

# Python程序设计报告

设计题目：汽车管家

学生姓名：张渊

专 业：计算机科学与技术

班 级：22-0

学 号：2022210000

指导教师：马学森

完成日期：2023-11-11

# （一）需求和规格说明

**问题描述：**设计实现一个汽车租赁管理系统。

**要求**：

① 车辆基本信息管理（车辆的添加、删除、修改）；

② 租车管理功能（租车、计费等）；

③ 统计功能（统计出租率、赚取金额）。

**编程任务：**

1.添加新的车辆

2.删除已有车辆

3.修改车辆信息

4.租用车辆

5.归还车辆

6.展示现有的车辆信息

7.统计目前的出租率和赚取金额

0.退出程序

# （二）设计

## 1．设计思想

车辆信息通过定义一个Car类来封装，其中包含车牌号、车品牌、租车价格、是否出租、租车时间等基本属性。系统管理功能通过定义一个RentCarManager类来实现，其中包含添加车辆、删除车辆、修改车辆信息、租车、还车、展示车辆信息、统计出租率和赚取金额等方法。  
 程序使用了菜单式交互方式，让用户选择要执行的具体操作。  
 在执行每个具体操作之前，程序会自动加载车辆信息，执行完毕后再保存车辆信息到文件，以保证汽车租赁系统的数据一致性。  
 统计功能中，出租率通过已出租车辆数除以总车辆数计算得出，出租费用总额则将所有已出租车辆的租车费用相加所得。  
 程序通过读取和保存车辆信息的文本文件来实现持久化数据存储。

## 2. 设计表示

1. 结点结构：

CarManager

self.car\_list #车辆列表

load\_cars\_from\_file(self) #从文件读取车辆信息

save\_cars\_to\_file(self)#将车辆信息保存到文件

add\_car(self)#添加车辆

remove\_car(self)#删除车辆

modify\_car(self)#修改车辆信息

rent\_car(self)#租车

return\_car(self)#归还车辆

show\_cars(self)#展示所有车辆信息

show\_stats(self)#展示出租率和赚取金额

Car

self.plate #车牌号

self.brand #品牌

self.price #租车价格

self.is\_rented #是否出租

self.time #租车时间

1. 类：（方法都定义在了类中）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类名** | **成员类别** | **类型** | **成员名** | **描述** |
| Car | 属性 | String | plate | 车牌号 |
| String | brand | 车品牌 |
| float | price | 租车价格 |
| bool | Is\_rented | 是否出租 |
| int | time | 租车时间 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类名** | **成员类别** | **类型** | **成员名** | **描述** |
| CarManager | 属性 | Car | self.car\_list | 车辆列表 |
| 方法 | void | load\_cars\_from\_file(self) | 从文件读取车辆信息 |
| void | save\_cars\_to\_file(self) | 将车辆信息保存到文件 |
| void | add\_car(self) | 添加车辆 |
| void | remove\_car(self) | 删除车辆 |
| void | modify\_car(self) | 修改车辆信息 |
| void | rent\_car(self) | 租车 |
| void | return\_car(self) | 归还车辆 |
| void | show\_cars(self) | 展示所有车辆信息 |
| void | show\_stats(self) | 展示出租率、赚取金额 |

## 3. 核心算法

|  |  |
| --- | --- |
| load\_cars\_from\_file(self) | 从文件读取车辆信息 |
| save\_cars\_to\_file(self) | 将车辆信息保存到文件 |

def load\_cars\_from\_file(self): *# 读取文件中的信息* try:  
 with open("cars.txt", "r") as ct:  
 for line in ct:  
 plate, brand, price = line.split()  
 self.car\_list.append(Car(plate, brand, float(price)))  
 except FileNotFoundError: *# 文件不存在时，将创建一个新文件。* print("文件不存在，将创建一个新文件。")  
 *# 创建一个新文件并继续* with open("cars.txt", "w") as ct:  
 pass *# 这里先不做任何操作，只是创建一个空文件*def save\_cars\_to\_file(self): *# 保存信息到文件中* with open("cars.txt", "w") as ct:  
 for car in self.car\_list:  
 *# 将车牌号 车的品牌 车的每小时租金存入文件录入系统* ct.write(f"{car.plate} {car.brand} {car.price}\n")

# （三）用户手册

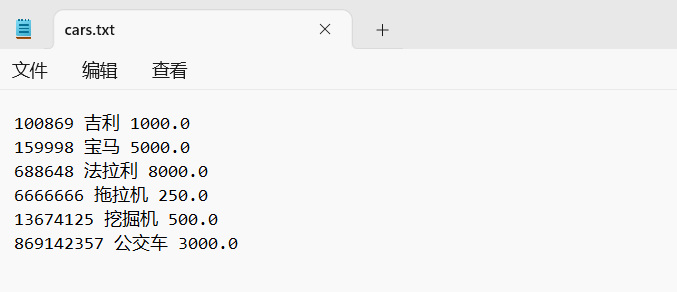
程序运行时，会提示用户输入想要执行的操作

1. 添加新的车辆 2.删除已有车辆3.修改车辆信息 4.租用车辆 5.归还车辆 6.展示现有的车辆信息 7.统计目前的出租率和赚取金额0.退出程序

在输入的过程中按照程序对应的提示执行即可

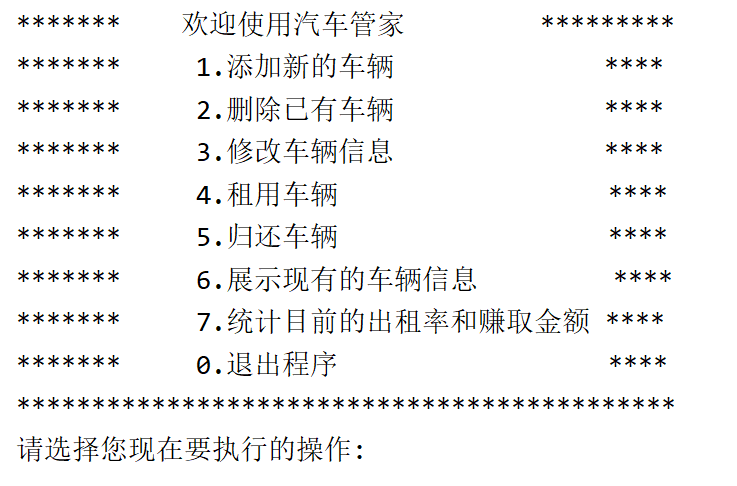
# （四）调试及测试

## 【1】 测试数据：(添加的车辆信息)



## 【2】 测试结果：

**主菜单**

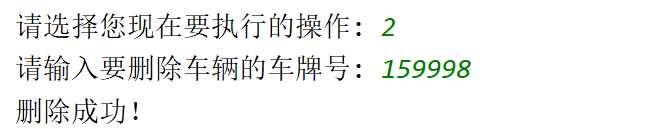


首先执行了6次1操作 一共加入了6辆汽车

1. 展示现有的车辆信息



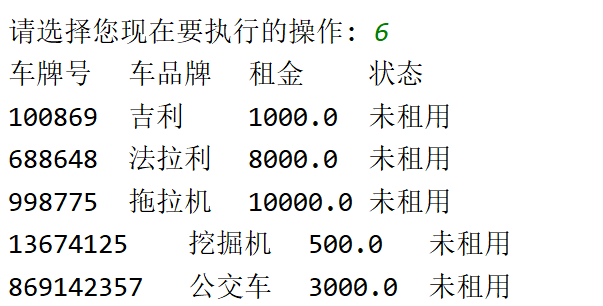
2.删除已有车辆



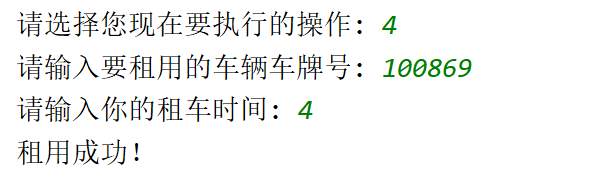


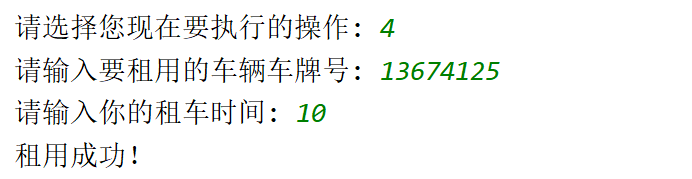
3.修改车辆信息





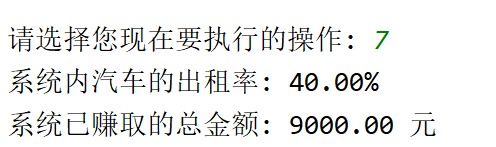
4.租用车辆



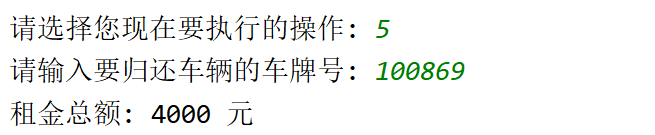




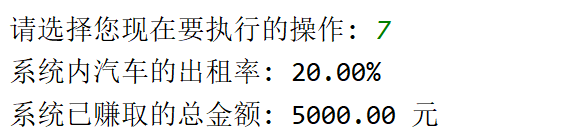
7.统计目前的出租率和赚取金额

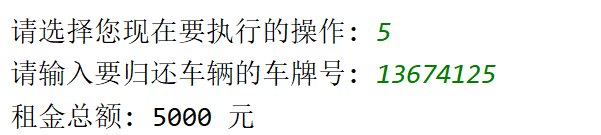


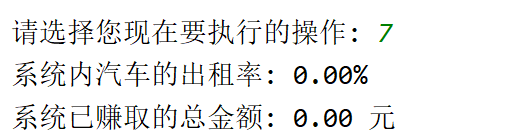
5.归还车辆

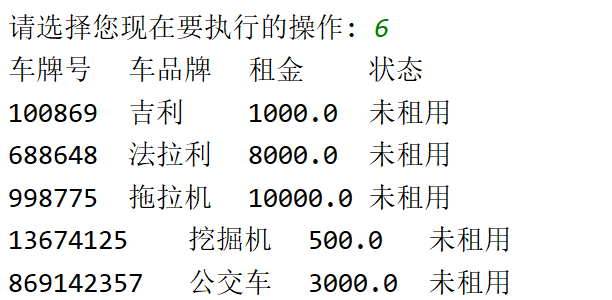




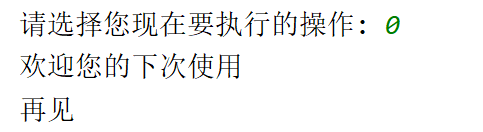








0.退出程序



## 【3】 进一步改进：

1. 为了解决计算收费的问题，我将car类里的计算时间的属性改为了局部变量，添加在对应的计算函数中，计算更加简便，便于程序的运行。

（2）为了解决出现错误无法回应的问题，我在对应的程序中添加了适当的判断，提高了代码的健壮性，使得出现错误可以作出相应的回应，很大程度上避免了程序的崩溃。

# (五) 感想

在完成这个大作业的过程中，我深深地感受到了软件开发的复杂性和挑战性。在设计和实现一个简单的汽车租赁管理系统时，需要考虑很多方面，比如如何设计合理的数据结构、如何实现各种复杂的业务逻辑、如何优化程序性能等等。

在我的设计中，租车和换车操作都需要对车辆的状态进行更新，这会导致频繁的数据操作和文件读写操作，从而降低程序的性能。为了解决这个问题，我通过数据操作以减少数据和文件系统的访问次数，从而提高程序的响应速度和性能表现。

我在统计功能的实现上也遇到了一些问题。在我的设计中，我选择了使用一些简单的算法和数据结构来实现租赁率和出租费用的统计功能，但是相比较复杂的实际情况，统计算法和数据结构显然还存在很大的改进空间。因此，在实际应用中，需要根据具体的业务需求来选择更加高效和精准的统计算法和数据结构。在编写程序的过程中，我多次因为没有按照Python的格式要求进行规范代码，导致出现了很多次bug。这些错误耗费了我很多时间，而且通常难以发现。让我深刻地感受到，学习一门新的语言，一定要按照语言的要求来，不能想当然地写代码。

总体而言，这个课程设计帮助我深入了解了软件开发的复杂性和挑战性，更加深入地理解了数据结构和算法的重要性，同时也让我在实践中不断提升自己的编程技能和解决问题的能力。我相信，在今后的实际工作中，这些经验和教训一定会对我的工作有所帮助。

**附录:**(如果直接粘贴，可能因为排版原因无法运行，如需运行需要运行py文件)

import os  
import os  
  
  
class Car: *# 汽车类 包含汽车的基本信息* def \_\_init\_\_(self, plate, brand, price):  
 self.plate = plate *# 车牌号* self.brand = brand *# 品牌* self.price = price *# 每个小时的租金* self.is\_rented = False *# 是否已经出租* self.time = 0 *# 使用时间*class CarManager:  
 def \_\_init\_\_(self):  
 self.car\_list = [] *# 存储汽车信息的列表，存储着不同Car的对象* def load\_cars\_from\_file(self): *# 读取文件中的信息* try:  
 with open("cars.txt", "r") as ct:  
 for line in ct:  
 plate, brand, price = line.split()  
 self.car\_list.append(Car(plate, brand, float(price)))  
 except FileNotFoundError: *# 文件不存在时，将创建一个新文件。* print("文件不存在，将创建一个新文件。")  
 *# 创建一个新文件并继续* with open("cars.txt", "w") as ct:  
 pass *# 这里先不做任何操作，只是创建一个空文件* def save\_cars\_to\_file(self): *# 保存信息到文件中* with open("cars.txt", "w") as ct:  
 for car in self.car\_list:  
 *# 将车牌号 车的品牌 车的每小时租金存入文件录入系统* ct.write(f"{car.plate} {car.brand} {car.price}\n")  
  
 def add\_car(self): *# 添加新的汽车* plate = input("请输入车辆的车牌号: ")  
 brand = input("请输入车辆品牌: ")  
 price = float(input("请输入租车价格: "))  
 self.car\_list.append(Car(plate, brand, price))  
 self.save\_cars\_to\_file()  
  
 def remove\_car(self): *# 依据车牌号删除汽车* plate = input("请输入要删除车辆的车牌号: ")  
 for car in self.car\_list:  
 if car.plate == plate:  
 self.car\_list.remove(car)  
 self.save\_cars\_to\_file()  
 print("删除成功！")  
 return  
 print("未找到该车辆！")  
  
 def modify\_car(self): *# 依据车牌号修改汽车* plate = input("请输入要修改车辆信息的车牌号: ")  
 for car in self.car\_list:  
 if car.plate == plate:  
 car.plate, car.brand, car.price = input("请输入新的车辆信息(车牌号、车品牌、租车价格，用空格分隔): ").split()  
 car.price = float(car.price)  
 self.save\_cars\_to\_file()  
 return  
 print("未找到该车辆！")  
  
 def rent\_car(self): *# 出租汽车* plate = input("请输入要租用的车辆车牌号: ")  
 for car in self.car\_list:  
 if car.plate == plate:  
 if car.is\_rented:  
 print("该车已被租用！")  
 else:  
 car.is\_rented = True  
 car.time = int(input("请输入你的租车时间: "))  
 print("租用成功！")  
 self.save\_cars\_to\_file()  
 return  
 print("未找到该车辆！")  
  
 def return\_car(self): *# 归还汽车* plate = input("请输入要归还车辆的车牌号: ")  
 for car in self.car\_list:  
 if car.plate == plate:  
 if car.is\_rented:  
 car.is\_rented = False *# 归还后将车变成未租用状态* total\_price = int(car.time \* car.price)  
 print(f"租金总额: {total\_price} 元")  
 self.save\_cars\_to\_file()  
 else:  
 print("该车未被租用，无需归还！")  
 return  
 print("未找到该车辆！")  
  
 def show\_cars(self): *# 展示现有的汽车信息* if not self.car\_list:  
 print("目前系统内还没有车辆")  
 else:  
 print("车牌号\t车品牌\t租金\t状态")  
 for car in self.car\_list:  
 print(f"{car.plate}\t{car.brand}\t{car.price}\t{'已租用' if car.is\_rented else '未租用'}")  
  
 def show\_stats(self): *# 展示目前的出租状况* total\_num = len(self.car\_list)  
 *# 已出租的车辆* rented\_num = sum(1 for car in self.car\_list if car.is\_rented)  
 *# 已赚取的总金额* total\_fee = sum(car.price \* car.time for car in self.car\_list if car.is\_rented)  
 rent\_rate = (rented\_num / total\_num) \* 100 if total\_num != 0 else 0  
 print(f"系统内汽车的出租率: {rent\_rate:.2f}%")  
 print(f"系统已赚取的总金额: {total\_fee:.2f} 元")  
  
  
def main():  
 manager = CarManager()  
 manager.load\_cars\_from\_file()  
 while True:  
 print("\*\*\*\*\*\*\* 欢迎使用汽车管家 \*\*\*\*\*\*\*\*\*")  
 print("\*\*\*\*\*\*\* 1.添加新的车辆 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 2.删除已有车辆 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 3.修改车辆信息 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 4.租用车辆 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 5.归还车辆 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 6.展示现有的车辆信息 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 7.统计目前的出租率和赚取金额 \*\*\*\*")  
 print("\*\*\*\*\*\*\* 0.退出程序 \*\*\*\*")  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 choice = input("请选择您现在要执行的操作: ")  
 if choice.isdigit(): *# 检查字符串是否由数字字符组成* choice = int(choice)  
 if 0 <= choice <= 7:  
 if choice == 1:  
 manager.add\_car()  
 elif choice == 2:  
 manager.remove\_car()  
 elif choice == 3:  
 manager.modify\_car()  
 elif choice == 4:  
 manager.rent\_car()  
 elif choice == 5:  
 manager.return\_car()  
 elif choice == 6:  
 manager.show\_cars()  
 elif choice == 7:  
 manager.show\_stats()  
 else:  
 manager.save\_cars\_to\_file()  
 print("欢迎您的下次使用\n再见")  
 break  
 else:  
 print("输入错误，请重新输入！")  
 else:  
 print("输入错误，请重新输入！")

main()