1 给出一个 Java ADT 题目

某公司拟设计和开发一个停车场管理系统，其基本需求陈述如下：

(1) 一个停车场有 n 个车位(n>=5)，不同停车场包含的车位数目不同。

(2) 一辆车进入停车场，如果该停车场有空车位且其宽度足以容纳车的宽度，则可以在此停车。

(3) 停在停车场里的车，随时可以驶离停车场，根据时间自动计费（每半小时 10 元，不足半小时按半小时计算）。

(4) 停车场管理员可以随时查看停车场的当前占用情况。

下图给出了一个包含 13 个停车位的小型停车场示例图，其中 1-8 号停车位较窄，9-13 号停车位较宽。在当前状态下，第 1、3、7、9、10 号车位被占用，其他车位空闲。

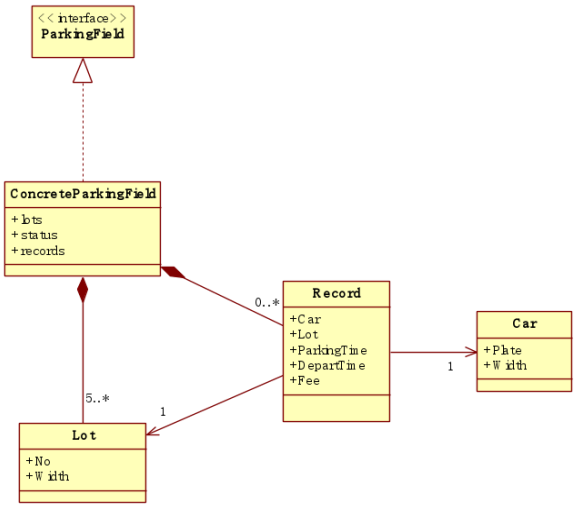
客户端程序的功能需求：

- 构造一个停车场
- 构造若干台车
- 依次将车停进停车场，可以指定车位，也可以不指定车位（随机指派）
- 随机将车驶离停车场，车辆驶离时给出入场时间、出场时间、费用金额
- 查看当前停车场的状态（目前每个车位停了什么车）

特殊情况：

- 停车进场的时候（两种情况）：该车辆已经在停车场里面了
- 停车进场的时候（不指定车位）：停车场已没有可供该车停车的位置
- 停车进场的时候（指定车位）：该车位已被占用、该车位过窄、没有该车位
- 驶离停车场的时候：该车并没有停在这里

2 回顾 ADT 设计习题课的结果



3 扩展 Car 至其他交通工具类型：可复用性

对上述设计进行扩展，考虑将来不仅可以停汽车，也可以停马车、摩托车、飞机（相当于将停车场扩展到了飞机场）等，并可在后续持续扩展其他事物。只要某个 lot 的宽度 (width) 大于某个对象的宽度，即可停在该位置。但是，除了牌照号和宽度之外，马车、摩托车、飞机等还具有与汽车不同的属性和方法。为此需如何修改当前设计？

4 使用 Factory Method 设计模式：可维护性

5 考虑 Car、Motor、Plane 的个性化

给子类 Car、Motor、Plane 增加个性化特征。

6 使用 State 设计模式管理停车物状态：可扩展性

需要扩展功能：每辆 Parkable 对象有两个状态：在停车场、在路上。客户端输入车牌号可查询状态。

7 使用 Decorator 设计模式：可复用性/可扩展性

有些停车场是政府设置的公共停车场（无人管理），有些停车场则是由专门的公司管理。上述 ConcreteParkingField 的 rep 不支持后者。在不改变 ConcreteParkingField 现有实现的情况下，能够做到：

(a) 在创建 ParkingField 对象时包含“公司”信息（String company 即可）；

(b) 车辆在此类停车场进行停车（调用 parking 方法）和驶离（调用 departure 方法）的时候，能够打印输出（System.Out）欢迎和告别信息（分别为：停车场 XX 欢迎车辆 YY、停车场 XX 祝车辆 YY 旅途愉快，其中 XX 表示停车场的名字，YY 表示车辆的车牌号）。

注意：不是说所有停车场都是带公司和欢迎信息的，所以之前设计的 ParkingField 还要保留，不能直接在 ConcreteParkingField 中修改 rep 和相关方法的逻辑，而是要“扩展”——牢记 OCP。

8 使用 Visitor 设计模式：可扩展性

考虑将来对 ParkingField 的功能扩展，使用 visitor 模式改造当前设计。例如要扩展的一个功能是统计停车场当前时刻占用比例(=已停车的车位数量　总车位数)。
在客户端代码中，给定一个 ParkingField 对象，如何使用你的 visitor 来统计当前时刻占用比例。

```
ParkingField pf = ...; //此处假设 pf 已成功创建
                       //并且进行了一系列 parking 和 depart 操作
double fullRatio = ...;
```

该功能可以直接扩展至 ParkingField 接口中，增加一个方法。但是没有解决将来扩展其他新方法的能力。

所以这里使用 visitor 设计模式。

9 使用 Iterator 设计模式：可维护性

ParkingField 需要具备遍历其中所停的所有 Car 对象的能力。拟使用以下形式的 client 端代码，按车辆所在停车位编号由小到大的次序，逐个读取所停车辆。请扩展现有设计方案(修改/扩展哪些 ADT)，在下方给出你的设计思路描述，必要时可给出关键代码示例或 UML 类图辅助说明。

```
Iterator<String> iterator = pf.iterator();
while (iterator.hasNext()) {
    String c = iterator.next();
    System.out.println("A car " + c + " is now parked in " + pf);
}
```

10 使用 Strategy 设计模式：可扩展性、可维护性

ParkingField 接口中定义了一个方法 void parking(String type, String plate, int width)，与另一个带有 num 车位号的 parking 方法相比，使用该方法的 client 端无需提供“停车位号码”信息，而是在方法内部自动进行空闲停车位的选择。现实中有不同的停车位选择方法，例如：

(1) 随机选择一个空闲的、宽度大于车辆宽度的停车位；

(2) 根据停车位编号，优先选择编号最小的空闲停车位，且其宽度大于车辆宽度。

使用 Strategy 设计模式改造现有设计：

在客户端代码调用 void parking(String type, String plate, int width)的时候，如何动态传入某个特定的停车位选择方法？

11 基于语法的输入：可维护性

停车场管理系统启动时，主程序读入外部文本文件，构造多个 ParkingField 对象。该文本文件遵循特定的语法格式，每个以 PF 开头的行代表一个 ParkingField 对象，语法说明如下所示。车位编号为从 1 开始的自然数，车位宽度=常数 20。

(1) PF::= 一个由<>括起来的字符串，分为三部分，分别代表停车场名字、最大车位数、公司名字，三部分之间由逗号“,”分割。

(2) 停车场名字 ::= 字符串，长度不限。可以由一个单词或多个单词构成，单词由字母或数字构成，单词之间只能用一个空格分开。

(3) 停车场最大车位数 ::= 自然数，其值最小为 5。不能为 012、0012 的形式，只能为 12 的形式。

(4) 公司名字 ::= 与停车场名字的语法规则一致，但可以为空。若该部分为空，表示该停车场没有公司管理 (即公共停车场)。

以下是三个例子:

```
PF::=<92 West Dazhi St,120,HIT>
PF::=<Expo820Roadside,10,>  //无公司管理的停车场，最后一个逗号之后为空
PF::=<73 Yellow River Rd,50,Harbin Institute of Technology>
```