**Vehicle Rental Management System Documentation**

**Overview**

The **Vehicle Rental Management System** is a comprehensive application designed to manage vehicle rentals using Object-Oriented Programming (OOP) principles, including encapsulation, inheritance, polymorphism, and abstraction. The system supports different vehicle types, tracks customer transactions, and manages vehicle availability within a rental agency.

**Key Features**

* **Abstraction**: Abstract base class Vehicle defines shared behavior for all vehicle types.
* **Inheritance**: Concrete vehicle classes (Car, Motorcycle, Truck) inherit from Vehicle.
* **Encapsulation**: Private fields and public methods control access to vehicle and customer data.
* **Polymorphism**: Interface Rentable ensures uniform rental behavior across different vehicle types.
* **Composition**: Relationships between Customer, RentalTransaction, and Vehicle classes enable complex functionality.

**Class Descriptions**

**1. Vehicle (Abstract Class)**

Represents a generic vehicle.

**Fields**

* String vehicleId: Unique identifier for the vehicle (immutable).
* String model: Vehicle model.
* double baseRentalRate: Base rental rate per day.
* boolean isAvailable: Availability status.

**Methods**

* **Abstract**: calculateRentalCost(int days), isAvailableForRental().
* **Concrete**: Getters, setters, and validation logic for fields.
* **Override**: toString() for formatted output.

**2. Concrete Vehicle Classes**

Each concrete class represents a specific type of vehicle with unique rental characteristics.

**a. Car**

* Includes an additional **insurance fee** of $15.
* Implements calculateRentalCost and rent behavior.

**b. Motorcycle**

* Implements calculateRentalCost and rent behavior.

**c. Truck**

* Implements calculateRentalCost and rent behavior.

**3. Customer**

Represents a customer renting vehicles.

**Fields**

* String customerId: Unique customer ID.
* String name: Customer name.
* List<RentalTransaction> rentalHistory: Record of completed rentals.

**Methods**

* Add rental transactions to history.
* Retrieve rental history.

**4. RentalTransaction**

Tracks details of individual rental transactions.

**Fields**

* Vehicle vehicle: The rented vehicle.
* int days: Number of rental days.

**Methods**

* Retrieve transaction details.

**5. RentalAgency**

Manages the vehicle fleet and rental operations.

**Fields**

* Map<String, Vehicle> fleet: Stores vehicles by their ID.

**Methods**

* Add vehicles to the fleet.
* Retrieve vehicles by ID.
* Generate a report of all vehicles.

**Interfaces**

**Rentable**

Defines rental behavior for vehicles.

**Methods**

* void rent(Customer customer, int days)
* void returnVehicle()

**Execution Flow**

1. **Initialize the system**:
   * Create a RentalAgency.
   * Add vehicles (Car, Motorcycle, Truck) to the agency's fleet.
2. **Customer rental**:
   * Create a Customer.
   * Rent a vehicle and track the transaction.
3. **Return and report**:
   * Mark the vehicle as available upon return.
   * Generate a fleet report showing all vehicles.

**Sample Test Cases**

**1. Vehicle Rental**

* Rent a Car for 5 days.
* Verify rental cost calculation.
* Check availability status before and after rental.

**2. Rental Transaction**

* Add a transaction for a Motorcycle.
* Validate that the transaction is stored in the customer's history.

**3. Fleet Management**

* Add vehicles to the fleet.
* Generate a report showing all vehicles and their statuses.

**Additional Features**

**Loyalty Program**

Implement a loyalty system to reward frequent renters.

**Custom Exceptions**

Define exceptions for scenarios like unavailable vehicles or invalid rental durations.

**Rating System**

Allow customers to rate vehicles and their rental experience.

**Conclusion**

This Vehicle Rental Management System demonstrates robust design principles and extensibility. It is a practical solution for managing vehicle rentals efficiently and can be enhanced with additional features to meet specific requirements.