Vehicle Rental Management System

Overview

The Vehicle Rental Management System is a robust application designed to manage the operations of a vehicle rental agency. This system demonstrates key Object-Oriented Programming (OOP) principles, including encapsulation, inheritance, polymorphism, abstraction, and composition.

**Features**

* **Vehicle Management**: Manage a diverse fleet of vehicles, including cars, motorcycles, and trucks.
* **Customer Management**: Track customer information and rental history.
* **Rental Transactions**: Facilitate vehicle rentals and returns with dynamic cost calculations.
* **Availability Tracking**: Monitor vehicle availability in real-time.
* **Loyalty Program (Bonus)**: Implemented via an interface for enhanced customer engagement.

Technologies Used

* **Programming Language**: Java (JDK 11+)
* **Build Tool**: Maven
* **IDE**: IntelliJ IDEA
* **Testing Framework**: JUnit 5

Installation

1. Clone the repository:
2. git clone https://github.com/DCIT201/oop-java-emmanuelcobbinah007
3. Navigate to the project directory.
4. Build the project using Maven to resolve dependencies:
5. mvn clean install
6. Open the project in IntelliJ IDEA or your preferred Java IDE.

**Usage**

Workflow

1. **Add Vehicles**: Populate the system with vehicles (cars, motorcycles, trucks) using the provided constructors.
2. **Add Customers**: Create customer profiles to enable rentals.
3. **Rent Vehicles**: Specify customer details and rental duration to rent a vehicle.
4. **Return Vehicles**: Process the return of vehicles and calculate total charges.
5. **Show All Vehicles**: Retrieve and display all vehicles currently in the system.

**Code Structure**

Core Classes

1. **Vehicle** (Abstract Class)
   * Base class for all vehicles, defining common attributes and behaviors.
   * Abstract methods:
     + calculateRentalCost(int days)
     + isAvailableForRental()
2. **Car, Motorcycle, Truck** (Concrete Subclasses)
   * Inherit from Vehicle and implement unique rental characteristics, such as base rates and specific rental rules.
3. **Customer**
   * Encapsulates customer details and rental history.
   * Provides eligibility checks for rentals.
4. **RentalAgency**
   * Manages the fleet of vehicles and customer transactions.
   * Implements business logic for rentals, returns, and reporting.
5. **RentalTransaction**
   * Tracks details of each rental, including the vehicle, customer, rental period, and cost.

**Key Methods**

Vehicle Class

* calculateRentalCost(int days): Abstract method implemented by subclasses to calculate rental costs.
* isAvailableForRental(): Checks vehicle availability.

RentalAgency Class

* addVehicle(Vehicle vehicle): Adds a vehicle to the fleet.
* rentVehicle(int vehicleId, Customer customer, int days): Processes a rental.
* returnVehicle(int vehicleId): Marks a vehicle as returned and calculates revenue.

**OOP Principles Demonstrated**

1. **Abstraction**: Abstract base class Vehicle with common methods.
2. **Encapsulation**: Private fields and controlled access through getters and setters.
3. **Inheritance**: Car, Motorcycle, and Truck extend Vehicle.
4. **Polymorphism**: Interfaces and method overriding for flexible design.
5. **Composition**: Classes like Customer and RentalTransaction complement the system design.

**Testing**

Unit tests are implemented using JUnit 5 to validate the functionality of all classes and methods.

Running Tests

1. Open the terminal in the project directory.
2. Execute the tests using Maven:
3. mvn test
4. Review the test results to ensure all cases pass successfully.

Future Enhancements

* **Graphical User Interface**: Introduce a user-friendly GUI.
* **Database Integration**: Add persistent storage for data.
* **Online Payment Support**: Enable secure payment processing.

**Credits**

This system was developed as part of the DCIT 201 Graded Assignment to demonstrate advanced OOP principles in Java. For inquiries, contact Emmanuel Cobbinah, [ekdcobbinah@st.ug.edu.gh](mailto:ekdcobbinah@st.ug.edu.gh).