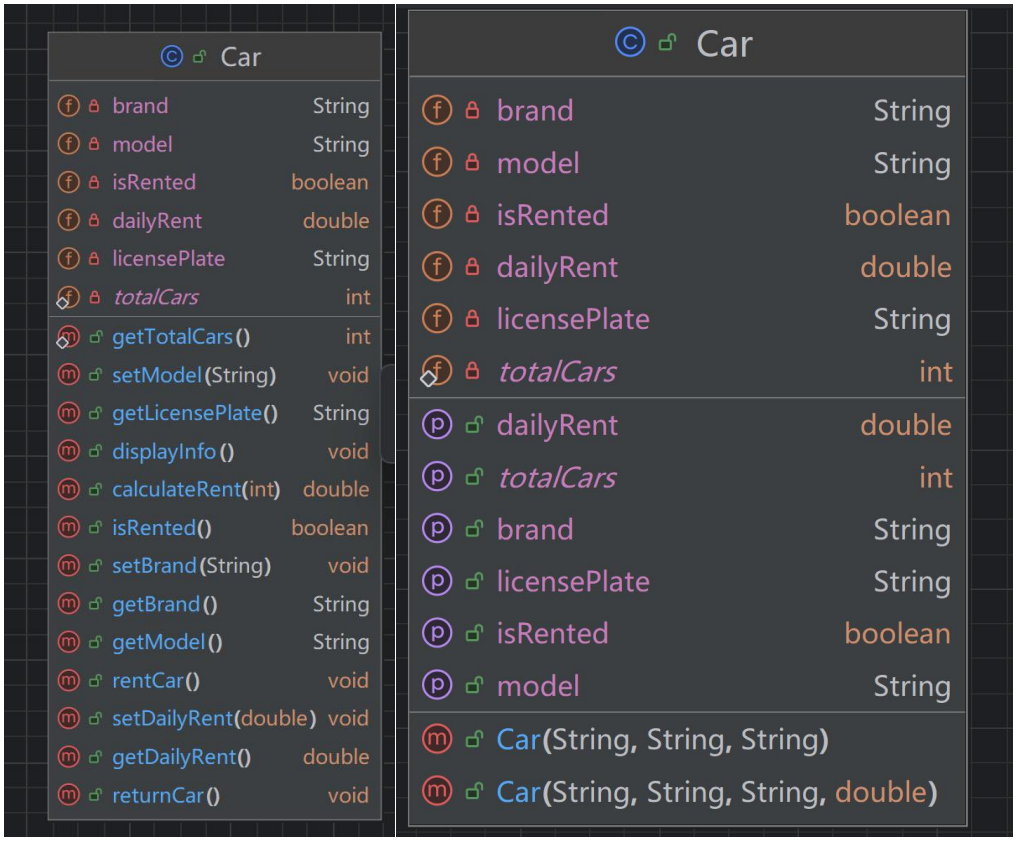
类图

以下展示了 Car 类的属性和方法，以及修饰符，锁的开关表示公共和私有。



核心代码

构造方法

以下代码展示了构造方法的重载，在第二个构造方法中使用 this() 调用其他构造方法，在第一个全参构造方法中调用 setDailyRent(dailyRent) 来检验数据有效性。

```
//构造方法
public Car(String licensePlate,String brand,String model,double dailyRent){
    this.licensePlate = licensePlate;
    this.brand = brand;
    this.model = brand;
    setDailyRent(dailyRent);
    this.isRented = false;
    totalCars++;
}
public Car(String licensePlate,String brand,String model){  1 个用法
    this(licensePlate,brand,model, dailyRent: 300.0);
}
```

```java
public void setDailyRent(double dailyRent){  2 个用法
    if (dailyRent>0){
        this.dailyRent=dailyRent;
    }else{
        System.out.println("日租金必须大于0，设置失败，保持原值。");
    }
}
```

关键业务方法

编写业务方法，用 if,else 语句，用 boolean 类型的属性判断是否租用，来达到租车和还车的业务。

```java
//业务方法
public void rentCar() {  2 个用法
    if (this.isRented) {
        System.out.println("车辆已租出，无法再次租用。");
    } else {
        this.isRented = true;
        System.out.println("车辆" + this.licensePlate + "租用成功。");
    }
}
public void returnCar() {  2 个用法
    if (!this.isRented) {
        System.out.println("车辆未被租用，无需归还。");
    } else {
        this.isRented = false;
        System.out.println("车辆" + this.licensePlate + "归还成功。");
    }
}
```

```java
public double calculateRent(int days) {  1 个用法
    return this.dailyRent * days;
}
```

测试类主要片段

创建了三个对象，分别调用不同构造方法。

```java
import com.rental.Car;

public class TestCar {
    public static void main(String[] args) {
        //创建至少3个Car对象
        Car car1 = new Car( licensePlate: "京A88888",  brand: "奥迪",  model: "A6L",  dailyRent: 500.0);
        Car car2 = new Car( licensePlate: "沪B66666",  brand: "宝马",  model: "5系");
        Car car3 = new Car( licensePlate: "粤C12345",  brand: "奔驰",  model: "E300L",  dailyRent: 600.0);
```

调用 displayInfo()展示所有车辆的信息。

```
//输出所有车辆信息
System.out.println("所有车辆信息如下");
car1.displayInfo();
car2.displayInfo();
car3.displayInfo();
System.out.println();
```

调用业务方法。

```
//对其中一辆车依次调用业务方法
System.out.println("对其中一辆车依次调用业务方法如下");
car1.rentCar();      // 第一次租用
car1.rentCar();      // 再次租用，查看提示
car1.returnCar();    // 第一次归还
car1.returnCar();    // 再次归还，查看提示
System.out.println();
```

调用计算租金的方法。

```
//调用 calculateRent(5) 计算费用
System.out.println("调用 calculateRent(5) 计算某辆车租用 5 天的费用");
double rent5Days = car3.calculateRent( days: 5);
System.out.println("车辆 " + car3.getLicensePlate() + " 租用5天的费用为: " + rent5Days + " 元");
System.out.println();
```

测试日租金设置的数据检验。

```
//尝试通过 setter 修改日租金为非法值
System.out.println("测试日租金合法性校验");
System.out.println("修改前 car2 的日租金: " + car2.getDailyRent());
car2.setDailyRent(-100); // 设置为非法值
System.out.println("修改后 car2 的日租金: " + car2.getDailyRent());
System.out.println();
```

输出总车辆数。

```
//输出总车辆数
System.out.println("本次运行共创建了 " + Car.getTotalCars() + " 辆车。");
```

运行结果截图

正常流程:

在终端输入指令 Javac TestCar.java 进行编译，输入 Java TestCar 后，得到正常运行结果。

```
D:\java\w3\src>java TestCar
所有车辆信息如下
车牌：京A88888，品牌：奥迪，型号：奥迪，日租金：500.0元/天，状态：可租
车牌：沪B66666，品牌：宝马，型号：宝马，日租金：300.0元/天，状态：可租
车牌：粤C12345，品牌：奔驰，型号：奔驰，日租金：600.0元/天，状态：可租

对其中一辆车依次调用业务方法如下
车辆京A88888租用成功。
车辆已租出，无法再次租用。
车辆京A88888归还成功。
车辆未被租用，无需归还。

调用 calculateRent(5) 计算某辆车租用 5 天的费用
车辆 粤C12345 租用5天的费用为：3000.0 元

测试日租金合法性校验
修改前 car2 的日租金：300.0
日租金必须大于0，设置失败，保持原值。
修改后 car2 的日租金：300.0

本次运行共创建了 3 辆车。
```

非法操作提示

```
:\java\w3\src>javac TestCar.java
estCar.java:1: 错误：程序包com.rental.Car与带有相同名称的类冲突
ackage com.rental.Car;

estCar.java:6: 错误：找不到符号
        Car car1 = new Car("京A88888", "奥迪", "A6L", 500.0);
        ^
  符号：   类 Car
  位置：类 TestCar
estCar.java:6: 错误：找不到符号
        Car car1 = new Car("京A88888", "奥迪", "A6L", 500.0);
                       ^
  符号：   类 Car
  位置：类 TestCar
estCar.java:7: 错误：找不到符号
        Car car2 = new Car("沪B66666", "宝马", "5系");
        ^
  符号：   类 Car
  位置：类 TestCar
estCar.java:7: 错误：找不到符号
        Car car2 = new Car("沪B66666", "宝马", "5系");
                       ^
  符号：   类 Car
  位置：类 TestCar
estCar.java:8: 错误：找不到符号
        Car car3 = new Car("粤C12345", "奔驰", "E300L", 600.0);
        ^
  符号：   类 Car
  位置：类 TestCar
```

```
):\java\w3\src>javac TestCar.java
TestCar.java:1: 错误: 无法访问Car
import com.rental.Car;
                  ^
  错误的源文件: .\com\rental\Car.java
    文件不包含类com.rental.Car
    请删除该文件或确保该文件位于正确的源路径子目录中。
```

```
D:\java\w3\src>javac TestCar.java
TestCar.java:1: 错误: 程序包com不存在
import com.rental;
           ^
TestCar.java:6: 错误: 找不到符号
        Car car1 = new Car("京A88888", "奥迪", "A6L", 500.0);
        ^
  符号:   类 Car
  位置: 类 TestCar
TestCar.java:6: 错误: 找不到符号
        Car car1 = new Car("京A88888", "奥迪", "A6L", 500.0);
                       ^
  符号:   类 Car
  位置: 类 TestCar
TestCar.java:7: 错误: 找不到符号
```

原因在于 Car 中的第一条语句(package com.rental;)使用中文全角分号 ； 的 Unicode 编码。在 Java 中，必须使用 英文半角分号 “;”。

```
D:\java\w3\src>javac TestCar.java
.\com\rental\Car.java:1: 错误: 非法字符: '\uff1b'
package com.rental;
                  ^
TestCar.java:33: 错误: 找不到符号
        System.out.println("修改前 car2 的日租金: " + car2.getDaily
Rent());
                                                    ^
  符号:   方法 getDailyRent()
  位置: 类型为Car的变量 car2
TestCar.java:34: 错误: 找不到符号
        car2.setDailyRent(-100); // 设置为非法值
            ^
  符号:   方法 setDailyRent(int)
  位置: 类型为Car的变量 car2
TestCar.java:35: 错误: 找不到符号
        System.out.println("修改后 car2 的日租金: " + car2.getDaily
Rent());
                                                    ^
  符号:   方法 getDailyRent()
  位置: 类型为Car的变量 car2
.\com\rental\Car.java:80: 错误: 找不到符号
                        ", 型号: " + this.model + ", 日租金: " +
 this.dailyRent +

^
  符号: 变量 dailyRent
5 个错误
```

```
D:\java\w3\src>javac TestCar.java
TestCar.java:33: 错误: 找不到符号
        System.out.println("修改前 car2 的日租金: " + car2.getDaily
Rent());
                                                    ^
  符号:   方法 getDailyRent()
  位置: 类型为Car的变量 car2
TestCar.java:34: 错误: 找不到符号
        car2.setDailyRent(-100); // 设置为非法值
            ^
  符号:   方法 setDailyRent(int)
  位置: 类型为Car的变量 car2
TestCar.java:35: 错误: 找不到符号
        System.out.println("修改后 car2 的日租金: " + car2.getDaily
Rent());
                                                    ^
  符号:   方法 getDailyRent()
  位置: 类型为Car的变量 car2
.\com\rental\Car.java:80: 错误: 找不到符号
                          ", 型号: " + this.model + ", 日租金: " +
 this.dailyRent +

^
  符号: 变量 dailyRent
4 个错误
```

变量名拼写错误：你可能声明了一个名字很像但不完全一样的变量，比如 `private double rent;` 或 `private int dailyRent;` (类型不同也可能导致问题)，然后在 getter/setter 或 `toString` 方法里却用了 `dailyRent` 。

出现命名错误，改正后错误消失。

实验总结

通过本次实验，我不仅掌握了 Java 类与对象的基本语法，更深刻体会到了封装的真正含义。

以前我认为封装只是简单的“私有变量 + 公有 Getter/Setter”，但通过本次对 isRented 属性的设计反思，我意识到：封装的本质是保护对象的内部状态不被非法篡改，并将业务逻辑内聚在对象内部。直接暴露 Setter 虽然方便，却牺牲了数据的安全性和业务的一致性。

此外，在解决编译错误的过程中，我学会了更耐心地阅读报错信息，并养成了“先保存、再清理、后编译”的良好开发习惯。利用 IDEA 生成类图的功能，也让我明白了可视化工具在理解复杂代码结构中的重要性。

本次实验为后续学习继承、多态以及更复杂的系统设计打下了坚实的基础。