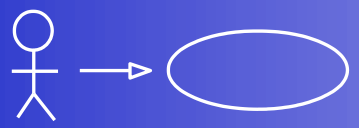
复杂领域模型的建模

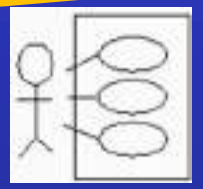
潘加宇



建模工作流

- 组织要解决什么问题——业务建模
- 为了解决组织的问题，待开发系统应提供什么功能和性能——需求

这就是建模！

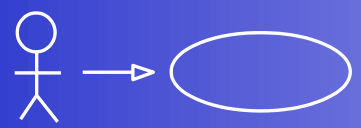
SRS 

提升销售

无意识→有意识
隐式知识→显式知识

- 为了提供功能，系统内部应该有什么样的核心机制——分析
- 为了满足性能，系统的核心机制如何用选定平台实现——设计

降低成本



核心域视角

系统	核心域概念	非核心域概念
文档处理器（如Microsoft Word）	文档、页、行、字……	CStringArray、CFileDialog、MSXML……
电子商务网站（如淘宝网）	商品、订单、会员……	、ActionForm、SessionFactory……

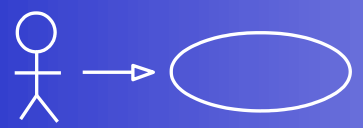
核心域和非核心域

核心域视角

- 根据愿景，这个不应该是系统的用例。
- 是的！我都写好了，运行一下给你看，系统确实提供了这个用例。

- 这两个类关系不应该是泛化，而是关联。
- 是泛化，不信我打开代码给你看，或者逆向工程转出类图给你看。

颠倒的思维：模型是代码的概述

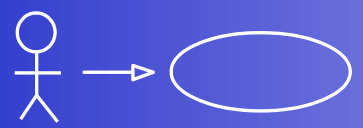


核心域视角

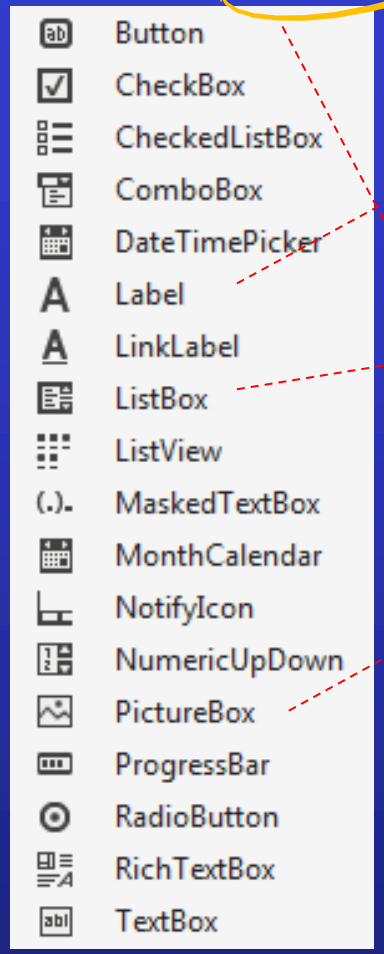
为了理清这个问题，Steve Mellor和我分别针对人们使用UML的特征归纳出三种模式：草稿、蓝图和编程语言。目前为止最常用的一种，是把UML当作草稿（UML as sketch），至少从我的“偏见”看是这样。这种用法中，开发人员使用UML 协助沟通系统的某些方面。在把UML当作蓝图时，你可以从正向工程或逆向工程两个方向使用草稿。正向工程在编写代码之前画UML图，而从已有代码建造UML图，目的是帮助理解代码。

--Martin Fowler《UML精粹》

名人也一样有误解



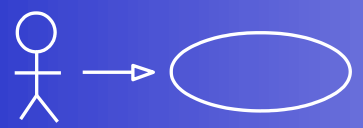
复用




```
public class Button : ButtonBase, IButtonControl {

    /// <include file='doc\Button.uex' path='docs/doc[@for="Button.dialo
    /// <devdoc>
    ///     The dialog result that will be sent to the parent dialog for
    ///     we are clicked.
    /// </devdoc>
    private DialogResult dialogResult;

    /// <include file='doc\Button.uex' path='docs/doc[@for="Button.dialo
    /// <devdoc>
    ///     For buttons whose FaltStyle = FlatStyle.Flat, this property
    ///     of the border around the button.
    /// </devdoc>
    private Size systemSize = new Size(Int32.MinValue, Int32.MinValue);

    /// <include file='doc\Button.uex' path='docs/doc[@for="Button.Butto
    /// <devdoc>
    ///    <para>
    ///       Initializes a new instance of the <see cref='System.Windo
    ///       class.
    ///    </para>
    /// </devdoc>
    public Button() : base() {
        // Buttons shouldn't respond to right clicks, so we need to do a
```

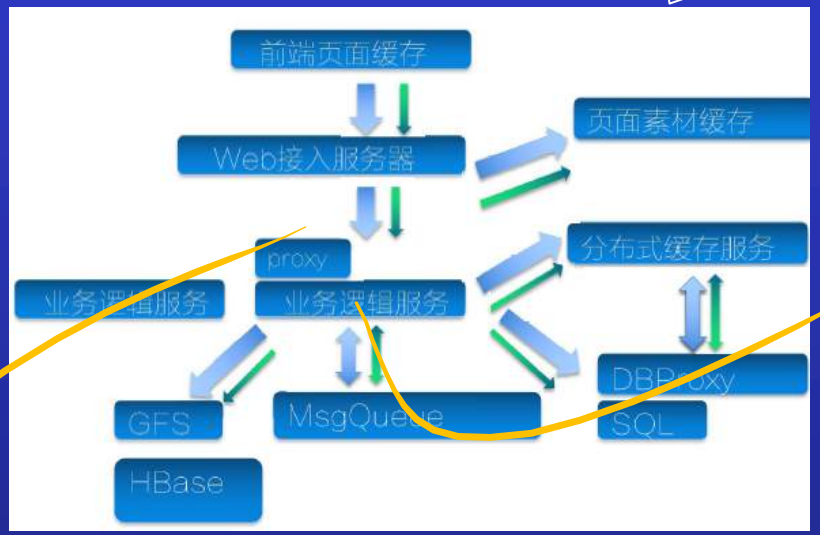
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

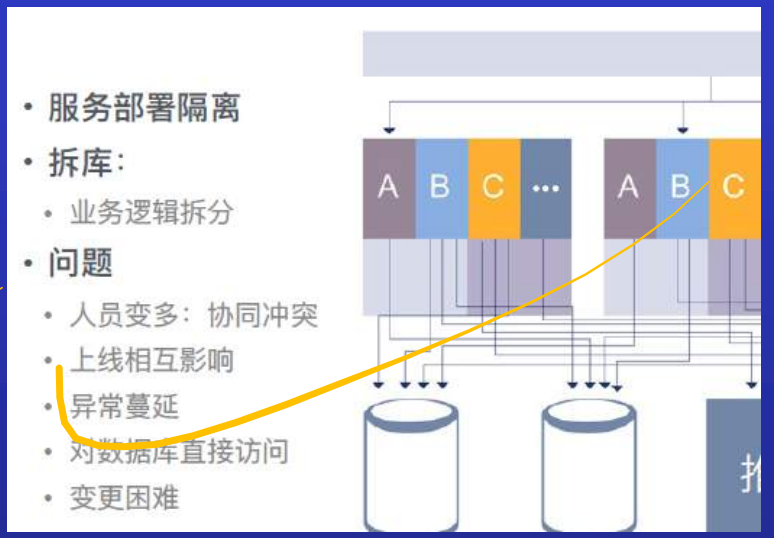
基础设施领域的复用已经做得不错



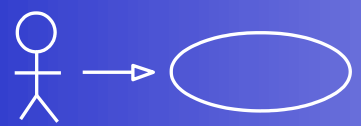
复用

2016年某技术大会的幻灯
看得出他们来自不同公司吗？



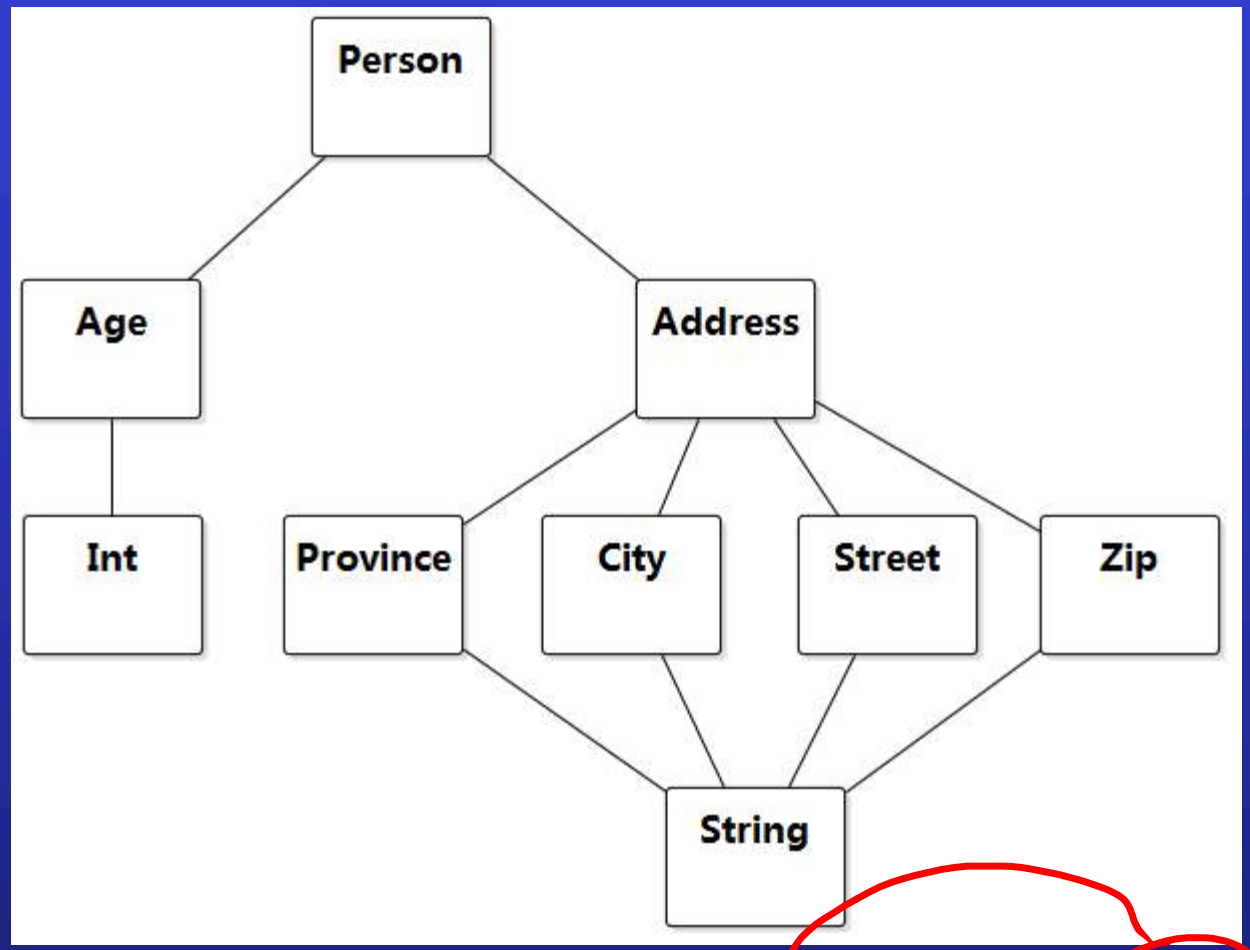


核心域的复用还未起步



复用

难，才有钱赚



领域

基础设施

基础语义

复用难度增加

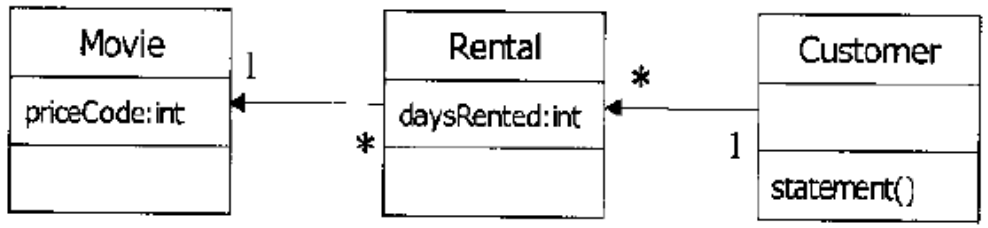
负载增加

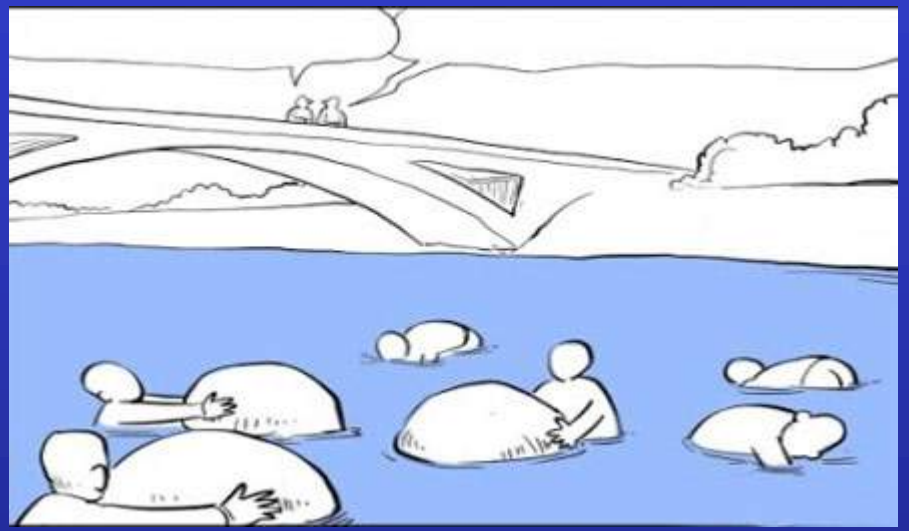
核心域的复用更难

域的分离

实例非常简单。这是一个影片出租店用的程序，计算每一位顾客的消费金额并打印报表（statement）。操作者告诉程序：顾客租了哪些影片、租期多长，程序便根据租赁时间和影片类型算出费用。影片分为三类：普通片、儿童片和新片。除了计算费用，还要为常客计算点数：点数会随着「租片种类是否为新片」而有不同。

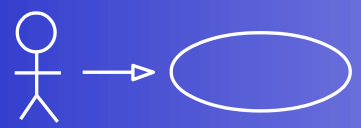
我以数个 classes 表现这个例子中的元素。图 1.1 是一张 UML class diagram（类图），用以显示这些 classes。我会逐一列出这些 classes 的代码。





简单设计？敏捷设计？

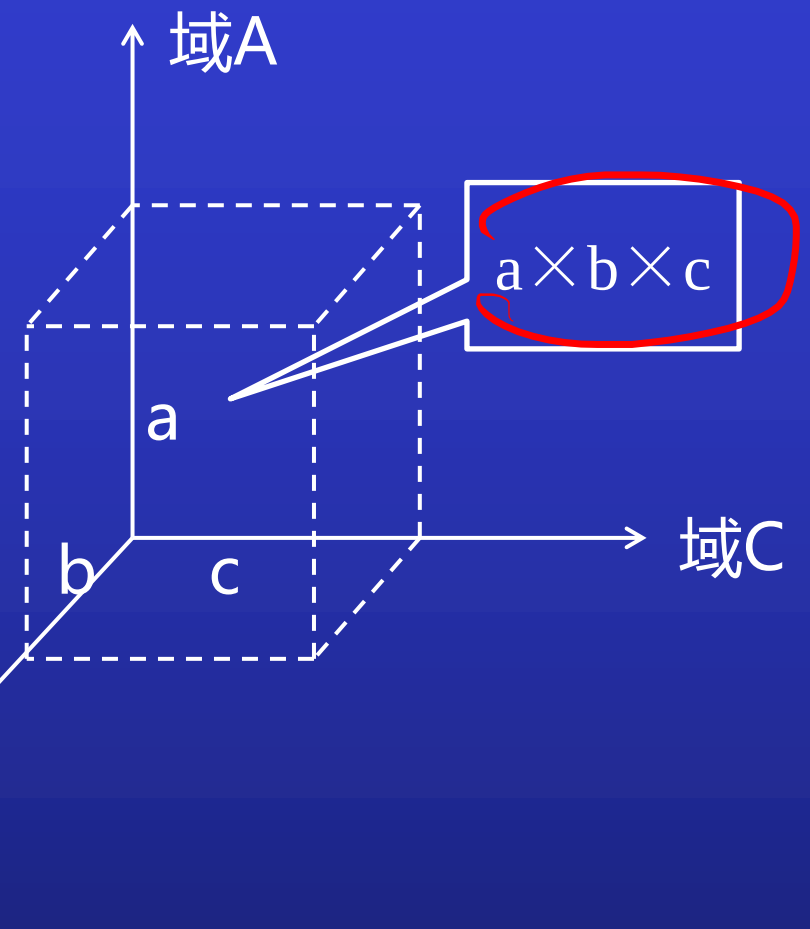
硬要摸着石头过河！



域的分离

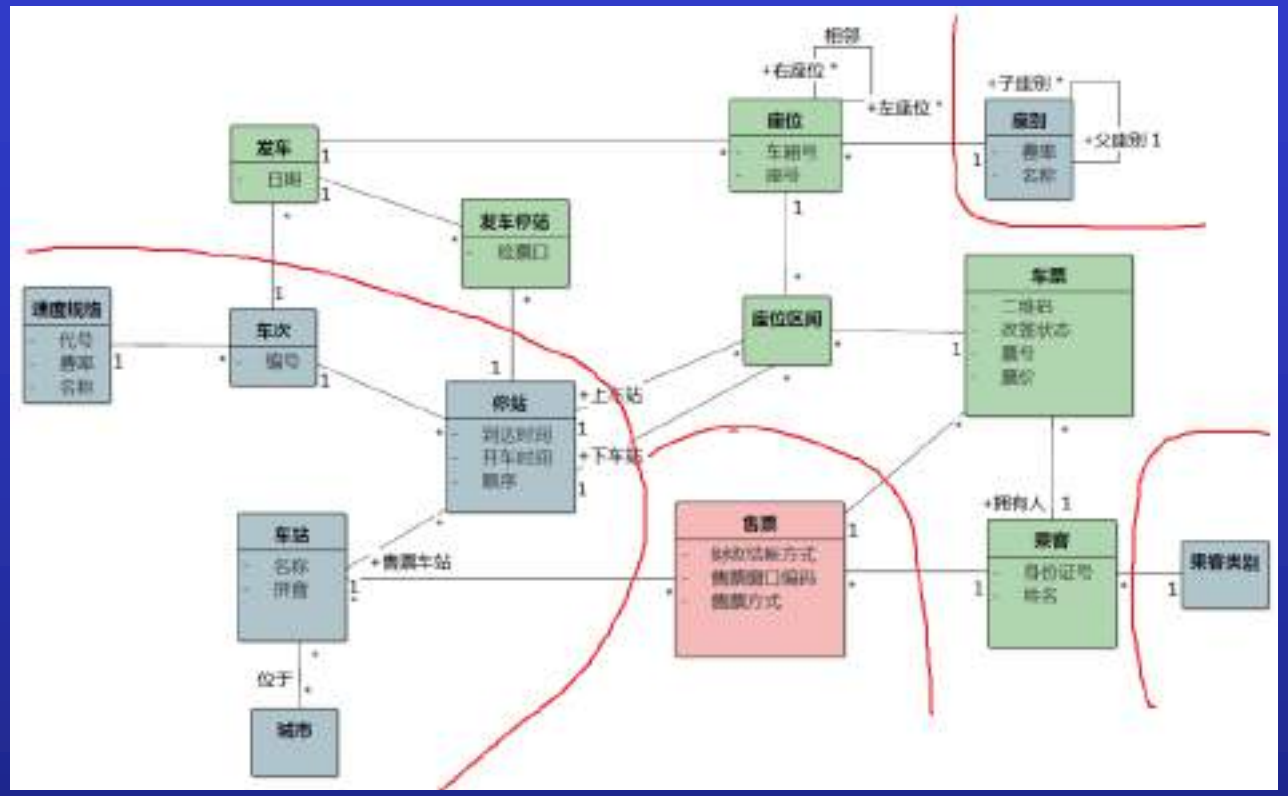
- 跨抽象级别的映射绝不孤立
- 过早混杂，a+b+c→ a×b×c
- 无法深入思考核心域问题

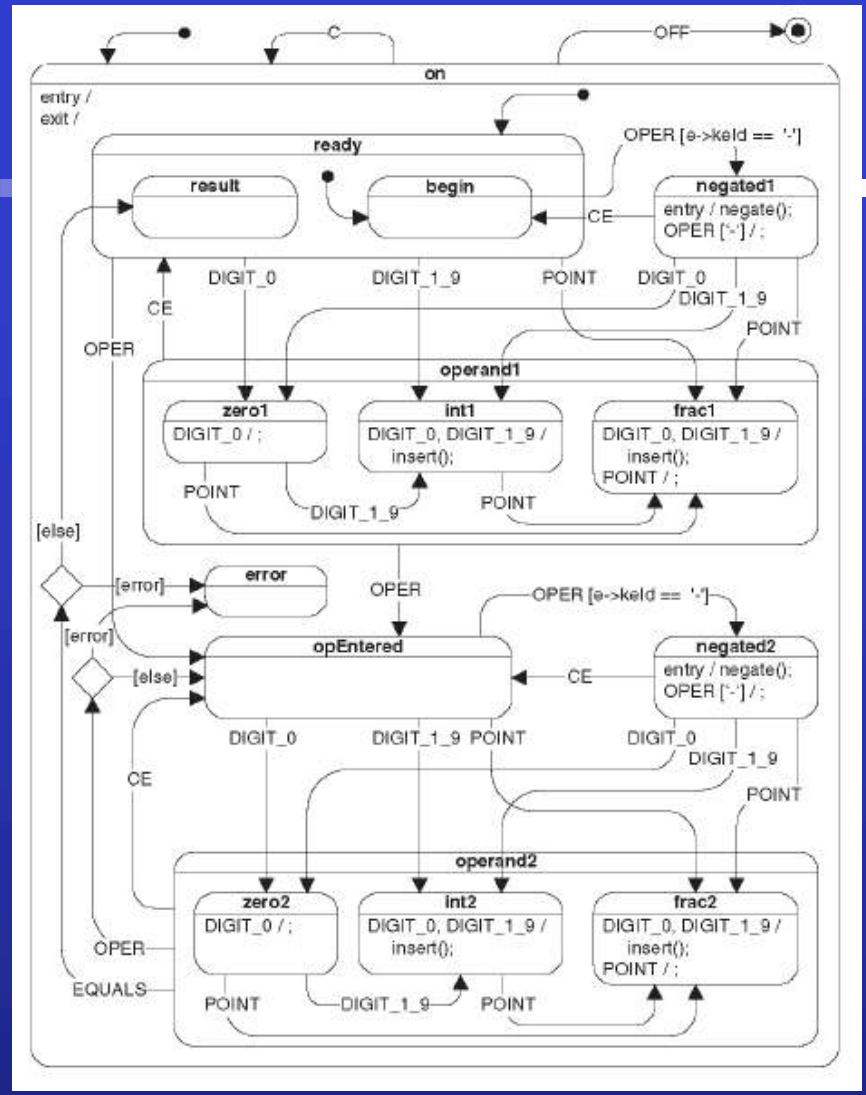
- 分离核心域和非核心域
- 找到不同知识域之间的映射规律



分离核心域和非核心域

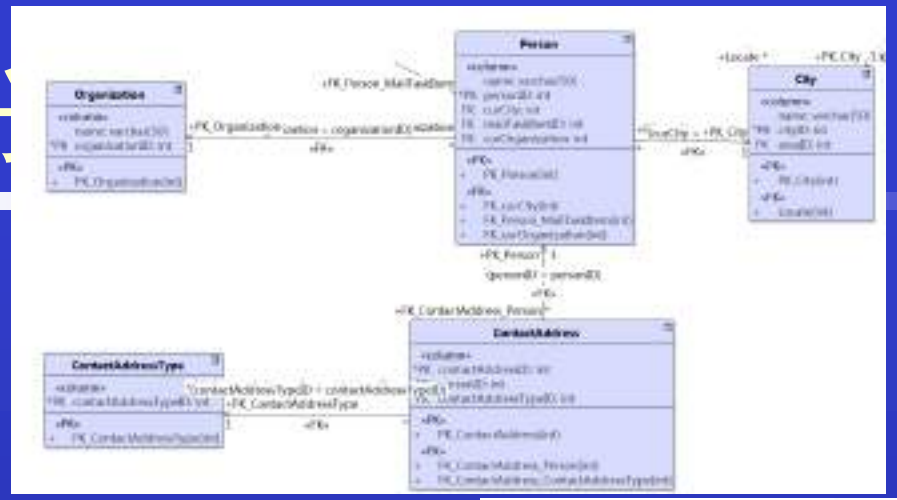
域的分离



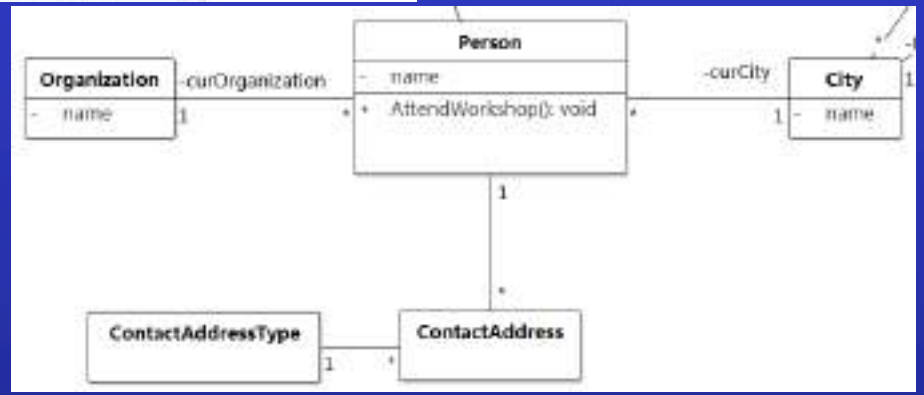


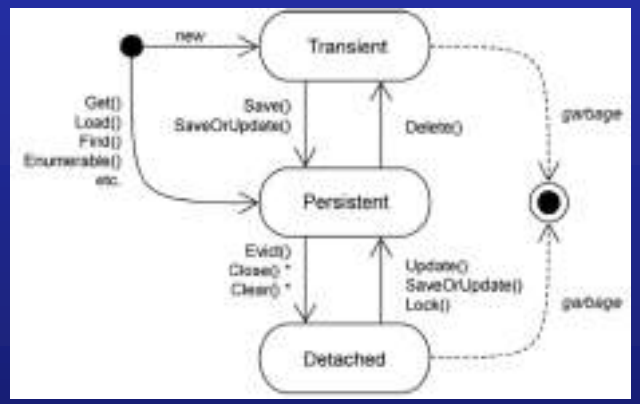
分离以深入思考

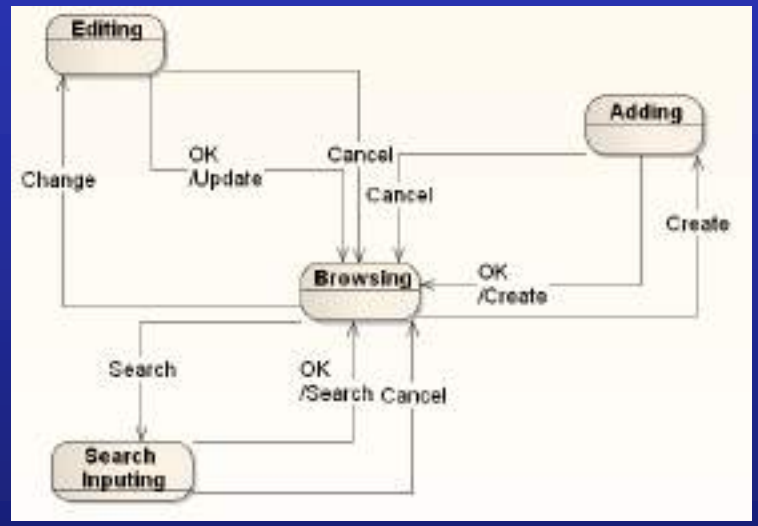
域的分



```
16 public class Person {
17
18     private string name;
19     private Organization curOrganization;
20     private City curCity;
21     private List<ContactAddress> m_ContactAddress;
22
23     public Person(){
24
25     }
26
27     ~Person(){
28
29     }
30
31     public void AttendWorkshop(){
32
```







映射

练习

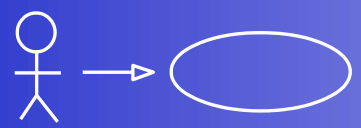
如果有人说“Linux代码超过千万行，也没有用UML建模、面向对象之类的啊？"，应该怎么回答比较好？

A）人和人不一样，搞操作系统的是天才，不能比。

B）操作系统领域的负载比较低。

C）其实是用了UML建模的，只不过没有公布出来。

D）因为Linux用了敏捷过程，敏捷以后就不用建模了。



练习

《***》杂志曾经刊登一篇译文，作者在白板上画了一个类图，然后开始掰着指头数这个类图缺什么，“没考虑到持久化”，“没考虑到对象的创建”……然后得出结论：画这个类图不如直接编码。请根据以上知识评价以上观点。

A）不同意。作者不了解核心域和非核心域分离的重要。

B）不同意。这个图会越来越细，逐渐添加作者认为缺少的那些东西。

C）同意。Talk is cheap. Show me the code.

D）同意。代码才是最终结果，其他事情都是浪费。

练习

以下是网络上较流行的描述"最小可行产品"（minimum viable product）开发过程的图（图片来自http://www.nickmilton.com/2015/07/lean-km-and-minimum-viable-product.html）。

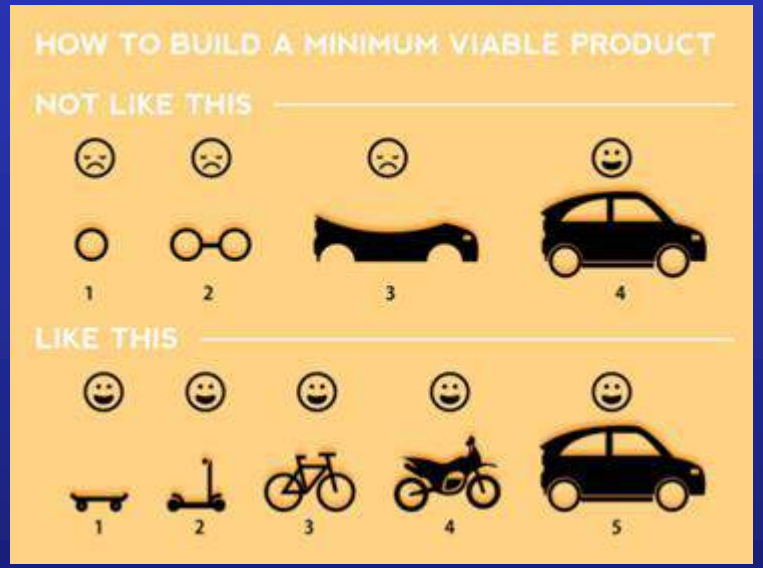
从以上知识出发，该图作者可能存在的认识上的最大错误是：

A) 认为造汽车应该先从轮子造起。

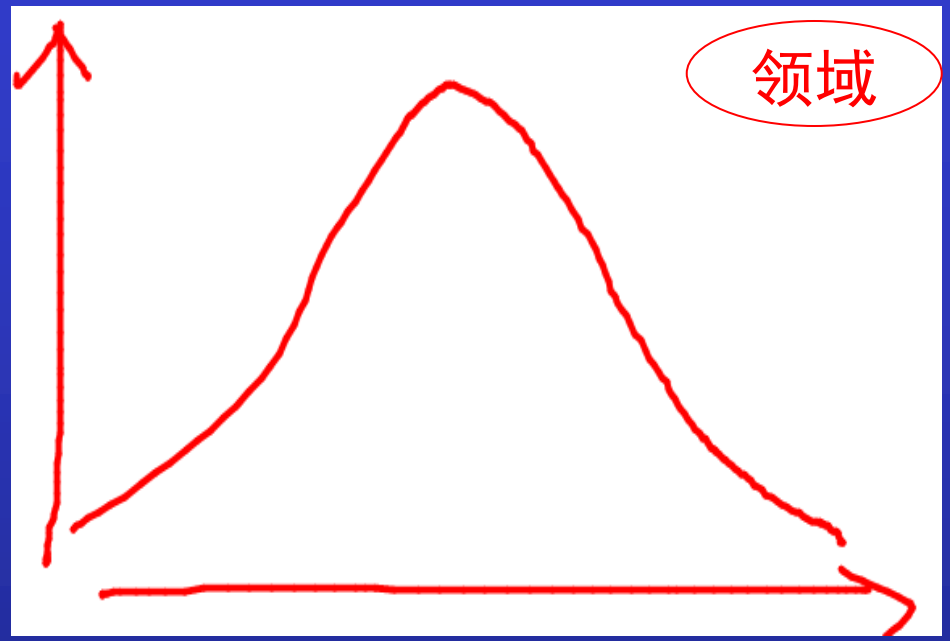
B) 认为造滑板车一定比造汽车简单。

C) 认为应该小步改进，先给客户一个滑板车也是改进。

D) 认为客户目前停止不动，随便给个什么车都是救命。



核心域建模



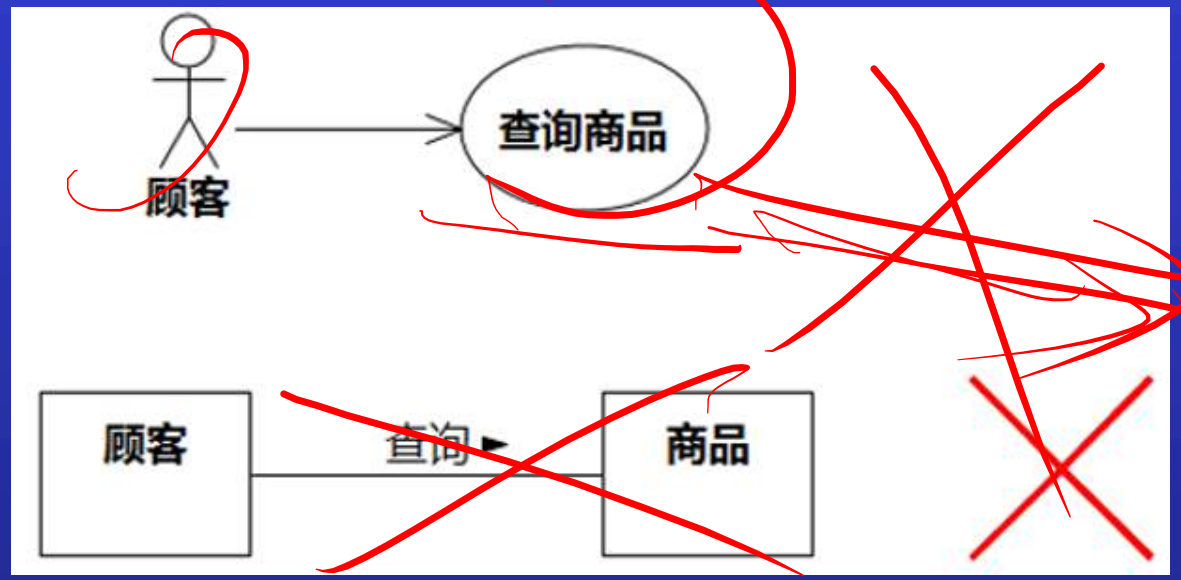
文章本天成
妙手偶得之

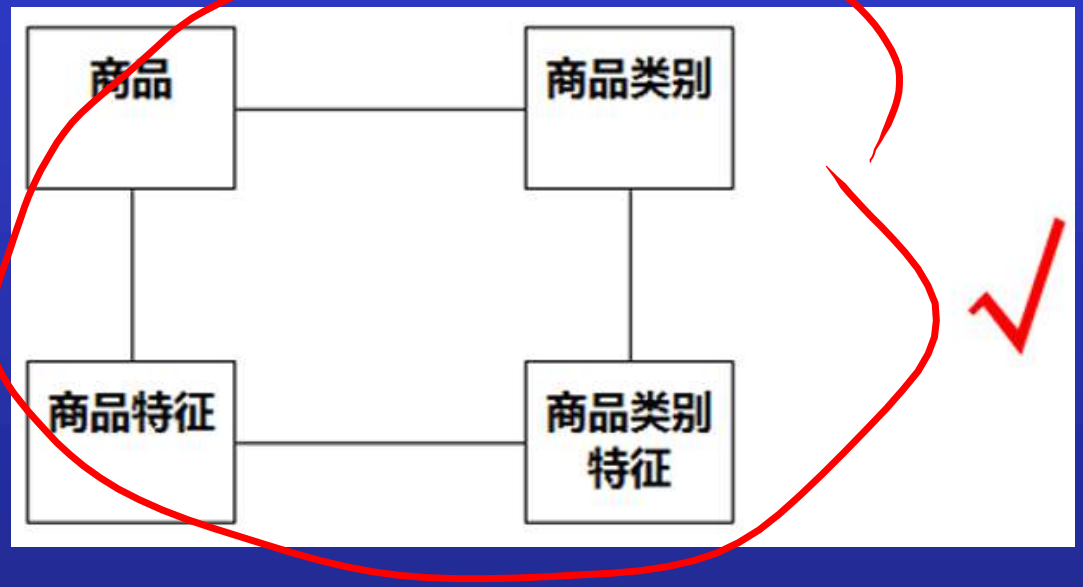
不是“设计”
而是“描述”

体现领域的真正味道
扭曲的映射难以应变

恰当抽象--抓住领域内涵应对需求变更

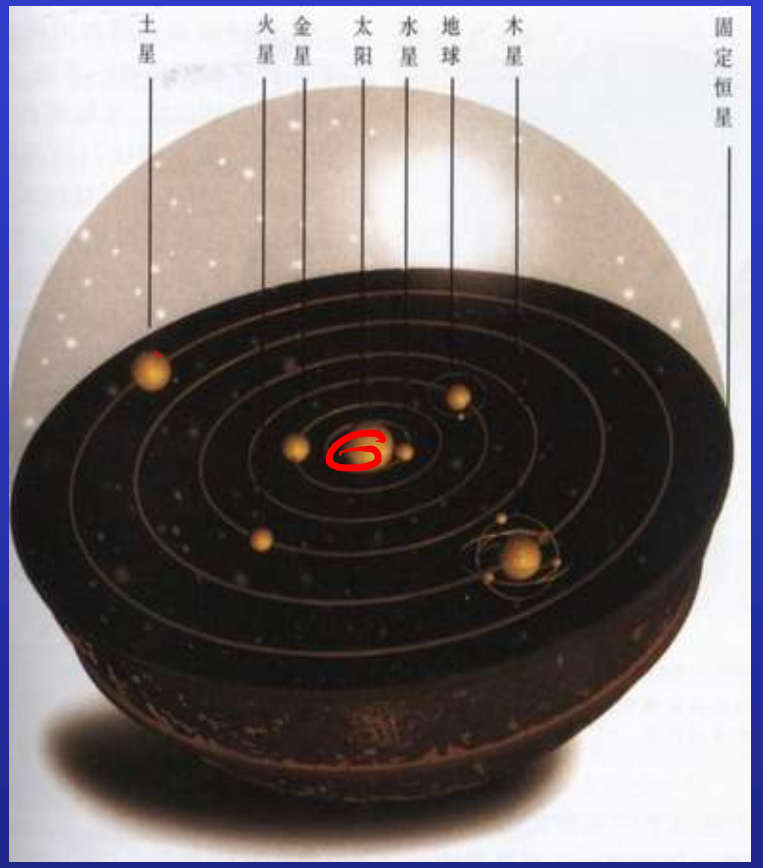
核心域建模





错误：照猫画虎，类图像用例图

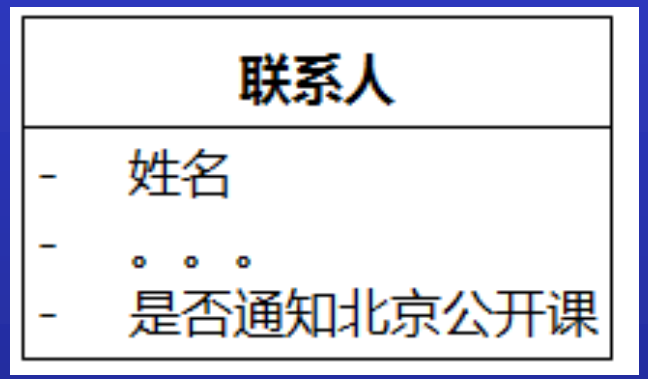
核心域建模

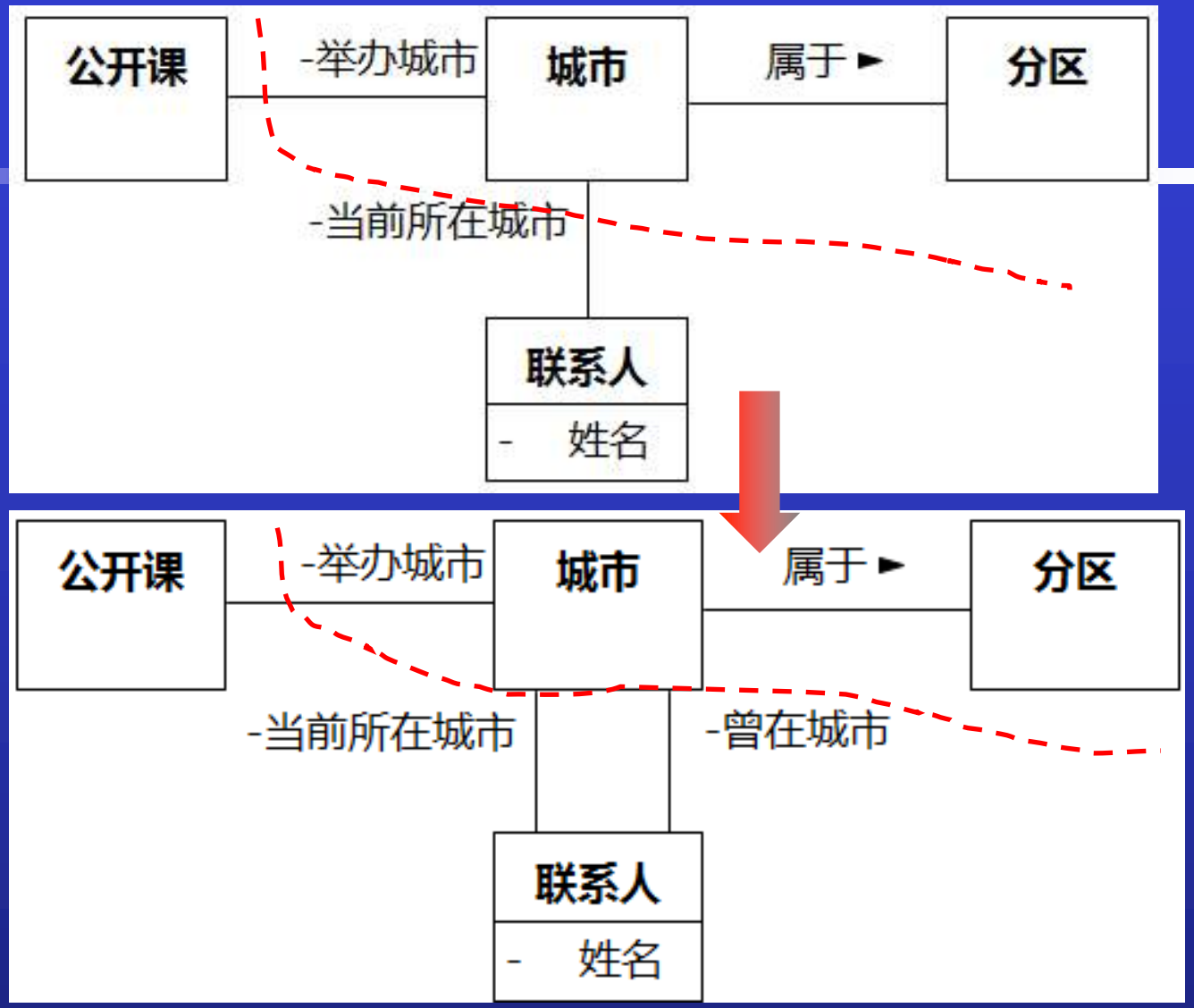


越来越复杂时
应该寻找更高级的规律

寻找现象背后的知识

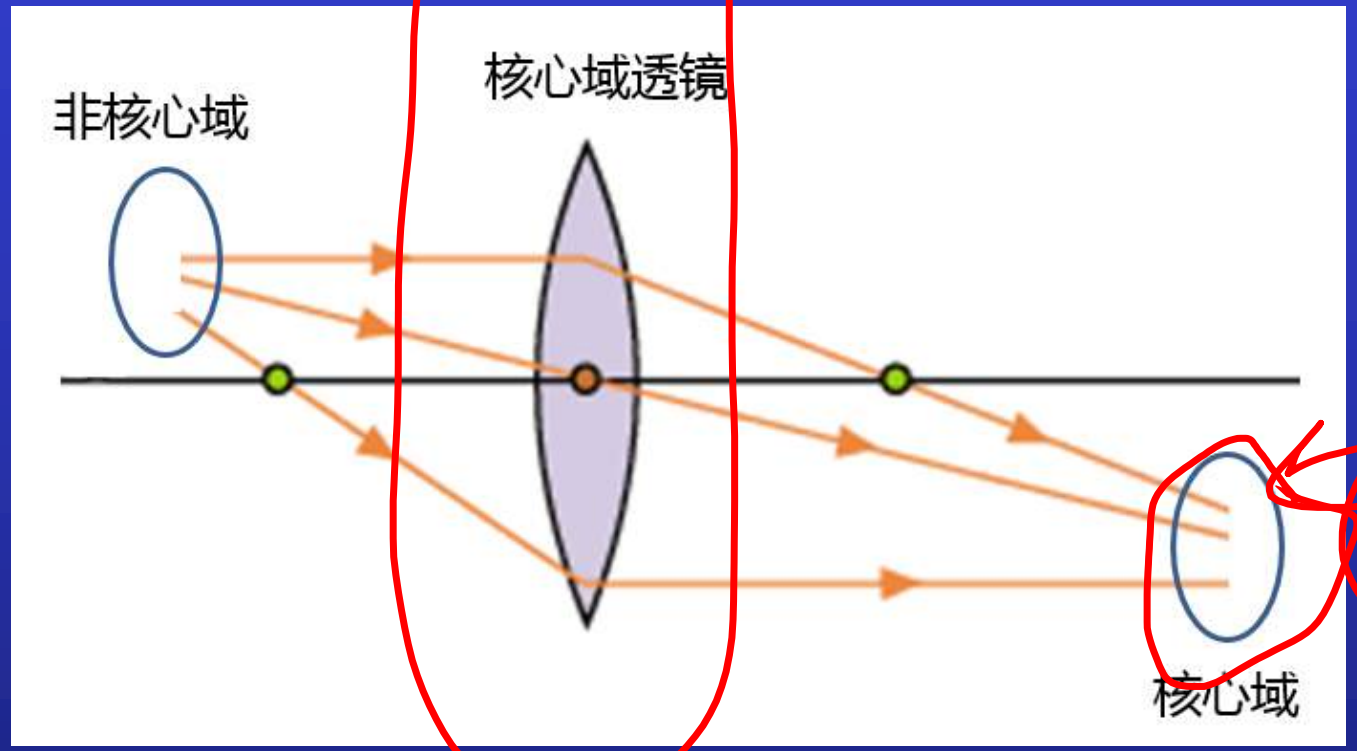
核心域建模





探索深层模型

核心域建模



概念	基础设施域	社交域
电子邮件	SMTP、POP3/IMAP	关系链、信息载体

核心域透镜

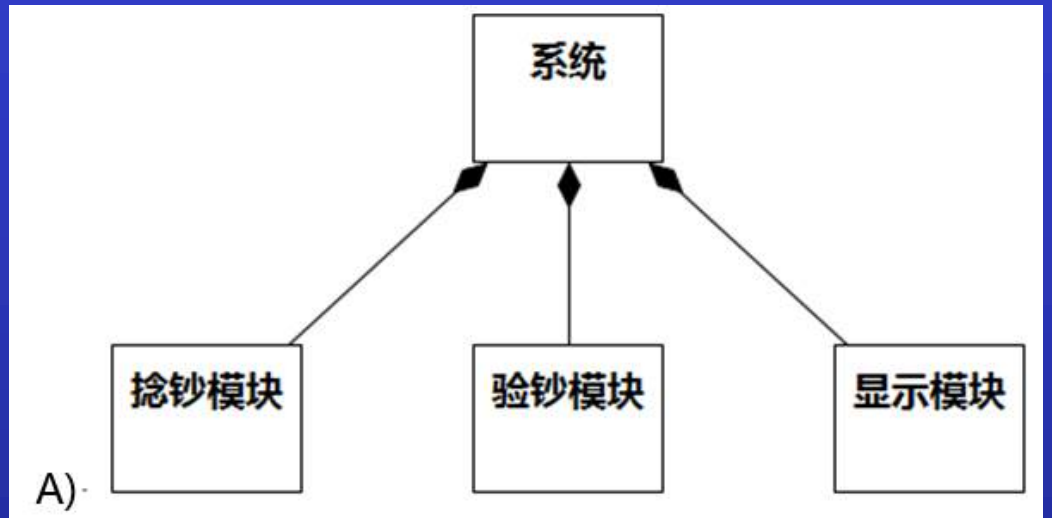
核心域建模

原描述	映射后的核心域概念	原描述过去变体	原描述将来变体（猜想）
Powerpoint	演示工具	黑板、玻璃幻灯片、赛璐珞幻灯片	全息
检查IP地址	检查重复听课学生	看脸、看签名	检查新的协议地址 检查大脑芯片标识
点击“开始”按钮	开始考试	观察到考生开始书写	?
向数据库“答题”表添加一条答题记录	答题	在答卷上涂黑一格	?

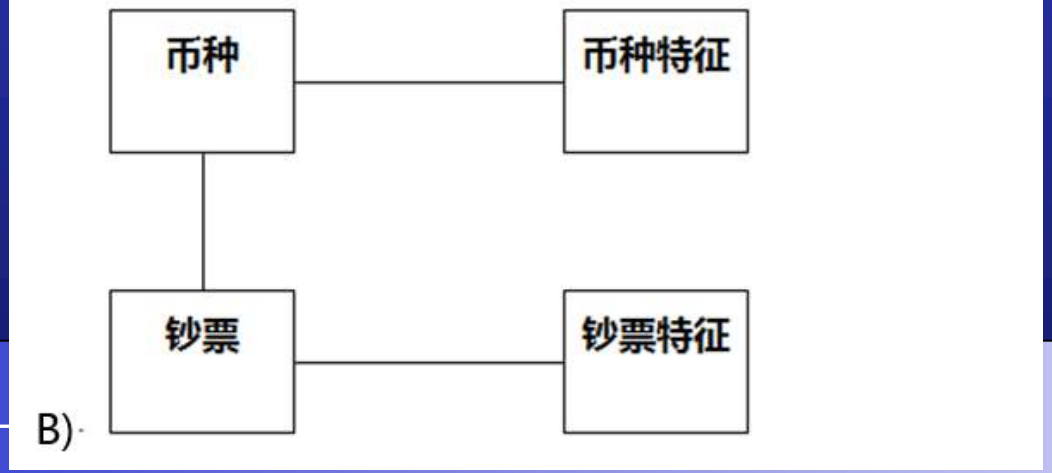
练习

- 要实现验钞机的“验钞”功能，恰当的抽象是

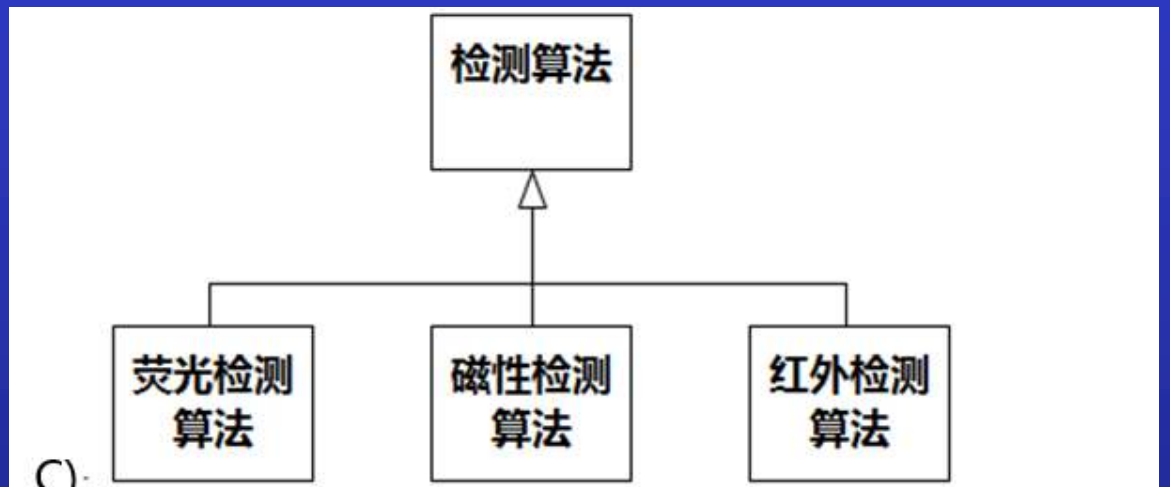
A)



B)

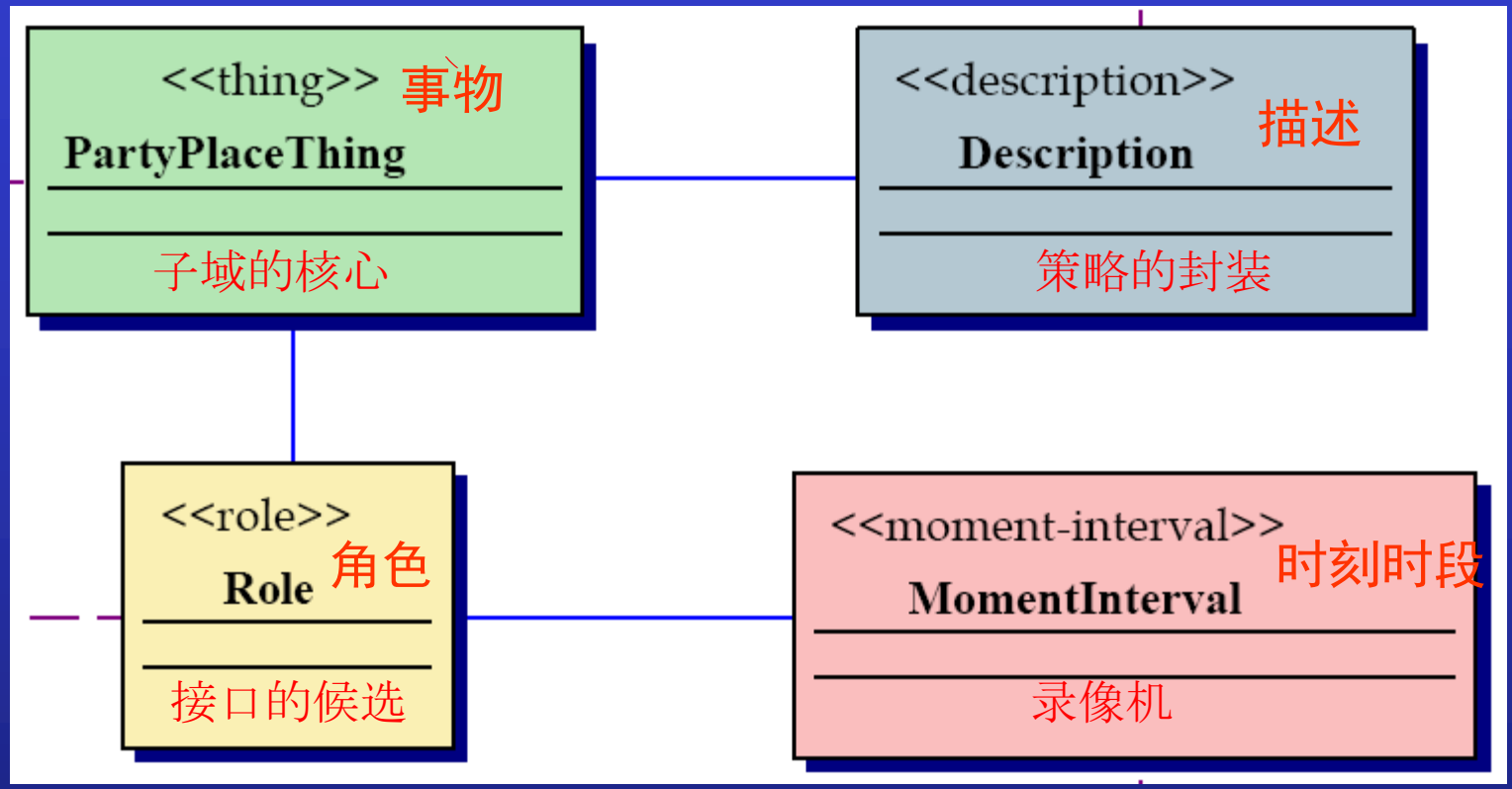


C)



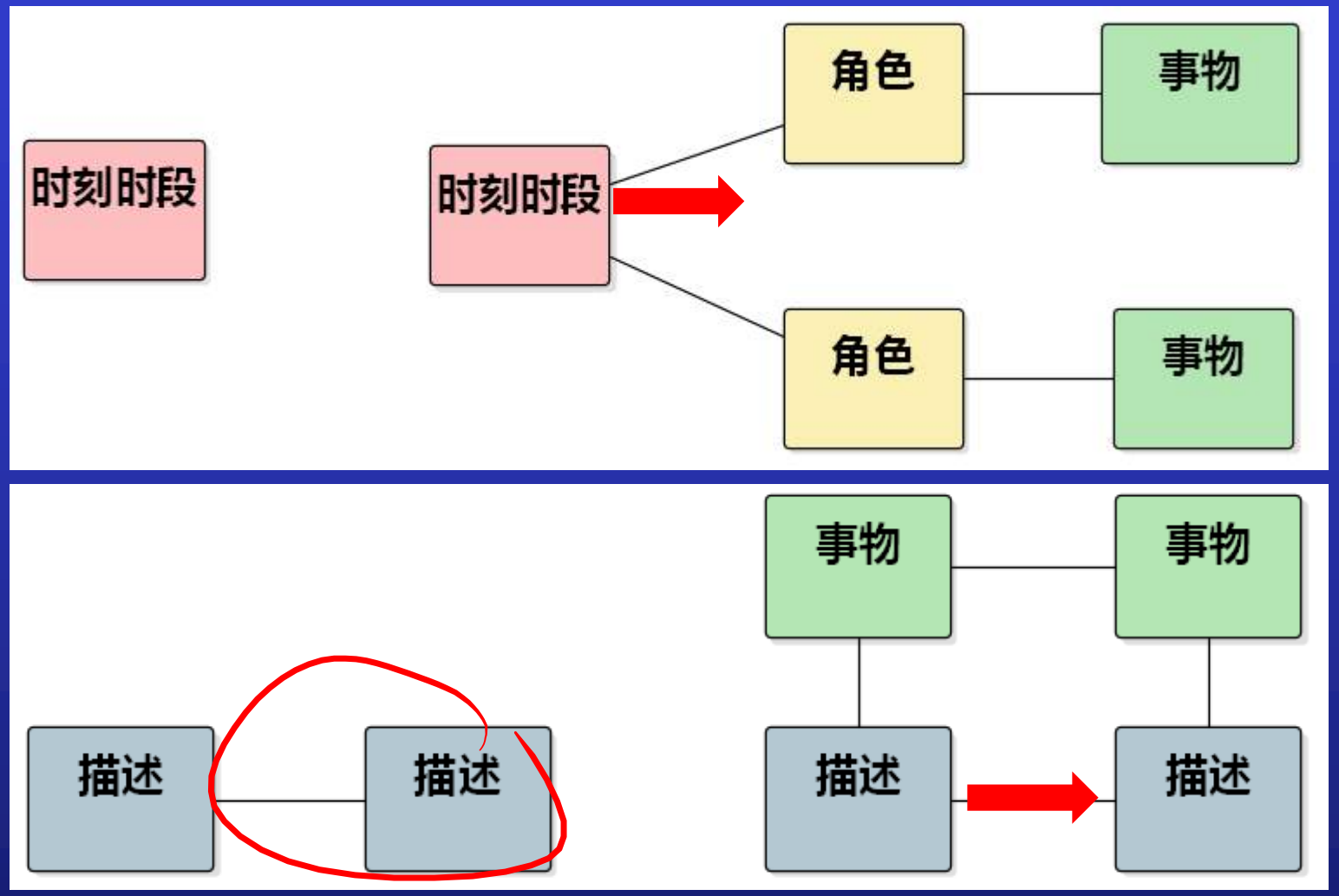
D)

借助颜色



彩色建模架构型（archetype）

借助颜色



从时刻时段开始

从描述开始

借助模式

By Purpose		Creational	Structural	Behavioral
By Scope	Class	• Factory Method	• Adapter (class)	• Interpreter • Template Method
	Object	• Abstract Factory • Builder • Prototype • Singleton	• Adapter (object) • Bridge • Composite • Decorator • Façade • Flyweight • Proxy	• Chain of Responsibility • Command • Iterator • Mediator • Memento • Observer • State • Strategy • Visitor

非最有用

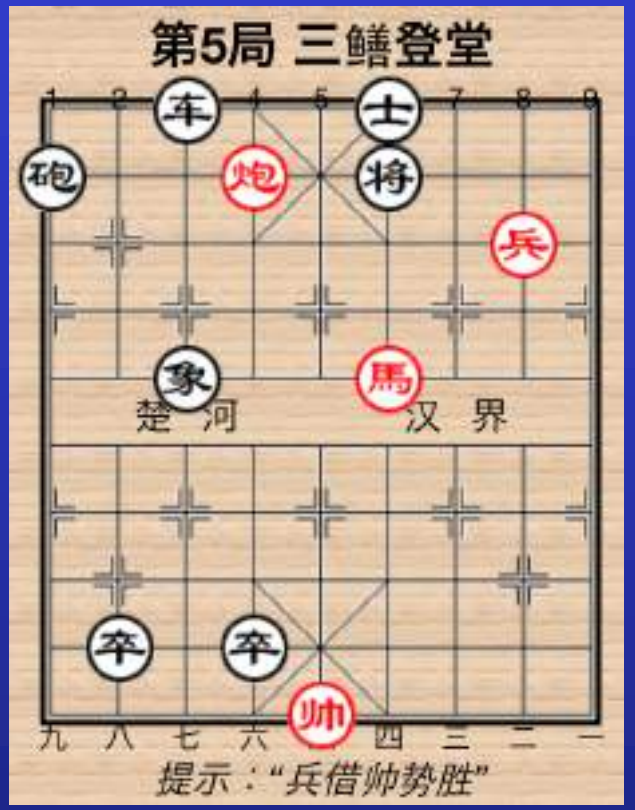
非最本质

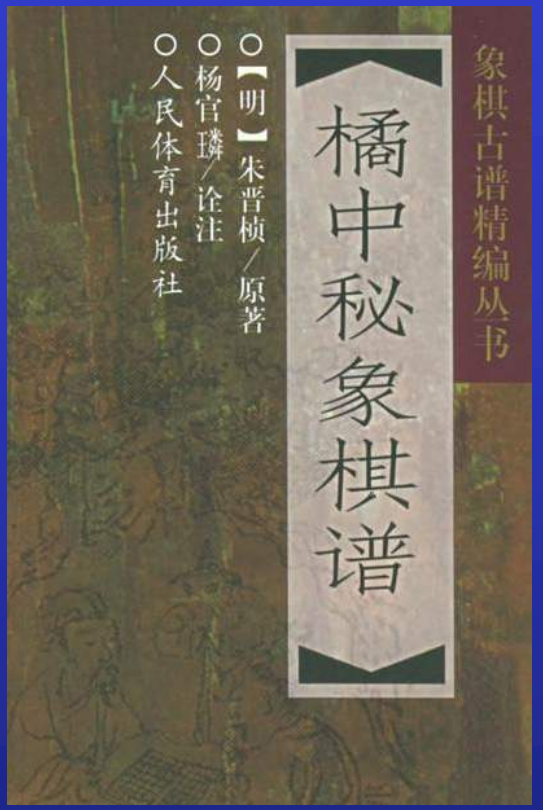
较早（非最早）成书

书因GoF最出名

模式不仅是GoF23模式！

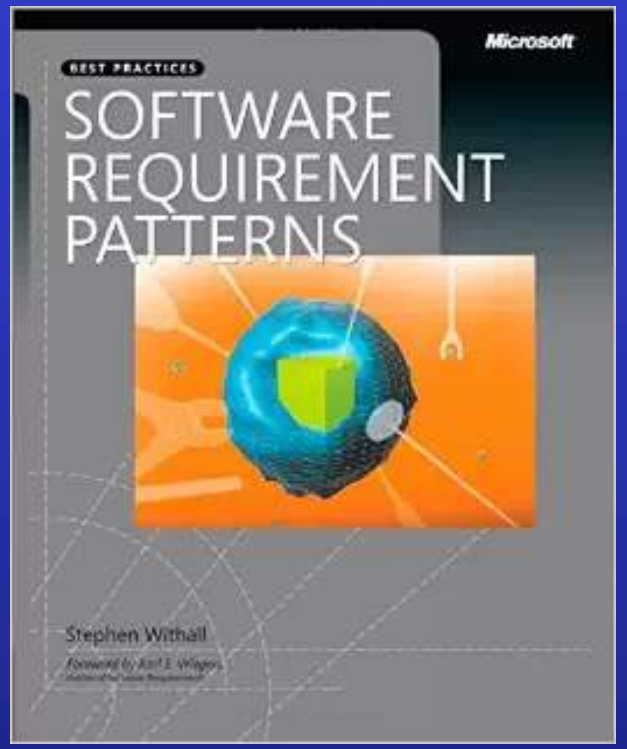
借助模式





最早的定式汇编而已

借助模式

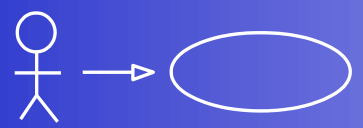


Template(s)

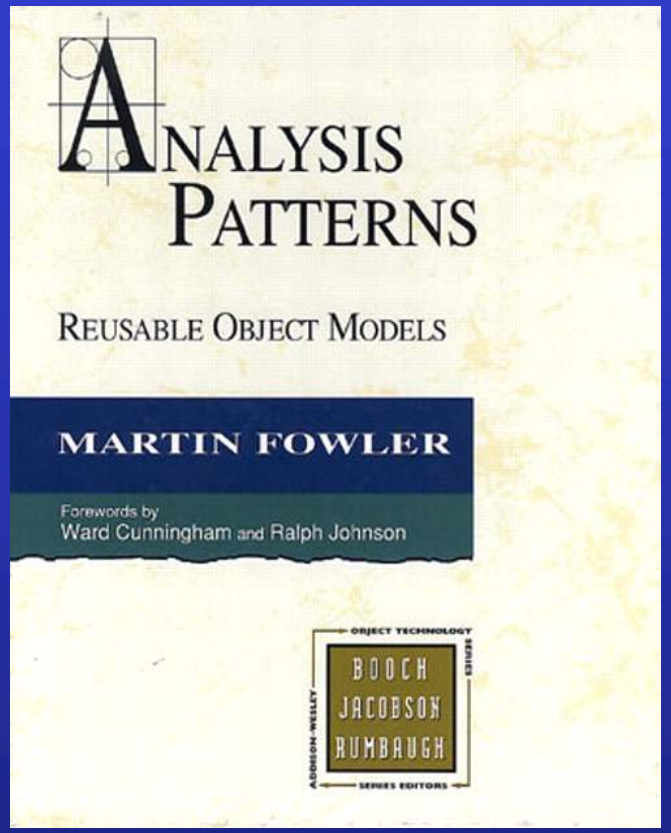
Open table as spreadsheet

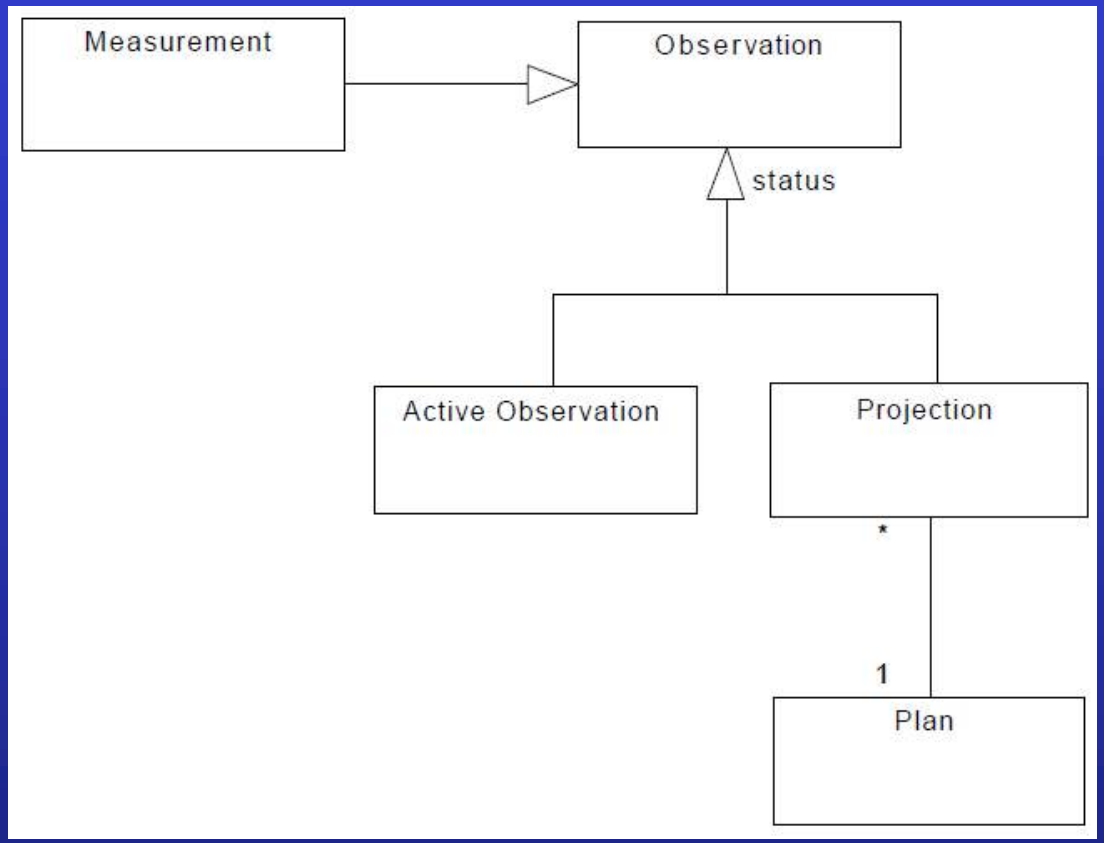
Summary	Definition
«*Operation type*» response time	Each «*Operation type*» shall have a response time of no more than «*Tolerable length of time*» fro «*Timing boundary end*» [when using «*Indicative hardware set-up*»]. This figure is based on «*Justi* [This requirement does not apply to «*Exceptional cases*».] [«*High load caveat*».] [The motivation for this requirement is «*Motivation*».]

需求模式



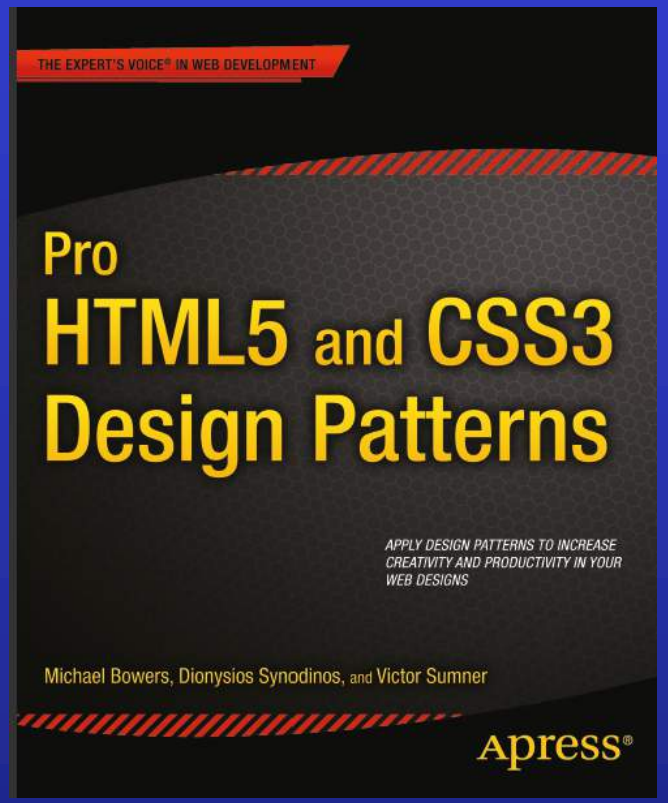
借助模式

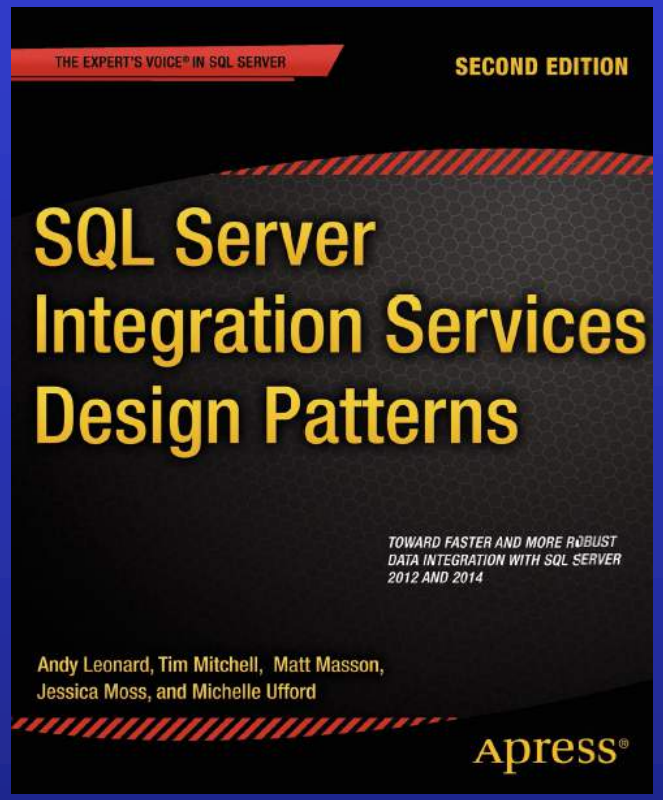

ANALYSIS PATTERNS
REUSABLE OBJECT MODELS
MARTIN FOWLER
Forewords by
Ward Cunningham and Ralph Johnson
OBJECT TECHNOLOGY SERIES
BOOCH
JACOBSON
RUMBAUGH
ADDISON-WESLEY
SERIES EDITORS

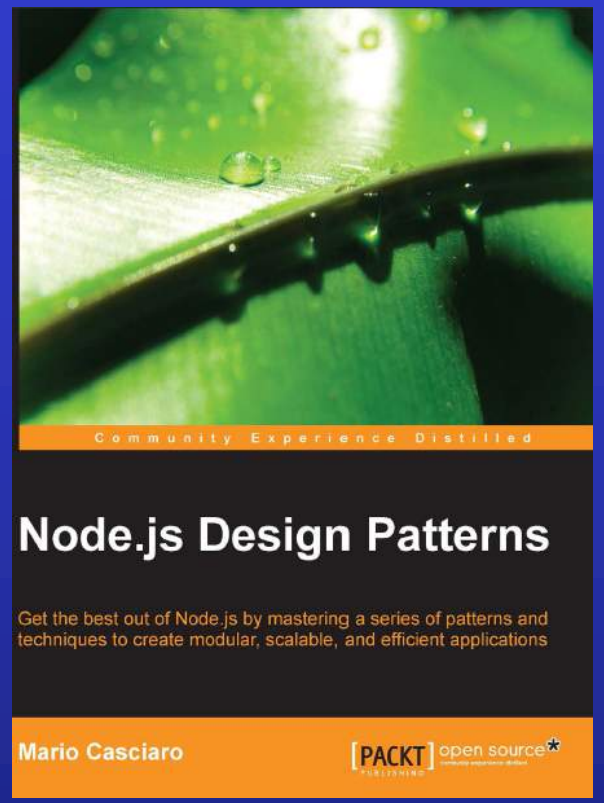

Measurement
Observation
status
Active Observation
Projection
*
1
Plan

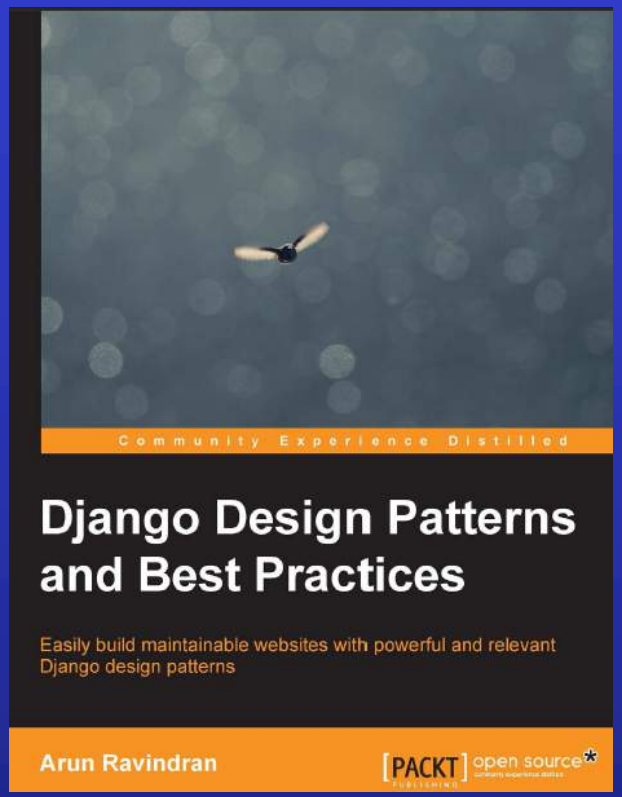
分析模式

借助模式

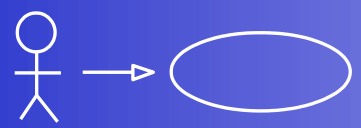

THE EXPERT'S VOICE® IN WEB DEVELOPMENT
Pro
HTML5 and CSS3
Design Patterns
APPLY DESIGN PATTERNS TO INCREASE CREATIVITY AND PRODUCTIVITY IN YOUR WEB DESIGNS
Michael Bowers, Dionysios Synodinos, and Victor Sumner
Apress®


THE EXPERT'S VOICE® IN SQL SERVER
SECOND EDITION
SQL Server
Integration Services
Design Patterns
TOWARD FASTER AND MORE ROBUST DATA INTEGRATION WITH SQL SERVER 2012 AND 2014
Andy Leonard, Tim Mitchell, Matt Masson,
Jessica Moss, and Michelle Ufford
Apress®


Community Experience Distilled
Node.js Design Patterns
Get the best out of Node.js by mastering a series of patterns and techniques to create modular, scalable, and efficient applications
Mario Casciaro
[PACKT] open source*


Community Experience Distilled
Django Design Patterns
and Best Practices
Easily build maintainable websites with powerful and relevant Django design patterns
Arun Ravindran
[PACKT] open source*

设计模式



借助模式

POSA系列（1-5）　PLoPD系列（1-5）

模式汇编

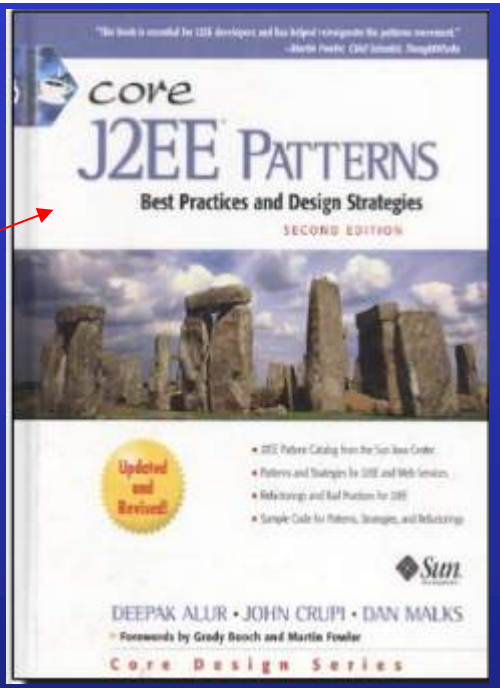
借助模式

- 系统所聚焦核心域的分析模式
- 系统所使用非核心域的设计模式

Composing analysis patterns to build complex models: Flight reservation

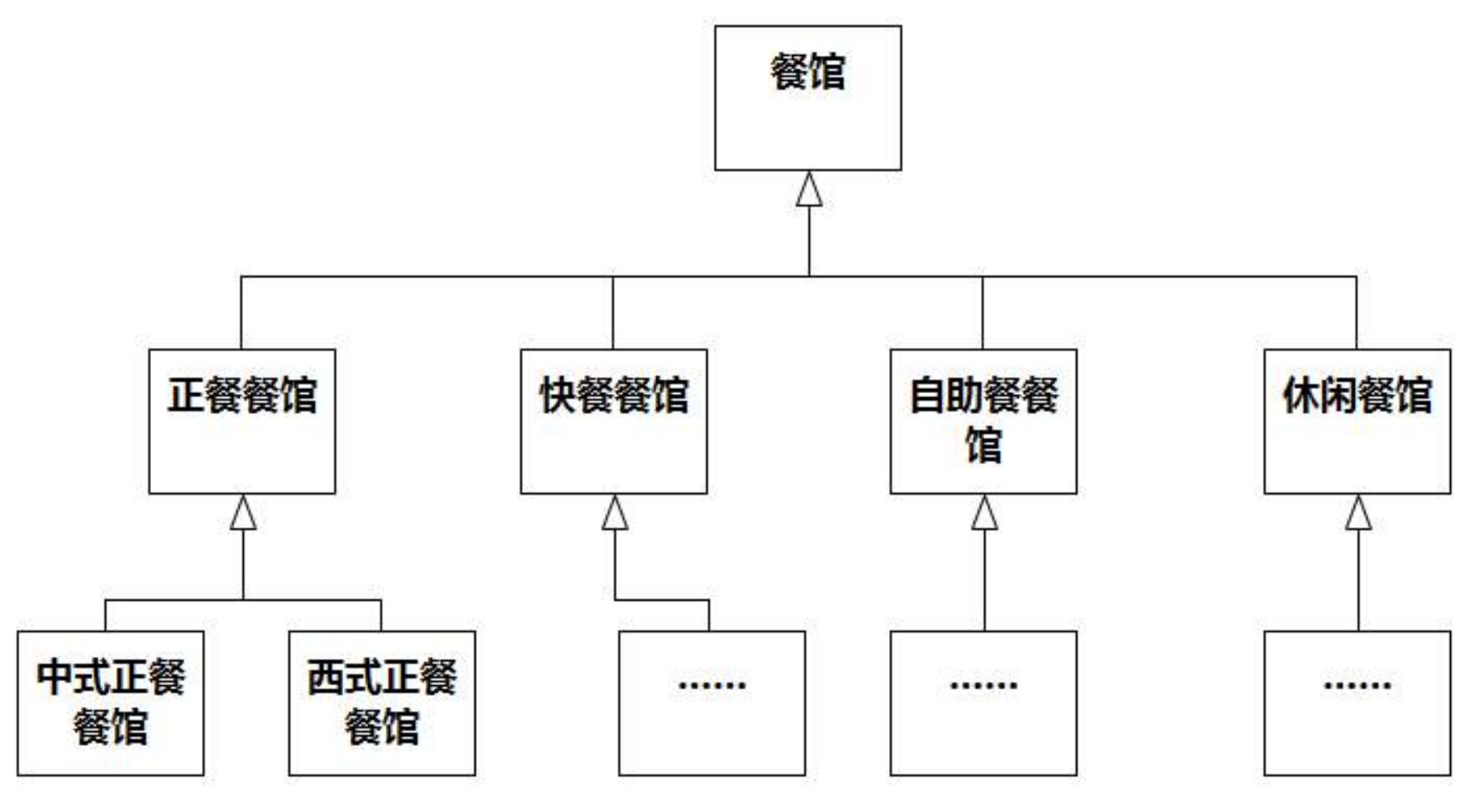
在航空客运领域，收益管理是指航空公司通过预测和优化等科学手段把产品按不同的价格适时地卖给不同类型的旅客，使每一航班的每一航段的每一座位以最好的价格出售，从而获得最大的利润。它是以数学、经济学、统计和运筹学原理为基础而建立起来的科学，是正确预测市场、优化子舱位结构、动态控制票价的手段，能够有效的控制航空公司运行成本。

J2EE (Java 2 Enterprise Edition)是一种利用 Java 2 平台来简化企业解决方案的开发、部署和管理相关的复杂问题的体系结构。J2EE 体系结构提供中间层集成框架用来满足无需太多费用而又需要高可用性、高可靠性以及可扩展性的应用需求。设计模式描述了通用的、简单的和可重用的解决方案的核心，在基于 J2EE 体系的应用系统的开

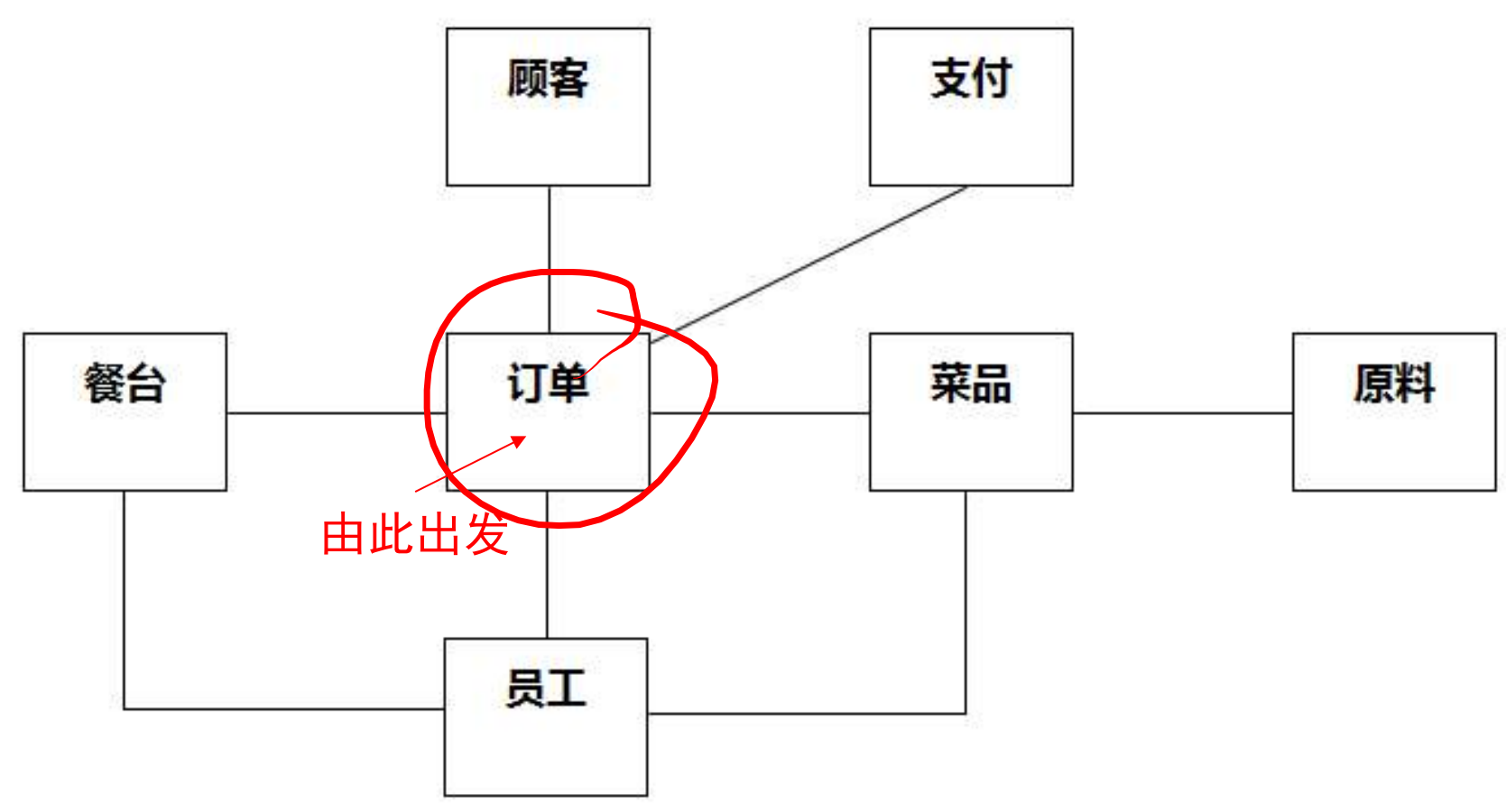


重点应学习的模式

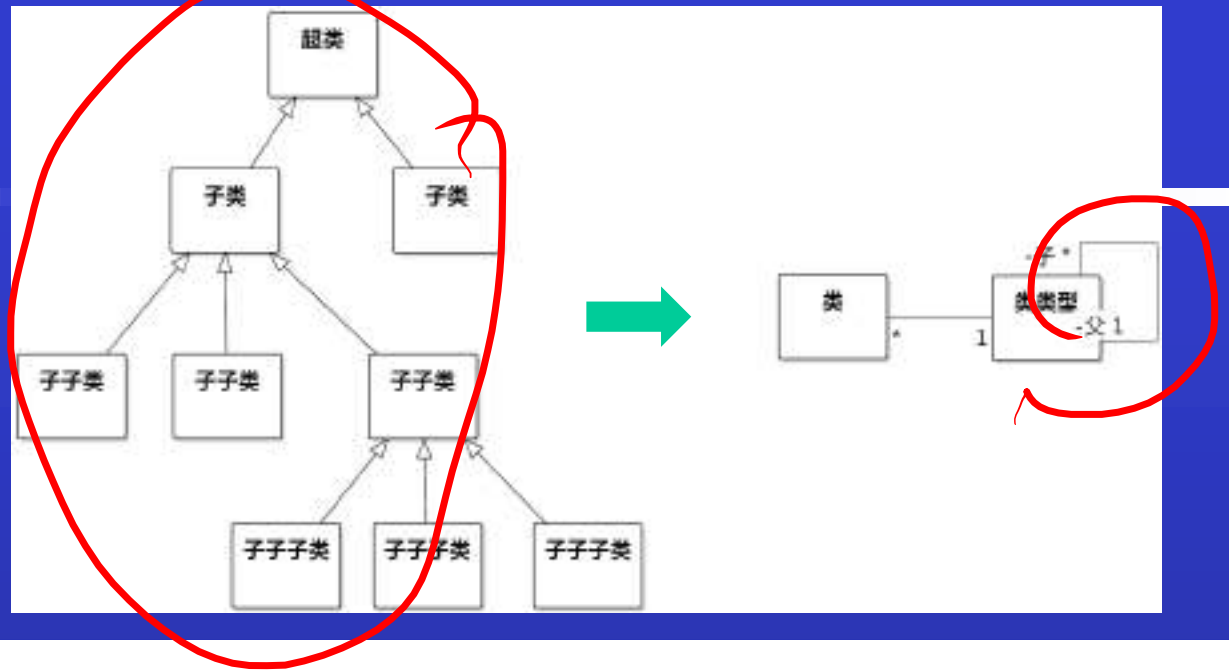
餐馆实例

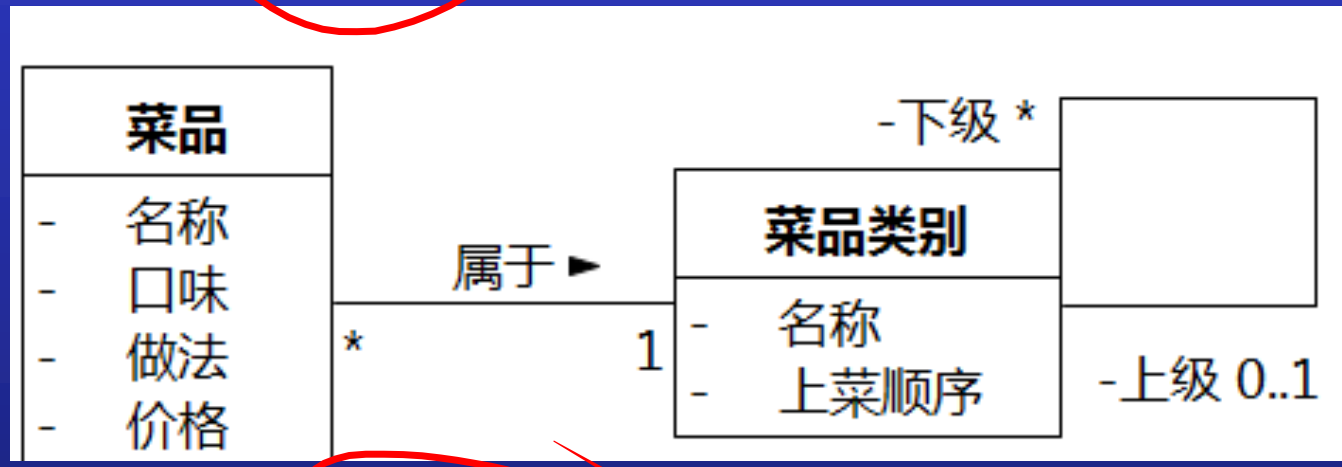


要素



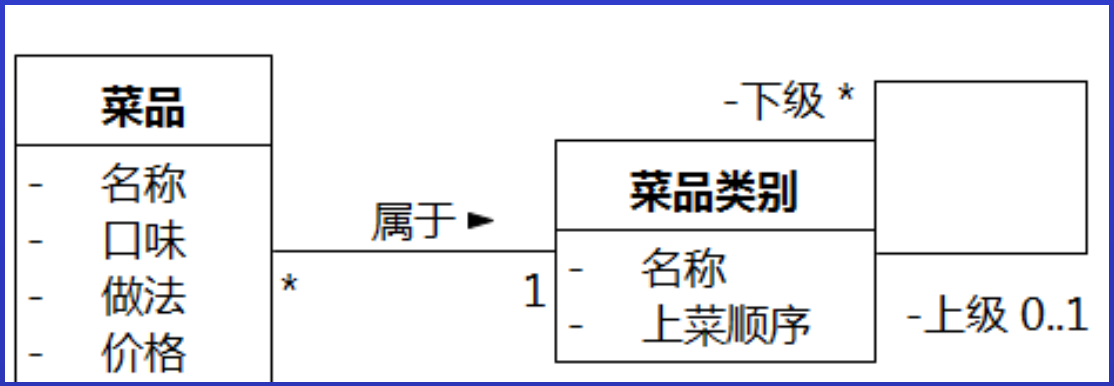
菜品

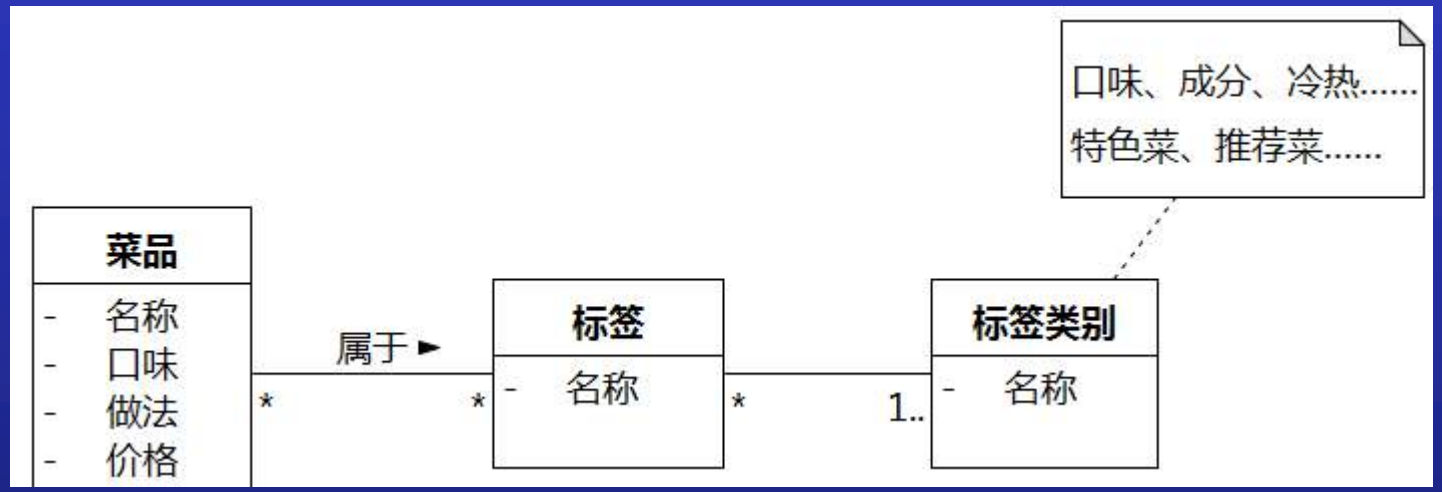




菜品和类别-应用分析模式-物品

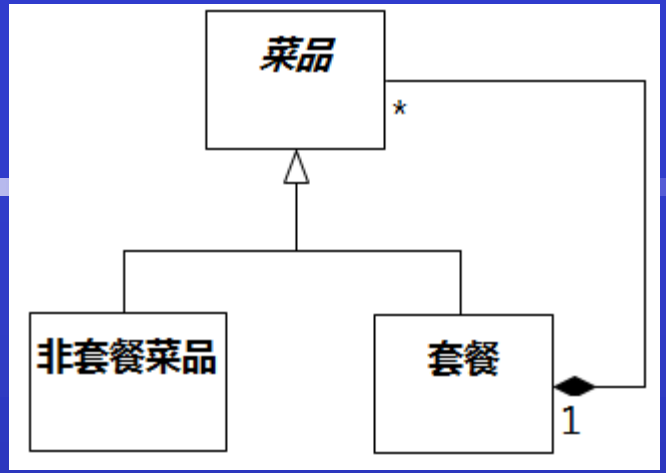
菜品

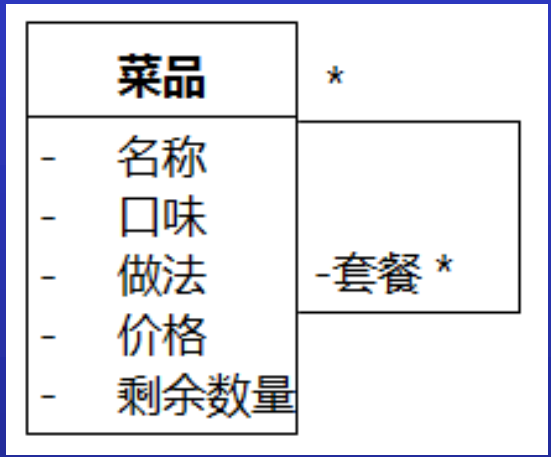

菜品
- 名称
- 口味
- 做法
- 价格
属于
*
1
-下级 *
菜品类别
- 名称
- 上菜顺序
-上级 0..1


口味、成分、冷热.....
特色菜、推荐菜.....
菜品
- 名称
- 口味
- 做法
- 价格
属于
*
*
标签
- 名称
*
1..
标签类别
- 名称

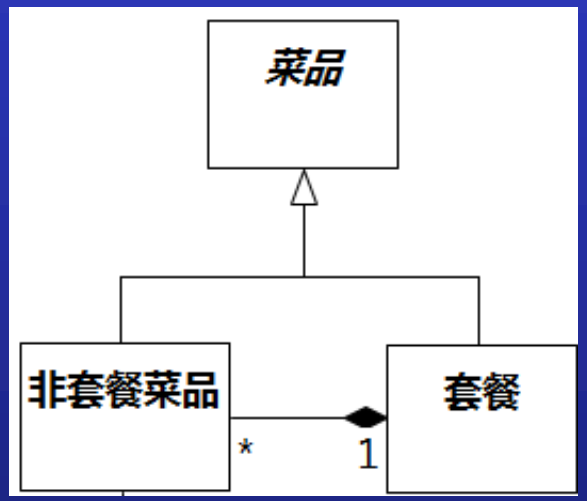
标签取代类别

菜品



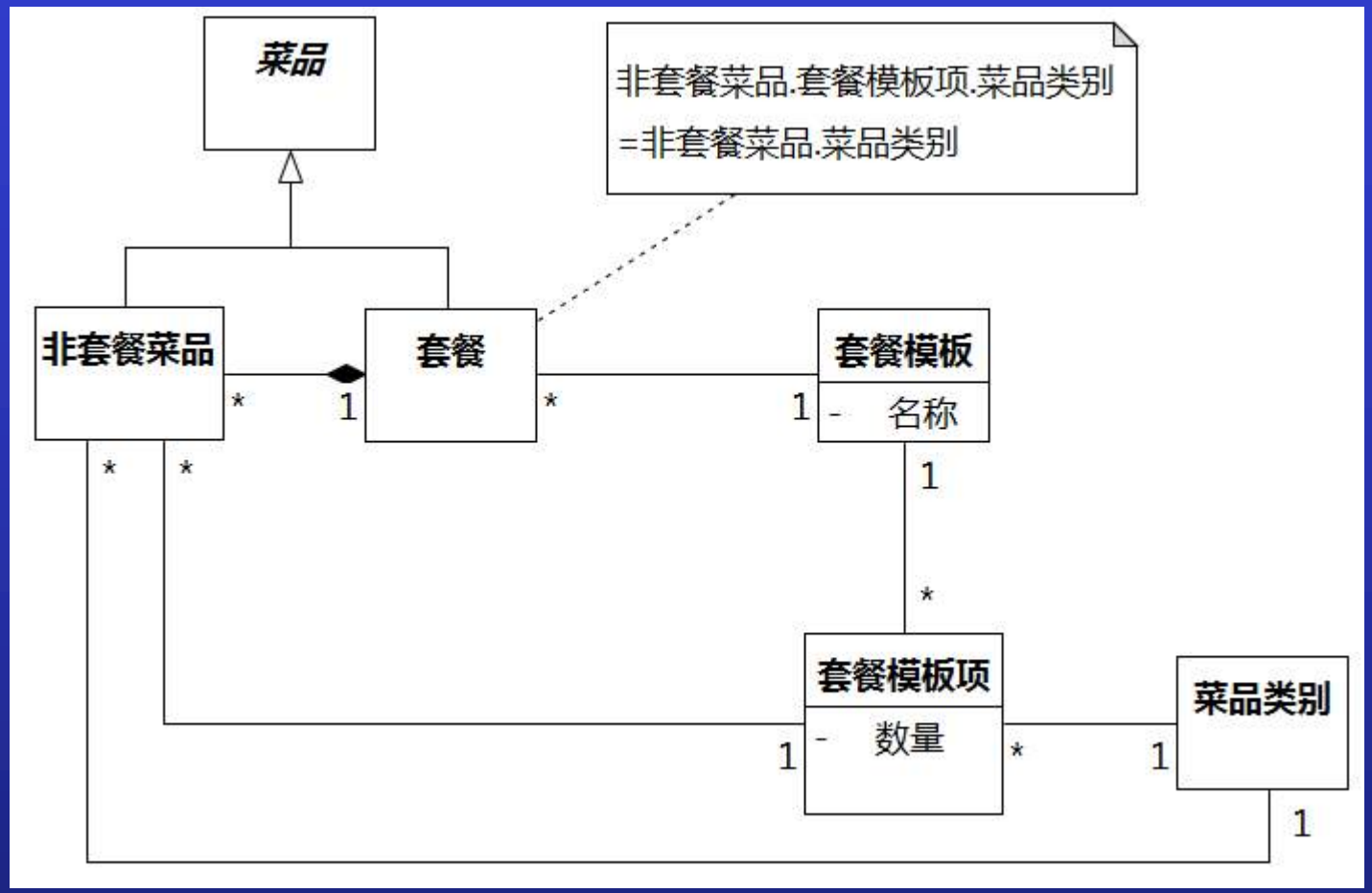


组合模式



套餐-应用GoF模式-组合

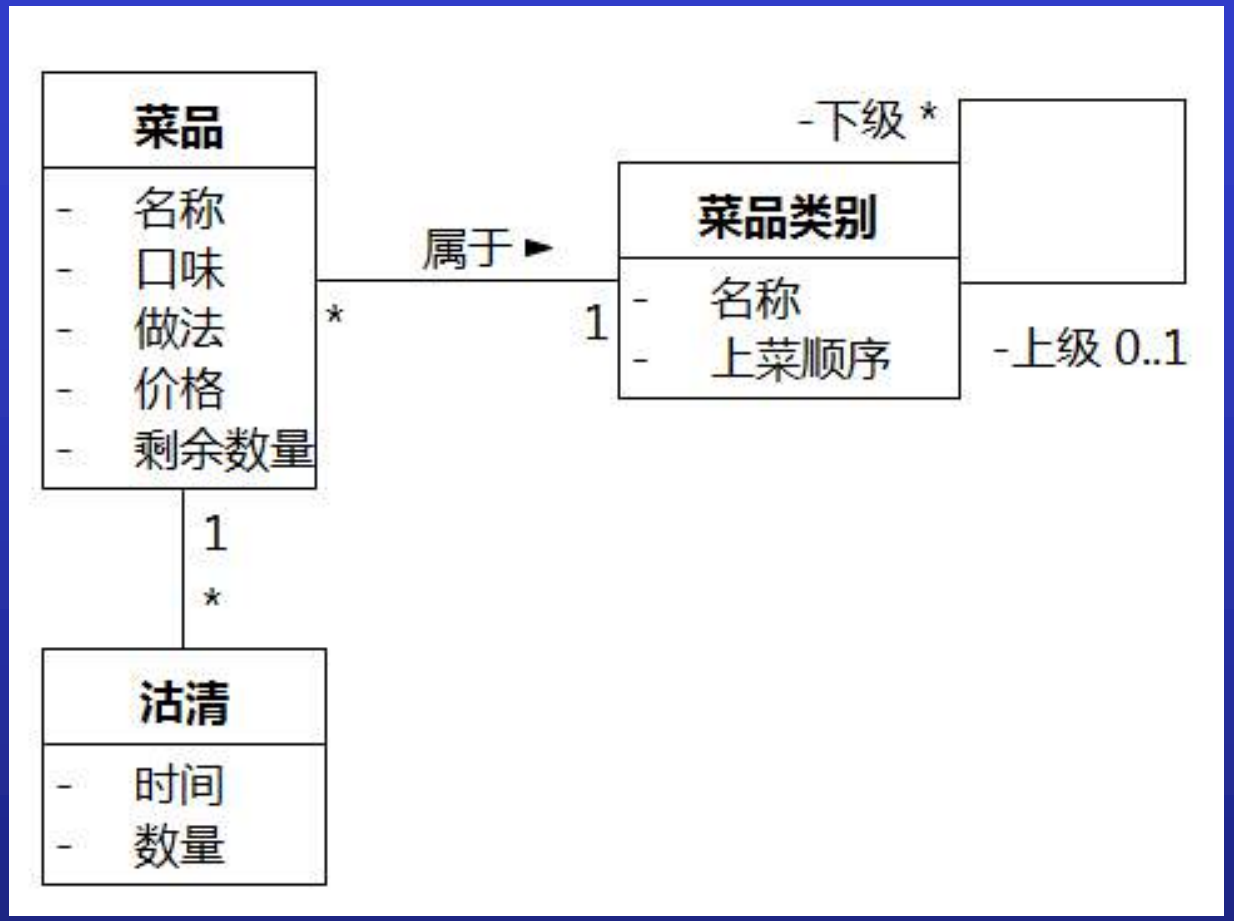
菜品

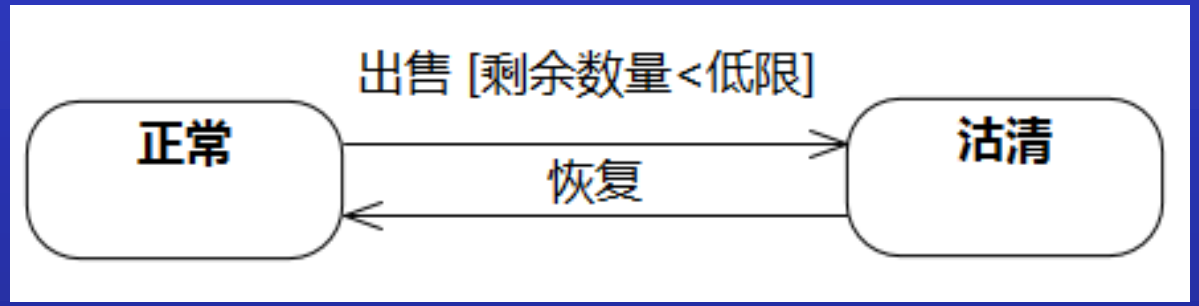

菜品
非套餐菜品.套餐模板项.菜品类别
=非套餐菜品.菜品类别
非套餐菜品
套餐
套餐模板
- 名称
套餐模板项
- 数量
菜品类别
*
1


二荤一素 20 元
二荤一素 15 元
鼓汁排骨 青椒腊肉 青葱炒腐竹
黑椒牛肉 蒜苗青椒炒蛋 莲藕片
香煎秋刀鱼
花生焖鸡脚 肉末蒸豆腐 炒青菜

按规则组合的套餐

菜品

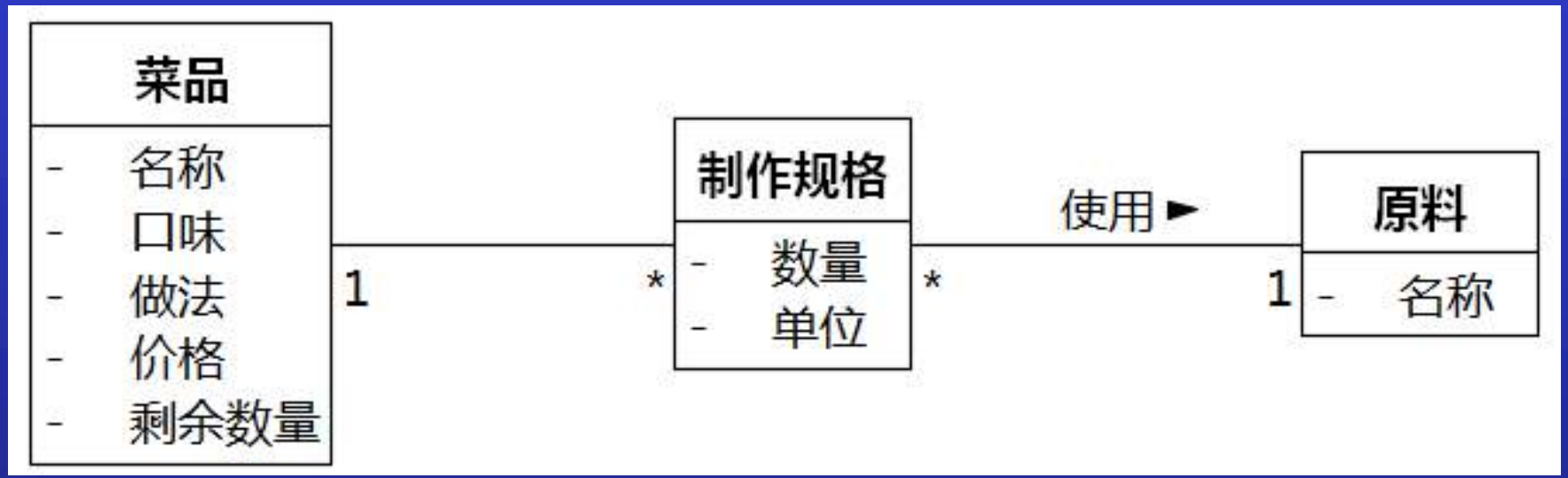

菜品
- 名称
- 口味
- 做法
- 价格
- 剩余数量
属于
*
1
菜品类别
- 名称
- 上菜顺序
-下级 *
-上级 0..1
1
*
沽清
- 时间
- 数量


出售 [剩余数量<低限]
正常
恢复
沽清

菜品状态

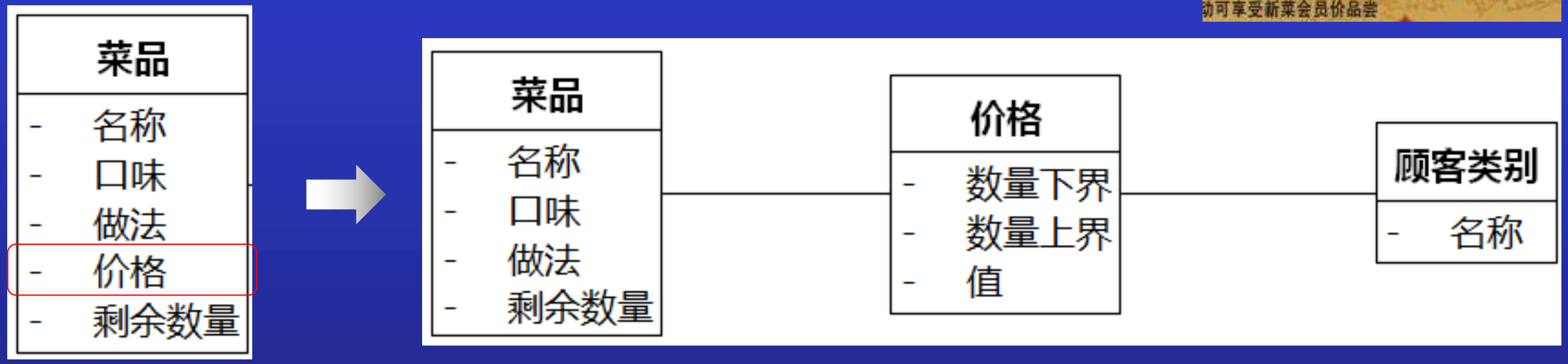
沽清

菜品



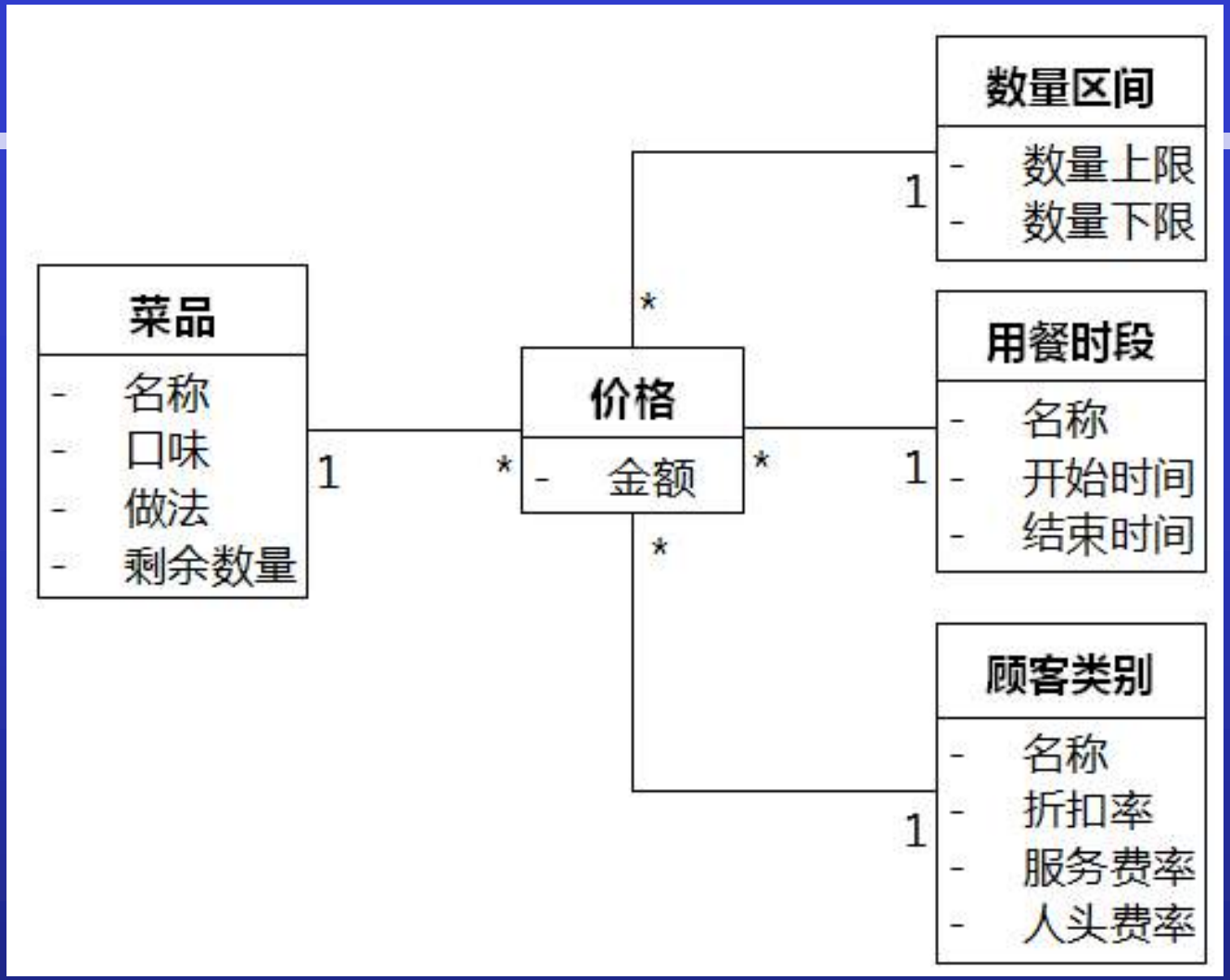
原料

菜品



价格

菜品


数量区间
- 数量上限
- 数量下限
1
菜品
- 名称
- 口味
- 做法
- 剩余数量
1
*
价格
- 金额
*
*
用餐时段
- 名称
- 开始时间
- 结束时间
1
*
顾客类别
- 名称
- 折扣率
- 服务费率
- 人头费率
1

更多价格因子

菜品

值	价格因子可选项ID	价格因子类别ID
黄金	1	1
白金	2	1
黑金	3	1
周末	4	2
生日	5	2

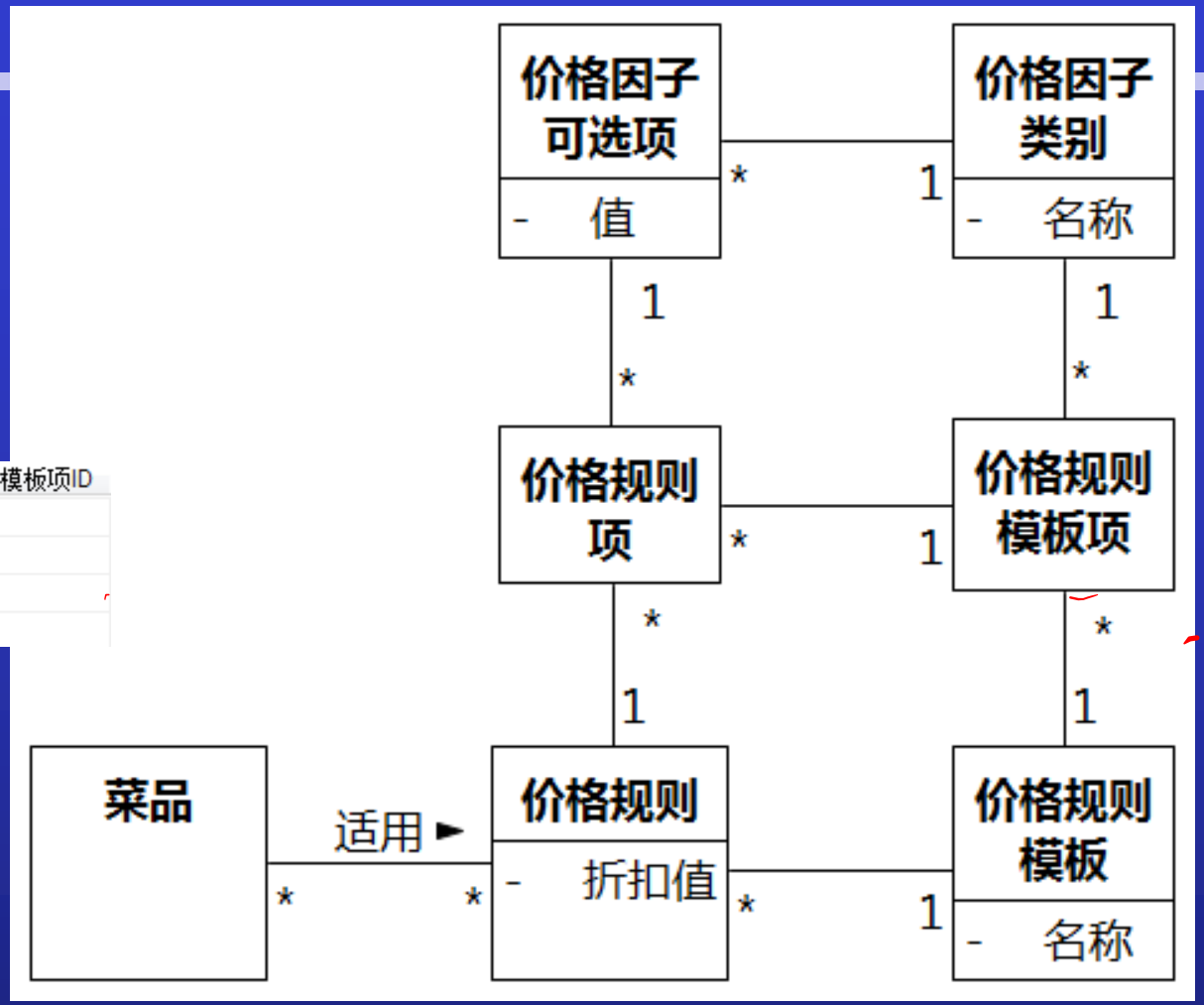
名称	价格因子类别ID
会员等级	1
假日等级	2
城市	3

价格规则项ID	价格因子可选项ID	价格规则ID	价格规则模板项ID
1	3	1	1
2	5	1	2
3	1	2	1
4	9	2	2

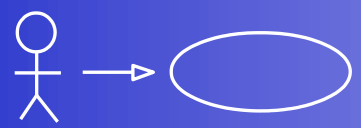
价格规则模板项ID	价格因子类别ID	价格规则模板ID
1	1	1
2	3	1

折扣值	价格规则ID	价格规则模板ID
0.89	1	1
0.88	2	1

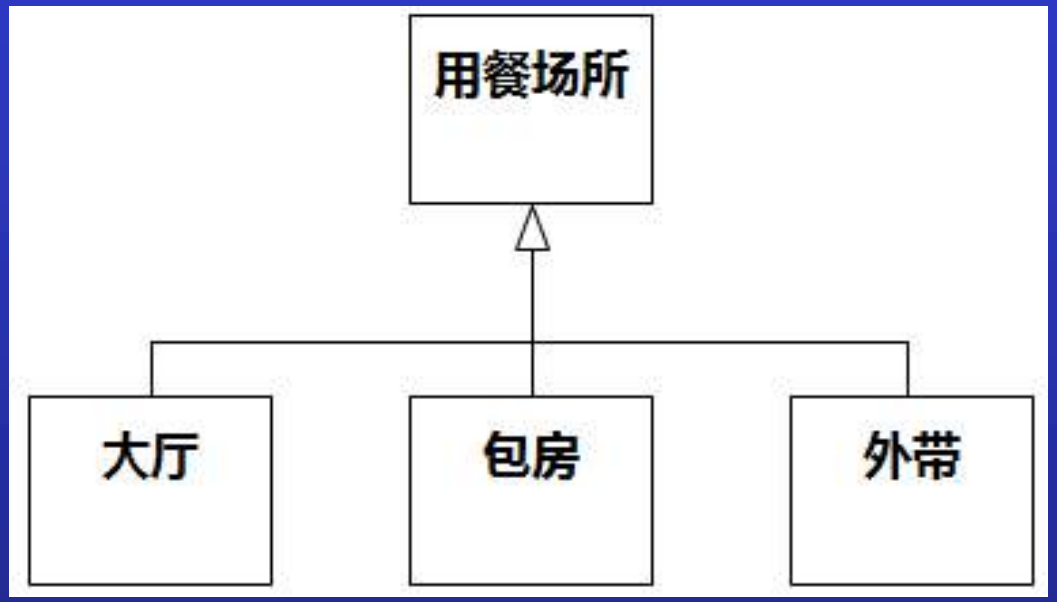
名称	价格规则模板ID
会员假期打折策略	1

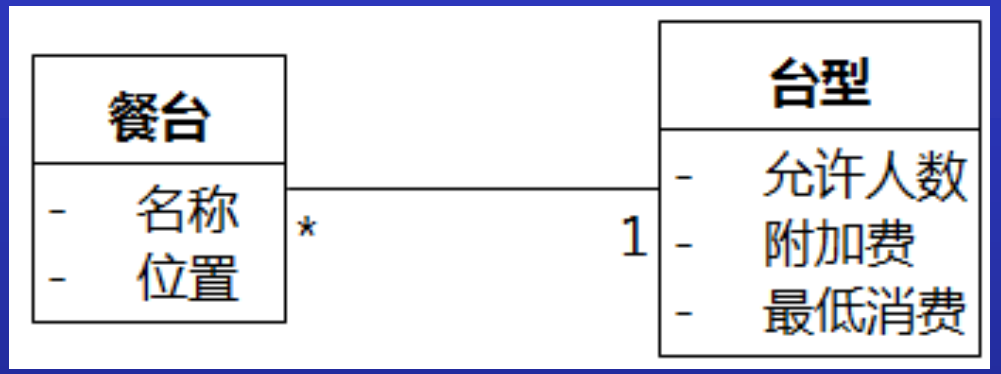


应用分析模式–物品

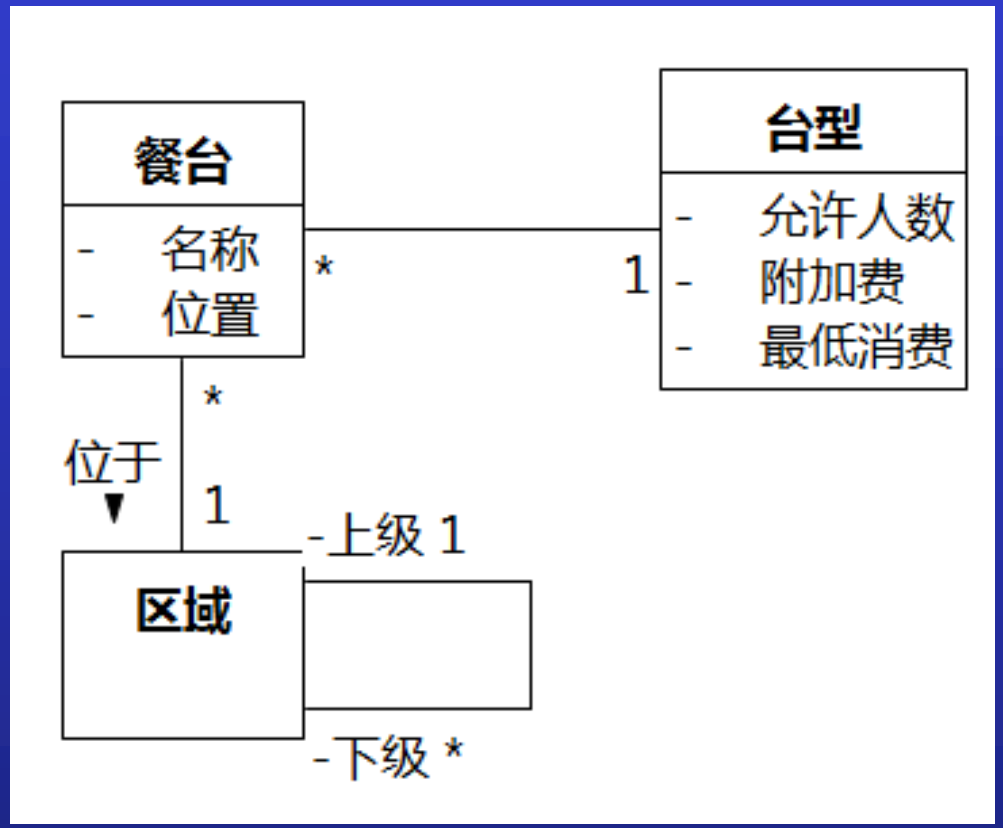


餐台



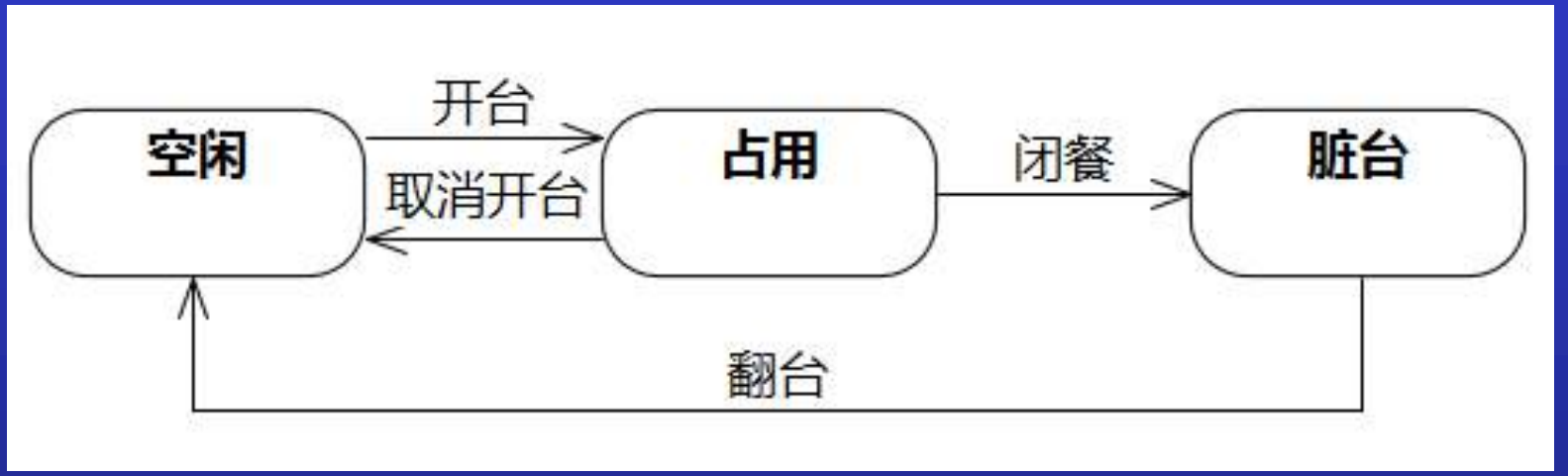


餐台



区域

餐台



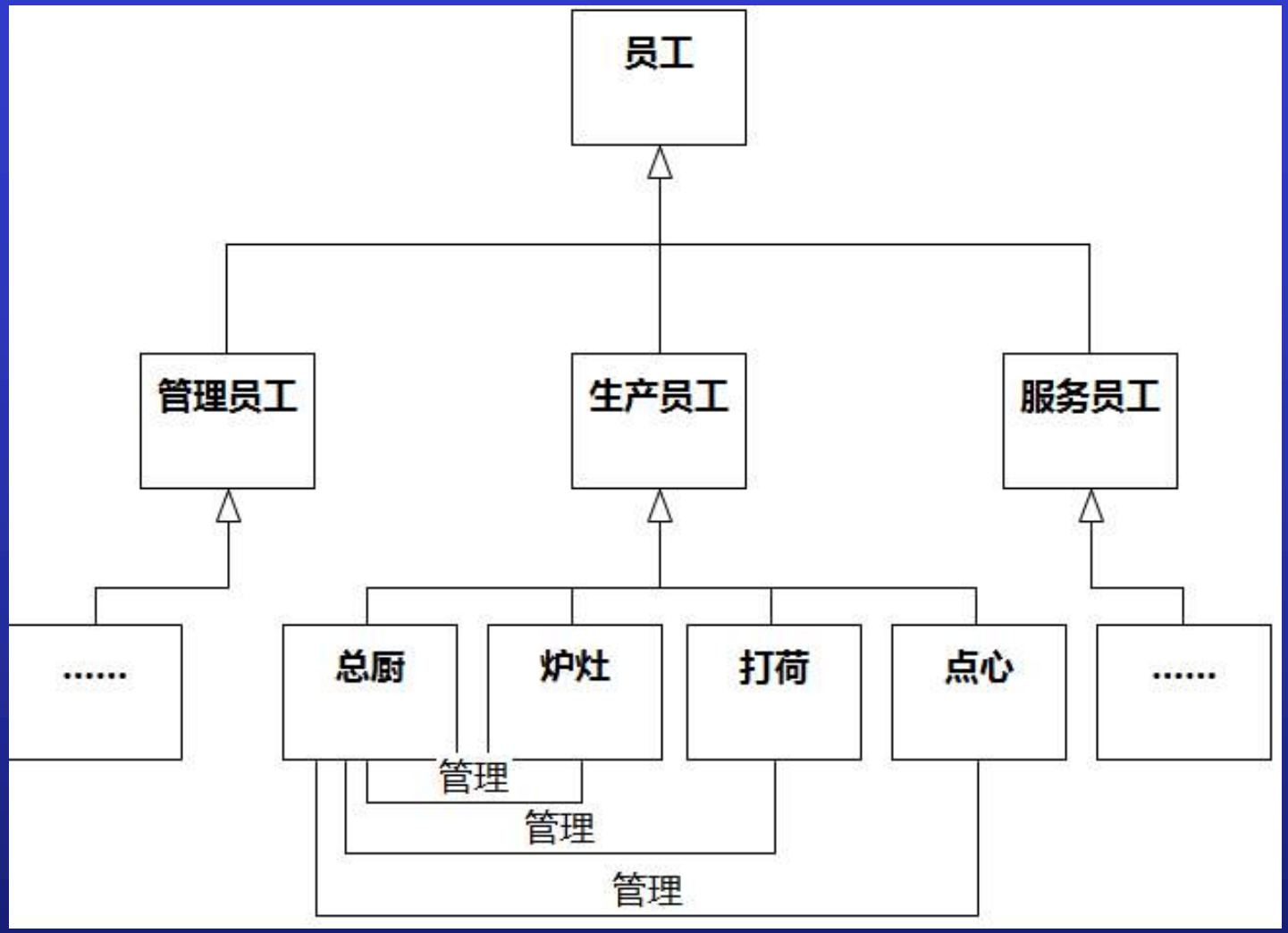
也可以两个分区：

干净-脏

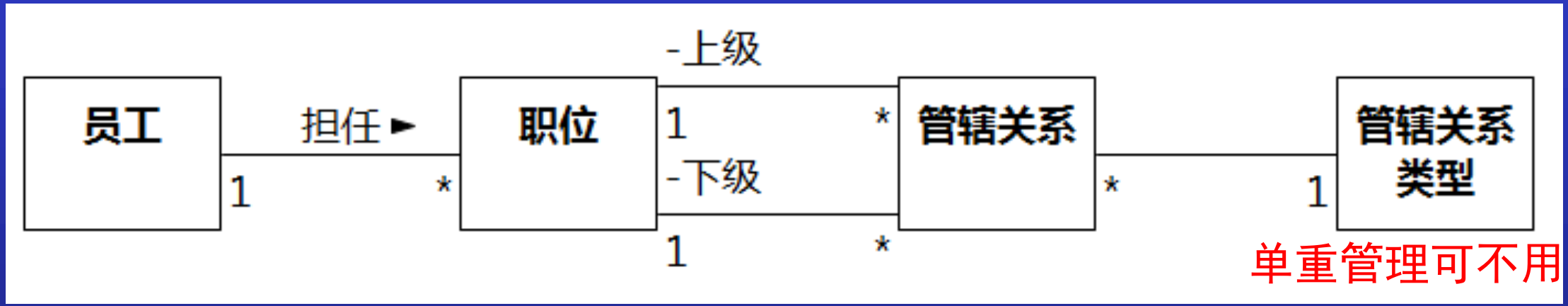
空闲-占用

状态

员工

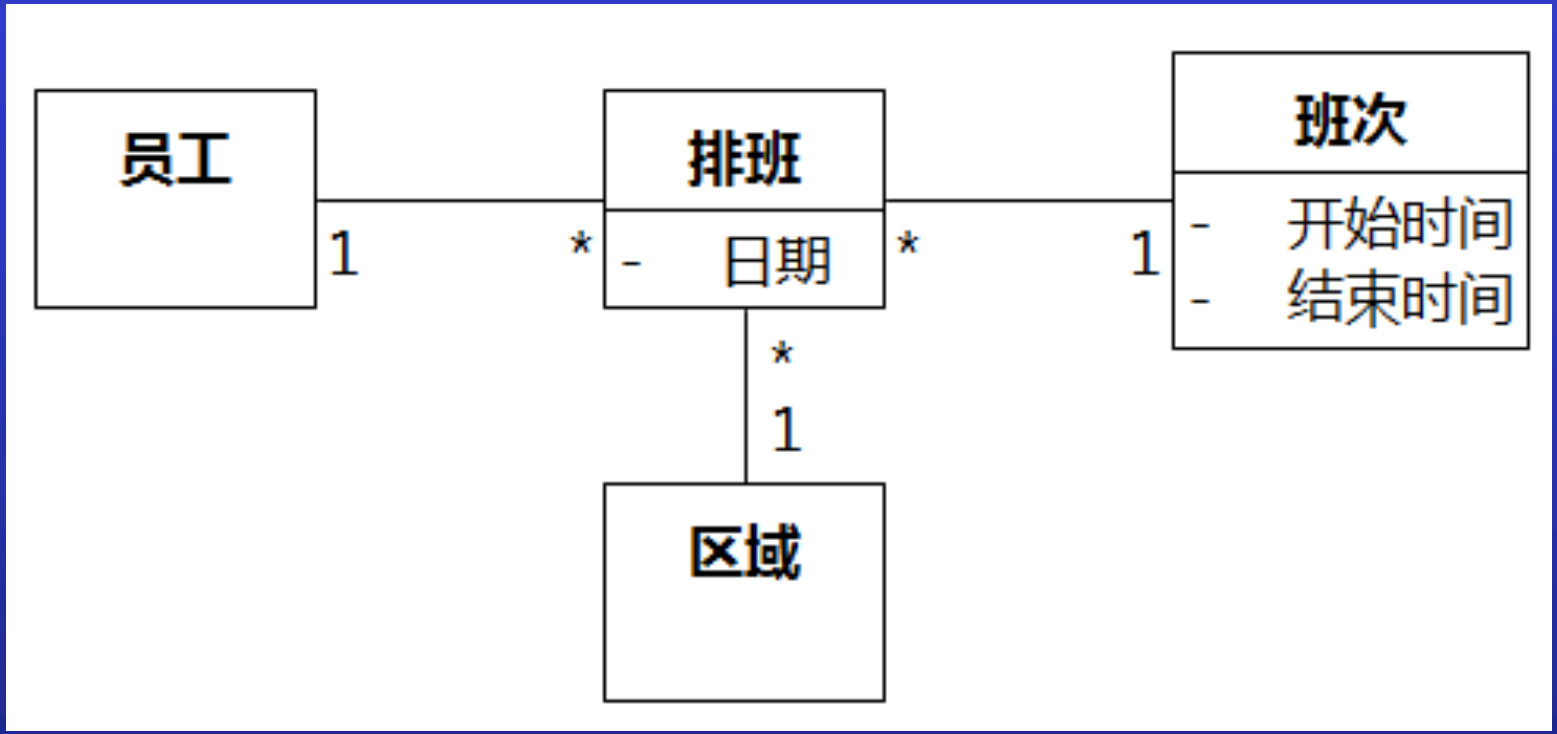


员工



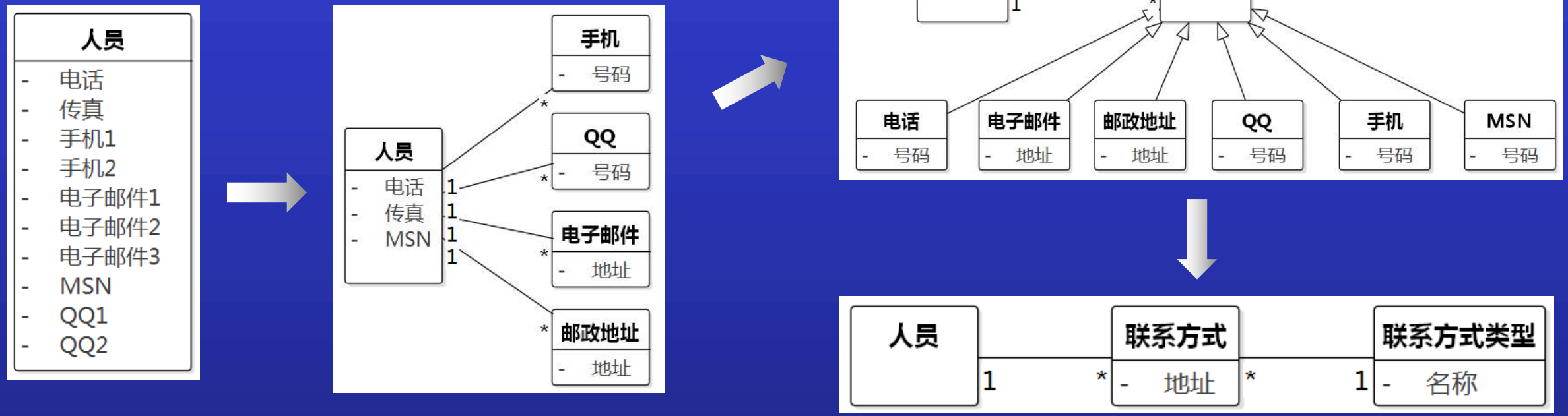
应用分析模式–组织

员工

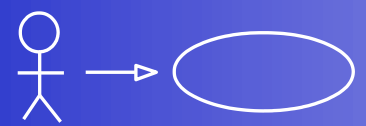


排班

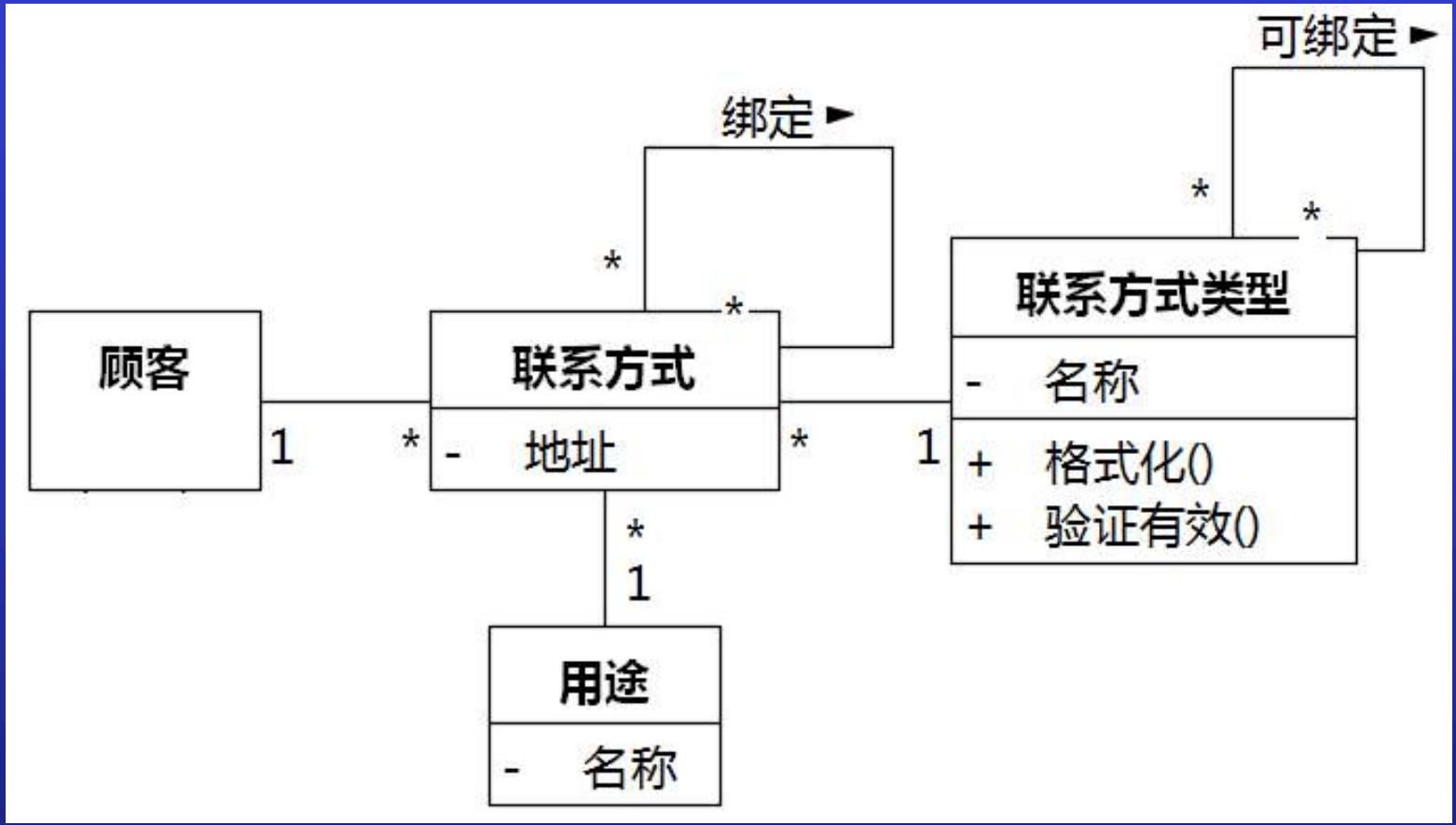
顾客


人员
- 电话
- 传真
- 手机1
- 手机2
- 电子邮件1
- 电子邮件2
- 电子邮件3
- MSN
- QQ1
- QQ2
人员
- 电话
- 传真
- MSN
手机
- 号码
QQ
- 号码
电子邮件
- 地址
邮政地址
- 地址
1
*
人员
联系方式
电话
- 号码
电子邮件
- 地址
邮政地址
- 地址
QQ
- 号码
手机
- 号码
MSN
- 号码
人员
联系方式
- 地址
联系方式类型
- 名称

应用分析模式–人员

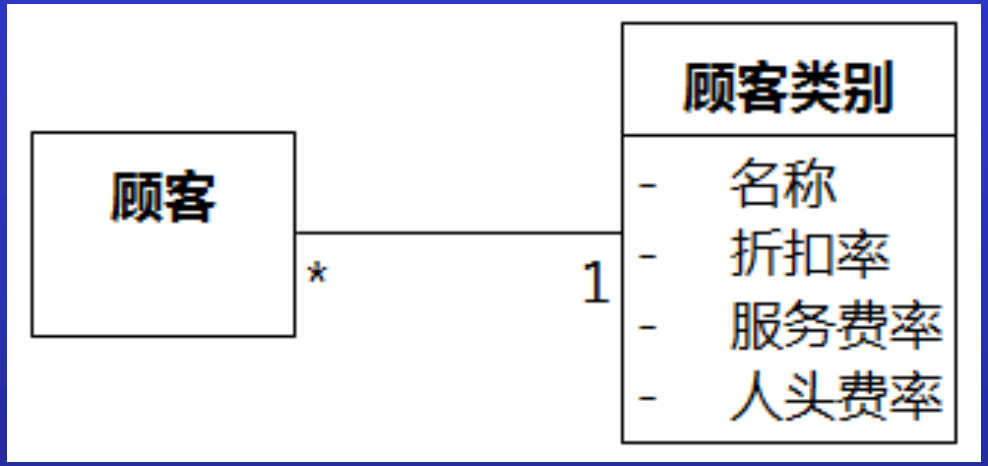


顾客



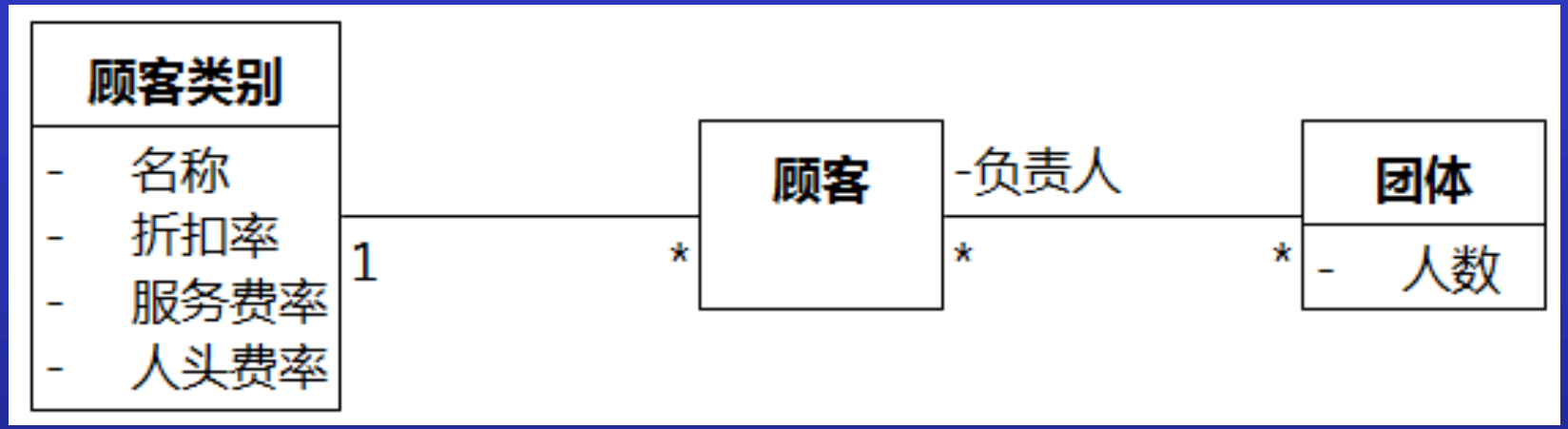
应用分析模式–人员

顾客



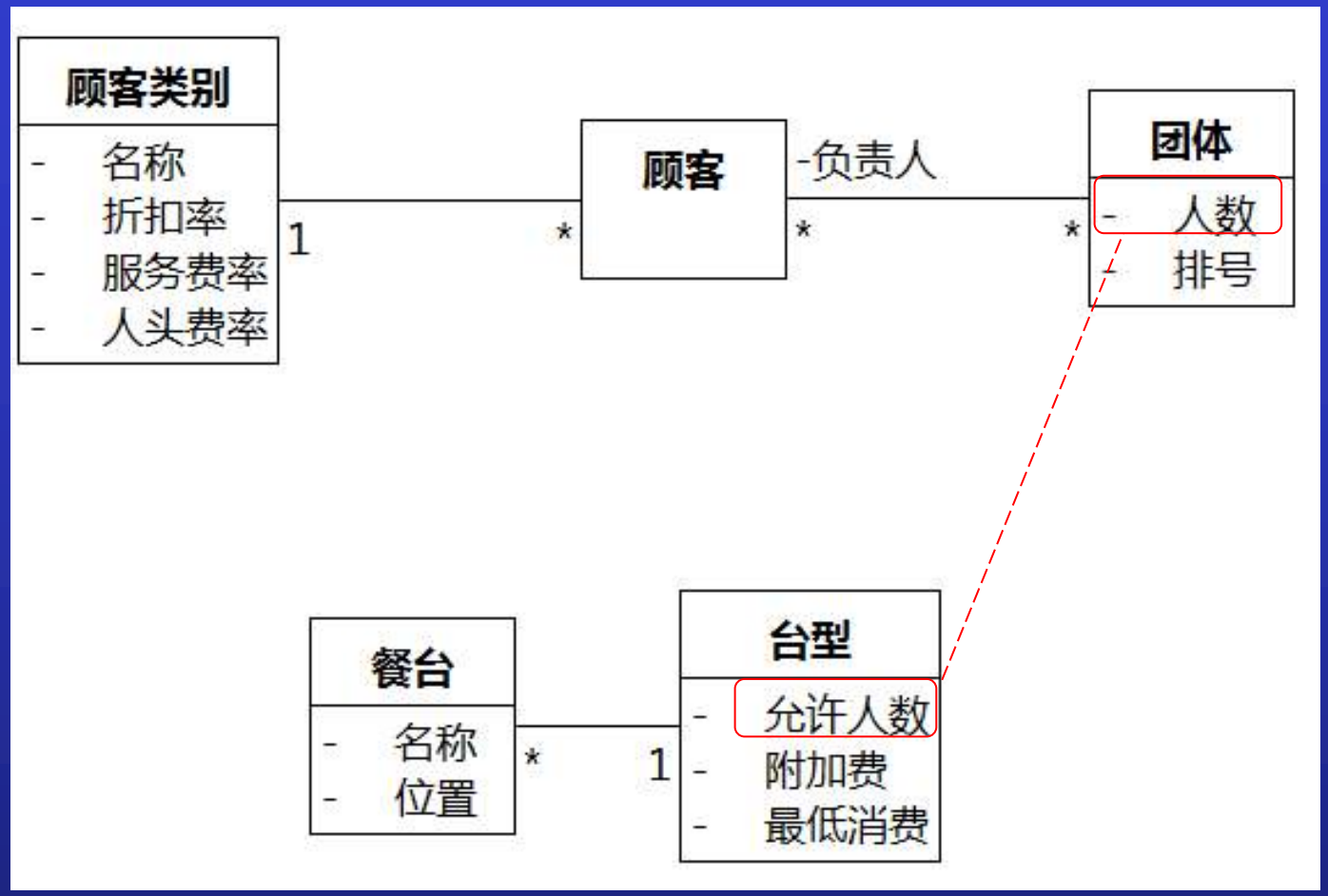
顾客类别

顾客



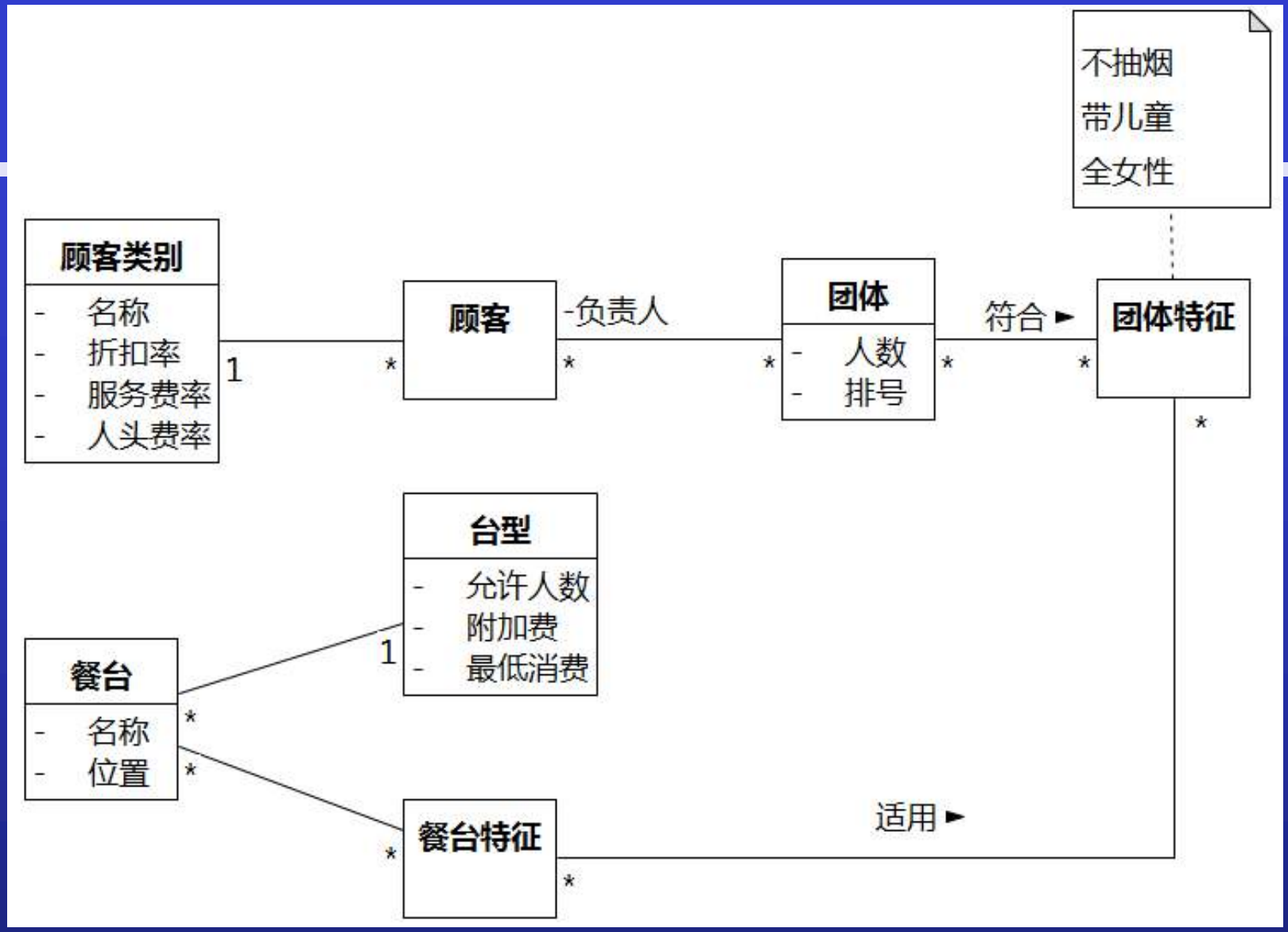
就餐团体

顾客

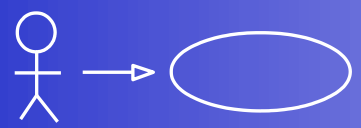


就餐团体

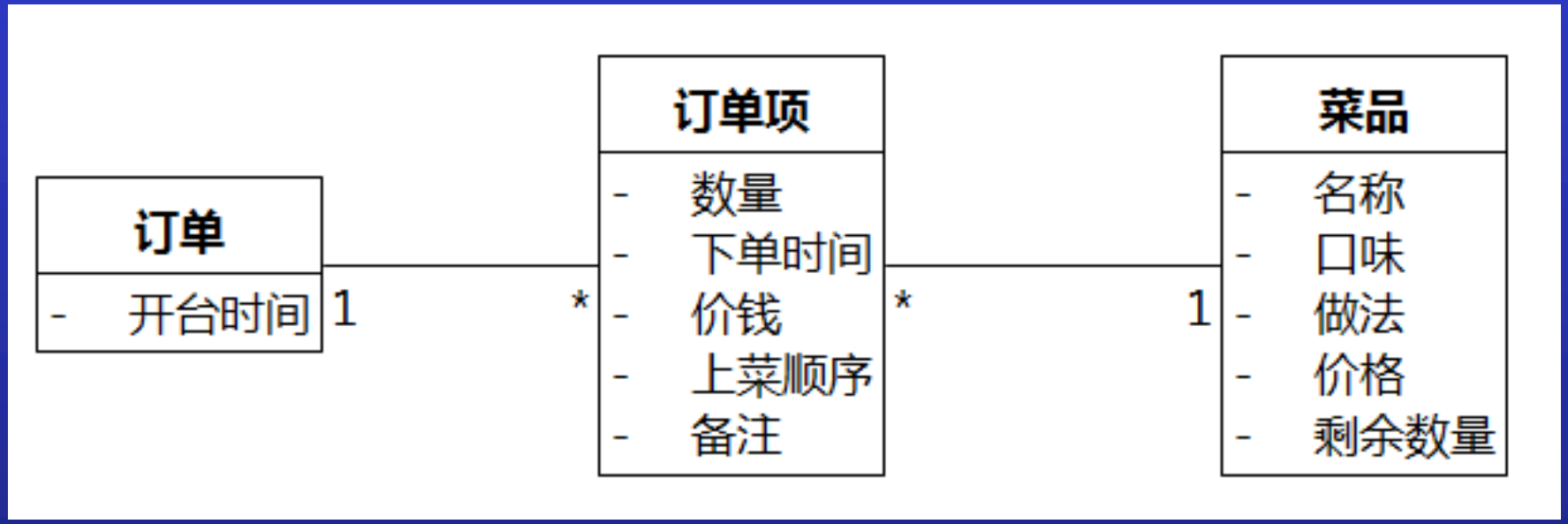
顾客


不抽烟
带儿童
全女性
顾客类别
- 名称
- 折扣率
- 服务费率
- 人头费率
1
*
顾客
-负责人
*
*
团体
- 人数
- 排号
*
符合 ►
*
团体特征
*
台型
- 允许人数
- 附加费
- 最低消费
1
餐台
- 名称
- 位置
*
*
*
餐台特征
*
适用 ►

进一步扩展



订单

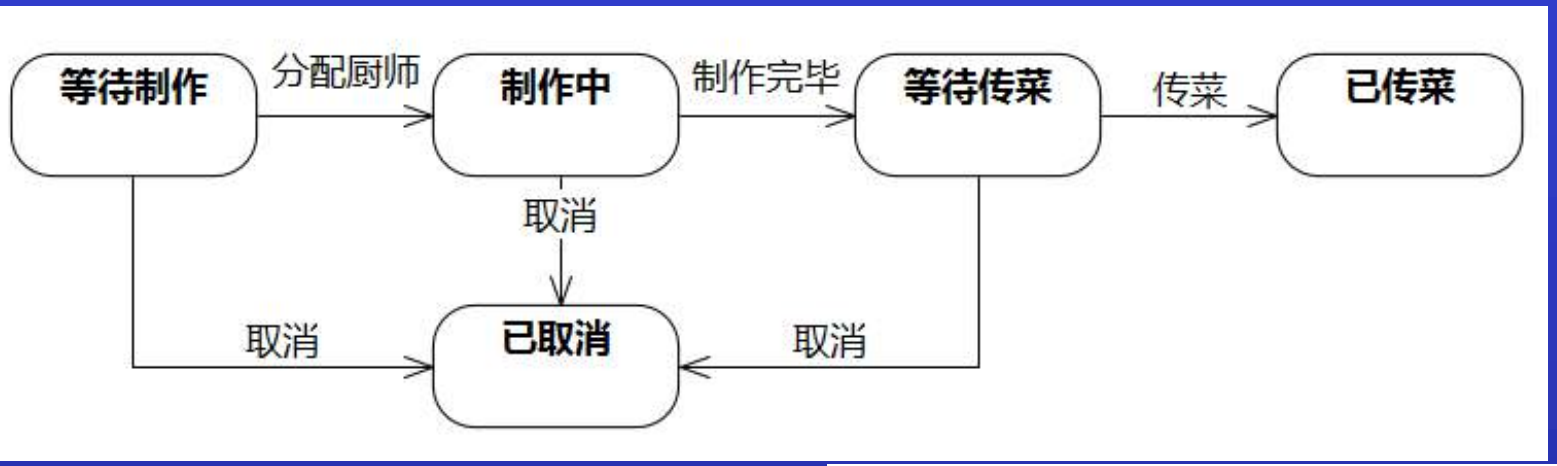


12桌的梅菜扣肉

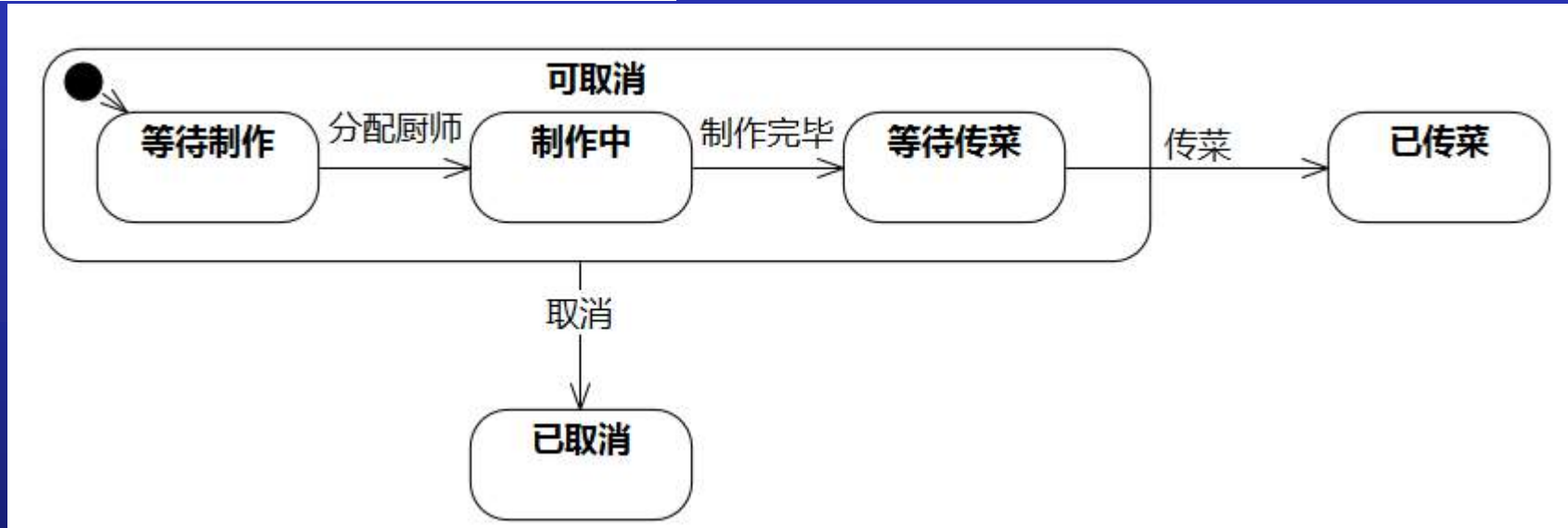
13桌的梅菜扣肉

和菜品的关系

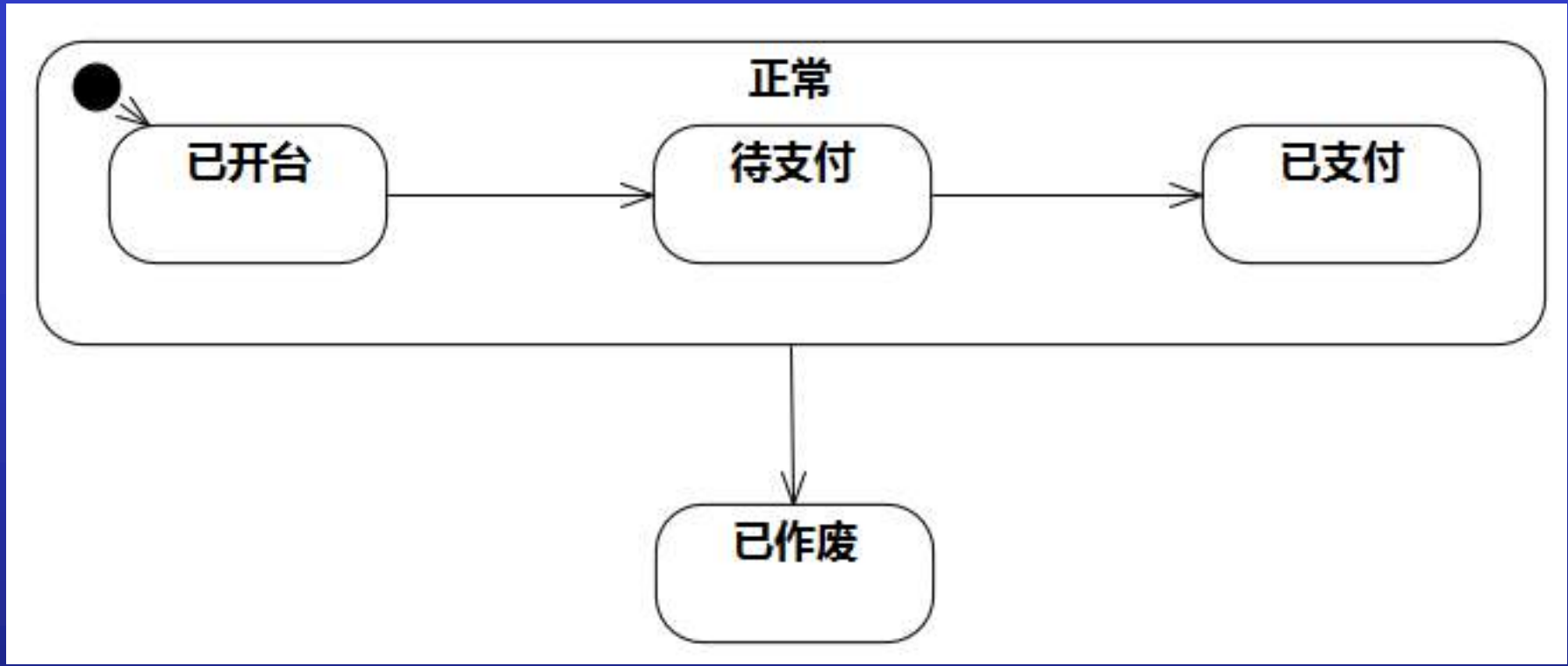
订单


等待制作
分配厨师
制作中
制作完毕
等待传菜
传菜
已传菜
取消
取消
已取消
取消

订单项的状态

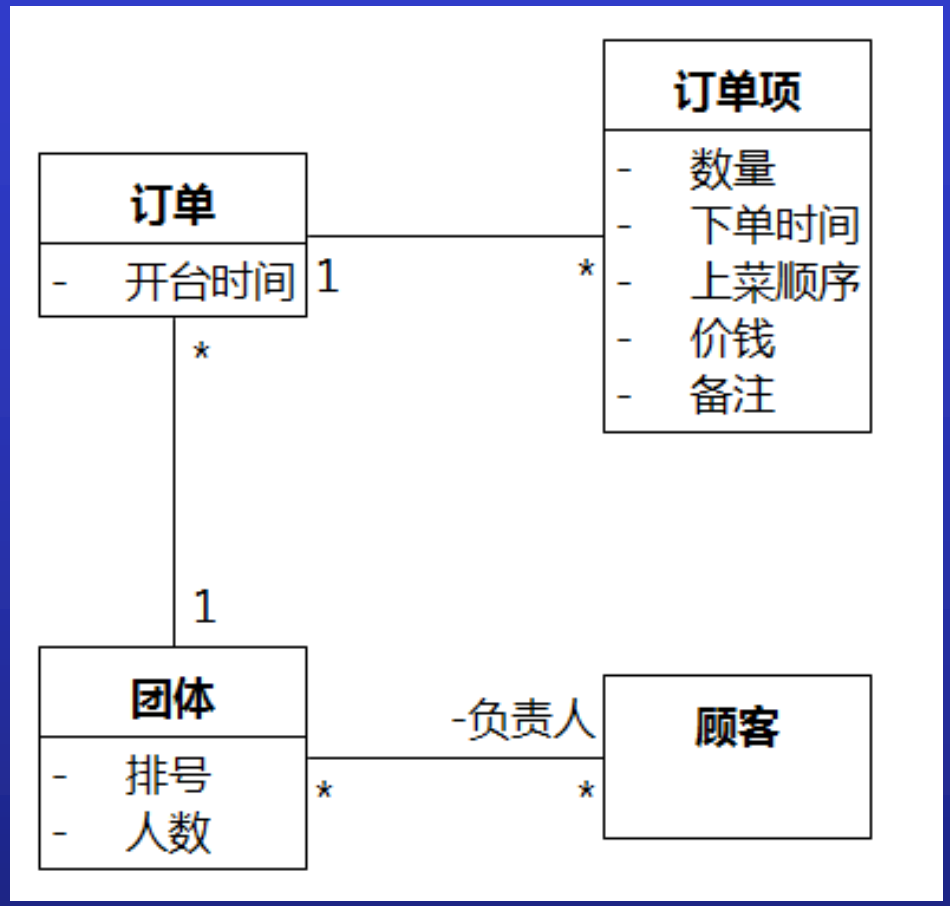

可取消
等待制作
分配厨师
制作中
制作完毕
等待传菜
传菜
已传菜
取消
已取消

订单

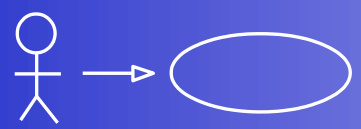


订单的状态

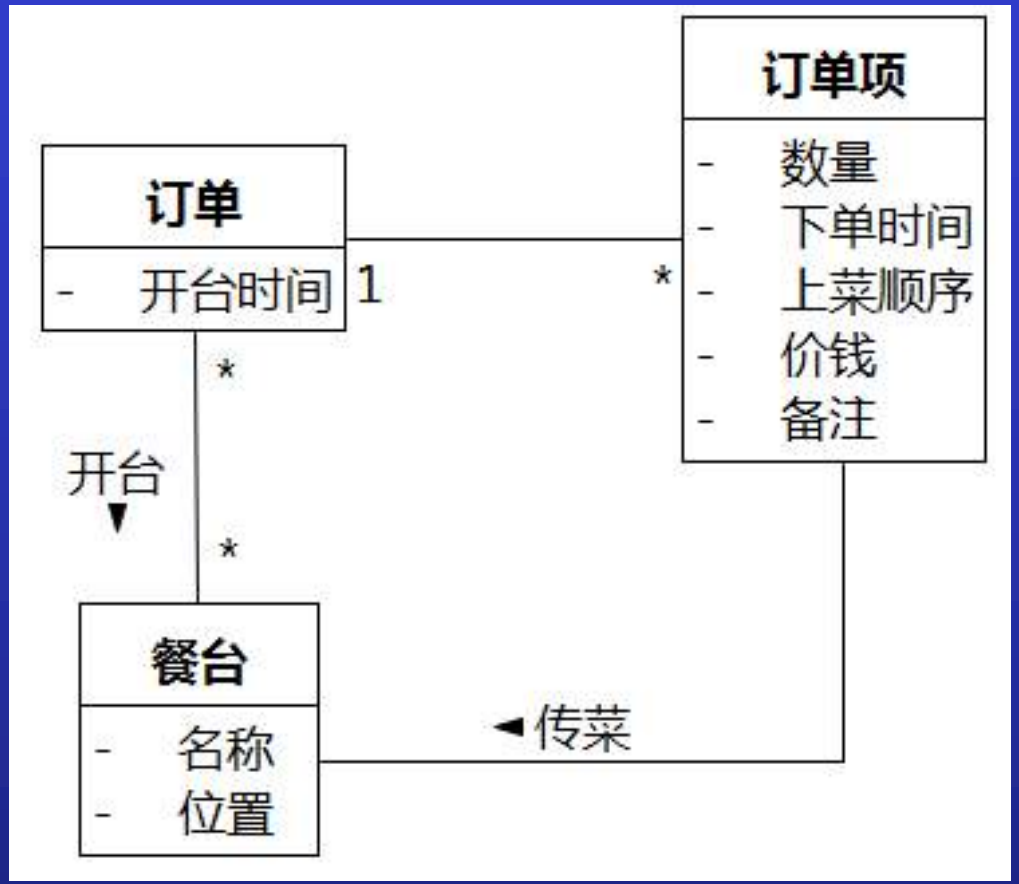
订单



和顾客的关系



订单



和餐台的关系