**Object-Oriented Programming Exercise**

**Exercise Objective**

Design and implement a simple system to demonstrate key object-oriented programming concepts such as interface, package/namespace, public, protected, private, method overloading, method overriding, dynamic binding, abstract classes, and UML class diagrams.

**Scenario**

You are required to create a simple simulation of a vehicle rental system. The system should manage vehicles and customers, with different types of vehicles and customers having unique behaviors and attributes.

**Instructions**

1. **Define an Abstract Class Vehicle:**
   * Attributes:
     + licensePlate (String, private)
     + make (String, private)
     + model (String, private)
     + rentalPrice (double, protected)
   * Methods:
     + getLicensePlate(): returns the license plate of the vehicle
     + getMake(): returns the make of the vehicle
     + getModel(): returns the model of the vehicle
     + getRentalPrice(): returns the rental price of the vehicle
     + Abstract Method calculateRentalCost(int days): returns a double representing the rental cost based on the number of days
2. **Define Subclasses of Vehicle:**
   * Car:
     + Additional Attribute: type (String, private)
     + Override calculateRentalCost(int days): Cars have a fixed daily rental cost
   * Truck:
     + Additional Attribute: capacity (int, private)
     + Override calculateRentalCost(int days): Trucks have a fixed daily rental cost plus an additional cost based on capacity
3. **Define an Interface Customer:**
   * Methods:
     + getName(): returns the name of the customer
     + rentVehicle(Vehicle vehicle, int days): processes the rental of a vehicle for a given number of days
     + getRentedVehicles(): returns a list of vehicles rented by the customer
4. **Define Implementations of the Customer Interface:**
   * RegularCustomer:
     + Attributes: name (String, private)
     + Implement Methods:
       - getName(): returns the name of the customer
       - rentVehicle(Vehicle vehicle, int days): adds the vehicle to the customer's rented vehicles
       - getRentedVehicles(): returns the list of rented vehicles
   * CorporateCustomer:
     + Additional Attribute: companyName (String, private)
     + Override rentVehicle(Vehicle vehicle, int days): Corporate customers get a discount on all rentals
5. **Define the Main Class VehicleRental:**
   * Methods:
     + main(String[] args): Create instances of vehicles and customers, and demonstrate the rental process, displaying the vehicles and their rental costs. Use lists to manage the rented vehicles and display them.

**UML Class Diagram**

Refer to the [PlantUML diagram](https://github.com/verisoft-ai/oop-ex/blob/master/images/oop-version2.png).

**Submission**

1. Upload your code to a public repository on GitHub.
2. Ensure your code is well-documented with comments explaining key parts.
3. Create a test case in the main method of the VehicleRental class to demonstrate the functionality.
4. Submit the GitHub repository link.

This exercise is designed to evaluate your understanding and implementation of object-oriented programming concepts. Good luck!