuery = "DELETE FROM Payment WHERE leaseID IN (SELECT leaseID FROM Lease WHERE customerID = ?)";
149     try (PreparedStatement stat = connection.prepareStatement(deletePaymentsQuery)) {
150         stat.setInt(1, customerID);
151         stat.executeUpdate();
152     } catch (SQLException e) {
153         System.out.println("Error while deleting payments: " + e.getMessage());
154         e.printStackTrace();
155     }
156     // Step 2: Delete the leases associated with the customer
157     String deleteLeasesQuery = "DELETE FROM Lease WHERE customerID = ?";
158     try (PreparedStatement stat = connection.prepareStatement(deleteLeasesQuery)) {
159         stat.setInt(1, customerID);
160         stat.executeUpdate();
161     } catch (SQLException e) {
162         System.out.println("Error while deleting leases: " + e.getMessage());
163         e.printStackTrace();
164     }
165 }
166 */

```

currental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE

```

124     return null;
125 }
126 }

127 /**
128 * --- Customer Management ---
129 */

130 /**
131 * @Override
132 * public void addCustomer(Customer customer) {
133     String query = "INSERT INTO Customer (firstName, lastName, email, phoneNumber) VALUES (?, ?, ?, ?)";
134     try (PreparedStatement stat = connection.prepareStatement(query)) {
135         stat.setString(1, customer.getFirstName());
136         stat.setString(2, customer.getLastName());
137         stat.setString(3, customer.getEmail());
138         stat.setString(4, customer.getPhoneNumber());
139         stat.executeUpdate();
140     } catch (SQLException e) {
141         e.printStackTrace();
142     }
143 }

144 /**
145 * @Override
146 * public void removeCustomer(int customerID) {
147     // Step 1: Delete the payments associated with the customer
148     String deletePaymentsQuery = "DELETE FROM Payment WHERE leaseID IN (SELECT leaseID FROM Lease WHERE customerID = ?)";
149     try (PreparedStatement stat = connection.prepareStatement(deletePaymentsQuery)) {
150         stat.setInt(1, customerID);
151         stat.executeUpdate();
152     } catch (SQLException e) {
153         System.out.println("Error while deleting payments: " + e.getMessage());
154         e.printStackTrace();
155     }
156     // Step 2: Delete the leases associated with the customer
157     String deleteLeasesQuery = "DELETE FROM Lease WHERE customerID = ?";
158     try (PreparedStatement stat = connection.prepareStatement(deleteLeasesQuery)) {
159         stat.setInt(1, customerID);
160         stat.executeUpdate();
161     } catch (SQLException e) {
162         System.out.println("Error while deleting leases: " + e.getMessage());
163         e.printStackTrace();
164     }
165 }
166 */

```

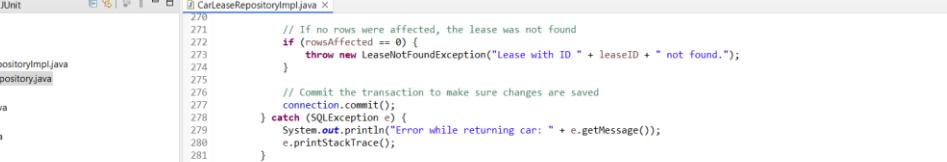
```

carrental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer JUnit CarLeaseRepositoryImpl.java X
carrental
  + src/main/java
    + dao
      + CarLeaseRepositoryImpl.java
      + ICarLeaseRepository.java
    + entity
      + Customer.java
      + Lease.java
      + Payment.java
      + Vehicle.java
    + exception
      + CarNotFoundException.java
      + CustomerNotFoundException.java
      + LeaseNotFoundException.java
    + main
      + MainModule.java
    + util
      + DBConnectionUtil.java
      + PropertyUtil.java
  + src/main/resources
  + src/test/java
  + src/test/resources
  + IRE System Library [JavaSE-1.8]
  + Maven Dependencies
  + src
  + target
  + + obproperties
  + pom.xml
+ CarRentalSystem
+ HospitalManagement

139     stmt.setInt(1, customerID);
140     stmt.executeUpdate();
141   } catch (SQLException e) {
142     System.out.println("Error while deleting leases: " + e.getMessage());
143     e.printStackTrace();
144   }
145
146   // Step 3: Now delete the customer
147   String query = "DELETE FROM Customer WHERE customerID = ?";
148   try (PreparedStatement stat = connection.prepareStatement(query)) {
149     stat.setInt(1, customerID);
150     int rowsAffected = stat.executeUpdate();
151     if (rowsAffected == 0) {
152       throw new CustomerNotFoundException("Customer with ID " + customerID + " not found.");
153     }
154   } catch (SQLException e) {
155     System.out.println("Error while deleting customer: " + e.getMessage());
156     e.printStackTrace();
157   }
158 }

182+ @Override
183 public List<Customer> listCustomers() {
184   List<Customer> customers = new ArrayList<>();
185   String query = "SELECT * FROM Customer";
186   try (PreparedStatement stat = connection.prepareStatement(query)) {
187     ResultSet rs = stat.executeQuery();
188     while (rs.next()) {
189       customers.add(mapCustomer(rs));
190     }
191   } catch (SQLException e) {
192     e.printStackTrace();
193   }
194   return customers;
195 }
196
197+ @Override
198 public Customer findCustomerById(int customerID) {
199   String query = "SELECT * FROM Customer WHERE customerID = ?";
200   try (PreparedStatement stat = connection.prepareStatement(query)) {
201     stat.setInt(1, customerID);
202     ResultSet rs = stat.executeQuery();
203     if (rs.next()) {
204       return mapCustomer(rs);
205     } else {
206       throw new CustomerNotFoundException("Customer with ID " + customerID + " not found.");
207     }
208   } catch (SQLException e) {
209     e.printStackTrace();
210   }
211   return null;
212 }
213
214 // --- Lease Management ---
215
216+ @Override
217 public Lease createLease(int carID, int carID, Date startDate, Date endDate) {
218   String leaseQuery = "INSERT INTO Lease (vehicleID, customerID, startDate, endDate, type) VALUES (?, ?, ?, ?, ?)";
219   String updateCarStatusQuery = "UPDATE Vehicle SET status = 'notAvailable' WHERE vehicleID = ?";
220
221   try {
222     // Start a transaction
223     connection.setAutoCommit(false);
224
225     // Create the lease
226     try (PreparedStatement stat = connection.prepareStatement(leaseQuery, Statement.RETURN_GENERATED_KEYS)) {
227       stat.setInt(1, carID);
228       stat.setInt(2, customerID);
229       stat.setDate(3, startDate);
230       stat.setDate(4, endDate);
231       stat.setString(5, calculateLeaseType(startDate, endDate));
232       stat.executeUpdate();
233
234     // Retrieve the generated leaseID
235     ResultSet rs = stat.getGeneratedKeys();
236     int leaseID = 0;
237     if (rs.next()) {
238       leaseID = rs.getInt(1);
239     }
240
241     // Update the car status to 'notAvailable'
242     try (PreparedStatement updateStat = connection.prepareStatement(updateCarStatusQuery)) {
243       updateStat.setInt(1, carID);
244       updateStat.executeUpdate();
245     }
246
247     // Commit the transaction
248     connection.commit();
249
250     return new Lease(leaseID, carID, customerID, startDate, endDate, calculateLeaseType(startDate, endDate));
251   } catch (SQLException e) {
252     connection.rollback(); // Rollback if any exception occurs
253     throw e;
254   }
255 } catch (SQLException e) {
256   e.printStackTrace();
257   return null;
258 }
259
260
261
262
263+ @Override
264 public void returnCar(int leaseID) {
265   // Update the vehicle's status to 'available'
266   String query = "UPDATE Vehicle SET status = 'available' WHERE vehicleID = (SELECT vehicleID FROM Lease WHERE leaseID = ?)";
267   try (PreparedStatement stat = connection.prepareStatement(query)) {
268     stat.setInt(1, leaseID);
269     int rowsAffected = stat.executeUpdate();
270
271   // If no rows were affected, the lease was not found
272   if (rowsAffected < 1) {
273     throw new LeaseNotFoundException("Lease with ID " + leaseID + " not found.");
274   }
275 }

```



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "current/src/main/java".
- Java Editor:** The active editor displays the code for `CarLeaseRepositoryImpl.java`. The code implements the `CarLeaseRepository` interface, handling lease operations like listing active leases and leases with history.

```
current/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
src/main/java
  +-- dao
    +-- CarLeaseRepositoryImpl.java
    +-- ICarLeaseRepository.java
  +-- entity
    +-- Customer.java
    +-- Lease.java
    +-- Payment.java
    +-- Vehicle.java
  +-- exception
    +-- CanNotFoundException.java
    +-- CustomerNotFoundException.java
    +-- LeaseNotFoundException.java
  +-- main
    +-- MainModule.java
  +-- util
    +-- DBConnectionUtil.java
    +-- PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml
CarRentalSystem
HospitalManagement

CarLeaseRepositoryImpl.java x
270
271     // If no rows were affected, the lease was not found
272     if (rowsAffected == 0) {
273         throw new LeaseNotFoundException("Lease with ID " + leaseID + " not found.");
274     }
275
276     // Commit the transaction to make sure changes are saved
277     connection.commit();
278 } catch (SQLException e) {
279     System.out.println("Error while returning car: " + e.getMessage());
280     e.printStackTrace();
281 }
282 }
283
284
285 * @Override
286 public List<Lease> listActiveLeases() {
287     List<Lease> activeLeases = new ArrayList<>();
288     String query = "SELECT * FROM Lease WHERE endDate > CURDATE()";
289     try (PreparedStatement stmt = connection.prepareStatement(query)) {
290         ResultSet rs = stmt.executeQuery();
291         while (rs.next()) {
292             activeLeases.add(mapLease(rs));
293         }
294     } catch (SQLException e) {
295         e.printStackTrace();
296     }
297     return activeLeases;
298 }
299
300 * @Override
301 public List<Lease> listLeaseHistory() {
302     List<Lease> leases = new ArrayList<>();
303     String query = "SELECT * FROM Lease";
304     try (PreparedStatement stmt = connection.prepareStatement(query)) {
305         ResultSet rs = stmt.executeQuery();
306         while (rs.next()) {
307             leases.add(mapLease(rs));
308         }
309     }
```



The screenshot shows the Eclipse IDE interface with the Java editor open to the file `CarLeaseRepositoryImpl.java`. The code implements a database connection and performs lease validation and payment recording.

```
current/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
Carental
src/main/java
  dao
    CarLeaseRepositoryImpl.java
    ICarLeaseRepository.java
entity
  Customer.java
  Lease.java
  Payment.java
  Vehicle.java
exception
  ClassNotFoundException.java
  CustomNotFoundException.java
  LeaseNotFoundException.java
main
  MainModule.java
util
  DBConnectionUtil.java
  PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml
CarRentalSystem
HospitalManagement

CarLeaseRepositoryImpl.java x
  306       while (rs.next()) {
  307           leases.add(mapLease(rs));
  308       }
  309   } catch (SQLException e) {
  310       e.printStackTrace();
  311   }
  312   return leases;
  313 }
  314
  315 // --- Payment Handling ---
  316
  317 @Override
  318 public void recordPayment(int leaseID, double amount) {
  319     // Validate lease ID exists
  320     String validateLeaseID = "SELECT leaseID FROM Lease WHERE leaseID = ?";
  321     try (PreparedStatement validateStat = connection.prepareStatement(validateLeaseQuery)) {
  322         validateStat.setInt(1, leaseID);
  323         ResultSet rs = validateStat.executeQuery();
  324         if (!rs.next()) {
  325             throw new IllegalArgumentException("Lease ID " + leaseID + " does not exist.");
  326         }
  327     } catch (SQLException e) {
  328         System.out.println("Error while validating lease ID: " + e.getMessage());
  329         e.printStackTrace();
  330         return;
  331     }
  332
  333     // Insert the payment record
  334     String query = "INSERT INTO Payment (leaseID, paymentDate, amount) VALUES (?, CURDATE()), ?";
  335     try (PreparedStatement stat = connection.prepareStatement(query)) {
  336         stat.setInt(1, leaseID);
  337         stat.setDouble(2, amount);
  338         stat.executeUpdate();
  339         System.out.println("Payment of " + amount + " recorded successfully for Lease ID: " + leaseID);
  340     } catch (SQLException e) {
  341         System.out.println("Error while recording payment: " + e.getMessage());
  342         e.printStackTrace();
  343     }
  344 }
```

carrental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE

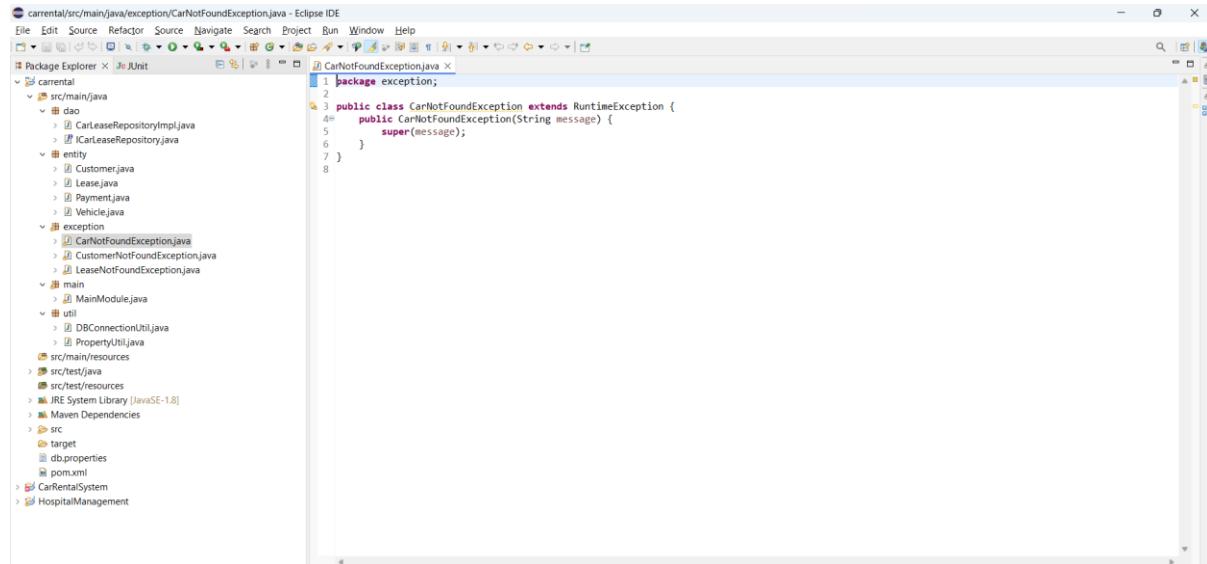
```
340@override
341    public List<Payment> getPaymentHistory(int customerID) {
342        List<Payment> payments = new ArrayList<>();
343        String query = "SELECT * FROM Payment WHERE leaseID IN (SELECT leaseID FROM Lease WHERE customerID = ?)";
344        try (PreparedStatement stat = connection.prepareStatement(query)) {
345            stat.setInt(1, customerID);
346            ResultSet rs = stat.executeQuery();
347            while (rs.next()) {
348                payments.add(mapPayment(rs));
349            }
350        } catch (SQLException e) {
351            e.printStackTrace();
352        }
353        return payments;
354    }
355
356    @Override
357    public double calculateTotalRevenue() {
358        String query = "SELECT SUM(amount) AS totalRevenue FROM Payment";
359        try (PreparedStatement stat = connection.prepareStatement(query)) {
360            ResultSet rs = stat.executeQuery();
361            if (rs.next()) {
362                return rs.getDouble("totalRevenue");
363            }
364        } catch (SQLException e) {
365            e.printStackTrace();
366        }
367        return 0;
368    }
369
370    // --- Helper Methods ---
371
372    private Vehicle mapVehicle(ResultSet rs) throws SQLException {
373        return new Vehicle(
374            rs.getInt("vehicleID"),
375            rs.getString("make"),
376            rs.getString("model"),
377            rs.getInt("year"));
378    }
379
380    private Customer mapCustomer(ResultSet rs) throws SQLException {
381        return new Customer(
382            rs.getInt("customerID"),
383            rs.getString("firstName"),
384            rs.getString("lastName"),
385            rs.getString("email"),
386            rs.getString("phoneNumber"));
387    }
388
389    private Lease maplease(ResultSet rs) throws SQLException {
390        return new Lease(
391            rs.getInt("leaseID"),
392            rs.getInt("vehicleID"),
393            rs.getInt("customerID"),
394            rs.getDate("startDate"),
395            rs.getDate("endDate"),
396            rs.getString("type"));
397    }
398
399    private Payment mapPayment(ResultSet rs) throws SQLException {
400        return new Payment(
401            rs.getInt("paymentID"),
402            rs.getInt("leaseID"),
403            rs.getDate("paymentDate"),
404            rs.getDouble("amount"));
405    }
406
407    private String calculateLeaseType(Date startDate, Date endDate) {
408        long difference = endDate.getTime() - startDate.getTime();
409        long days = difference / (1000 * 60 * 60 * 24);
410        return days > 30 ? "Monthly" : "Daily";
411    }
412}
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
```

carrental/src/main/java/dao/CarLeaseRepositoryImpl.java - Eclipse IDE

```
390
391    private Customer mapCustomer(ResultSet rs) throws SQLException {
392        return new Customer(
393            rs.getInt("customerID"),
394            rs.getString("firstName"),
395            rs.getString("lastName"),
396            rs.getString("email"),
397            rs.getString("phoneNumber"));
398    }
399
400    private Lease maplease(ResultSet rs) throws SQLException {
401        return new Lease(
402            rs.getInt("leaseID"),
403            rs.getInt("vehicleID"),
404            rs.getInt("customerID"),
405            rs.getDate("startDate"),
406            rs.getDate("endDate"),
407            rs.getString("type"));
408    }
409
410    private Payment mapPayment(ResultSet rs) throws SQLException {
411        return new Payment(
412            rs.getInt("paymentID"),
413            rs.getInt("leaseID"),
414            rs.getDate("paymentDate"),
415            rs.getDouble("amount"));
416    }
417
418    private String calculateLeaseType(Date startDate, Date endDate) {
419        long difference = endDate.getTime() - startDate.getTime();
420        long days = difference / (1000 * 60 * 60 * 24);
421        return days > 30 ? "Monthly" : "Daily";
422    }
423
424
425
426
427
```

exception Package :

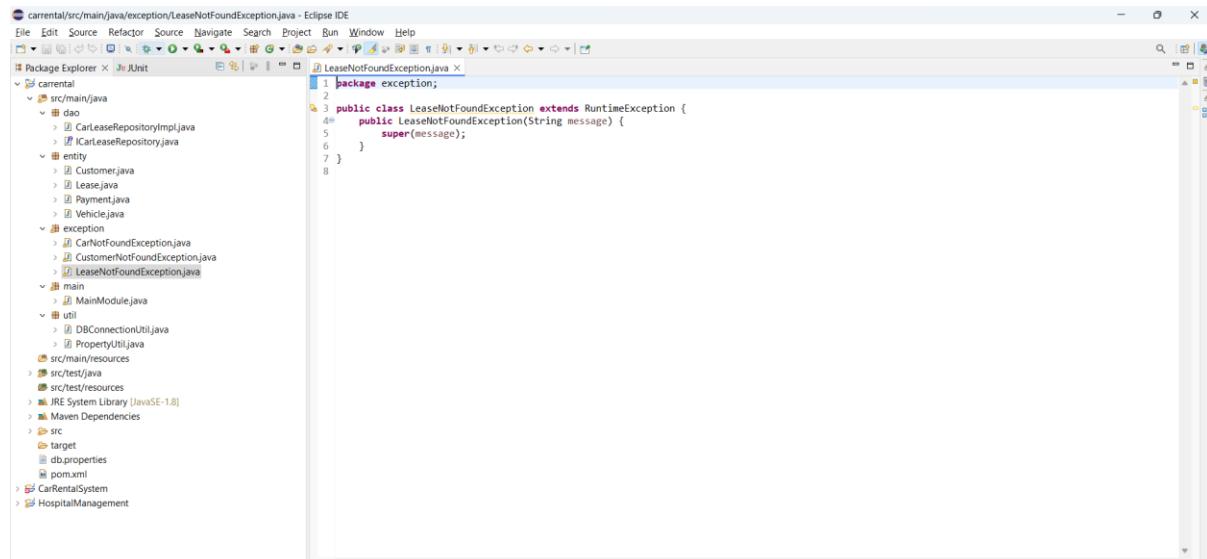
CarNotFoundException.java:



```
currental/src/main/java/exception/CarNotFoundException.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer JUnit
src/main/java
  dao
    CarLeaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundException.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml
CarRentalSystem
HospitalManagement
```

```
CarNotFoundException.java
1 package exception;
2
3 public class CarNotFoundException extends RuntimeException {
4   public CarNotFoundException(String message) {
5     super(message);
6   }
7 }
```

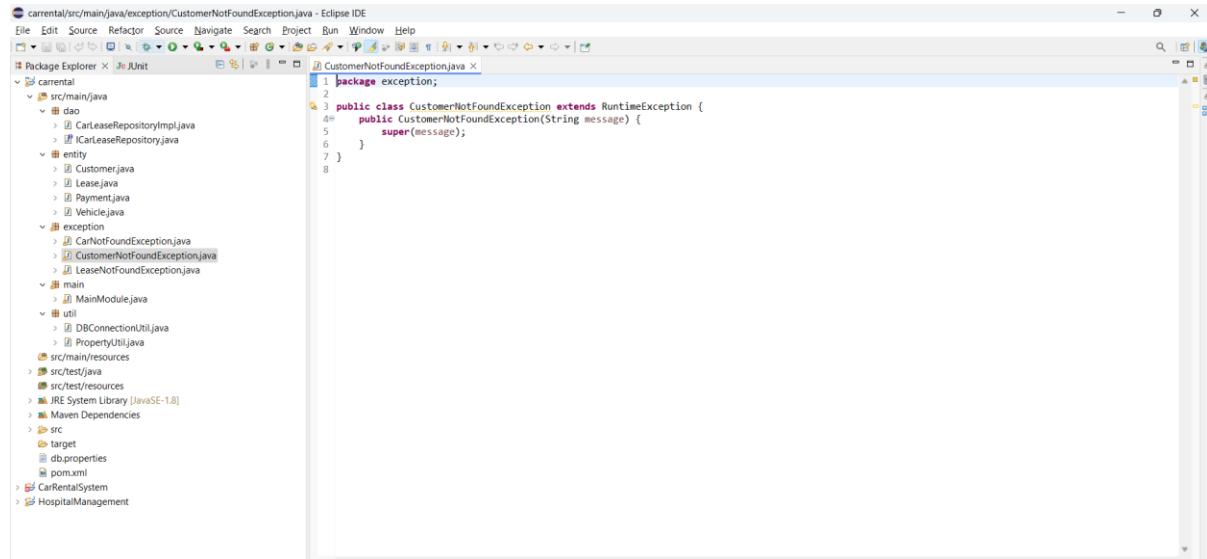
LeaseNotFoundException.java



```
currental/src/main/java/exception/LeaseNotFoundException.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer JUnit
src/main/java
  dao
    CarLeaseRepositoryImpl.java
    ICarLeaseRepository.java
  entity
    Customer.java
    Lease.java
    Payment.java
    Vehicle.java
  exception
    CarNotFoundException.java
    CustomerNotFoundException.java
    LeaseNotFoundException.java
  main
    MainModule.java
  util
    DBConnectionUtil.java
    PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml
CarRentalSystem
HospitalManagement
```

```
LeaseNotFoundException.java
1 package exception;
2
3 public class LeaseNotFoundException extends RuntimeException {
4   public LeaseNotFoundException(String message) {
5     super(message);
6   }
7 }
```

CustomerNotFoundException.java

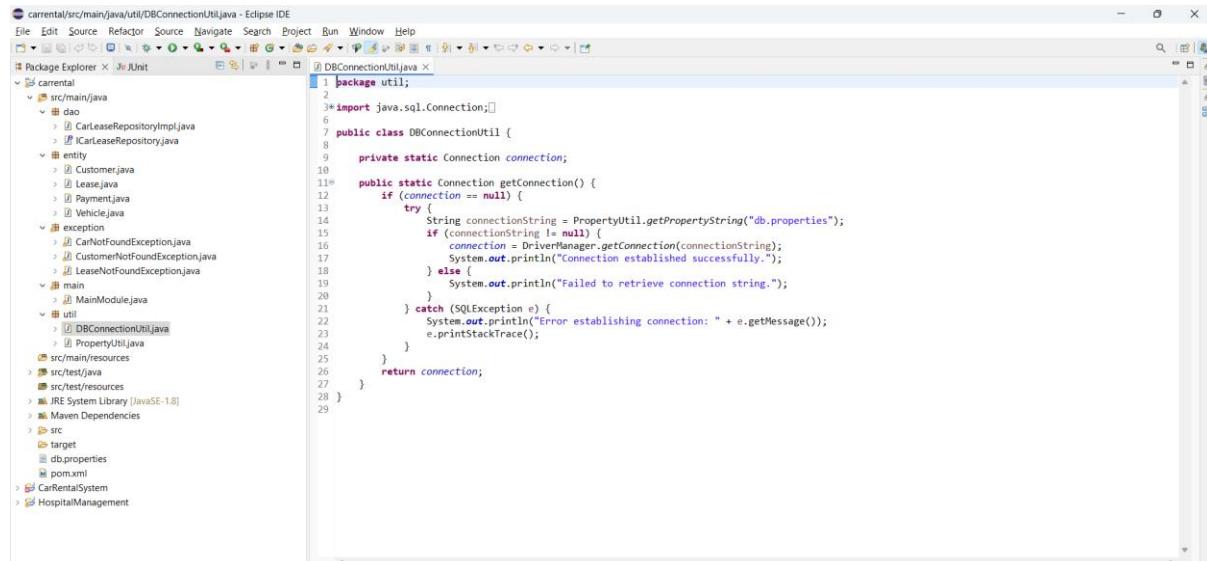


The screenshot shows the Eclipse IDE interface with the 'CustomerNotFoundException.java' file open in the central editor window. The code defines a class 'CustomerNotFoundException' that extends 'RuntimeException'. The package is 'exception'. The code includes imports for 'java.lang.RuntimeException' and 'exception.CustomerNotFoundException'. The class has a constructor that takes a string message and calls the super constructor.

```
1 package exception;
2
3 public class CustomerNotFoundException extends RuntimeException {
4     public CustomerNotFoundException(String message) {
5         super(message);
6     }
7 }
```

util Package:

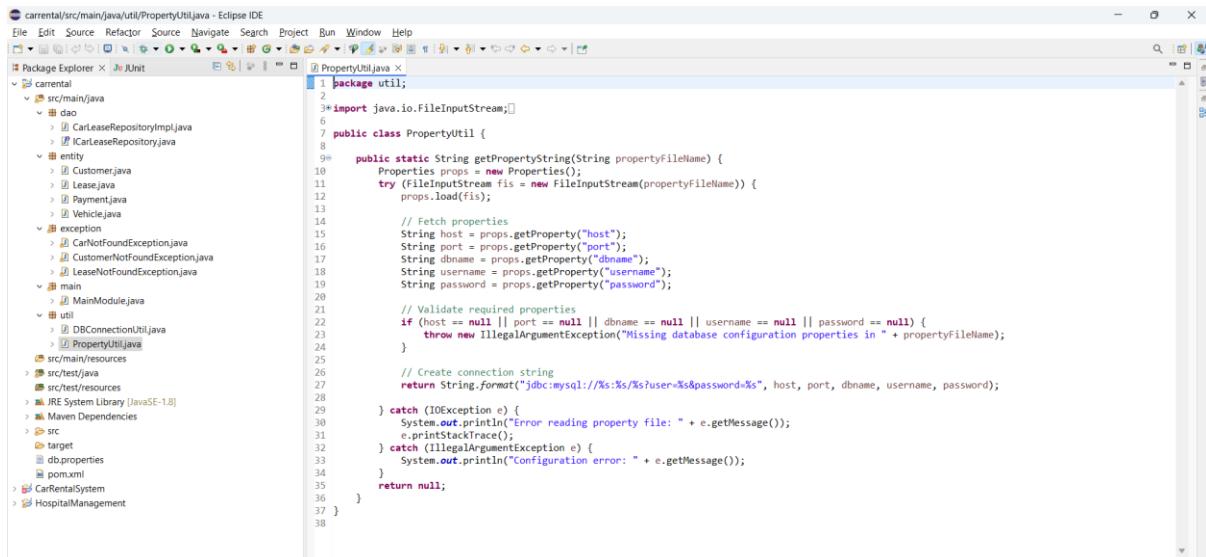
DBConnectionUtil.java:



The screenshot shows the Eclipse IDE interface with the 'DBConnectionUtil.java' file open in the central editor window. The code defines a static utility class 'DBConnectionUtil' with a single static method 'getConnection'. This method retrieves a connection from a static variable 'connection'. If 'connection' is null, it tries to establish a new connection using the 'DriverManager.getConnection' method, passing in a connection string from the 'db.properties' file. If successful, it prints a success message to the console; if not, it prints a failure message. It then returns the connection.

```
1 package util;
2
3 import java.sql.Connection;
4
5 public class DBConnectionUtil {
6     private static Connection connection;
7
8     public static Connection getConnection() {
9         if (connection == null) {
10             try {
11                 String connectionString = PropertyUtil.getPropertyString("db.properties");
12                 if (connectionString != null) {
13                     connection = DriverManager.getConnection(connectionString);
14                     System.out.println("Connection established successfully.");
15                 } else {
16                     System.out.println("Failed to retrieve connection string.");
17                 }
18             } catch (SQLException e) {
19                 System.out.println("Error establishing connection: " + e.getMessage());
20                 e.printStackTrace();
21             }
22         }
23         return connection;
24     }
25 }
26
27 }
```

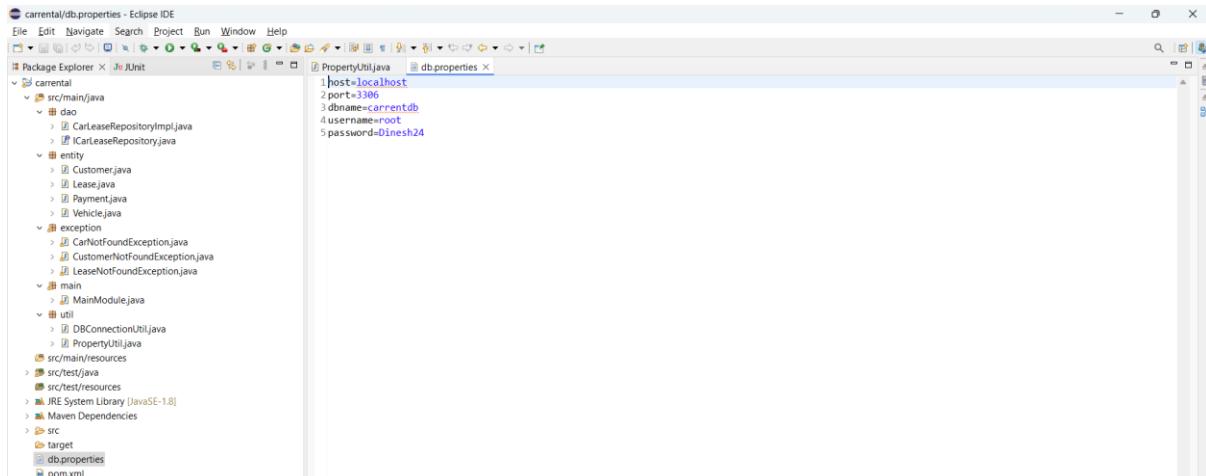
PropertyUtil.java:



```
currental/src/main/java/util/PropertyUtil.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
src/main/java
  currental
    dao
      CarLeaseRepositoryImpl.java
      ICarLeaseRepository.java
    entity
      Customer.java
      Lease.java
      Payment.java
      Vehicle.java
    exception
      CarNotFoundException.java
      CustomerNotFoundException.java
      LeaseNotFoundException.java
    main
      MainModule.java
    util
      DBConnectionUtil.java
      PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml
CarRentalSystem
HospitalManagement

PropertyUtil.java
1 package util;
2
3 import java.io.FileInputStream;
4
5 public class PropertyUtil {
6
7     public static String getPropertyString(String propertyFileName) {
8         Properties props = new Properties();
9         try (FileInputStream fis = new FileInputStream(propertyFileName)) {
10             props.load(fis);
11
12             // Fetch properties
13             String host = props.getProperty("host");
14             String port = props.getProperty("port");
15             String dbname = props.getProperty("dbname");
16             String username = props.getProperty("username");
17             String password = props.getProperty("password");
18
19             // Validate required properties
20             if (host == null || port == null || dbname == null || username == null || password == null) {
21                 throw new IllegalArgumentException("Missing database configuration properties in " + propertyFileName);
22             }
23
24             // Create connection string
25             return String.format("jdbc:mysql://%s:%s?user=%s&password=%s", host, port, dbname, username, password);
26
27         } catch (IOException e) {
28             System.out.println("Error reading property file: " + e.getMessage());
29             e.printStackTrace();
30         } catch (IllegalArgumentException e) {
31             System.out.println("Configuration error: " + e.getMessage());
32         }
33     }
34
35     return null;
36 }
37
38
```

db.properties (file in root folder)

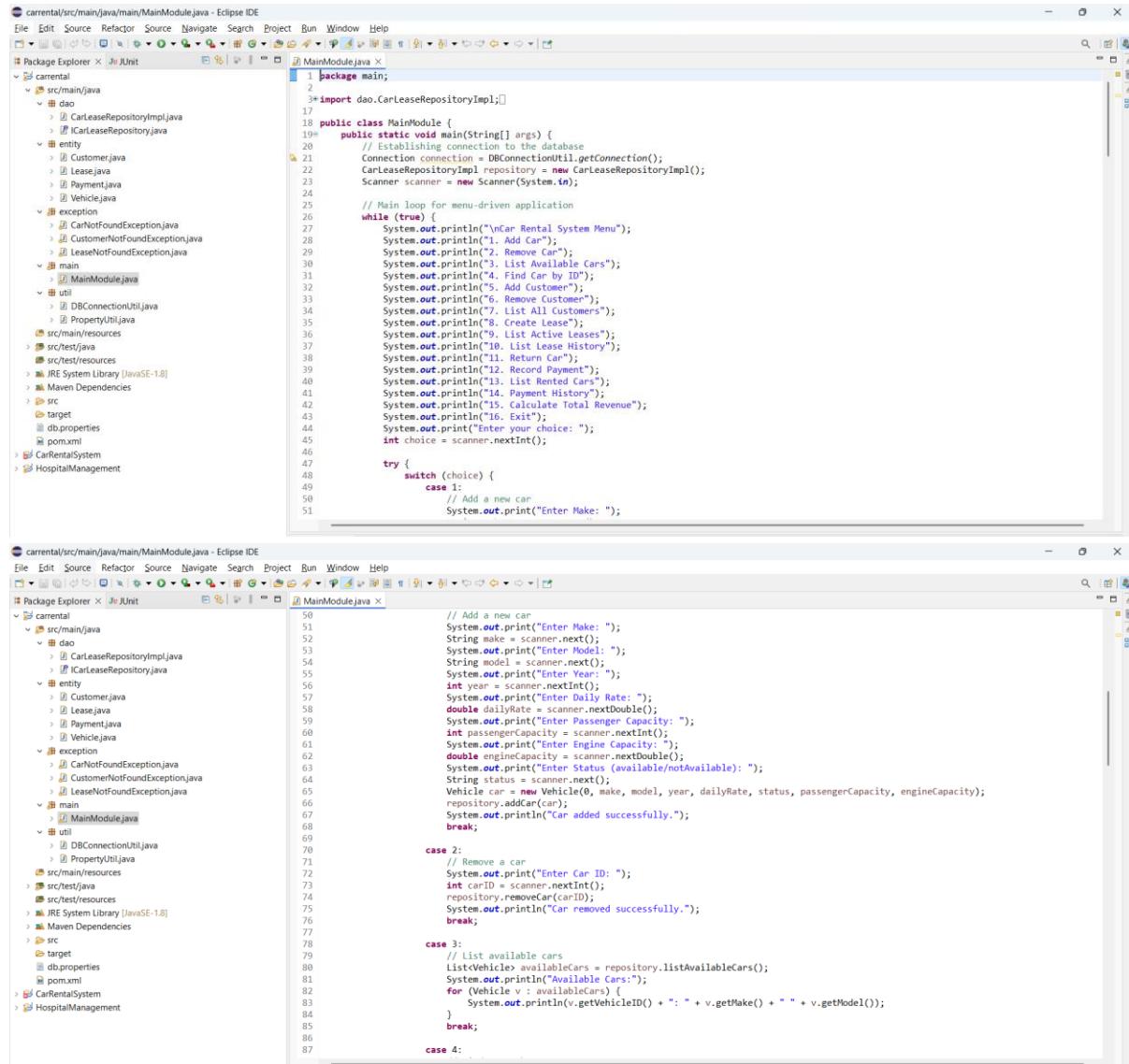


```
currental/db.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer X JUnit
src/main/java
  currental
    dao
      CarLeaseRepositoryImpl.java
      ICarLeaseRepository.java
    entity
      Customer.java
      Lease.java
      Payment.java
      Vehicle.java
    exception
      CarNotFoundException.java
      CustomerNotFoundException.java
      LeaseNotFoundException.java
    main
      MainModule.java
    util
      DBConnectionUtil.java
      PropertyUtil.java
src/main/resources
src/test/java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
target
db.properties
pom.xml

db.properties
1 host=localhost
2 port=3306
3 dbname=Carrentdb
4 username=root
5 password=Dinesh24
```

main Package:

MainModule.java:



The image shows two side-by-side views of the Eclipse IDE interface, both displaying the same Java code for `MainModule.java`. The code is a menu-driven application for a car rental system.

```
package main;
import dao.CarLeaseRepositoryImpl;
public class MainModule {
    public static void main(String[] args) {
        // Establishing connection to the database
        Connection connection = DBConnectionUtil.getConnection();
        CarLeaseRepositoryImpl repository = new CarLeaseRepositoryImpl();
        Scanner scanner = new Scanner(System.in);

        // Main loop for menu-driven application
        while (true) {
            System.out.println("\nCar Rental System Menu");
            System.out.println("1. Add Car");
            System.out.println("2. Remove Car");
            System.out.println("3. List Available Cars");
            System.out.println("4. Add Customer");
            System.out.println("5. Add Payment");
            System.out.println("6. Remove Customer");
            System.out.println("7. List All Customers");
            System.out.println("8. Create Lease");
            System.out.println("9. List Active Leases");
            System.out.println("10. List Lease History");
            System.out.println("11. Return Car");
            System.out.println("12. Record Payment");
            System.out.println("13. Calculate Revenue");
            System.out.println("14. Payment History");
            System.out.println("15. Calculate Total Revenue");
            System.out.println("16. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

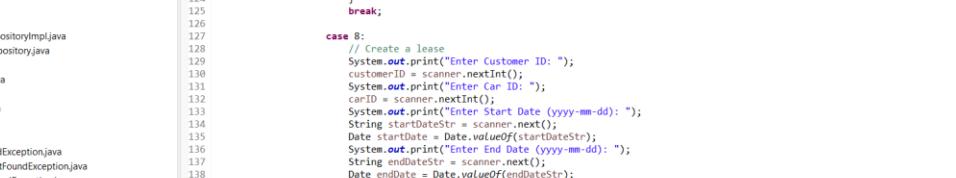
            try {
                switch (choice) {
                    case 1:
                        // Add a new car
                        System.out.print("Enter Make: ");
                        String make = scanner.nextLine();
                        System.out.print("Enter Model: ");
                        String model = scanner.nextLine();
                        System.out.print("Enter Year: ");
                        int year = scanner.nextInt();
                        System.out.print("Enter Daily Rate: ");
                        double dailyRate = scanner.nextDouble();
                        System.out.print("Enter Passenger Capacity: ");
                        int passengerCapacity = scanner.nextInt();
                        System.out.print("Enter Engine Capacity: ");
                        double engineCapacity = scanner.nextDouble();
                        System.out.print("Enter Status (available/notAvailable): ");
                        String status = scanner.nextLine();
                        Vehicle car = new Vehicle(0, make, model, year, dailyRate, status, passengerCapacity, engineCapacity);
                        repository.addCar(car);
                        System.out.println("Car added successfully.");
                        break;
                    case 2:
                        // Remove a car
                        System.out.print("Enter Car ID: ");
                        int carID = scanner.nextInt();
                        repository.removeCar(carID);
                        System.out.println("Car removed successfully.");
                        break;
                    case 3:
                        // List available cars
                        List<Vehicle> availableCars = repository.listAvailableCars();
                        System.out.println("Available Cars:");
                        for (Vehicle v : availableCars) {
                            System.out.println(v.getVehicleID() + " " + v.getMake() + " " + v.getModel());
                        }
                        break;
                    case 4:
                }
            }
        }
    }
}
```

```
current/src/main/java/Main/MainModule.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit
MainModule.java
case 4:
    // Find a car by ID
    System.out.print("Enter Car ID: ");
    carID = scanner.nextInt();
    Vehicle foundCar = repository.findCarById(carID);
    System.out.println("Car found: " + foundCar.getMake() + " " + foundCar.getModel());
    break;

case 5:
    // Add a new customer
    System.out.print("Enter First Name: ");
    String firstName = scanner.next();
    System.out.print("Enter Last Name: ");
    String lastName = scanner.next();
    System.out.print("Enter Email: ");
    String email = scanner.next();
    System.out.print("Enter Phone Number: ");
    String phoneNumber = scanner.next();
    Customer customer = new Customer(0, firstName, lastName, email, phoneNumber);
    repository.addCustomer(customer);
    System.out.println("Customer added successfully.");
    break;

case 6:
    // Remove a customer
    System.out.print("Enter Customer ID: ");
    int customerId = scanner.nextInt();
    repository.removeCustomer(customerId);
    System.out.println("Customer removed successfully.");
    break;

case 7:
    // List all customers
    List<Customer> customers = repository.listCustomers();
    System.out.println("Customers:");
    for (Customer c : customers) {
        System.out.println(c.getCustomerId() + ": " + c.getFirstName() + " " + c.getLastName());
    }
}
```



The screenshot shows the Eclipse IDE interface with the following details:

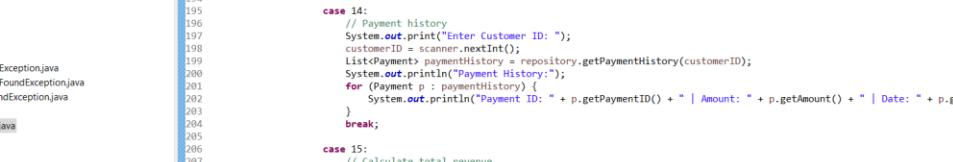
- Project Explorer:** Shows the project structure with packages like `com.cental`, `com.cental.dao`, `com.cental.entity`, `com.cental.exception`, `com.cental.main`, `com.cental.util`, and resources like `src/main/resources`.
- MainModule.java:** The current file being edited, containing Java code for a car rental system.
- Code Content:** The code includes several `case` statements corresponding to different lease creation scenarios. Each case prints customer and car details, creates a lease object, and prints a success message. The code uses `System.out.println` for output.

```
current/src/main/java/MainModule.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer X JUnit X MainModule.java X
124     System.out.println("Customer ID: " + l.getCustomerID() + " | Car ID: " + l.getCarID());
125 }
126 break;
127
case 8:
128     // Create a lease
129     System.out.print("Enter Customer ID: ");
130     customerID = scanner.nextInt();
131     System.out.print("Enter Car ID: ");
132     carID = scanner.nextInt();
133     System.out.print("Enter Start Date (yyyy-mm-dd): ");
134     String startDateStr = scanner.nextLine();
135     Date startDate = Date.valueOf(startDateStr);
136     System.out.print("Enter End Date (yyyy-mm-dd): ");
137     String endDateStr = scanner.nextLine();
138     Date endDate = Date.valueOf(endDateStr);
139     Lease lease = repository.createLease(customerID, carID, startDate, endDate);
140     System.out.println("Lease created successfully. Lease ID: " + lease.getLeaseID());
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
case 9:
162     // List active leases
163     List<Lease> activeLeases = repository.listActiveLeases();
164     System.out.println("Active Leases:");
165     for (Lease l : activeLeases) {
166         System.out.println("Lease ID: " + l.getLeaseID() + " | Customer ID: " + l.getCustomerID());
167     }
168     break;
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
427
428
429
429
430
431
432
433
434
435
436
437
437
438
439
439
440
441
442
443
444
445
446
447
447
448
449
449
450
451
452
453
454
455
456
457
457
458
459
459
460
461
462
463
464
465
465
466
467
467
468
469
469
470
471
472
473
473
474
475
475
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1
```

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** Shows the project structure with packages like `src/main/java`, `src/test/java`, and `src/main/resources`. It also lists JRE System Library [JavaSE-1.8], Maven Dependencies, and two external projects: `CarentalSystem` and `HospitalManagement`.
- Java Editor View:** Displays the `MainModule.java` file. The code handles various cases (11 through 14) for car lease operations, including printing lease IDs, recording payments, listing rented cars, and handling exceptions.
- Top Bar:** Shows the menu bar with options like File, Edit, Source, Refactor, Source, Navigate, Search, Project, Run, Window, Help.

```
159         break;
160
161     case 11:
162         // Return a car
163         System.out.print("Enter Lease ID: ");
164         int leaseID = scanner.nextInt();
165         repository.returnCar(leaseID);
166         System.out.println("Car returned successfully.");
167         break;
168
169     case 12:
170         // Record a payment
171         System.out.print("Enter Lease ID: ");
172         leaseID = scanner.nextInt();
173         System.out.print("Enter Payment Amount: ");
174         double amount = scanner.nextDouble();
175         repository.recordPayment(leaseID, amount);
176         System.out.println("Payment recorded successfully.");
177         break;
178
179     case 13:
180         try {
181             List<Vehicle> rentedCars = repository.listRentedCars();
182             System.out.println("Rented Cars:");
183             if (rentedCars.isEmpty()) {
184                 System.out.println("No cars are currently rented.");
185             } else {
186                 for (Vehicle car1 : rentedCars) {
187                     System.out.println(car1);
188                 }
189             }
190         } catch (Exception e) {
191             System.err.println("Error while retrieving rented cars: " + e.getMessage());
192         }
193         break;
194
195     case 14:
196         // Payment history
197         System.out.println("Customer and vehicle/Carson Customer ID: ");
```



The screenshot shows the Eclipse IDE interface with the title bar "currental/src/main/java/main/MainModule.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Source, Navigate, Project, Run, Window, Help. The toolbar has icons for New, Open, Save, Cut, Copy, Paste, Find, Replace, and others. The left sidebar is the Package Explorer showing the project structure:

- currental
- src/main/java
 - dao
 - entity
 - exception
 - CarNotFoundException.java
 - CustomerNotFoundException.java
 - LeaseNotFoundException.java
 - main
 - MainModule.java
 - util
- src/main/resources
- src/test/java
 - CarLeaseRepositoryImplTest.java
- src/test/resources
- IREE System Library [JavaSE-1.8]
- Maven Dependencies
- src
- target
- db.properties
- pom.xml
- CarentalSystem
- HospitalManagement

The main editor window displays the MainModule.java code:

```
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231

case 14:
    System.out.println("Payment history");
    customerID = scanner.nextInt();
    List<Payment> paymentHistory = repository.getPaymentHistory(customerID);
    System.out.println("Payment History:");
    for (Payment p : paymentHistory) {
        System.out.println("Payment ID: " + p.getPaymentID() + " | Amount: " + p.getAmount() + " | Date: " + p.getPaymentDate());
    }
    break;

case 15:
    // Calculate total revenue
    double totalRevenue = repository.calculateTotalRevenue();
    System.out.println("Total Revenue: " + totalRevenue);
    break;

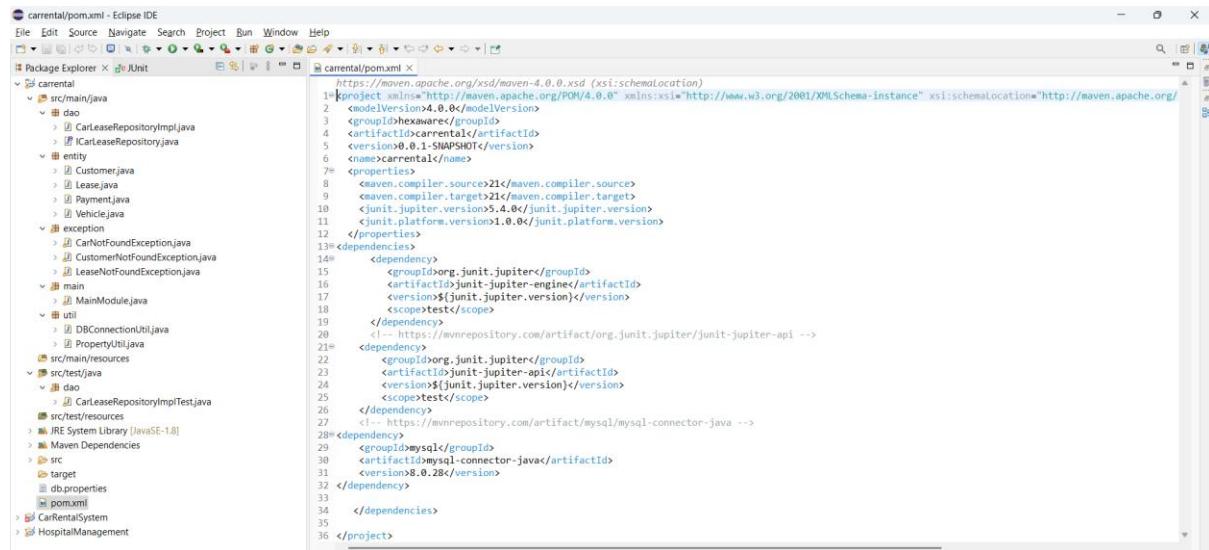
case 16:
    System.out.println("Exiting the system...");
    scanner.close();
    System.exit(0);
    break;

default:
    System.out.println("Invalid choice. Please try again.");
}

} catch (CarNotFoundException | CustomerNotFoundException | LeaseNotFoundException e) {
    System.err.println("Error: " + e.getMessage());
} catch (Exception e) {
    System.err.println("Unexpected error: " + e.getMessage());
    e.printStackTrace();
}
}
```

Pom.xml file for dependencies:

Pom.xml:



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/4.0.0 http://www.w3.org/2001/XMLSchema-instance">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>carental</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Car Rental System</name>
    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <junit.jupiter.version>5.8.0</junit.jupiter.version>
        <junit.platform.version>1.0.0</junit.platform.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>${junit.jupiter.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>mysql:mysql-connector-java</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.28</version>
        </dependency>
    </dependencies>
</project>
```

Output (By running MainModule.java):

1)



```
MainModule (2) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (23 Nov 2024, 1:28:21 pm) [pid: 19424]
Connection established successfully.

Car Rental System Menu
1. Add Car
2. Remove Car
3. List Available Cars
4. Find Car by ID
5. Add Customer
6. Remove Customer
7. List All Customers
8. Create Lease
9. List Active Leases
10. List Lease History
11. Return Car
12. Record Payment
13. List Rented Cars
14. Payment History
15. Calculate Total Revenue
16. Exit
Enter your choice: 1
Enter Make: Tata
Enter Model: 21x
Enter Year: 2024
Enter Daily Rate: 20
Enter Passenger Capacity: 5
Enter Engine Capacity: 2
Enter Status (available/notAvailable): available
Car added successfully.
```

2)

```
Enter your choice: 2
Enter Car ID: 1
Car removed successfully.
```

3)

```
| Enter your choice: 3
| Available Cars:
| 2: Honda Civic
| 9: Tata 21x
```

4)

```
| Enter your choice: 4
| Enter Car ID: 2
| Car found: Honda Civic
```

5)

```
| Enter your choice: 5
| Enter First Name: Dinesh
| Enter Last Name: Saravanan
| Enter Email: saravana1974dp@gmail.com
| Enter Phone Number: 6369505351
| Customer added successfully.
```

6)

```
| Enter your choice: 6
| Enter Customer ID: 1
| Customer removed successfully.
```

7)

```
| Enter your choice: 7
| Customers:
| 2: Jane Smith
| 9: Dinesh Saravanan
```

8)

```
| Enter your choice: 8
| Enter Customer ID: 9
| Enter Car ID: 2
| Enter Start Date (yyyy-mm-dd): 2024-12-25
| Enter End Date (yyyy-mm-dd): 2024-12-30
| Lease created successfully. Lease ID: 56
```

9)

```
| Enter your choice: 9
| Active Leases:
| Lease ID: 55 | Customer ID: 2
| Lease ID: 56 | Customer ID: 9
```

10)

```
| Enter your choice: 10
| Lease History:
| Lease ID: 55 | Customer ID: 2
| Lease ID: 56 | Customer ID: 9
```

11)

```
| Enter your choice: 11
| Enter Lease ID: 56
| Car returned successfully.
```

12)

```
| Enter your choice: 12
| Enter Lease ID: 56
| Enter Payment Amount: 500
| Payment of 500.0 recorded successfully for Lease ID: 56
| Payment recorded successfully.
```

13)

```
| Enter your choice: 13
| No cars are currently rented.
| Rented Cars:
| No cars are currently rented.
```

14)

```
| Enter your choice: 14
| Enter Customer ID: 9
| Payment History:
| Payment ID: 58 | Amount: 500.0 | Date: 2024-11-23
```

15)

```
| Enter your choice: 15  
| Total Revenue: 500.0
```

Exception Handling:

Throwing Exception where Customer Id , Car Id , Lease Id are invalid.

1)Customer Id Invalid-

```
| Enter your choice: 6  
| Enter Customer ID: 6669  
| Error: Customer with ID 6669 not found.
```

2)Lease Id Invalid-

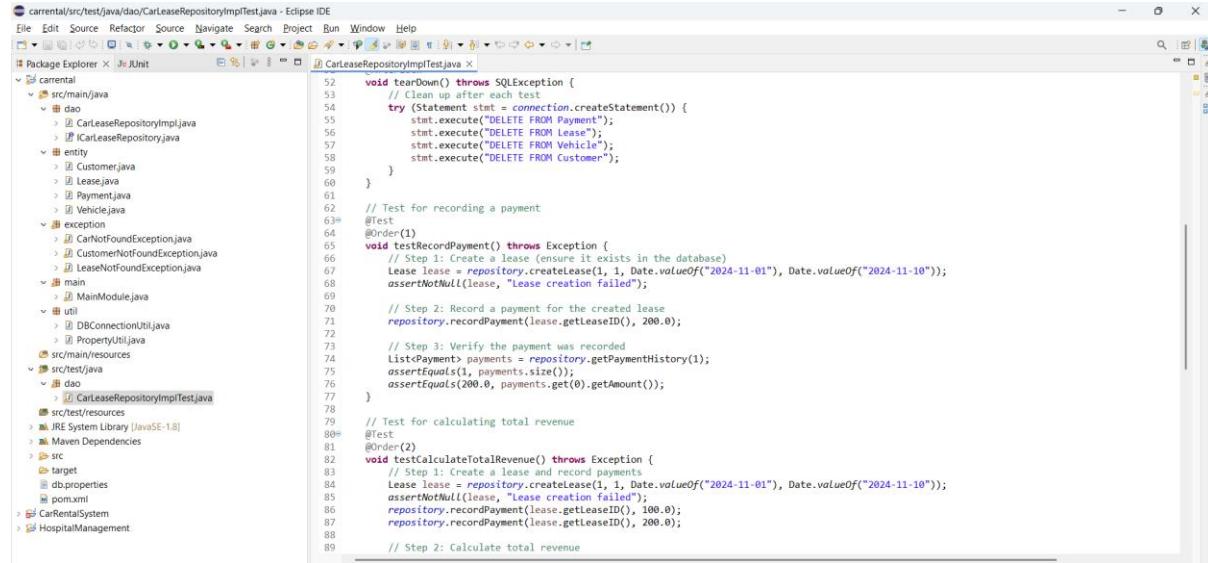
```
| Enter your choice: 11  
| Enter Lease ID: 10  
| Error: Lease with ID 10 not found.
```

3)Car Id Invalid-

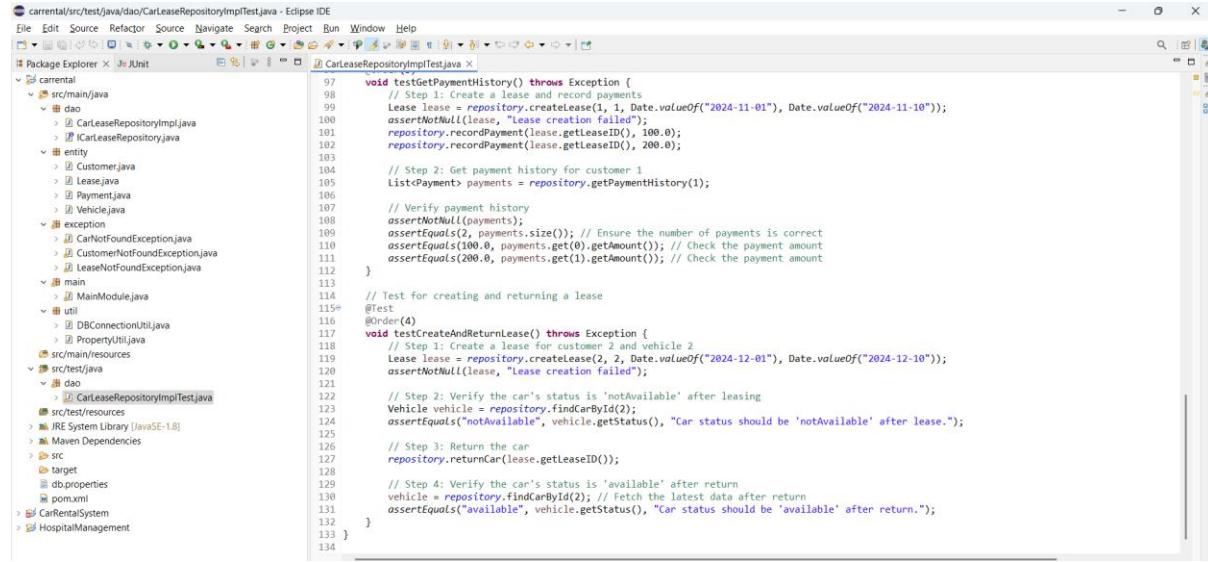
```
| Enter your choice: 4  
| Enter Car ID: 555  
| Error: Car with ID 555 not found.
```

Junit test case for CarLeaseRepositoryImpl:

CarLeaseRepositoryImplTest.java:



```
carrental/src/test/java/dao/CarLeaseRepositoryImplTest.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer × JUnit
src/main/java
  currental
    dao
      CarLeaseRepositoryImpl.java
      ICarLeaseRepository.java
    entity
      Customer.java
      Lease.java
      Payment.java
      Vehicle.java
    exception
      CarNotFoundExcption.java
      CustomerNotFoundException.java
      LeaseNotFoundException.java
    main
      MainModule.java
    util
      DBConnectionUtil.java
      PropertyUtil.java
src/main/resources
src/test/java
  currental
    dao
      CarLeaseRepositoryImplTest.java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
  CarRentalSystem
  HospitalManagement
52     void tearDown() throws SQLException {
53         // Clean up after each test
54         try (Statement stat = connection.createStatement()) {
55             stat.execute("DELETE FROM Payment");
56             stat.execute("DELETE FROM Lease");
57             stat.execute("DELETE FROM Vehicle");
58             stat.execute("DELETE FROM Customer");
59         }
60     }
61
62     // Test for recording a payment
63     @Test
64     @Order(1)
65     void testRecordPayment() throws Exception {
66         // Step 1: Create a lease (ensure it exists in the database)
67         Lease lease = repository.createLease(1, 1, Date.valueOf("2024-11-01"), Date.valueOf("2024-11-10"));
68         assertNotNull(lease, "Lease creation failed");
69
70         // Step 2: Record a payment for the created lease
71         repository.recordPayment(lease.getLeaseID(), 200.0);
72
73         // Step 3: Verify the payment was recorded
74         List<Payment> payments = repository.getPaymentHistory(1);
75         assertEquals(1, payments.size());
76         assertEquals(200.0, payments.get(0).getAmount());
77     }
78
79     // Test for calculating total revenue
80     @Test
81     @Order(2)
82     void testCalculateTotalRevenue() throws Exception {
83         // Step 1: Create a lease and record payments
84         Lease lease = repository.createLease(1, 1, Date.valueOf("2024-11-01"), Date.valueOf("2024-11-10"));
85         assertNotNull(lease, "Lease creation failed");
86         repository.recordPayment(lease.getLeaseID(), 100.0);
87         repository.recordPayment(lease.getLeaseID(), 200.0);
88
89         // Step 2: Calculate total revenue
90     }
91 }
```



```
carrental/src/test/java/dao/CarLeaseRepositoryImplTest.java - Eclipse IDE
File Edit Source Refactor Source Navigate Search Project Run Window Help
Package Explorer × JUnit
src/main/java
  currental
    dao
      CarLeaseRepositoryImpl.java
      ICarLeaseRepository.java
    entity
      Customer.java
      Lease.java
      Payment.java
      Vehicle.java
    exception
      CarNotFoundExcption.java
      CustomerNotFoundException.java
      LeaseNotFoundException.java
    main
      MainModule.java
    util
      DBConnectionUtil.java
      PropertyUtil.java
src/main/resources
src/test/java
  currental
    dao
      CarLeaseRepositoryImplTest.java
src/test/resources
IREE System Library [JavaSE-1.8]
Maven Dependencies
src
  target
    db.properties
    pom.xml
  CarRentalSystem
  HospitalManagement
97     void testGetPaymentHistory() throws Exception {
98         // Step 1: Create a lease and record payments
99         Lease lease = repository.createLease(1, 1, Date.valueOf("2024-11-01"), Date.valueOf("2024-11-10"));
100        assertNotNull(lease, "Lease creation failed");
101        repository.recordPayment(lease.getLeaseID(), 100.0);
102        repository.recordPayment(lease.getLeaseID(), 200.0);
103
104         // Step 2: Get payment history for customer 1
105         List<Payment> payments = repository.getPaymentHistory(1);
106
107         // Verify payment history
108         assertEquals(2, payments.size()); // Ensure the number of payments is correct
109         assertEquals(100.0, payments.get(0).getAmount()); // Check the payment amount
110         assertEquals(200.0, payments.get(1).getAmount()); // Check the payment amount
111     }
112
113     // Test for creating and returning a lease
114     @Test
115     @Order(4)
116     void testCreateAndReturnLease() throws Exception {
117         // Step 1: Create a lease for customer 2 and vehicle 2
118         Lease lease = repository.createLease(2, 2, Date.valueOf("2024-12-01"), Date.valueOf("2024-12-10"));
119         assertNotNull(lease, "Lease creation failed");
120
121         // Step 2: Verify the car's status is 'notAvailable' after leasing
122         Vehicle vehicle = repository.findCarById(2);
123         assertEquals("notAvailable", vehicle.getStatus(), "Car status should be 'notAvailable' after lease.");
124
125         // Step 3: Return the car
126         repository.returnCar(lease.getLeaseID());
127
128         // Step 4: Verify the car's status is 'available' after return
129         vehicle = repository.findCarById(2); // Fetch the latest data after return
130         assertEquals("available", vehicle.getStatus(), "Car status should be 'available' after return.");
131
132     }
133 }
```

Test Case 4/4 Passed:

Test Cases:

- 1) testRecordPayment() - Test for recording a payment.
- 2) testCalculateTotalRevenue() - Test for calculating total revenue.
- 3) testGetPaymentHistory() - Test for getting payment history.
- 4) testCreateAndReturnLease() - Test for creating and returning a lease.

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure with files like CarLeaseRepositoryImplTest.java.
- Code Editor:** Displays the Java test code for `CarLeaseRepositoryImplTest`. The code includes four test methods: `testGetPaymentHistory()`, `testCreateAndReturnLease()`, and two others that are partially visible.
- Java Console:** Located at the bottom, it shows the command-line output of the test run:

```
<terminated> CarLeaseRepositoryImplTest [JUnit] C:\Program Files\Java\jdk-22\bin\javaw.exe (23 Nov 2024, 1:26:12 pm – 1:26:13 pm) [pid: 13240]
Connection established successfully.
Payment of 200.0 recorded successfully for Lease ID: 52
Payment of 100.0 recorded successfully for Lease ID: 53
Payment of 200.0 recorded successfully for Lease ID: 53
Payment of 100.0 recorded successfully for Lease ID: 54
Payment of 200.0 recorded successfully for Lease ID: 54
```