import java.time.LocalDate;

import java.time.temporal.ChronoUnit;

import java.util.ArrayList;

import java.util.List;

public class CarRentalSystem {

public static class Car {

private String id;

private String model;

private String brand;

private boolean isAvailable;

private double dailyRate;

public Car(String id, String model, String brand, double dailyRate) {

this.id = id;

this.model = model;

this.brand = brand;

this.dailyRate = dailyRate;

this.isAvailable = true;

}

public String getId() { return id; }

public String getModel() { return model; }

public String getBrand() { return brand; }

public boolean isAvailable() { return isAvailable; }

public double getDailyRate() { return dailyRate; }

public void rent() {

if (isAvailable) {

isAvailable = false;

System.out.println(model + " has been rented successfully!");

} else {

System.out.println(model + " is already rented.");

}

}

public void returnCar() {

isAvailable = true;

System.out.println(model + " is now available for rent again.");

}

@Override

public String toString() {

return brand + " " + model + " [ID: " + id + "] - ksh" + dailyRate + "/day, Available: " + (isAvailable ? "Yes" : "No");

}

}

public static class Customer {

private String id;

private String name;

private String email;

private List<Rental> rentalHistory;

public Customer(String id, String name, String email) {

this.id = id;

this.name = name;

this.email = email;

this.rentalHistory = new ArrayList<>();

}

public String getId() { return id; }

public String getName() { return name; }

public String getEmail() { return email; }

public List<Rental> getRentalHistory() { return rentalHistory; }

public void addRental(Rental rental) {

rentalHistory.add(rental);

System.out.println("Rental added to " + name + "'s history.");

}

@Override

public String toString() {

return name + " [ID: " + id + "]";

}

}

public static class Rental {

private String id;

private Car car;

private Customer customer;

private LocalDate startDate;

private LocalDate endDate;

private double totalCost;

public Rental(String id, Car car, Customer customer, LocalDate startDate, LocalDate endDate) {

this.id = id;

this.car = car;

this.customer = customer;

this.startDate = startDate;

this.endDate = endDate;

this.totalCost = calculateTotalCost();

}

private double calculateTotalCost() {

long days = ChronoUnit.DAYS.between(startDate, endDate);

return days \* car.getDailyRate();

}

public Car getCar() { return car; }

public Customer getCustomer() { return customer; }

public LocalDate getStartDate() { return startDate; }

public LocalDate getEndDate() { return endDate; }

public double getTotalCost() { return totalCost; }

@Override

public String toString() {

return "Rental ID: " + id + ", Car: " + car.getModel() + ", Rented by: " + customer.getName() + ", Total Cost: ksh" + totalCost;

}

}

public static class RentalAgency {

private List<Car> carFleet;

private List<Customer> customerList;

private List<Rental> rentalList;

public RentalAgency() {

this.carFleet = new ArrayList<>();

this.customerList = new ArrayList<>();

this.rentalList = new ArrayList<>();

}

public void addCar(Car car) {

carFleet.add(car);

System.out.println(car.getModel() + " has been added to the fleet.");

}

public void addCustomer(Customer customer) {

customerList.add(customer);

System.out.println(customer.getName() + " has been registered.");

}

public void viewAvailableCars() {

System.out.println("Available Cars:");

for (Car car : carFleet) {

if (car.isAvailable()) {

System.out.println(car);

}

}

}

public Rental rentCar(String carId, String customerId, LocalDate startDate, LocalDate endDate) {

Car car = findCarById(carId);

Customer customer = findCustomerById(customerId);

if (car != null && customer != null && car.isAvailable()) {

Rental rental = new Rental("R" + (rentalList.size() + 1), car, customer, startDate, endDate);

rentalList.add(rental);

customer.addRental(rental);

car.rent();

return rental;

} else {

System.out.println("Unable to rent the car. Either it's unavailable or invalid customer/car ID.");

return null;

}

}

public void returnCar(Rental rental) {

rental.getCar().returnCar();

System.out.println("Rental completed: " + rental);

}

private Car findCarById(String carId) {

for (Car car : carFleet) {

if (car.getId().equals(carId)) {

return car;

}

}

System.out.println("Car with ID " + carId + " not found.");

return null;

}

private Customer findCustomerById(String customerId) {

for (Customer customer : customerList) {

if (customer.getId().equals(customerId)) {

return customer;

}

}

System.out.println("Customer with ID " + customerId + " not found.");

return null;

}

}

public static void main(String[] args) {

// Creating the rental agency

RentalAgency agency = new RentalAgency();

agency.addCar(new Car("1", "Camry", "Toyota", 150.0));

agency.addCar(new Car("2", "NV350", "Nissan Van", 130.0));

agency.addCustomer(new Customer("1", "Joshua", "joshua@gmail.com"));

agency.viewAvailableCars();

Rental rental = agency.rentCar("1", "1", LocalDate.now(), LocalDate.now().plusDays(5));

if (rental != null) {

agency.returnCar(rental);

}

agency.viewAvailableCars();

}

}