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：

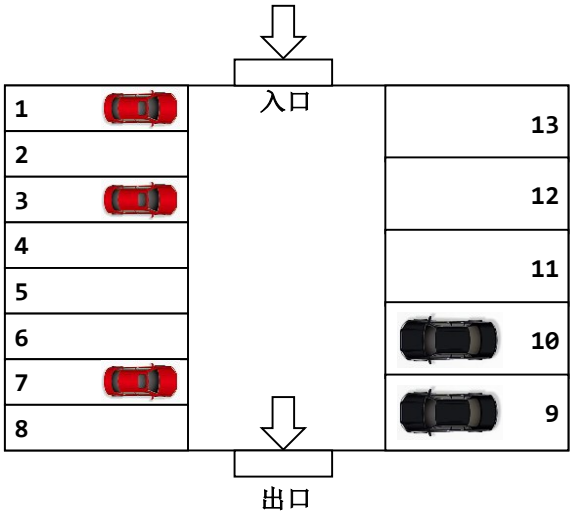
(1) 一个停车场有 n 个车位(n>=5)，不同停车场包含的车位数目不同。

(2) 有车牌号的车可进入停车场，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（学生自行考虑解决，或者等第六章学完之后再回头解决）：

- 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 设计思路

- 使用 OOP 的思路，寻找这里的“名词”：
  - 停车场：车位数目
  - 停车位：编号、宽度
  - 车：车牌号、宽度
  - 一次停车（从入场到出场）：车、入场时间、出场时间、所在停车位、费用
- 动词：
  - 构造“停车场”
  - 构造“车”
  - 停车
  - 驶离

- 计费
- 查看状态

- 哪些名词可以作为 Object？哪些名词作为其他 Object 的属性？
  - 停车场：车位数目、具体哪些车位
  - 停车位：编号、宽度
  - 车：车牌号、宽度
  - 一次停车（从入场到出场）：车、入场时间、出场时间、所在停车位、费用——可变的，不是一下子能构造出来（进入的时候要记录时间/车位、出去的时候要记录时间，目的是为了计算费用）
- 设计原则：尽可能缩小 mutable 的范围

一次停车（从入场到出场）：车、入场时间、出场时间、所在停车位、费用

这些可变信息在哪里管理？

单独造一个 mutable 的 class Record，单独管理它。

但是，谁来负责管理所有的停车记录？客户端程序？no，客户端只面向具体功能，不能让其管理这些。

1. 造一个类，专门负责管理停车记录。——缺点：多了一个类，稍微复杂
2. 让“车”管理自己的状态：停在哪个车场的车位、何时进、何时出。——不好的地方：车的自身固有属性与车的状态，前者不可变，后者可变，混在一起
3. 让“车位”管理状态：谁目前停在“我”的位置上？——不好的地

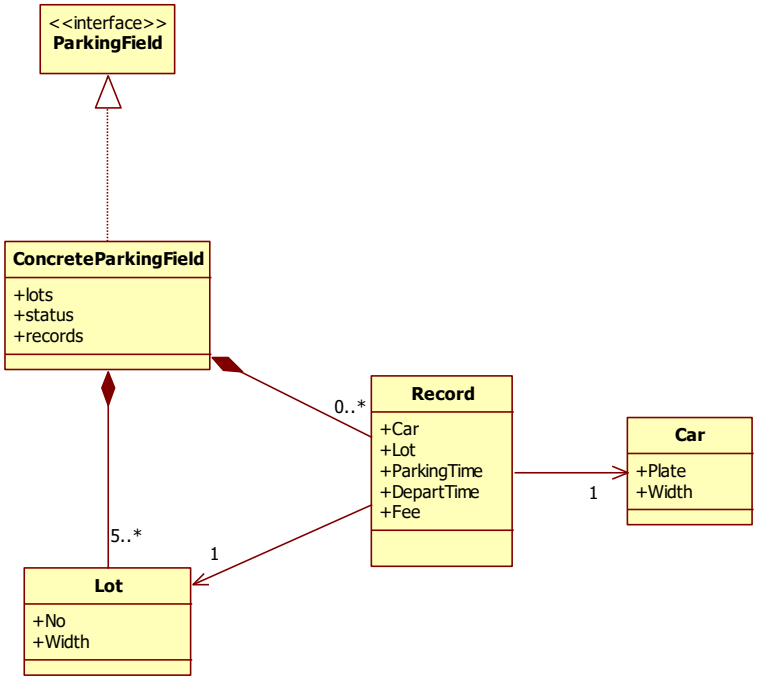
方：与上类似，可变与不可变混在一起

4. 让“”停车场“管理状态：谁目前正停在我这里。——不好的地方：类似

任何一种实现方式都可以。

本次课：按第 4 种方式。

- ADT 设计：接口还是抽象类、具体类?
  - 本例中，只有一个应用，不存在复用，可以直接写具体类
  - 但是，考虑到将来可能面临的变化，以及不同的实现方式，最好按接口方式设计 ADT
  - 接口：ParkingField
  - 类: ConcreteParkingField、Lot、Car、Record (后三者比较简单，Lot 和 Car 都是 immutable 的，不用接口)
- 设计方案



3 设计 ParkingField 接口，撰写 spec

设计方法

撰写 spec

识别 ADT 的四种操作类型

功能需求：

- 构造一个停车场——constructor，见下一节
- 构造若干台车——跟 ParkingField 无关
- 依次将车停进停车场，可以指定车位，也可以不指定车位——mutator，两个方法，parking(Car c, Lot lot)、parking(Car c)，增加停车记录。第一个方法的 Lot 参数，是否有必要暴露给客户端使用？建议改为 int num，用停车位编号。同样的道理，如果不希望把 Car 暴露出去，可改成 parking(String plate, int width, int num), parking(String plate, int width)
- 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——mutator， double depart(String plate)，增加离场记录，并计算费用。如果觉得两项功能耦合在一起，可以拆开：void depart(String plate)、double calcFee()。但计算费用这个方法无需对外暴露，所以不能作为接口的公共方法。
- 查看当前停车场的状态——observer，xxx status()。希望返回什么数据结构？最适合的应该是 Map 结构，Key 是车位，Value 是停的车，Value

可以为空，所以 Map<Integer, String> status()，用车位编号作为 key，用车牌号作为 value。

汇总所有的接口方法：

create() --- 静态工厂方法，constructor

void parking(String plate, int width, int lot) --- mutator，按车位停车

void parking(String plate, int width) ---- mutator，随机分配车位停车

double depart(String plate) ----- mutator，离场计费

Map<Integer, String> status() --- observer

分别设计它们的 spec

需求中的特殊情况怎么处理？

- 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

建议的处理方式：

- 更强的规约：更弱的前置前置、更强的后置条件，让 client 使用更容易，把责任放在实现者身上。

- 太弱的 spec，client 不放心、不敢用（因为没有给出足够的承诺)。开发者应尽可能考虑各种特殊情况，在 post-condition 给出处理措施。
- 惯用做法是：不限定太强的 precondition，而是在 postcondition 中抛出异常：输入不合法。
- 例如：针对“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“这种情况，无法限定 client 已知停车场是否已满，所以不能在 pre 里写，而是交给 post 来处理。
- 所以，可以针对上述特殊情况，在 post 里使用异常，告知 client。
- 考虑到目前尚未学习如何定义异常，这里先采用通用的异常类 Exception。

Git 提交到 v0.1 design parkingfield interface and specs

代码见 ParkingField.java

4 为 ParkingField 接口增加静态工厂方法

增加静态工厂方法，创建实例

```
    /**
     * 创建一个新的停车场
     *
     * @param nos 各停车位的编号，均为自然数，且无重复
     * @param widths 各停车位的宽度，包含的元素数量=nos
中元素数量 且>=5
```

```
     * @return  一个停车场对象，包含了widths.length个车
位，各车位的宽度
     *与nos中相应数字一致，且各车位上均未有停车
     * @throws  如果违反nos和widths不合法
     */
    public static ParkingField create(int[] nos, int[] widths)
                                                          throws
Exception {
        return new ConcreteParkingField(nos, widths);
    }
```

目前先写出调用具体类 ConcreteParkingField 的构造函数，具体实现一会再说。

参数为什么这么设计？

- 车场需要包含一组车位，最开始没有停车，所以构造时需要知道多少个车位、每个车位宽度如何。目前用了两个数组，规定了它们的 pre-condition。
- 也可以用其他参数策略，例如两个 List、一个 Map。Map 中的 key 是车位号，value 是车位宽度，Map 可天然保证 key 不重复
- 在 ParkingField 中增加了一个新的 create()方法，参数不同，是 overload

```
    /**
     * 创建一个新的停车场
     *
```

```
     * @param lots key是车位编号（自然数），value是车位宽度（自然数），lots长度>=5，
     * @return 一个停车场对象，包含了lots.size()个车位，各车位的编号与宽度与lots中的KV一致，且各车位上均未有停车
     * @throws 如果lots不合法
     */
    public static ParkingField create(Map<Integer, Integer> lots)
                                                        thro
ws Exception {
        return new ConcreteParkingField(lots);
    }
```

5 针对 ParkingField 接口设计测试用例

根据各个方法的 spec 设计测试用例

撰写 Testing Strategy

写出测试用例代码

静态工厂方法和实例方法的测试，分开。让测试类的职责更清晰。

针对静态工厂方法：ParkingFieldStaticTest.java

调用 create(..,..)或 create(..)创建一个新停车场对象

如何观察它？目前的接口方法里没有足够的观察者。

要能够检查所有 post-condition 是否满足。

要观察什么：（1）有几个车位；（2）参数里的每个“车位编号+车位宽度“是否包含；（3）每个车位上是否没有停车

已有的一个 observer 方法 public Map<Integer, String> status()，可以用来间接实现(1)(3)，但无法实现(2)。

所以给 ParkingField 增加三个 observer 方法：

public int getNumberOfLots(); //非必须，可以用 status().size()

public boolean isLotInParkingField(int num, int width);

public boolean isEmpty(); //非必须，可以检查 status()返回值的每个 value 是否为 “”。

将它们补充到接口里，写出 spec。

如何为 create 设计测试用例？——以下都用 create(Map ..)为例

- 按车位数量划分：=0，<5, =5, >5
- 按车位号划分：全部自然数、包含非自然数（0 或负数）
- 按车位宽度划分：全部自然数、包含 0 或负数
- 特殊情况：车位宽度都一样、车位宽度不一样 //非必须

在类的开始位置写 testing strategy

使用“每个取值至少覆盖一次”策略或“笛卡尔积全覆盖”

设计测试用例

- 空
- <1,10><2,10><3,10><4,10>
- <1,10><2,10><3,10><4,10><5,10>
- <1,10><2,10><3,10><4,10><5,10><6,10>
- <1,10><2,0><3,10><4,10><5,10><6,10>
- <1,10><2,-10><3,10><4,10><5,10><6,10>

为每个测试用例写测试函数，函数前写覆盖了哪个分类

针对其他方法：ParkingFieldInstanceTest.java

针对 parking 函数，输入车牌号、车宽、车位号。按上述过程设计测试用例

前提：用 create()静态方法创建一个车场

观察其结果是否正确：

- 车牌号为plate的车辆，之前没停在车场，执行后停在了车位号为num的车位上
- 该车位宽度大于等于车宽度
- 其他车位的状态不变

增加观察者方法?

- boolean isParkingOn(String plate, int width, int num)　车辆 plate 是否停在车位 num 上。该函数非必须，可以对 status()返回的 KV 进行查询得到结果。
- int getLotWidth(int num)　返回某车位宽度，判断宽度是否足够该车。
- 针对“其他车位状态不变”，可以在 parking()调用前后分别调用 status()返回的 KV 对，逐个作比较。也可以先取出每个车位上的停车情况，执行 parking 后比较结果是否变化，为此需要增加观察者方法 String getCarOnLot(int num)，返回停在 num 车位上的车牌号，没有车则返回 “”。

在类的开始位置写 testing strategy、设计测试用例

- 按 plate 划分：该车已经停在该停车场、该车未在停车
- 按 num 划分：该车位是车场的合法车位、不是合法车位
- 按 num 划分：该车位合法，已被其他车占用、未被占用
- 按 num 和 width 划分：车位宽度不超过 plate 车宽度，等于车宽度、大于车宽度

设计测试用例：

- 车场：<1,10><2,15><3,20><4,20><5,20>
- parking(HA001,10,1)　覆盖车未在停车场、车位合法且未被占用，车宽=位宽

- parking(HA001,10,2)　覆盖车已在停车场、车位合法且未被占用
- parking(HA002,15,6>　覆盖车未在停车场、车位不合法
- parking(HA002,15,3>　覆盖车未在停车场、车位合法且未被占用，车宽<位宽
- parking(HA003,20,3>　覆盖车未在停车场、车位合法且已被占用
- parking(HA003,20,2>　覆盖车未在停车场、车位合法且未被占用，车宽>位宽

为每个测试用例写测试函数，函数前写覆盖了哪个分类。

类中给出了前两个测试用例的方法，其他由学生自己完成。

6 编写具体实现类 ConcreteParkingField

设计 rep

Rep 里要表达什么？内部存储数据结构

要存储什么？——

- 车位数：int 即可
- 哪些车位：集合类 List, Set 等
- 当前车位占用情况（哪个车位停了哪个车、或者为空）——Map<>，key 为车位，value 为车
- 停进来的车何时进入的、何时离场的~~、在哪个车位停~~（要计费）——List<Record>

说明：车位数可以不用单独保存，可以从其他属性中间接得到

所以，一种可行的 rep：

- List<Lot> lots; //一组车位
- Map<Lot, Car> status; //占用情况。如果某 lot 上没有车，那么存什么？null 不好，因为当 status.get(lot)返回 null 的时候，无法判断是因为没停车还是因为没有这个 lot。因此没有车的时候不要放在 status 里。
- List<Record> records; //停车记录

//你是否还有其他方案？可以写出不同的实现类

设计 AF、RI

```
//Rep invariants:
//   lots.size() >= 5;
//   lots.size() >= status.size();
//   status中的每个key均在lots中出现;
//   status中的values中不存在重复;
//   status中的value，该Car的宽度小于等于相应的key（车位）的宽度;
//   对records中的每个Record对象，如果record.getTimeOut()为空，
//         则<record.getLot(),record.getCar()>必定出现在status中
```

```
//Abstraction function:
//   代表着一个停车场，该停车场有lots.size()个车位
```

```
    //  lots中每个元素l代表着一个车位，其编号和宽度是l.getNumber()和
l.getWidth()
    //  l上停的车是status.get(l)
    //  该车场的所有停车记录是records，其中的元素r表示一个停车记录，表明：
    //  车辆r.getCar()在r.getTimeIn()时刻停到了r.getLot()车位上，在
r.getTimeOut()离开，花费r.getFee()元
```

实现 checkRep()

按上面 RI，逐条翻译过来即可

将 checkRep()加到每个方法 return 之前。

```
    private void checkRep() {

        assert lots.size() >= 5;
        assert lots.size() >= status.size();

        Set<Car> parkingCars = new HashSet<>();
        for(Lot lot : status.keySet()) {
            Car car = status.get(lot);
            assert lots.contains(lot);
            assert ! parkingCars.contains(car);
            parkingCars.add(car);
```

```
            assert lot.getWidth() >= car.getWidth();
        }

        for(Record record : records) {
            if(record.getTimeOut() == null) {
                assert status.containsKey(record.getLot());
                assert status.get(record.getLot()).equals(record.getCar());
            }
        }
    }
```

实现构造函数

目前有两个构造函数，以 ConcreteParkingField(Map<Integer, Integer> lots) 为例

在构造函数中，要造出车位，填充到 Rep 中

Status 和 record 在刚启动时为空

实现接口方法

根据 spec 和 rep 逐个实现即可，都不困难

override equals()、hashCode()

ParkingField 是 mutable 的，所以先不需要写 override 这两个

toString()

例子：某 ParkingField 对象的有 5 个停车位，状态如下表所示。

停车位编号	停车位宽度	当前所停车辆
1	200	车牌号：AB001，宽度 180
2	180	空闲
3	200	车牌号：CD002，宽度 180
4	170	车牌号：EF003，宽度 160
5	190	空闲

从上表看出，该停车场目前有 60%的停车位已停车（=3/5)。针对该例子，你所写的 toString()的输出结果应为：

The parking field has total number of lots: 5

Now 60% lots are occupied

Lot 1 (200): Car AB001

Lot 2 (180): Free

Lot 3 (200): Car CD002

Lot 4 (170): Car EF003

Lot 5 (190): Free

这是个基本的 observer。

7 编写辅助类 Car、Lot

Immutable 类

设计 rep，是否有多种表示?

设计 RI，与 Rep 密切相关

实现 checkRep()

设计方法

override equals()、hashCode()、toString()

学生自行完成

8 编写客户端程序

某个停车场管理系统，实现一系列功能

```
Map<Integer, Integer> lots = new HashMap<>();
lots.put(1, 10);
lots.put(2, 15);
ParkingField pf = ParkingField.create(lots);

pf.parking("HA001", 10, 1);
pf.parking("HA002", 10, 2);
System.out.println(pf);
```

9 绘制 Snapshot Diagram

针对客户端程序运行到某个时刻，想象内存状态，绘制 snapshot

在以上主程序执行完之后：

- 首先有一个 lots，是个 Map，先画出来，分别指向各个 Integer；
- 然后创建了一个 pf，看它的构造函数，包含三个部分：lots, status, records，都是 final 的，双线。这三个对象是 mutable 的，所以单线圈，pf 也是单线圈。
- lots 有两个对象，指向 Lot，Lot 内部分别指向 number 和 width，都是直接的值（int）
- status 目前为空
- records 目前也为空
- 然后执行 parking，看它的代码，首先创建了一个 Car 对象，指向了一个 String 双线圈和一个 width 值（无圈）
- 然后在 status 里加入了一个 KV 元素，<Lot, Car>，分别指向 lot 对象和 car 对象
- 在 records 里加了一个元素，该元素也指向 lot、car 对象，并增加了一个 Calendar 对象（停车时间，单线圈，mutable），离场时间目前指向 null，fee 指向 0。

10 表示泄露与安全性

考察 ParkingField、Car、Lot、Record 这些 ADT，考虑客户端代码，判断这些 ADT 是否存在表示泄露？列出可能存在表示泄露的地方，分析其潜在风险，并给出其修改策略。

分别检查每个函数

给出几个表示泄露的客户端例子，以及如何通过防御式拷贝策略修改 ADT 设计。

```
private Calendar timeOut = null;
public Calendar getTimeOut() {
    return timeOut;
}
```

11 执行 JUnit 测试

以及查看测试覆盖度 Eclemma

12 使用 SpotBugs 查看代码潜在风险

例如

Car doesn't define a hashCode() method but is used in a hashed data structure [Scariest(1), High confidence]

Comparison of String objects using == or != [Troubling(11), Normal confidence]

所有内容结束后，git commit 到 v2.0 ADT design for parking field