# S1阶段项目-汽车租赁系统

## 本章任务

使用面向对象(OOP)的编程思想，构建“汽车租赁系统”。

## 功能需求

1. 管理员模块
   1. 注册(将所有注册管理员对象信息使用一个全局静态的Map<String,Admin>

保存,只要程序不结束,该用户Map就一直存在!)

* 1. 登录
  2. 修改密码

1. 汽车管理模块
   1. 汽车信息浏览(单部车详情浏览、多车概览)
   2. 新车入库(多线程，要求同步)
   3. 汽车报废(多线程，要求同步)
2. 租赁管理模块
   1. 汽车出租
   2. 汽车归还
   3. 出租、归还记录浏览，并将日志记录在XML文件中(任意路径均可)，

可采取以下格式:

<records>

<record id=”1”>

<carid>川A3908U</carid>

<pid>511203198708150321</pid> <!—租车人身份证号 -->

<type>1</type> <!-- 1代表”出租”, 2代表”归还” -->

<date>2018年5月1日 09:10:12</date> <!-- 租赁交易日期 -->

</record>

<record id=”2”>

<carid>鲁B120XR</carid>

<pid>372203198801151121</pid> <!—租车人身份证号 -->

<type>0</type> <!-- 1代表”出租”, 2代表”归还” -->

<date>2018年5月2日 13:20:02</date> <!-- 租赁交易日期 -->

</record>

<!-- 其它租赁记录……该日志XML文件是”追加模式” -->

</records>

## 类的设计

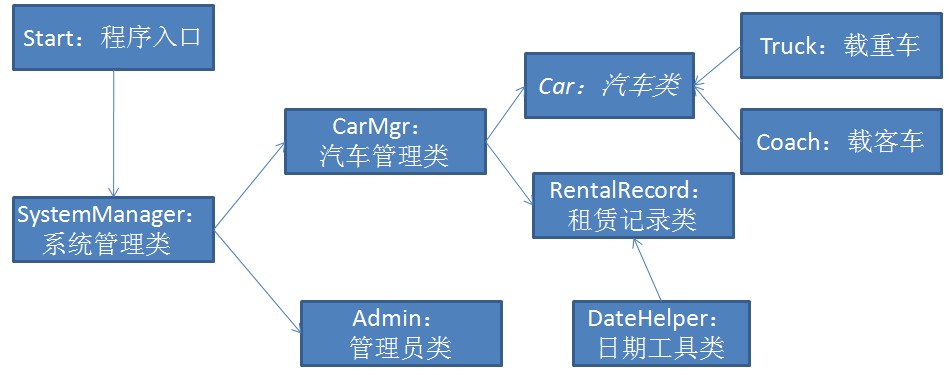


图7.3.1 汽车租赁系统类设计图

### SystemManager类的设计

SystemManager类是整个“汽车租赁管理系统的抽象”，该类的实例代表一个“系统对象”，有一个work方法，调用后，整个系统开始工作。

SystemManager类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | Admin admin; | 代表整个系统中的管理员 |
| CarMgr carMgr = new CarMgr(this); | 代表系统中的汽车管理对象 |
| 方法 | public void work() | 开始工作的方法，显示系统名称，进入登录环节，登录环节可以调用Admin类的静态方法Admin login(String loingId, String loginPsw); |
| private void showMenu() | 登录成功后，进入后台管理主菜单，包括密码修改、汽车管理、租赁管理 |

### Admin类的设计

Admin类是对系统使用者管理员角色的抽象，该类的实例代表一个“成功登录的管理员对象”，包含了账号、密码、管理员姓名等属性，还包含登录的静态方法，成功登录后返回当前的管理员对象，该对象拥有修改密码的方法。（无论登录也好还是修改密码也好，都基于隐藏在Admin类私有的，静态的load和save方法，来读取配置文件中管理员的信息）

Admin的信息都存储在data\admin.txt中

Admin类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **private static ArrayList<Admin> adminList;** | 代表系统中所有管理员对象的列表 |
| **private String loginId;** | 代表管理员的账号（对应的get、set方法不再类的方法中赘述） |
| **private String loginPsw;** | 代表管理员的密码（对应的get、set方法不再类的方法中赘述） |
| **private String name;** | 代表管理员的姓名（对应的get、set方法不再类的方法中赘述） |
| 方法 | **public static Admin login(String loginId, String loginPsw)** | 静态的登录方法，根据账号和密码查询管理员信息对象，没有查找到返回null |
| **private void modifyPsw()** | 管理员对象的修改密码的方法 |
| **public boolean load()** | 读取admin.txt文件，加载所有的Admin对象信息 |
| **public boolean save()** | 存储adminList中的所有Admin对象信息 |

### Car类的设计

Car类是对系统中租赁的汽车的抽象，该类是一个抽象类，包含了一系列汽车的属性，还包含一个根据日租金计算租赁费用的方法。

Car的信息都存储在data\car.txt中

Car类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **public static final String *TYPE\_ZAIKECHE* = "载客车";** | |
| **public static final String *TYPE\_ZAIHUOCHE* = "载货车";** | |
| **public static final String *STATE\_BAOFEI* = "报废";** | |
| **public static final String *STATE\_KONGXIAN* = "空闲";** | |
| **public static final String *STATE\_YICHUZU* = "已出租";** | |
| **private String id;** | 车牌号 |
| **private String engine;** | 发动机号 |
| **private String name;** | 车名 |
| **private String color;** | 颜色 |
| **private Date product\_date;** | 出厂日期 |
| **private int miles;** | 里程 |
| **private String oil\_type;** | 燃油类型 |
| **private int rent\_security;** | 租车押金 |
| **private int rent\_money\_day;** | 日租金 |
| **private String state;** | 状况（空闲，已出租，报废） |
| 方法 | **public int calcRental(int days)** | 根据日租金以及出租天数来计算基本租金 |

### Coach类的设计

Coach类是对系统中租赁的“载客车的抽象”，该类继承了Car类，增加了一个载客数seatCount属性，还重写了父类中计算租金的方法，除了父类中基本的天数乘以日租金的基本租金外，还根据载客人数来额外增加租金，规则如下：

4座及4座以下，没有额外租金

5座到7座，增收基本租金的20%

8座到22座，增收基本租金的100%

23座及以上，增收基本租金的200%

Coach类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **private int seatCount;** | 载客数 |
| 方法 | **public int calcRental(int days)** | 基本租金+额外租金 |

### Truck类的设计

Truck类是对系统中租赁的“载货车的抽象”，该类继承了Car类，增加了一个载货数weight属性，还重写了父类中计算租金的方法，除了父类中基本的天数乘以日租金的基本租金外，还根据载货吨数来额外增加租金，规则如下：

1吨及1吨以下，没有额外租金

1吨以上到3吨以下，增收基本租金的50%

3吨以上到5吨以下，增收基本租金的100%

5吨以上到10吨以下，增收基本租金的200%

10吨以上，增收基本租金的400%

Truck类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **private int weight;** | 载货数 |
| 方法 | **public int calcRental(int days)** | 基本租金+额外租金 |

### RentalRecord类的设计

RentalRecord类是对系统中“汽车租、还记录的抽象”，该类包含了租车人的信息、租车的押金、租车日期、还车日期、租车的实际天数、租车等办理人、租车的收入等属性。

RentalRecord的信息都存储在data\record.txt中

RentalRecord类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **private String customer;** | 租车人 |
| **private String pid;** | 租车人身份证号码 |
| **private String tel;** | 租车人联系电话 |
| **private int yj;** | 实缴押金 |
| **private int ts;** | 预计租车天数 |
| **private String rentDate;** | 租车日期 |
| **private String returnDate = "未归还";** | 还车日期 |
| **private int sjts = 0;** | 实际租车天数 |
| **private int sr = 0;** | 收入 |
| **private String carCph;** | 租用的汽车车牌号 |
| **private String adminName;** | 办理租车手续的管理员姓名 |

### CarMgr类的设计

CarMgr类是对系统中“汽车相关操作的抽象”，该类是整个系统中最复杂的类，它对汽车入库、报废、浏览，汽车的租借、归还、租赁记录浏览，而支持这些操作的底层就是对汽车信息和租赁信息的读取和存储。

CarMgr类的成员如下：

|  |  |  |
| --- | --- | --- |
| 属性 | **private ArrayList<Car> carList = new ArrayList<Car>(); //汽车列表** | |
| **private ArrayList<RentalRecord> recordList = new ArrayList<RentalRecord>(); //租赁记录列表** | |
| **private SystemManager sys;** | 系统管理对象，通过CarMgr的构造方法传入 |
| 方法 | **private void load()** | 加载汽车信息 |
| **private boolean save()** | 存储汽车信息 |
| **public void showCarMenu()** | 显示“汽车管理”菜单 |
| **private void carView()** | 汽车信息浏览 |
| **private void carAdd()** | 汽车入库 |
| **private void carStop()** | 汽车报废 |
| **private boolean loadRecord()** | 加载汽车租赁记录信息 |
| **private boolean saveRecord()** | 存储汽车租赁记录信息 |
| **public void showRentalMenu()** | 显示“租赁管理”菜单 |
| **public void carRent()** | 汽车出租 |
| **public void carReturn()** | 汽车归还 |
| **public void recordView()** | 租赁记录信息浏览 |

### DateHelper类的设计

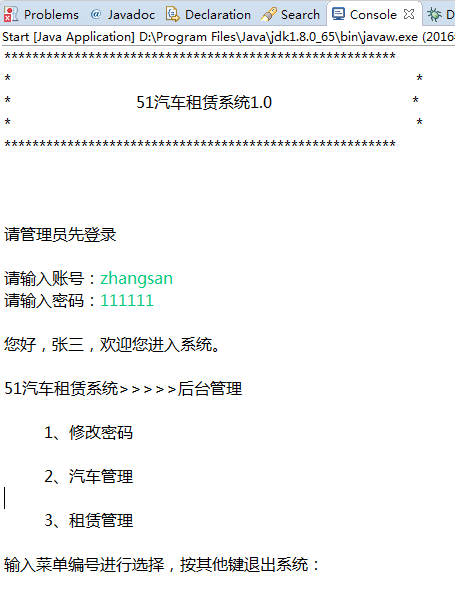
DateHelper类是一个关于日期时间的帮助类，主要是为了简化日期时间java.util.Date类的简化和计算日期天数差，提供了几个静态的方法。

DateHelper类的成员如下：

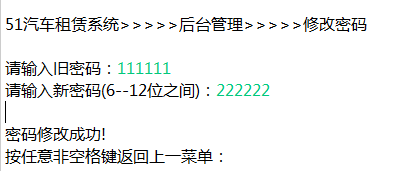
|  |  |  |
| --- | --- | --- |
| 属性 | **private static SimpleDateFormat *sdf*=new SimpleDateFormat("yyyy-MM-dd");** | 指定模板的日期格式化类对象 |
| 方法 | **public static String getDate()** | 获取当前日期的格式化字符串形式 |
| **public static String getDate(Date date)** | 获取指定日期的格式化字符串形式 |
| **public static int getDayDiff(String date1, String date2)** | 获取两个字符串格式日期的天数差 |

## 4 系统运行截图

### 4.1 登录和主菜单



### 4.2 管理员修改密码

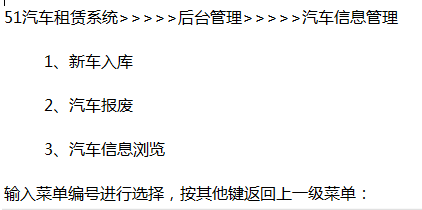


### 4.3 汽车管理

#### 4.3.1 汽车管理菜单

所有汽车的信息可以保存在\*.txt中，然后用IO进行读写

比如：每行1条汽车数据，按照汽车类的属性从左到右依次保存，每个属性用#分隔即可



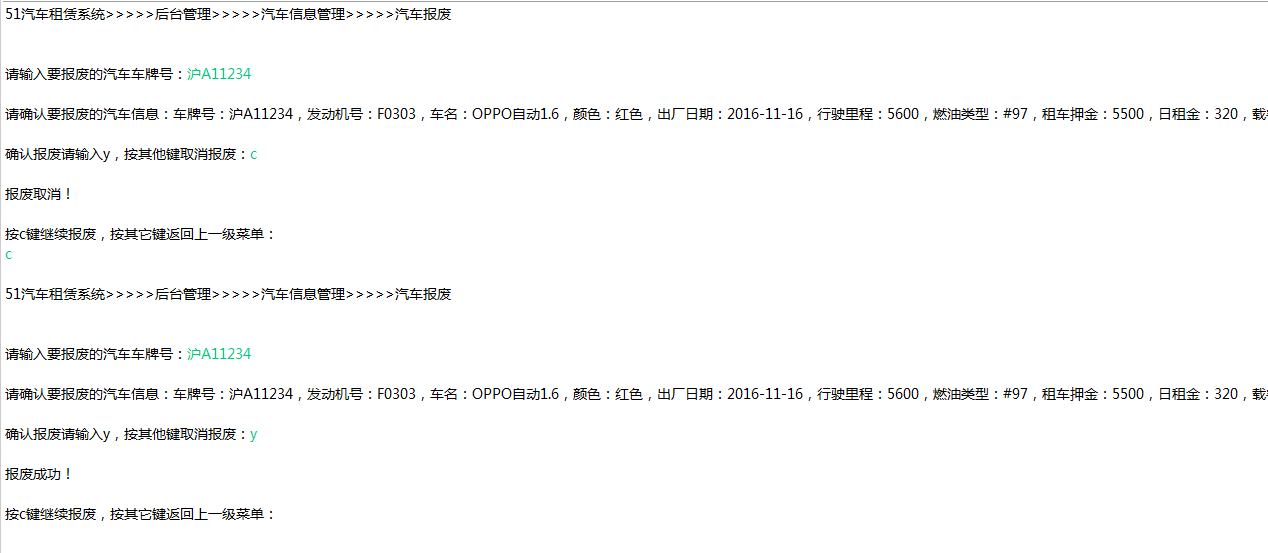
#### 4.3.2 新车入库



#### 4.3.3 汽车信息浏览

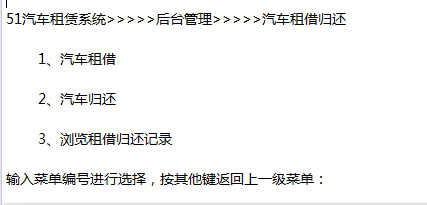


#### 4.3.4 汽车报废



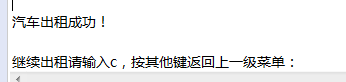
### 4.4 租赁管理

#### 4.4.1 租赁管理菜单

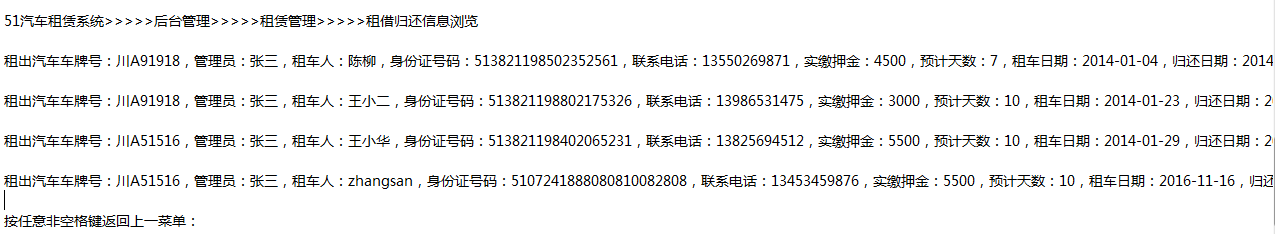


#### 4.4.2 汽车租借





#### 4.4.3 浏览租借归还记录





#### 4.4.4 汽车归还

