@Override

public double calculateRentalCost(int days) {

double cost = days \* getBaseRentalRate();

if (hasGPS) cost += 15.0 \* days; // Additional GPS cost

return cost;

}

@Override

public boolean isAvailableForRental() {

return isAvailable();

}

@Override

public void rent(Customer customer, int days) {

if (!isAvailableForRental()) {

throw new IllegalStateException("Car is not available for rental.");

}

setAvailable(false);

System.out.println("Car rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

setAvailable(true);

System.out.println("Car returned.");

}

}

Motor cycle class

package com.rental.vehicles;

import com.rental.interfaces.Rentable;

public class Motorcycle extends Vehicle implements Rentable {

private final boolean requiresHelmet;

public Motorcycle(String vehicleId, String model, double baseRentalRate, boolean requiresHelmet) {

super(vehicleId, model, baseRentalRate);

this.requiresHelmet = requiresHelmet;

}

@Override

public double calculateRentalCost(int days) {

return days \* getBaseRentalRate() + (requiresHelmet ? 10.0 \* days : 0);

}

@Override

public boolean isAvailableForRental() {

return isAvailable();

}

@Override

public void rent(Customer customer, int days) {

if (!isAvailableForRental()) {

throw new IllegalStateException("Motorcycle is not available for rental.");

}

setAvailable(false);

System.out.println("Motorcycle rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

setAvailable(true);

System.out.println("Motorcycle returned.");

}

}

Truck class

package com.rental.vehicles;

import com.rental.interfaces.Rentable;

public class Truck extends Vehicle implements Rentable {

private final double loadCapacity;

public Truck(String vehicleId, String model, double baseRentalRate, double loadCapacity) {

super(vehicleId, model, baseRentalRate);

if (loadCapacity <= 0) {throw new IllegalArgumentException("Load capacity must be positive.");

}

this.loadCapacity = loadCapacity;

}

@Override

public double calculateRentalCost(int days) {

return days \* getBaseRentalRate() + (loadCapacity > 1000 ? 50.0 \* days : 20.0 \* days);

}

@Override

public boolean isAvailableForRental() {

return isAvailable();

}

@Override

public void rent(Customer customer, int days) {

if (!isAvailableForRental()) {

throw new IllegalStateException("Truck is not available for rental.");

}

setAvailable(false);

System.out.println("Truck rented to " + customer.getName() + " for " + days + " days.");

}

@Override

public void returnVehicle() {

setAvailable(true);

System.out.println("Truck returned.");

}

}

Rental interface

package com.rental.interfaces;

import com.rental.customers.Customer;

public interface Rentable {

void rent(Customer customer, int days);

void returnVehicle();

}