# Vehicle Rental Management System

## Overview

The **Vehicle Rental Management System** is a console-based application designed to manage vehicle rentals efficiently. It allows customers to rent and return vehicles, while administrators (admins) can manage the vehicle fleet, including adding new vehicles and displaying the fleet details. The system exemplifies key Object-Oriented Programming (OOP) principles such as **encapsulation**, **inheritance**, **polymorphism**, and **abstraction**.

## Classes

### 1. ****Vehicle****

An abstract base class representing a generic vehicle.

#### Fields:

* private String vehicleId: The unique ID of the vehicle.
* private String model: The model of the vehicle.
* private double baseRentalRate: The base rental rate per day.
* private boolean isAvailable: Indicates whether the vehicle is available for rental.

#### Constructor:

* public Vehicle(String vehicleId, String model, double baseRentalRate): Initializes the vehicle with the given ID, model, and base rental rate.

#### Methods:

* public String getVehicleId(): Returns the vehicle ID.
* public String getModel(): Returns the vehicle model.
* public double getBaseRentalRate(): Returns the base rental rate.
* public boolean isAvailable(): Returns whether the vehicle is available.
* public void setAvailable(boolean isAvailable): Sets the availability of the vehicle.
* public abstract double calculateRentalCost(int days): Calculates the rental cost for the given number of days.
* public abstract boolean isAvailableForRental(): Checks if the vehicle is available for rental.

### 2. ****Car, Motorcycle, Truck****

Concrete classes that extend the **Vehicle** class and implement the **Rentable** interface.

#### Constructor:

* public Car(String vehicleId, String model, double baseRentalRate): Initializes a car with the given ID, model, and base rental rate.
* public Motorcycle(String vehicleId, String model, double baseRentalRate): Initializes a motorcycle with the given ID, model, and base rental rate.
* public Truck(String vehicleId, String model, double baseRentalRate): Initializes a truck with the given ID, model, and base rental rate.

#### Methods:

* public double calculateRentalCost(int days): Calculates the rental cost for the given number of days.
* public boolean isAvailableForRental(): Checks if the vehicle is available for rental.
* public void rent(Customer customer, int days): Rents the vehicle to the customer for the given number of days.
* public void returnVehicle(): Returns the vehicle.

### 3. ****Rentable****

An interface that defines methods for renting and returning vehicles.

#### Methods:

* void rent(Customer customer, int days): Rents the vehicle to the customer for the given number of days.
* void returnVehicle(): Returns the vehicle.

### 4. ****Customer****

A class representing a customer.

#### Fields:

* private final String name: The name of the customer.

#### Constructor:

* public Customer(String name): Initializes the customer with the given name.

#### Methods:

* public String getName(): Returns the name of the customer.

### 5. ****Admin****

A class representing the admin responsible for managing the vehicle fleet.

#### Fields:

* private List<Vehicle> vehicleFleet: A list of vehicles in the fleet.
* private String password: The admin password.

#### Constructor:

* public Admin(): Initializes the admin with a default password and prepopulates the fleet with some vehicles.

#### Methods:

* private void prepopulateFleet(): Prepopulates the fleet with some vehicles.
* public boolean verifyPassword(String inputPassword): Verifies the admin password.
* public void addVehicle(Vehicle vehicle): Adds a vehicle to the fleet.
* public void displayFleet(): Displays all vehicles in the fleet.
* public void displayAvailableVehicles(): Displays all available vehicles for rent.
* public Vehicle getVehicleById(String vehicleId): Returns the vehicle with the given ID.

### 6. ****Main****

The main class that contains the entry point of the application.

#### Method:

* public static void main(String[] args): The main method that runs the application.

## Usage

### Running the Application

1. Execute the **Main** class to start the application.
2. The main menu provides the following options:
   * **Rent a Vehicle**: Displays all available vehicles and allows the user to rent a vehicle by entering the vehicle ID, customer name, and rental days.
   * **Return a Vehicle**: Allows the user to return a rented vehicle by entering the vehicle ID.
   * **Add a Vehicle (Admin)**: Admins can add a new vehicle to the fleet by entering the vehicle type, ID, model, and base rental rate. Admin password verification is required.
   * **Display Vehicle Fleet (Admin)**: Displays all vehicles in the fleet. Admin password verification is required.
   * **Exit**: Exits the application.

## Example Workflow

1. **Renting a Vehicle**:
   * A customer selects a vehicle, provides their name, and specifies the number of rental days.
   * The system calculates the rental cost and marks the vehicle as unavailable.
2. **Returning a Vehicle**:
   * The customer enters the vehicle ID, and the system marks the vehicle as available again.
3. **Admin Functionalities**:
   * Admins can manage the fleet by adding new vehicles and viewing all vehicles, with or without filtering by availability.

## Conclusion

The **Vehicle Rental Management System** provides a complete solution for managing vehicle rentals, highlighting key OOP concepts. With its intuitive interface, it simplifies the rental process for customers while offering robust management features for admins.