## 一．问题描述

用面向对象方法和OOP语言，实现一个停车场活动的仿真程序。需求在文档中已详细给出，在这里不在赘述。

## 系统设计

#### 2.1．静态模型

##### 2.1.1 系统类图

图太大，见附件系统设计图。

##### 2.1.2 类图说明

（1）Access类，关卡类，入口类和出口类的父类。包含出入口皆有的属性和操作。

（2）Entrance类，入口类，继承自关卡类，除了继承父类所有属性方法之外，还具有特殊功能，即显示提示信息，例如当场内车数量已满时，在入口处有警示信息。另外系统中只存在一个入口对象，所以采用Singleton模式。提供getInstance方法来获取此对象。

（3）Exit类，出口类，继承自关卡类。同样也采用Singleton模式。

（4）DriveLaneManager类，车道管理类。系统中唯一存在一个对象。

（5）SpaceManager类，车位管理类。系统中唯一存在一个对象。

（6）ParkingLot类，车场类，主要由DriveLaneManager对象和SpaceManager对象组合而成。完成对整个车场的管理。

（7）SpaceDao接口，车位存取接口。包含对车位管理的所有方法。有两个子类来实现。

（8）SpaceDaoMemory类，实现SpaceDao接口。将车位信息情况存储至内存中。

（9）SpaceDaoDb类，实现SpaceDao接口。可以将车位信息存储到数据库中，以实现系统的可扩充，可移植性。本系统暂时未具体实现此种方式。

（10）Employee类，员工信息实体类。

（11）EmployeeDao类，员工信息存取类。用来初始化员工信息，以及存取。

（12）Car类，车辆抽象类。由于车辆在不同时刻有不同的状态，但他们有共同的属性和方法。抽象类Car就封装了不同状态车的共性。含有抽象方法nextStep(),表示该状态的车下一秒的动作。

（13）LogInfo类，日志信息类。封装每条日志信息。

（14）WriteLogInfo类，写日志信息类。用于生成日志文件的类。

（15）CarFactory类，工厂类，随机产生需要进场的车。

（16）EntranceReadCar类，入口读卡状态的车辆类。继承抽象类Car。

（17）EntrancePassCar类，入口正通过状态的车辆类。继承抽象类Car。

（18）SearchSpaceCar类，寻找车位的车辆类。继承抽象类Car。

（19）ToSpaceCar类，从车道要停到车位的车辆类。继承抽象类Car。

（20）OnSpaceCar类，停放在车位状态下的车辆类。继承抽象类Car。

（21）ToLaneCar类，从车位离开到车道的车辆类。继承抽象类Car。

（22）InLaneCar类，在车道上驶向出口的车辆类。继承抽象类Car。

（23）ExitReadCar类，出口读卡车俩类。继承抽象类Car。

（24）ExitPassCar类，通过出口的车辆类。继承抽象类Car。

（25）DeadCar类，无效的将被删除的车辆类。包括入口读卡失败或离开的车辆。

##### 2.1.3 核心结构

如系统类图所示，Park类包含Entrance（入口），Exit(出口)，ParkingLot（场内），WriteLogInfo(写日志类)， cars(所有车辆集合)。仿真时候调用simulate()函数，对cars中的每个车辆对象调用其nextStep()方法。刷新车辆状态（可能车辆对象状态改变，也可能该车辆对象消亡，而产生新状态下的车辆对象，例如SearchSpaceCar对象找到停车位后，他可能会消亡，而新的车辆对象ToSpaceCar诞生）。那么对所有的车辆对象每秒钟刷新一次，则可实现车场整体的动态仿真。

#### 2.2．动态模型

##### 2.2.1 序列图

图太大，见附件中系统设计图—序列图。

##### 2.2.2 说明

此序列图主要描述一辆车有入场到出场的生命周期的变迁。

从程序的角度我们描述一下这个变迁过程，首先在Park中cars线性表中存储着每个活动状态的车辆对象指针（其实是java中的引用，只是为了描述方便，采用指针一词 ）。如下图所示。首先carFactory产生一个EntranceReadCar车对象，指针指向该对像。每秒调用其nextStep()方法，若读卡成功，并且栏杆已经抬起，则产生新的EntrancePassCar，原指针又指向EntrancePassCar。EntrancePassCar每秒调用其nextStep()方法改变其通过的状态，当完全通过后，产生新的车对象SearchSpaceCar。原指针又指向SearchSpaceCar。SearchSpaceCar调用其nextStep()方法刷新他在车道上的位置，前进过程中寻找空车位，当找到空车位时。新的ToSpaceCar诞生，原指针指向ToSpaceCar。这是一个过渡性质的对象。两秒后当车移动到车位上面时，OnSpaceCar对象诞生。原指针又指向OnSpaceCar。OnSpaceCar每秒调用其nextStep()方法判断其停车时间是否已到，若到，并且前方车道空闲，则诞生新的ToLaneCar对象，原指针又指向ToLaneCar。两秒后，车移动到车道上，InLaneCar对象诞生，原指针又指向InLaneCar对象。InLaneCar调用自己的nextStep()方法，每秒刷新自己在车道上面的位置，当他到达出口时，产生新的ExitReadCar对象，开始读卡。原指针指向ExitReadCar。当验证合法，并且栏杆打开后，ExitPassCar诞生，指针指向ExitPassCar。ExitPassCar每秒调用其nextStep()方法，每秒刷新器通过状态，当完全通过时，DeadCar诞生。系统将其移除。这就是一辆车整个生命周期，每变更一次状态，原车对象消亡，指针指向新的状态车对象。至于原车对象占用的内存，将被垃圾收集器回收。当然也可自己释放。



一辆车生命周期示意图

## 三．仿真流程

程序入口为Park类中的main方法，简便起见通过修改n,e即可改变车场容量和员工多少。当然也可从控制台获得。运行java application即可。

## 四．实现部分

#### 4.1 模块实现

如下图所示，共有五个包。

1. edu.xidian.liyan.log 封装写日志功能。
2. edu.xidian.liyan.park 停车场主类。
3. edu.xidian.liyan.park.access 封装出入口功能类。
4. edu.xidian.liyan.park.car 封装车辆类等。
5. edu.xidian.liyan.park.drivelane 封装车道上活动等相关类。
6. edu.xidian.liyan.park.parkinglot 封转车场内部管理等功能。
7. edu.xidian.liyan.park.space 封装车位管理功能类。

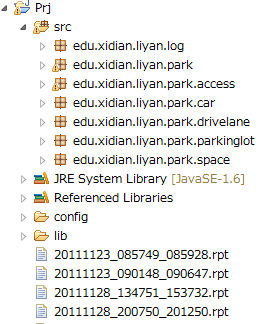


图4.1 模块设计

#### 4.2 详细实现

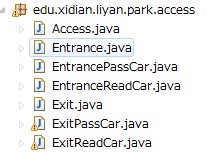
##### 4.2.1 日志管理模块 edu.xidian.liyan.log

C:\Users\Administrator\AppData\Roaming\Tencent\Users\644730737\QQ\WinTemp\RichOle\WB{9%CZQ5LVU(_UWAYLT7ME.jpg

LogInfo类封装每条日志信息。WriteLogInfo类将他的属性中存储的若干日志信息按要求格式输出到日志文件中。如下图。

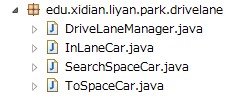
C:\Users\Administrator\AppData\Roaming\Tencent\Users\644730737\QQ\WinTemp\RichOle\PA0CHO[M5`H5XZ7DM[`DFBX.jpg

##### 4.2.2 出入口管理类 edu.xidian.liyan.park.access



Entrance和Exit继承自Access.分别实现对入出口的管理。EntranceReadCar,EntrancePassCar,ExitReadCar,ExitPassCar分别是在出入口四种状态的车辆。

##### 4.2.3 车道管理 edu.xidian.liyan.park.drivelane

****

DriveLaneManager用来实现对车道的管理。InLaneCar,SearchSpaceCar,ToSpaceCar分别是在车道上三种状态的车辆。

##### 4.2.4 车位管理 edu.xidian.liyan.park.space

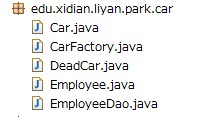


SpaceDaoDb,SpaceDaoMemory实现了接口SpaceDao,分别实现在数据库和内存中对车位的存储及管理。简单起见。本系统只实现了后者。

##### 4.2.5 车场管理edu.xidian.liyan.park.parkinglot

ParkingLog类，由SpaceManager和DriveLaneManager组合而成，完成对车场内部的管理。

##### 4.3.6 车辆管理 edu.xidian.liyan.park.car



Car抽象类，规定了作为一个车所应有的大部分属性和功能。CarFactory专用于生成入场的车对象。DeadCar将被系统移除的车对象类。Employee和EmployeeDao用于管理车辆ID.

##### 4.2.7 edu.xidian.liyan.park

包含类Park作为系统的主控类。内有simulate函数来实现仿真。