实验报告

实验目的：

设计一个 Car 类，用于模拟租车公司对车辆的基本管理，并编写 TestCar 类，用于测试 Car 类的属性和方法。

类图（手绘或用工具绘制，展示类名、属性、方法及访问修饰符）：

核心代码（构造方法、关键业务方法、测试类主要片段）：

方法一：

```
public Car(String licensePlate,String brand,String model,double DailyRent){
    this.licensePlate=licensePlate;
    this.brand=brand;
    this.model=model;
    this.DailyRent=DailyRent;
    this.isRented=false;
    totalcars++;
}
```

方法二：

```
public Car(String licensePlate,String brand,String model){
    this(licensePlate,brand,model, DailyRent: 300.0);
```

方法三：

```
public Car(){  1 个用法
    this( licensePlate: "FV 6666", brand: "Toyota", model: "Nissan", DailyRent: 500);
```

业务方法：

```
//构造静态getTotalCars()方法
public static int getTotalcars(){  1 个用法
    return totalcars;
}
//建立getter  licensePlate brand model dailyRent isRented()method
public String getLicensePlate(){  1 个用法
    return licensePlate;
}
public String getBrand() { return brand; }
public String getModel() { return model; }

public double getDailyRent() { return DailyRent; }
public boolean isRented() { return isRented; }

//建立setter
public void setBrand(String brand) { this.brand=brand; }

public void setModel(String model) { this.model = model; }
public void setDailyRent(double DailyRent){  1 个用法
    if (DailyRent>0){
        this.DailyRent=DailyRent;
    }
    else{
        System.out.println("非法租金,修改失败");
    }
}
public void rentCar(){  2 个用法
    if(isRented==true){
        System.out.println("车辆已租出，无法再次租用");
    }
    else{
        isRented=true;
    }
}
```

```
public void returnCar(){  2 个用法
    if(isRented==false){
        System.out.println("车辆未被租用，无需归还");
    }
    else{
        isRented=false;
    }
}
public double calculateRent(int days){  1 个用法
    double money=days*DailyRent;
    return money;
}
```

测试类主要片段：

```java
public class TestCar {
    //编写静态方法displayInfo()打印车辆信息
    public static void displayInfo(Car car){     //static是声明静态  3个用法
        System.out.println("车辆信息:车牌号"+car.getLicensePlate()+" 品牌"+car.getBrand()+" 型号"+car.getModel()+" 日租金(元/天)"+car.getDailyRent());
    }
    public static void main(String[] args){
        Car mine1=new Car( licensePlate: "FU 0001", brand: "Crown", model: "Wilsaf", DailyRent: 200.0);
        Car mine2=new Car( licensePlate: "FU 8888", brand: "Toyota", model: "Alpha");
        Car mine3=new Car();
        displayInfo(mine1);
        displayInfo(mine2);
        displayInfo(mine3);
        //调用两次rentCar()并查看提示
        System.out.println("--调用两次rentCar()并查看提示--");
        mine1.rentCar();
        System.out.println(mine1.isRented());
        mine1.rentCar();
        //调用两次returnCar()并查看提示
        System.out.println("--调用两次returnCar()并查看提示--");
        mine1.returnCar();
        System.out.println(mine1.isRented());
        mine1.returnCar();
        //计算5天的租金
        System.out.println("--计算5天的租金--");
        System.out.println("您的总租金为:"+mine1.calculateRent( days: 5)+"元");
        //用setter修改租金为-100查看提示
        System.out.println("--用setter修改租金为-100查看提示--");
        System.out.println("当前日租金为:"+mine1.getDailyRent()+"元");
        mine1.setDailyRent(-100);
        //查看日租金是否保持原值
        System.out.println("--查看日租金是否保持原值--");
        System.out.println("当前日租金为:"+mine1.getDailyRent()+"元");
        //输出车辆总数
        System.out.println("--输出车辆总数--");
        System.out.println("车辆总数为:"+Car.getTotalcars());
        }
```

运行结果截图：

```
车辆信息:车牌号FU 0001 品牌Crown 型号Wilsaf 日租金(元/天)200.0
车辆信息:车牌号FU 8888 品牌Toyota 型号Alpha 日租金(元/天)300.0
车辆信息:车牌号FV 6666 品牌Toyota 型号Nissan 日租金(元/天)500.0
--调用两次rentCar()并查看提示--
true
车辆已租出，无法再次租用
--调用两次returnCar()并查看提示--
false
车辆未被租用，无需归还
--计算5天的租金--
您的总租金为:1000.0元
--用setter修改租金为-100查看提示--
当前日租金为:200.0元
非法租金,修改失败
--查看日租金是否保持原值--
当前日租金为:200.0元
--输出车辆总数--
车辆总数为:3
```

实验总结（封装带来的好处、遇到的问题及解决方法）：

好处：以保证数据安全，防止内部数据被外界随意篡改，并且可

以隐藏方法的内部细节，使用者只需要调用方法，无需过多考虑，并且可以多次调用构造的方法，较方便。

遇到的问题：调用了非静态方法，报错

解决方法：把报错截图发给 ai，它让我把定义方法的那一行加一个 static，把它变成静态方法，这样就可以在测试类主程序中调用了。