Vehicle Rental Application

This project is a vehicle rental system simulator. The user can rent a bike or a car and return the vehicle after a certain period, receiving a bill. The project consists of two main parts: the first part involves defining classes and functions, and the second part is the main program flow where these classes and functions are used. Let's analyze the general structure and functionality of the code.

1. Class and Function Definitions (First Part)

a. VehicleRent Class (Parent Class)

- **Purpose:** This class provides the basic functionalities for the vehicle rental system. It manages vehicle stock and handles hourly or daily rental operations.
- **Methods:**
    - displayStock(): Displays the available vehicle stock.
    - rentHourly(n): Processes hourly rentals.
    - rentDaily(n): Processes daily rentals.
    - returnVehicle(request, brand): Handles vehicle return and bill calculation.

b. CarRent and BikeRent Classes (Child Classes)

- The CarRent and BikeRent classes inherit from the VehicleRent class.
- **CarRent**: Includes an additional discount function (discount) for car rentals.
- **BikeRent**: Uses the basic rental functions and does not include any discounts.

c. Customer Class

- **Purpose:** This class holds customer-related information and manages customer requests and returns.
- **Methods:**
    - requestVehicle(brand): Requests the number of vehicles the customer wants to rent.
    - returnVehicle(brand): Returns the rental time and information when the customer returns the vehicle.

2. Main Program (Second Part)

In the main program, a menu is presented to the user where they can rent a bike or car. Let's examine the flow of the program.

a. Main Menu

- **Options:**
    - **Bike Menu:** Handles bike rental operations.
    - **Car Menu:** Handles car rental operations.
    - **Exit:** Exits the program.

b. Submenus

- **Bike Menu:**
    - Allows the user to display stock, rent on an hourly/daily basis, and return a bike.
- **Car Menu:**
    - Allows the user to rent or return a car.

c. Customer Interaction

In both menus, the user can rent vehicles, check the available stock, and return rented vehicles to receive a bill.

3. General Structure

The code is modular and functionally well-organized. The VehicleRent class provides a basic structure for rental operations, which is extended by child classes. The Customer class ensures a clear separation of concerns by managing user interactions and rentals.

4. Logic and Functionality

- The rental operations are based on two models: hourly and daily. The prices differ for bikes and cars, and daily rentals are more cost-effective on an hourly basis.
- When vehicles are returned, the bill is calculated, and discounts are applied based on the number of vehicles rented.
- User interactions are handled through terminal inputs, with basic error handling in place (e.g., ValueError and checks for negative numbers).