**Lecture 2 - 管理类OOD: Parking Lot & Restaurant - Singleton Pattern**

* 管理类 - 题目后面都可以接上三个字: 管理员
  + e.g.: Gym, Parking lot, Restaurant, Library, Super market, Hotel
  + 设计一个模拟/代替管理员日常工作的系统
* 解题方法
  + Clarify: 除了题目中问的名词外，还需要从管理的名词来考
  + Core object: 有进有出
  + Cases: 从管理员角度考虑
    - Reserve: 预定
    - Serve: 服务
    - Checkout: 买单
  + Class: 经常可以使用**收据**的形式，来保管信息
  + Correctness

**Clarify**

* What
  + 关键词: Parking lot, Vehicle, Parking Spot
  + Parking lot: 考虑多层的Parking lot, 没有错层Graphical user interface

    Description automatically generated
  + Vehicle: 考虑三种大小的车
    - 如何设计停车场支持停不同大小的车？

两种方案，一种是大车占用两个车位，第二种也可以设计专门的大车停车位，优缺点如下图

Graphical user interface, application

Description automatically generated

* + 不考虑残疾人停车位/充电车位
* How
  + 规则1: 如何停车
    - 一定要从**停车场**的角度来考虑,而不是车的角度
      * 停车场: 开进停车场 -> 返回一个能停的地方 ->停进一个位 置
      * 车: 开进停车场 -> 经过每一个位置看看能不能停 -> 停进一个位置
  + 规则2: 付费
    - 免费还是付费
* Who: Optional

**Core Object**

* Parking Lot
* Parking spot
* Car, Bus, Motorcycle
* 映射关系
  + Parking has a list of Spot
  + 错误方法: spot里面有car,或者parking lot有list of car

**Cases - 站在管理员的角度想**

* Bus / Car / Motorcycle
  + N/A
* Parking Lot
  + Get available count (reserve)
  + Park vehicle (serve)
  + Clear spot (checkout)
  + Calculate price (checkout)
* Parking Spot
  + N/A

**Graphical user interface, application

Description automatically generated with medium confidence**

**Table

Description automatically generated把availableCount设成private的目的是为了封装，这样外部的类就不能自行修改这个量**

**别的类可以通过public的 getAvailableCount来获取这个量的值**

**A screenshot of a computer

Description automatically generated with low confidence**

**不好的方案：**

**Table

Description automatically generated with low confidence虽然能work但是非常低级**

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**不好的方案：根据不同车型找spot，这个extentability就会很差**

**Table

Description automatically generated**

**Table

Description automatically generated**

**Graphical user interface, text

Description automatically generated**

**vehicle的size变成#protectedText

Description automatically generated**

**可以看到如果是private那么子类里还得再次申明这个变量，但如果是protect就不用了**

**更新版的：**

**Graphical user interface, text, application

Description automatically generated**

**对parkinglot进一步的improve**

**Table

Description automatically generated**

**对于parkvehicle怎么知道是不是停成功？可以添加exception。**

Diagram

Description automatically generated with low confidence

**不把它返回值设置成true false的原因是因为二元t f有不同的解释，并不能代表很多，而且你还得写doc来解释，对于产业界不是很好的practice**

**Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, text, email

Description automatically generated**

**Table

Description automatically generated**

**Diagram

Description automatically generated**

**上面的方案虽然可以解决问题，但是缺点是不elegent**

**更好的practice是用receipt**

**Diagram

Description automatically generated**

**Text, letter

Description automatically generated**

**Class & Correctness**

* Draw UML per Use Case
* 从以下几方面检查:
  + Validate use cases (检查是否支持所有的use case)
  + Follow good practice (面试当中的加分项，展现一个程序员的经验)
  + S.O.L.I.D
  + Design pattern

加excption

Diagram

Description automatically generated with low confidence

这些exception是单独存在的类，不需要和其他类有connection

Text

Description automatically generated

Text, letter

Description automatically generated

solution1缺点：Logo

Description automatically generated with medium confidenceDiagram

Description automatically generated

Diagram

Description automatically generated

大致的流程就是一辆车进parkinglot，然后parkinglot这个class就去调用findSpotForVehicle（Vehicle v）来获取有哪些spot是可用的，然后通过parkVehicle这个method去产生ticket

followup：怎么考虑多线程

当你有多个线程的时候，多个线程会同时访问一个data，这个data也就是critical data，比如这个parking lot，critical data就是parkinglot class里的list<Level> levels就是一个critical data，比如一辆车正在3楼找spot，这个同时有其他车也正在找车位，那这个时候第一辆车在找spot的时候我们就应该把list<Level> levels这个data锁住，不让其他用户去access这个data，同时只能有一个用户去access这个data

**Design Pattern - Singleton**

public class ParkingLot {

private static ParkingLot \_instance = null;

private List<Level> levels;

private ParkingLost() {

levels = new ArrayList<List>();

}

public static synchronized ParkingLot getInstance() {

if (\_instance == null) {

\_instance = new ParkingLot();

}

return \_instance;

}

}

* synchronized getInstance()使得Singleton线程安全
* static \_instance 使得只有一个instance
* private ParkingLot()使得constructor只被调用一次

Graphical user interface, text, application, email

Description automatically generated

Diagram

Description automatically generatedText

Description automatically generated

Text

Description automatically generated

以上优点是线程安全，缺点是不够efficient，因为一个函数调用这个之后就block了其他的函数调用

最优解

Text

Description automatically generated

当一个method是static时，说明并不需要在run time就把它构造出来，在build time，也就是编译器在构造这个类的时候他就会把这个类里面的东西执行一遍，

final不会担心别的线程对他进行修改

full code: <https://github.com/pulkitent/parking-lot-lld-oop-ood/tree/master/src/main/java>