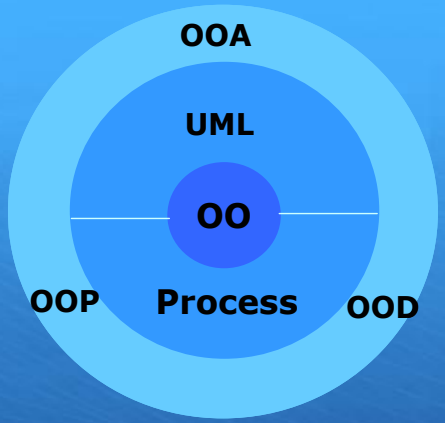
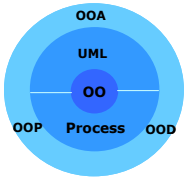


第6周

面向对象系统分析——类图

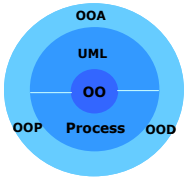
北京大学软件与微电子学院
蒋严冰 jyb@ss.pku.edu.cn

引言



+ 上周，讲述了如何捕获系统的功能需求。
+ 一个软件系统有静态结构方面，也有动态行为方面。
+ 本章从需求出发，结合问题域，详细介绍类图，用以对系统的静态结构建模。
+ 类图是面向对象建模的最重要的图。
+ 一个类图应该注重表达系统静态结构的一个方面，并且要与抽象的层次相一致。

什么是结构?

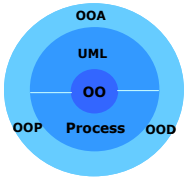


+ 让•皮亚杰（Jean Piaget，1896年8月9日 - 1980年9月16日），瑞士人，是近代最有名的儿童心理学家。他的认知发展理论成为了这个学科的典范。

+ 结构是由种种转换规律组成的体系.结构就是由具有整体性的若干转换规律组成的一个有自身调整性质的体系。

+ 结构的三个特征
  - 整体性：一个结构有若干成分组成，并且整体有部分不具有的特征
    - 代数结构
    - 类的类属性
    - 面向对象系统的功能
  - 转换性
    - 具有转换规律和法则，如运算。
  - 自身调整性
    - 结构的守恒和封闭性，一个结构所固有的各种转换不会越出结构的边界之外。

母结构



+ 布尔巴基学派是一个对现代数学有着极大影响的数学家的集体。其中大部分是法国数学家。他们的活动从20世纪30年代中期开始，曾先后在数学杂志上发表过一些文章，是布尔巴基学派认为数学只是研究结构的科学，因此只对抽象的数学结构感兴趣而对对象本身究竟是数、是形、是函数还是运算并不关心。

+ 代数结构
  - 群 环 体 域
+ 序结构
  - 格
+ 拓扑结构
  - 邻接性 连续性 界限
  - 图

To be continued…

提纲

- 1.对象与类
- 2.识别属性
- 3.识别服务
- 4.识别关系
- 5. 接口

1.对象与类

- 1.1 概念与表示法
- 1.2 识别对象与类的方法
- 1.3 审查与筛选
- 1.4 识别主动对象
- 1.5 类的命名
- 1.6 建立类图的对象层

1.1 概念与表示法

- 对象：是系统中用来描述客观事物的一个实体，是具有明确语义边界的实体；作为构成系统的一个基本单位,一个对象由一组属性和对这组属性进行操作的一组服务构成。

- 类：是具有相同属性、服务、关系和语义的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述，其内部包括属性和服务两个主要部分。

- 类和对象的关系——模板与实例；类的实例是对象。类的外延是其所产生的对象集。

罗素公理体系

+ 在类的公理体系中，有一些基本的概念是不加定义的，我们只能从其客观含义上给予解释，但这样的解释仅仅起到帮助理解这些概念。

+ 数学中研究的任何一个客体（对象）都称为一个**类**。类的概念是没有任何限制。类与类之间可能存在着一种称为**属于**的关系，类A属于类B记为$A \in B$，此时也称类A是类B的一个**元素**(简称为**元**)。

+ 我们可以把类理解成为是由若干元素组成的一个整体。一个类是否是另一个类的元素是完全确定的，这就是类元素的**确定性**。类A如果不是类B的元素，则称A**不属于**B，记为$A \notin B$。

+ 另一个不加定义的概念就是：类总是具有一定的**性质**，我们常以P(x)表示类x具有性质P。我们可以把性质理解为“关于类的一句表述”。

类的外延公理

+ **公理 I (外延公理)**

+ 公理 I 的含义是：两个类“相等”的充要条件是它们的元素完全相同，这就是说，类完全由其元素确定。类的所有元素可以通俗地称为它的**外延**，正因如此，公理 I 被称为外延公理。由此我们可以定义： $\forall A, B(A = B \iff \forall x(x \in A \iff x \in B))$

+ **定义1.1** 两个类A、B，如果它们的元素完全相同，则称这两个类是**相等**的，记为: $A = B$

类的内涵

+ 一般地说，类中的元素总是具有某种共同的性质的，这就是类的元素的**同质性**。

+ 一个类的所有元素所共同具有的、而且是这个类的元素所独有的性质(也就是说不是该类的元素就不具有该性质) 称为该类的**内涵**。

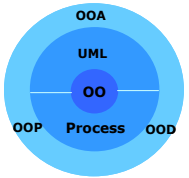
+ 类的内涵与外延之间存在着直观的“反比关系”：类的内涵越多，其外延越小；内涵越少，其外延越大。

罗素悖论

- 任给一个性质，满足该性质的所有类可以组成一个类。
- 但这样的企图将导致如下的悖论：
- **罗素悖论**
- 设性质P(x)表示“ $x \notin x$ ”，现假设由性质P确定了一个类A----也就是说“ $\forall x(x \in A \iff x \notin x)$ ”。
- 那么现在的问题是：是否成立？首先，若$A \in A$，则A是A的元素，那么A具有性质P，由性质P知$A \notin A$；其次，若 $A \notin A$ ，也就是说A具有性质P，而A是由所有具有性质P的类组成的，所以$A \in A$ 。
- 也就是说“A={x|x ∉ x}”。那么现在的问题是：A∈A是否成立？

According to Booch

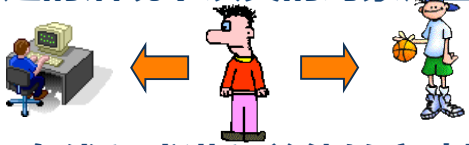
（Object-Oriented Analysis and Design with Applications）



- 对象
  - 是一个具有状态、行为和标识符的实体。结构和行为类似的对象定义在同一类中。
    - 对象不一定是现实世界的
    - 现实中的有些东西并不是对象
      - 没有明确的边界?
      - 不可触摸?
  - 状态：
    - 这个对象的所有属性（通常是静态的）以及每个属性当前值（通常是动态的）
  - 属性：
    - 内在的独特的特征、特点、品质或特性，使一个对象区别于别的对象
  - 行为：
    - 对象在状态改变和消息传递方面的动作和反应的方式
  - 标示符：
    - 对象的一个属性，区分对象与其他所有对象
- 类
  - 是一组对象，拥有共同的结构，共同的行为和共同的语义

+ 由一个类生成的一个对象可以扮演不同的角色。

+ 角色:一个类的一个角色是在特定的语境下该类的对象所呈现的行为。



+ 主动对象(active object): 是拥有线程或进程并能够启动控制活动的对象。是用于描述具有主动行为的事物。

+ 主动类:主动对象所属于的类叫做主动类。
+ 特征标记:服务的名称及其后的位于括号内的参数列表叫做特征标记（或基调）(signature)。
  - setColor(String aColor)

类的各种表示法

Window

Window

size: Area
visibility: Boolean

display ()
hide ()

Window
{abstract,
author=Joe,
status=tested}

+size: Area = (100,100)
#visibility: Boolean = true
+default-size: Rectangle
#maximum-size: Rectangle
-xptr: XWindow*

+display ()
+hide ()
+create ()
-attachXWindow(xwin:Xwindow*)

Reservation

operations

guarantee()
cancel ()
change (newDate: Date)

responsibilities

bill no-shows
match to available rooms

exceptions

invalid credit card

类名
{active}

类名

对象名：类名

对象名：类名

To be continued…

1.2 识别对象与类的方法

- 研究用户需求，明确系统责任
- 研究问题域
- 考虑系统边界
- 考虑系统责任
- 名词技术
- CRC卡片
- Booch对识别类的方法总结
- Peter Coad的四色类图

研究用户需求，明确系统责任

- 阅读：阅读一切与用户需求有关的书面材料
- 交流：澄清疑点，指导需求
- 调查：到现场调查
- 记录、整理：产生需求文档（补充用况图）
- 如果建立了用况图，那么跳过1．

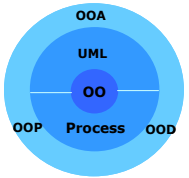
研究问题域

- 亲临现场调查，掌握第一手资料
- 听取问题域专家的见解
- 阅读与问题域有关的材料
- 借鉴相同或类似问题域已有的系统开发经验及文档

考虑系统边界(参与者)

- 参与者启发作为系统中的类的条件
  - 管理信息
  - 模拟行为
  - 建立通讯

考虑系统责任



- 对照系统责任所要求的每一项功能，查看是否可以由现有的对象完成这些功能。
- 如果发现某些功能在现有的任何对象中都不能提供，则可启发我们发现问题域中某些遗漏的对象。

名词技术

- Rusell J.Abbott率先发起，Grady Booch推广．
- 从名词到对象或类通常有一对一的映射。
- 用单个的专有名词或代词（Jim、他、她、雇员号5、我的工作站、我的家）以及直接引用的名词（第六个参赛者、第一百万次购买）识别对象。
- 用复数名词（人们、顾客们）以及普通名词（人、顾客）来识别类。
- 可以启发分析员发现对象的因素包括：人员、组织、物品、设备、事件（如索赔、上访、交易）、表格、日志、报告、结构等。

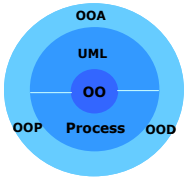
优缺点

- 优点
  - 叙述性语言可以被很好的理解
  - 名词到对象或类通常有一对一的关系
  - 简单\直接
- 缺点
  - 盲目性
  - 寻找的对象或类不全或错误
    - ?回滚事务
    - 软件将计算平均工资

CRC卡片（类 / 责任 / 协作）

- Rebecca Wirfs-Brock发明
- 方法：
  - 在一张CRC卡片上，记录类名并列出责任（提供那些服务）和协作者；
  - 识别应该加入到已存在的CRC卡片集合中的尚未发现的类．寻找已存在的类中的责任和还没有指派给某个责任的协作者．

举例 • *线路监控软件系统的CRC卡*



+ 某供电局准备开发线路监控软件系统，用于各条供电线路的情况。该系统由专职的管理员来操作。每条供电线路安装一个线路检测仪，每30秒采集1次该线路的信息（包括电压、电流）。每隔1小时，线路检测仪通过专线向线路监控软件系统传送该小时的数据，系统接受后，保存在系统中。

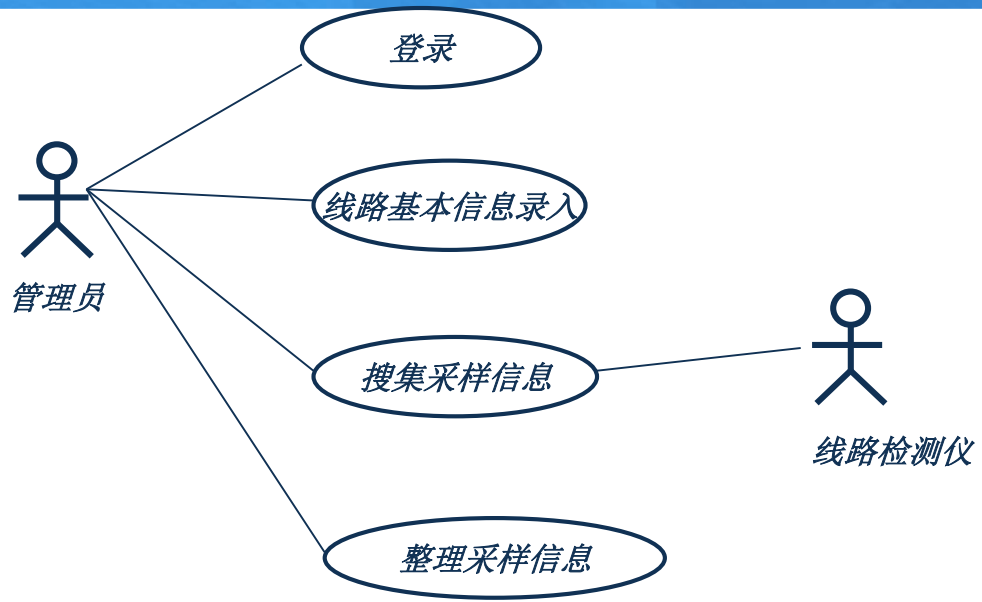
+ 管理员登陆系统后，可进行如下几项工作：

+ ①线路基本信息录入：录入每条线路的基本信息，包括线路号、位置等等。

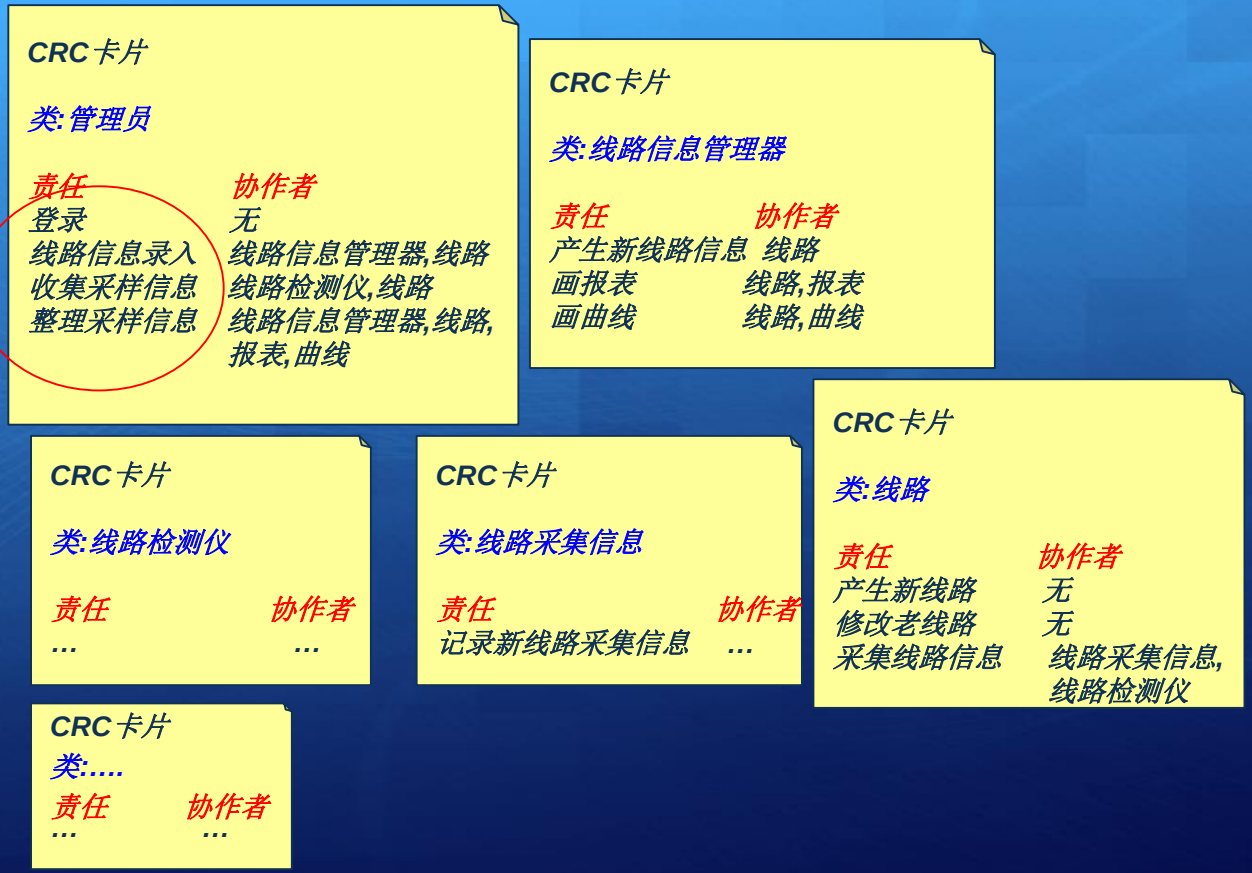
+ ②搜集采样信息：当系统与某线路的检测仪通讯时，系统提示管理员正在与该线路的检测仪通讯，并将检测仪所收集的该线路该时段的采样信息保存在系统中。

+ ③采样信息整理：可以根据这些数据生成该线路该日的数据报表与曲线，并能打印出来。


OOA
UML
OO
OOP
Process
OOD


登录
线路基本信息录入
搜集采样信息
整理采样信息
管理员
线路检测仪

CRC卡片

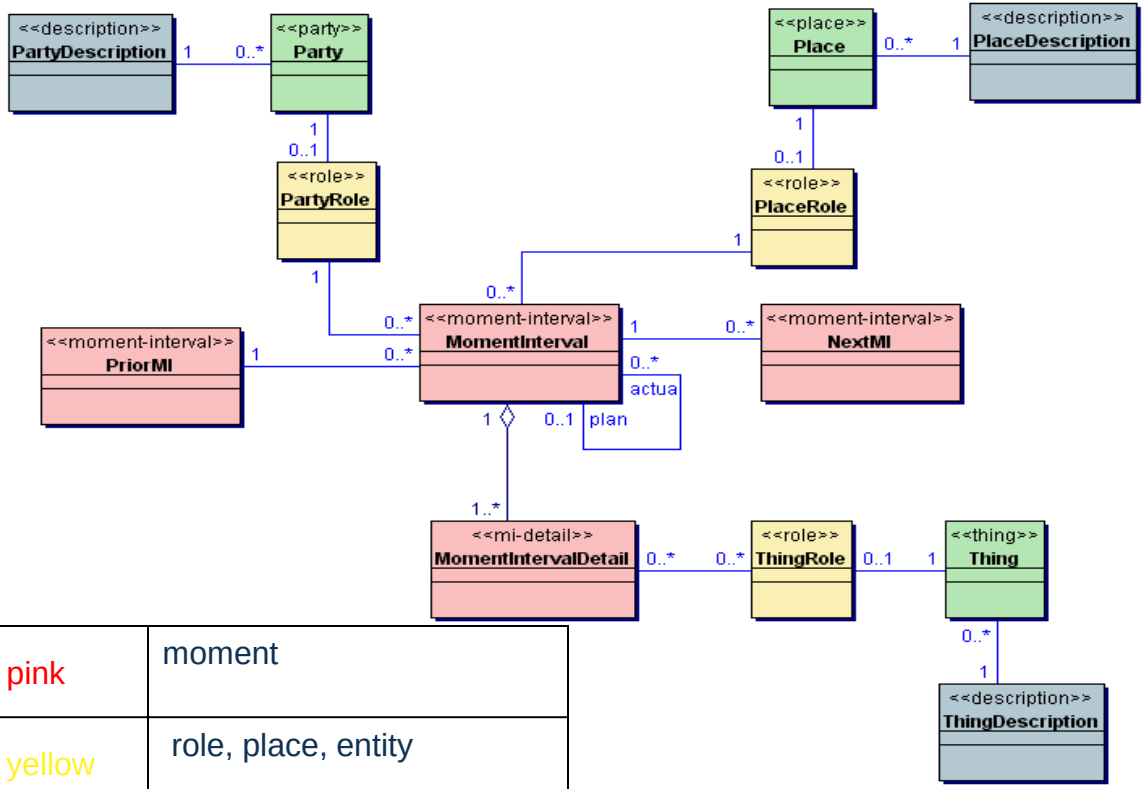

CRC卡片
类:管理员
责任
协作者
登录
无
线路信息录入
线路信息管理器,线路
收集采样信息
线路检测仪,线路
整理采样信息
线路信息管理器,线路,报表,曲线
CRC卡片
类:线路信息管理器
责任
协作者
产生新线路信息
线路
画报表
线路,报表
画曲线
线路,曲线
CRC卡片
类:线路检测仪
责任
协作者
…
…
CRC卡片
类:线路采集信息
责任
协作者
记录新线路采集信息
…
CRC卡片
类:线路
责任
协作者
产生新线路
无
修改老线路
无
采集线路信息
线路采集信息,线路检测仪
CRC卡片
类:....
责任
协作者
…
…

优缺点

- 容易使用
- 模拟通讯
- 适合于考虑和设计对象和类,而不是识别它们
- 开发者必须有一定的经验、创造力和直觉

Booch对识别类的方法总结

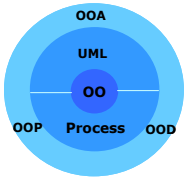
- 经典方法
  - Shlaher & Mellor : 实物 角色 事件 交互
  - Ross: 人 地点 物 组织 概念 事件
  - Coad & Yourdon: 结构 外部系统 设备 事件 用户角色 位置 组织机构单位
- 行为分析
  - 责任
  - 功能
  - 功能点：输出 查询 输入 文件 接口
- 领域分析
- Use case 分析
- CRC卡片
- 非正式英语
- 结构化分析



pink	moment
yellow	role, place, entity
green	thing, party, set
blue	catolog, list

Peter Coad的四色类图

1.3 审查与筛选



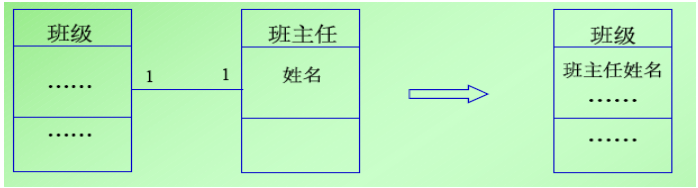
+ 1. 舍弃无用的对象

- 通过属性判断：
  - 是否通过属性记录了某些有用的信息？
- 通过服务判断：
  - 是否通过服务提供了某些有用的功能？
- 二者都不是——无用
- 在应用中,一个对象应该为一些其他的对象提供服务。

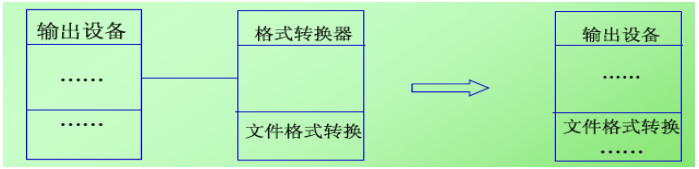
1.3 审查与筛选

+ 2. 精简对象

- **只有一个属性的对象**



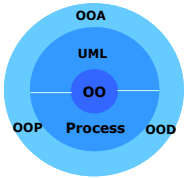
- **只有一个服务的对象**



- **总体上，对象必须具有多个属性和服务。也存在对象没有属性仅提供服务，或有属性无服务的情况。**

3. 与实现条件有关的对象，推迟到OOD考虑

- 系统责任所要求的某些功能
  - 例如系统安装、配置、信息备份、浏览——可能无法从问题域中找到相应的对象来提供这些功能，可在设计阶段考虑专门为它们增加一些对象，既把它们推迟到设计阶段考虑。
- 系统责任要求的某些功能可能与实现环境有关，也推迟到设计阶段考虑。
  - 与图形用户界面（GUI）系统、数据管理系统、硬件和操作系统有关的对象。



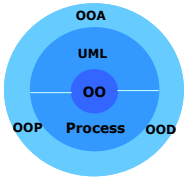
4. 名词筛选技术

- 相关性
  - 在问题域中名词不总是类或对象。
  - 对讨论需求是重要的
- 多个名词对应一个事物
  - 通常用几个不同的名词或名词短语描述同样的事。
  - 问题域中的某些事物实际上是另一种事物的附属品和一定意义上的抽象
- 用相同的名词捕获多个不同的概念

1.4 识别主动对象

- 考虑问题域和系统责任
  - 哪些对象需呈现主动行为？--按定义
- 从需求考虑系统的执行情况
  - 如果一切对象服务都是顺序执行的，那么首先执行的服务在哪个对象（唯一）？
  - 如果需要并发执行，每条控制线程的起点在哪个对象？这样的对象都是主动对象。
- 考虑系统边界
  - 哪些对象与参与者交互？
    - 如果一个交互是由参与者发起的，
    - 第一个处理该交互的对象是主动对象

1.5 类的命名



- 适合该类及其特殊类的全部对象实例
  - 汽车加摩托->机动车；还有马车->车辆
- 反映个体而不是群体
  - 书-书籍；船-船舶
- 使用名词，避免无意义的符号
- 使用问题域通用、规范的词汇
- 在中国：可用中、英文双重命名

1.6 建立类图的对象层

- 用类符号表示每个对象类
- 填写类描述模板
- 若发现新的属性与服务、关系，可以随时加到类符号中。

例题:习题管理系统

在一个公共习题库的支持下,使各科教师可以在系统中编写习题及其标准答案,并将编写的习题和答案加入题库;或者在题库中选取一组习题,组成一份作业或试卷,并在适当的时刻公布答案.学生可以在系统中完成教师布置的作业或试卷,也可以在题库中选择更多的题目练习.教师可以通过系统检查作业,学生可以在教师公布答案后对自己的练习进行核对.系统对题库管理,并检查教师和学生的权限.

教师

班

习题板

学生

习题解答

习题

练习本

试卷

试题解答

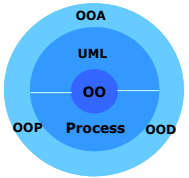
习题答案

题库

考试题板

To be continued…

2.识别属性



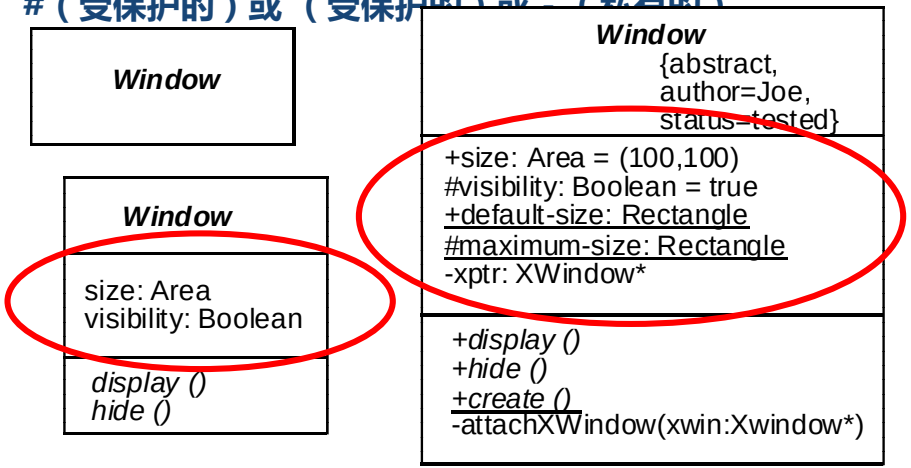
- 2.1 概念与表示法
- 2.2 属性的特征
- 2.3 识别属性
- 2.4 筛选
- 2.5 属性的命名和定位
- 2.6 属性的详细说明

2.1 概念与表示法

- 实例属性:是类的一个已命名的性质，它描述该性质的一个实例可以取的值的范围。
  - 抽象为属性的性质是与问题域相关的。
  - 从技术观点上，属性是一些变量（数据项或状态信息），包含它的每一个对象（实例）都具有自己的值。
  - 按照面向对象方法的封装原则，一个对象的属性和服务是紧密结合的，对象的属性只能由这个对象的服务存取。
- 类属性:是描述类的所有对象共同特征的一个数据项，对于任何对象实例，它的属性值都是相同的。

属性的表示法

- 类范围的属性与实例范围的属性。
  - 通过在类范围属性名和类型表达式画下划线的方式表示类范围的属性
- 属性可具有初始值和可见性
  - 可见性的值可为 中描述它们。可见性的值可为+（公有的）、（公有的）、#（受保护的）或（受保护的）或 -（私有的）

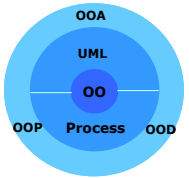


2.2 属性的特征

- Shlaer Mellor(1992)提出,Richard C.Lee增补
- 特征0：属性必须捕获与其对象所在的语义域相一致的特征。

- 特征1：任何时间一个实例为其每一个属性都精确地给出一个值。

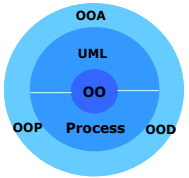
- 特征2：不能包含内部的结构。
  - 人的姓+名



- 特征3：属性必须是整个实体的特征。
  - 计算机的屏幕尺寸?显示器的屏幕尺寸
- 特征4：对象的属性必须与该对象相关。
  - 油罐 倒 油瓶 的容器中的容积属性
- 特征5：对象的属性值不能是与其有关的对象的值以其关系的值。
  - 人的薪水 结婚日期

To be continued…

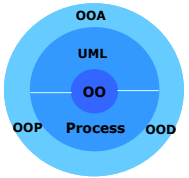
3.识别服务



- 3.1 概念与表示法
- 3.2 识别服务
- 3.3 审查与调整
- 3.4 识别对象的主动行为
- 3.5 服务的命名和定位
- 3.6 描述服务

3.1 概念与表示法

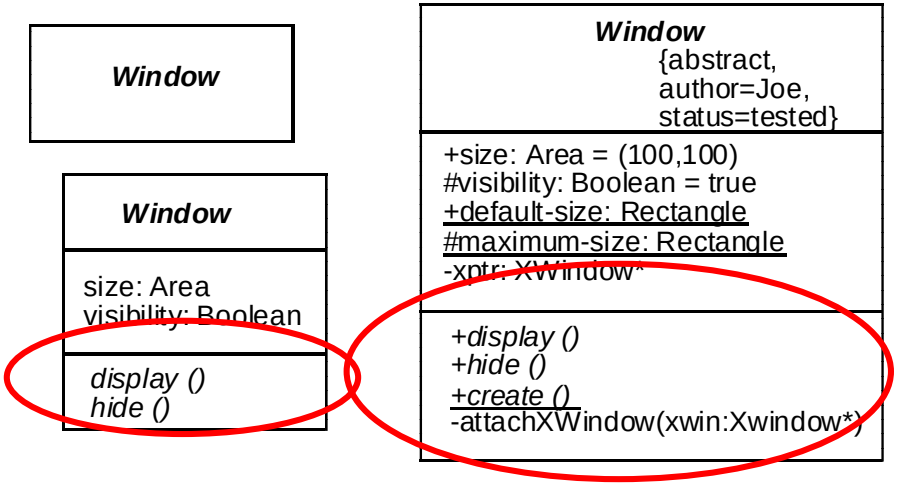
- 操作：是类的行为特征，用于描述为了引发相关行为的名称、类型、参数与约束。
  - 有名字和参数表; 有可见性和返回类型。
  - 可见性的取值为+(公有的)、#(受保护的)、-(私有的)或~(包内的)。
  - 可见性、参数表和返回类型可在类描述模板中描述。
  - 抽象操作:把在一个类中没有实现的操作（即没有提供方法）。
- 服务：一个类为其他类所做的工作。一个对象的服务是当其他对象借助消息传递机制请求它时，它愿意执行的所公布的或公开的工作。
- 方法：是操作的实现。当一个对象请求另一个对象的服务时对象完成的详细的动作集合（算法、过程）。



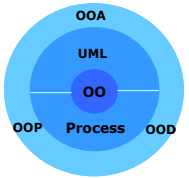
- 对象行为分类
  - 系统行为
    - 例：创建、删除、复制、转存
  - 对象自身的行为——算法简单的服务
    - 例：读、写属性值
  - 对象自身的行为——算法复杂的服务
    - 计算或监控

表示法

- 类范围的操作用带下划线的名字和类型表达式串表示。实例范围的操作是默认的，对其不用标记。
- 抽象操作带有标记“{abstract}”，或者把操作的特征标记写成斜体来表示它是抽象的。



3.2 识别服务



- OOA不考虑算法简单服务
  - 去创建、连接、访问、断开连接、删除等等。
- 考虑算法复杂的服务
  - 由对象提供的、在算法上复杂的业务服务（如要进行某些计算或监控操作）。

定义服务的策略与启发

- 考虑系统责任
  - 有哪些功能要求在本对象提供？
- 考虑问题域
  - 对象在问题域对应的事物有哪些行为？
- 分析对象状态
  - 在每种状态下对象可能发生什么行为？对象状态的转换，是由哪些服务引起的？
- 追踪服务的执行路线
  - 模拟服务的执行，并在整个系统中跟踪。
- 用动词识别服务。
- 识别出计算、监视和查询类的服务。

3.3 审查与调整

- 审查对象的每个服务
  - 是否真正有用
    - 是否直接提供系统责任所要求的某项功能？
  - 或者
    - 响应其它服务的请求间接地完成这种功能的某些局部操作？
  - 调整——取消无用的服务
  - 是不是高内聚的
    - 一个服务只完成一项单一的、完整的功能
  - 调整——拆分或合并

3.4 认识对象的主动行为

- 考虑问题域对象行为是被引发的，还是主动呈现的？
- 与参与者交互的对象服务
- 完成最外层功能的对象服务外层与内层是请求与被请求的关系
- 服务执行路线逆向追踪找到了主动服务就等于找到了主动对象。

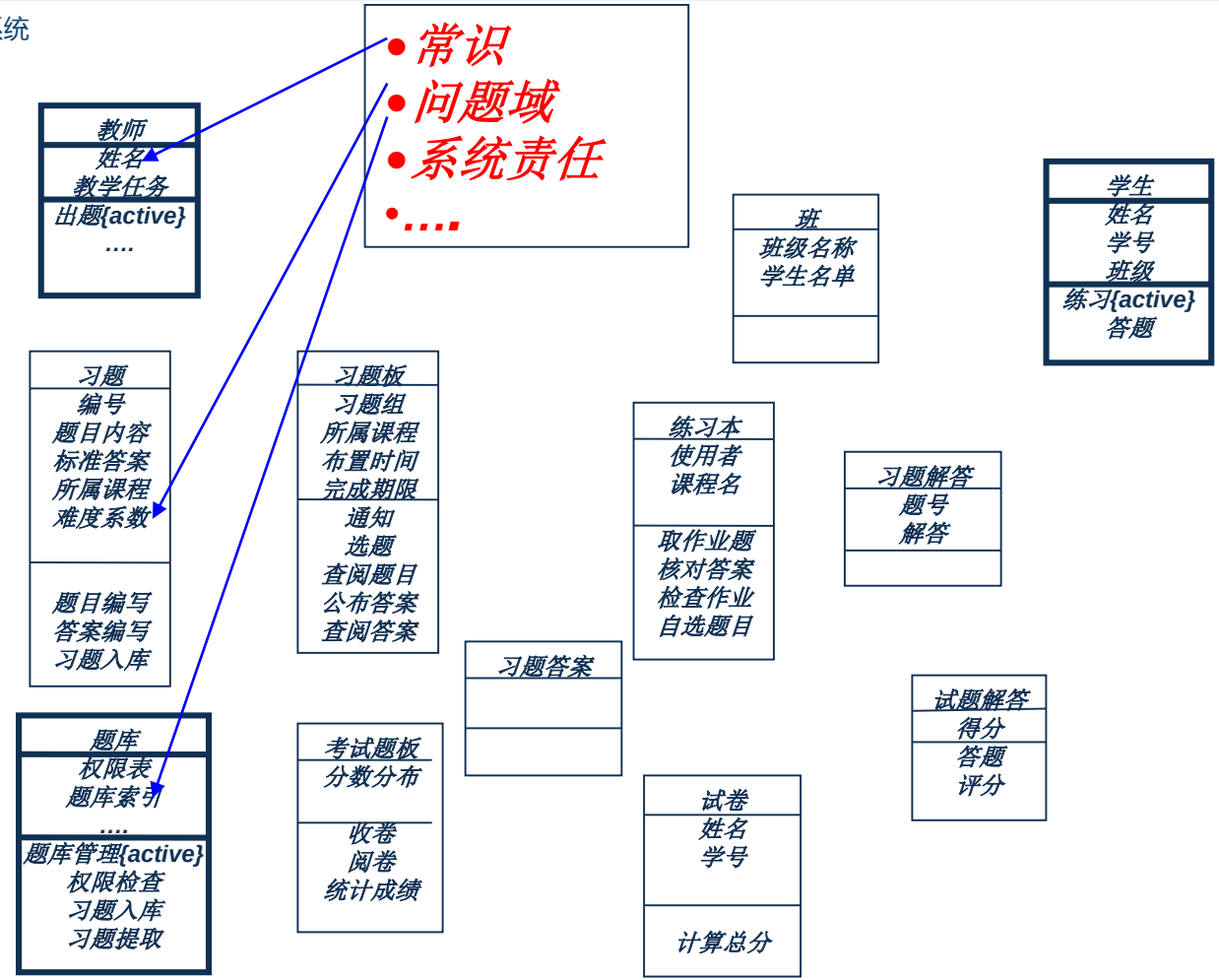
3.5 服务的命名和定位

- 命名：动词或动宾结构; 外向性
- 定位：
  - 与实际事物一致
    - 例：售货员——售货，商品——售出
  - 在泛化中的位置
    - ——适合类的全部对象实例

3.6 描述服务

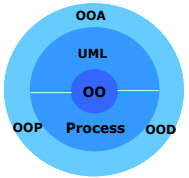
- 把每个对象的服务都填写到相应的类符号中。
  - 在类描述模板中，写出：
    - 说明服务的职责
    - 服务原型（消息的格式）
    - 消息发送（指出在这个服务执行时，需要请求哪些别的对象服务，即接收消息的对象类名以及执行这个消息的服务名）
    - 约束条件：如果该服务的执行有前置条件、后置条件，以及执行时间的要求等其它需要说明的事项，则在这里加以说明。
    - 实现服务的方法（文字、活动图或流程图）。

例题:题库管理系统


常识
问题域
系统责任
....
教师
姓名
教学任务
出题{active}
....
习题
编号
题目内容
标准答案
所属课程
难度系数
题目编写
答案编写
习题入库
题库
权限表
题库索引
....
题库管理{active}
权限检查
习题入库
习题提取
习题板
习题组
所属课程
布置时间
完成期限
通知
选题
查阅题目
公布答案
查阅答案
考试题板
分数分布
收卷
阅卷
统计成绩
习题答案
试卷
姓名
学号
计算总分
班
班级名称
学生名单
练习本
使用者
课程名
取作业题
核对答案
检查作业
自选题目
习题解答
题号
解答
试题解答
得分
答题
评分
学生
姓名
学号
班级
练习{active}
答题

To be continued…

4.定义关系

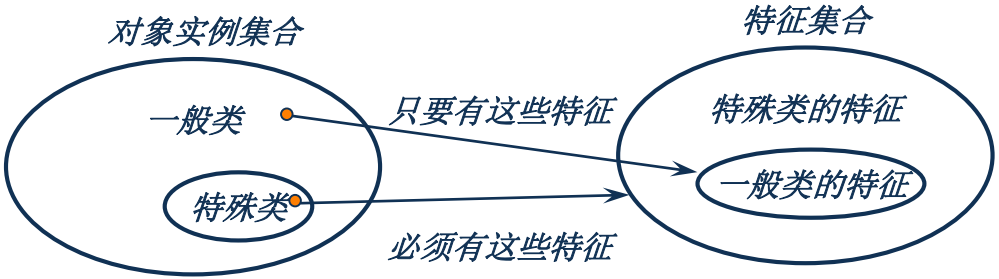


+ 4.1 泛化关系
+ 4.2 关联关系
+ 4.3 聚合关系
+ 4.4 依赖关系
+ 4.5 接口与实现

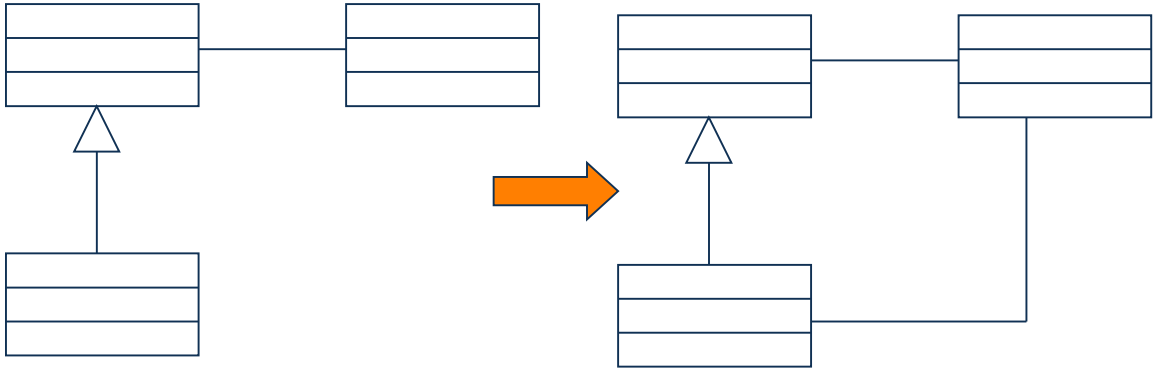
4.1 泛化

+ 定义

- 如果类A具有类B的全部属性和全部操作,而且还具有自己特有的一些属性或操作,则A叫B的特殊类,B叫做A的一般类,A与B之间的关系成为泛化关系.
- 如果类A的全部对象都是类B的对象,且类B中存在不属于类A的对象,则A是B的特殊类,B是A的一般类,A与B之间的关系称为泛化关系.

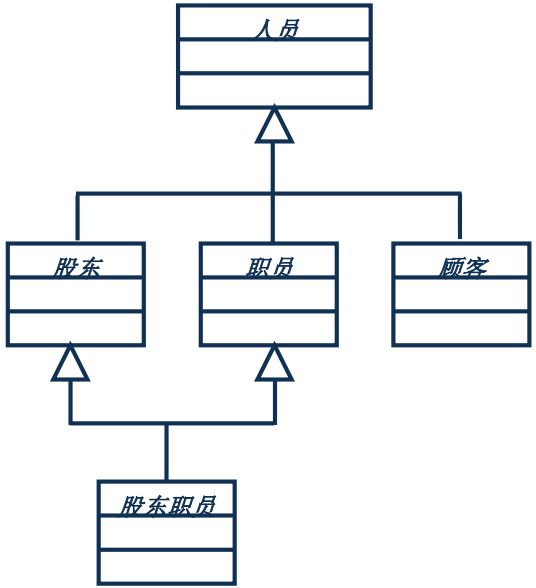


- 性质：
  - 后代将具有祖先的所有的关联。

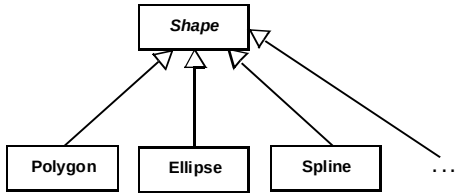


- a．反对称性
  - 如果对象A是对象B的后代，那么B将不会与A有“是一个”关系（对象B不是对象A的后代）。例如，Employee是一个Person，但并不是所有的人都是雇员。
- b．传递性
  - 如果对象A“是一个”对象B，对象B“是一个”对象C，那么对象A“是一个”对象C。
    - 例如，SalesPerson是一个Employee，那么SalesPerson也是一个Person。
  - 在OOA模型中建立泛化，是为了使系统模型更清晰地映射问题域中事物的分类关系。它把具有泛化关系的类组织在一起，可以简化我们对复杂系统的认识，从而增加了软件的可维护性和适应变化的灵活性。

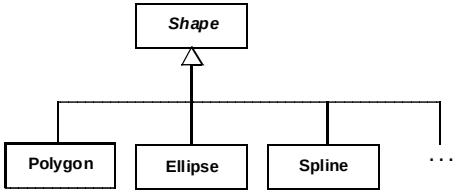
基本表示法



深入表示法(1.X)

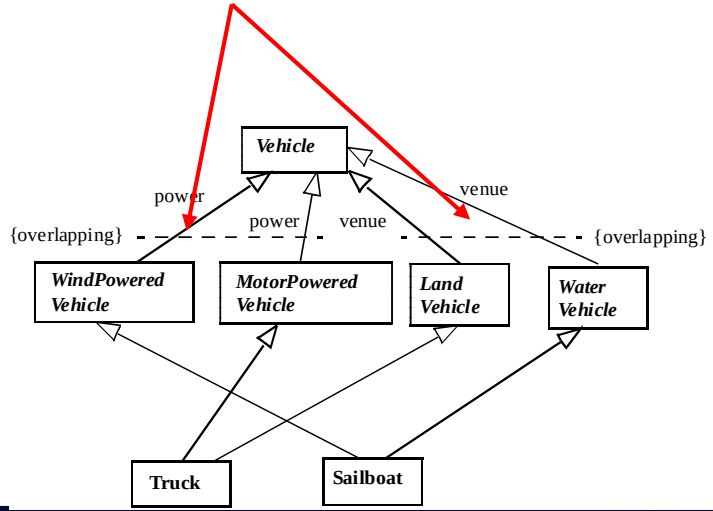


Separate Target Style

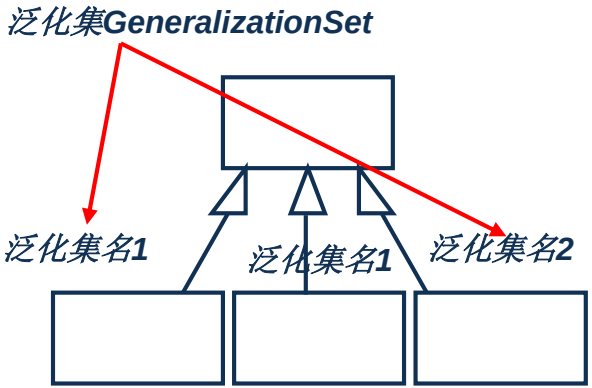


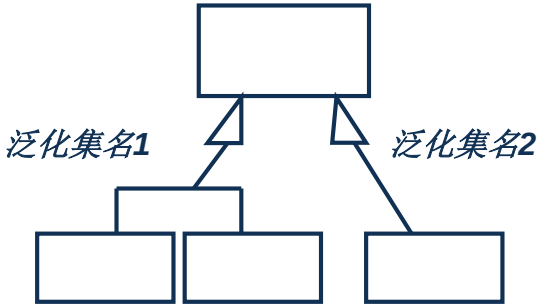
Shared Target Style

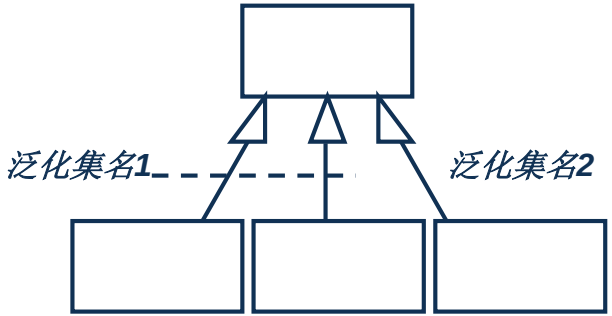
Discriminator(区分器)



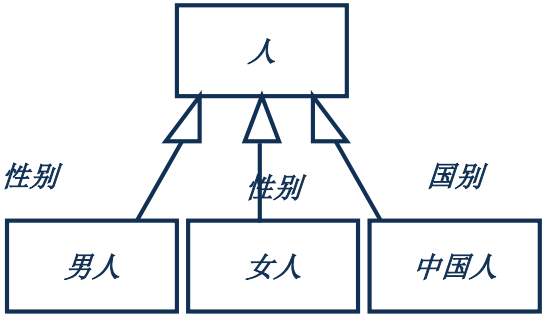
深入表示法(2.0)

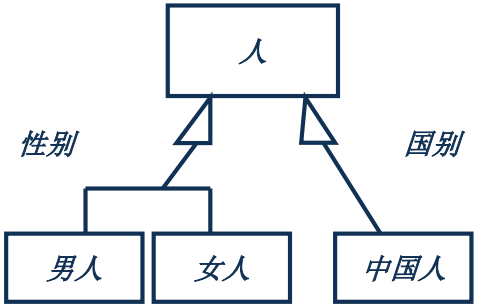

泛化集GeneralizationSet
泛化集名1
泛化集名1
泛化集名2

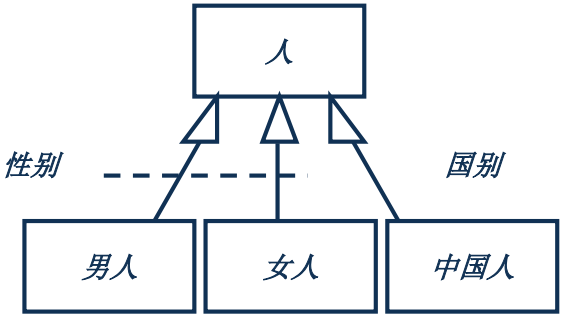

泛化集名1
泛化集名2


泛化集名1
泛化集名2

举例

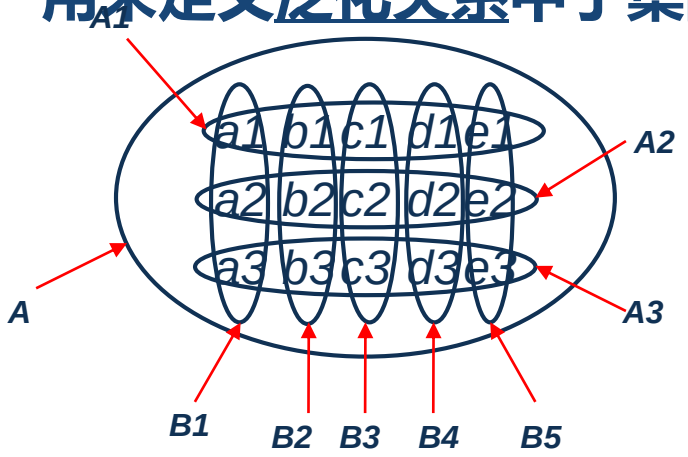


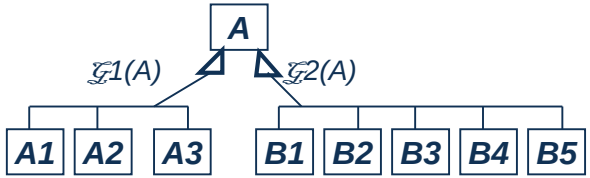




泛化集(GeneralizationSet)

- 用来定义泛化关系中子集的集合.





$A=\{a1,b1,c1,d1,e1,a2,b2,c2,d2,e2,a3,b3,c3,d3,e3\};$
$A1=\{a1,b1,c1,d1,e1\};$
$A2=\{a2,b2,c2,d2,e2\};$
$A3=\{a3,b3,c3,d3,e3\};$
$B1=\{a1,a2,a3\};$
$B2=\{b1,b2,b3\};$
$B3=\{c1,c2,c3\};$
$B4=\{d1,d2,d3\};$
$B5=\{e1,e2,e3\};$

$2^{A}=\{A1,A2,A3,B1,B2,B3,B4,B5,\ldots\ldots\};$

$G(A)=\{<A,A1>,<A,A2>,<A,A3>,<A,B1>,<A,B2>,<A,B3>,<A,B4>,<A,B5>\};$

$G1(A)=\{<A,A1>,<A,A2>,<A,A3>\};$

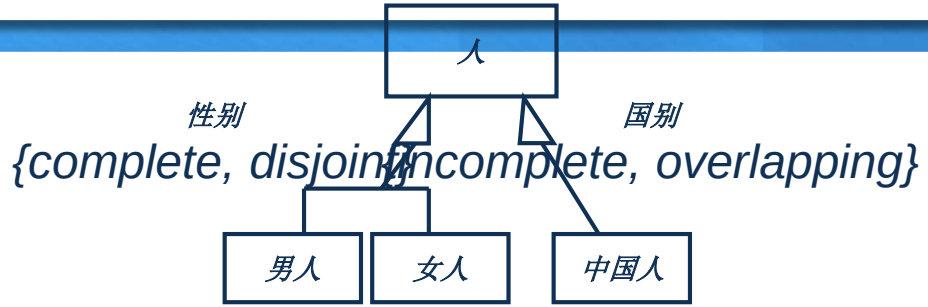
$G2(A)=\{<A,B1>,<A,B2>,<A,B3>,<A,B4>,<A,B5>\};$

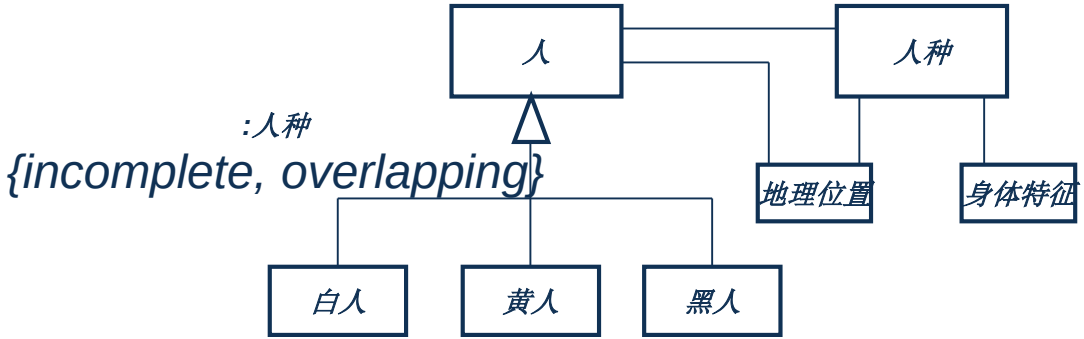
泛化集的约束

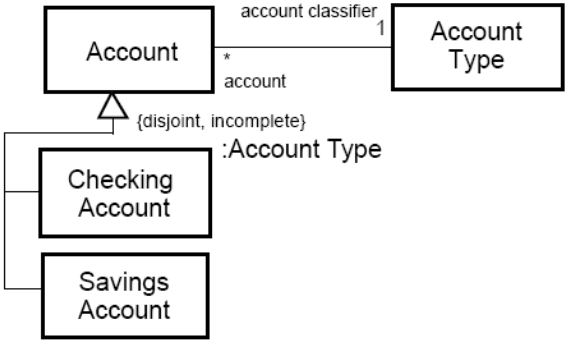
+ {complete, disjoint}
  - 表明泛化集是覆盖的,并且某些特殊类没有共同的实例.
+ {incomplete, disjoint}
  - 表明泛化集不是覆盖的,并且某些特殊类没有共同的实例.
+ {complete, overlapping}
  - 表明泛化集是覆盖的,并且某些特殊类具有共同的实例.
+ {incomplete, overlapping}
  - 表明泛化集不是覆盖的,并且某些特殊类具有共同的实例.
+ 默认 {incomplete, disjoint}

举例

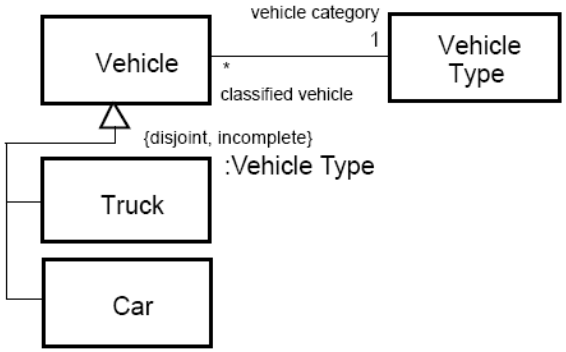

OOA
UML
OO
OOP
Process
OOD


人
性别
国别
{complete, disjoint}{incomplete, overlapping}
男人
女人
中国人

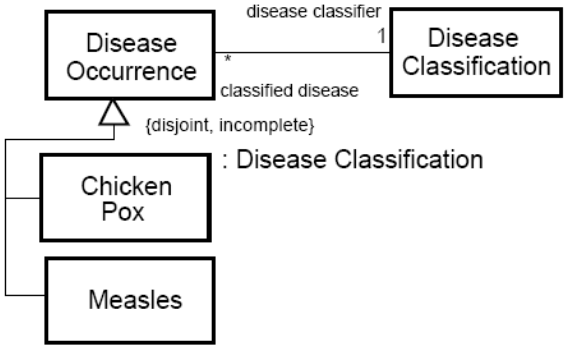

人
人种
:人种
{incomplete, overlapping}
地理位置
身体特征
白人
黄人
黑人



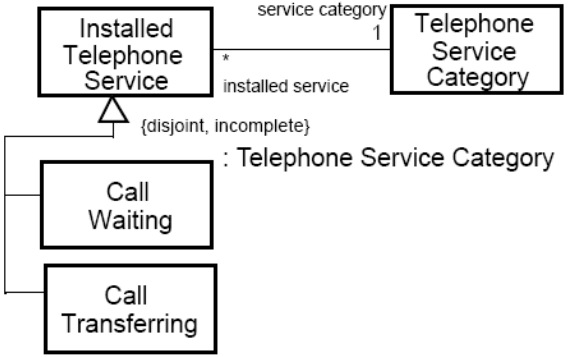
(a) Bank account/account type example



(b) Vehicle/vehicle type example



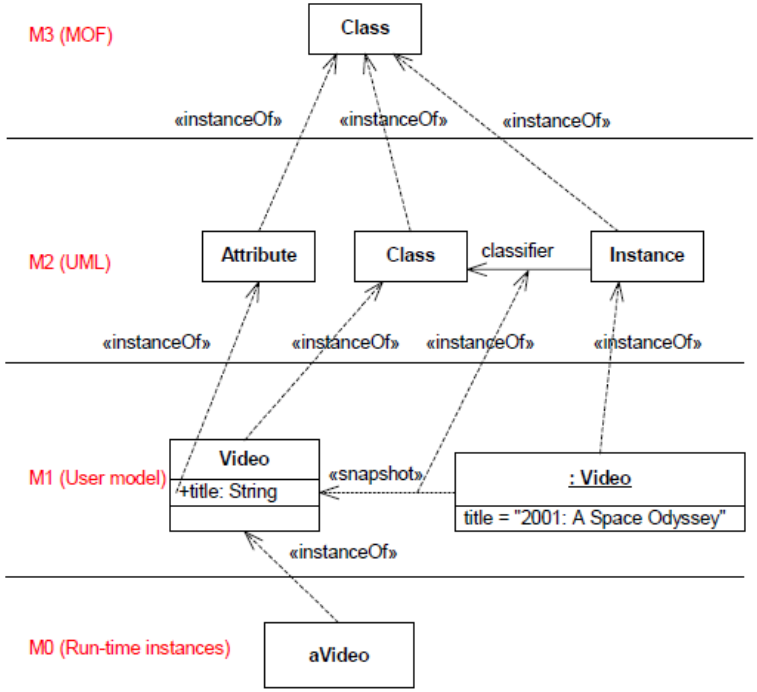
(c) Disease Occurrence/Disease Classification example



(d) Telephone service example

- *幂类型(power type)其实例是另外一个类的子类的类.*

元类 meta class

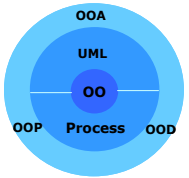


OMG Unified Modeling Language™ (OMG UML), Infrastructure
Version 2.2

识别泛化

- 1）学习当前领域的分类学知识
- 2）按常识考虑事物的分类
- 3）利用泛化的定义
  - 两个类
    - 集合包含
    - 特征包含
- 4）看两个类的对象之间是否有“是一个”关系
- 5）考察类的属性与服务
- 6）考虑领域范围内的复用

4）看两个类的对象之间是否有“是一个”关系



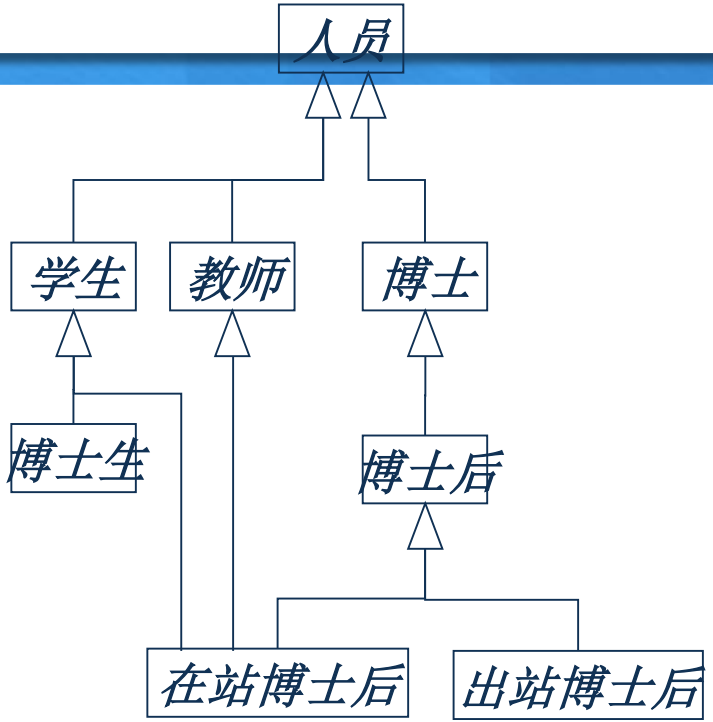
- **Is A 关系表**
  - **考察A是一个B,B是一个A?**
    - **若都是——同义**
    - **若都从来不是——没有泛化关系**
    - **A总是B,B有时是A,则A是一个B**

思考题

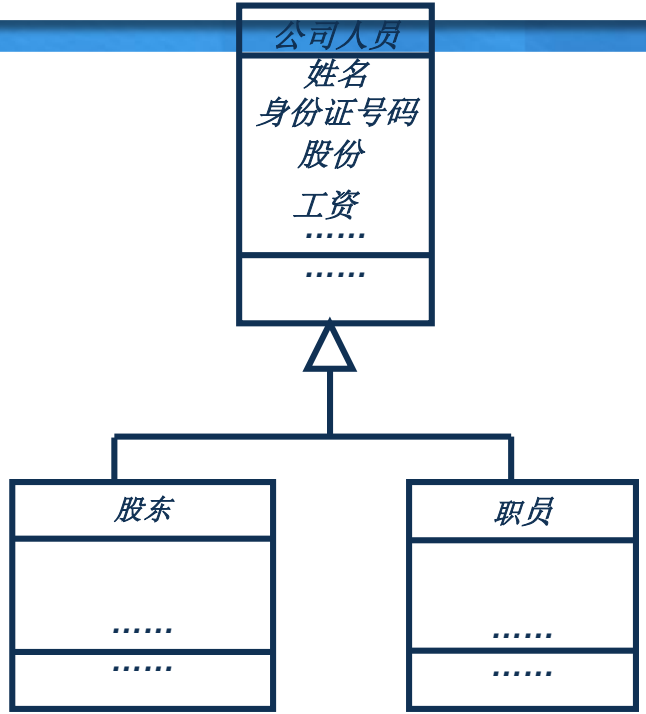
- 请画出以下类泛化关系
  - 学生
  - 博士生
  - 博士
  - 博士后
  - 在站博士后
  - 出站博士后
  - 教师
  - 人员

A是一个B?	人员	学生	教师	博士	博士生	博士后	在站博士后	出站博士后
人员	——	有时是	有时是	有时是	有时是	有时是	有时是	有时是
学生	总是	——	有时是	有时是	有时是	有时是	有时是	从不是
教师	总是	有时是	——	有时是	有时是	有时是	有时是	有时是
博士	总是	有时是	有时是	——	从不是	有时是	有时是	有时是
博士生	总是	总是	有时是	从不是	——	从不是	从不是	从不是
博士后	总是	有时是	有时是	总是	从不是	——	有时是	有时是
在站博士后	总是	总是	总是	总是	从不是	总是	——	从不是
出站博士后	总是	从不	有时是	总是	从不是	总是	从不是	——

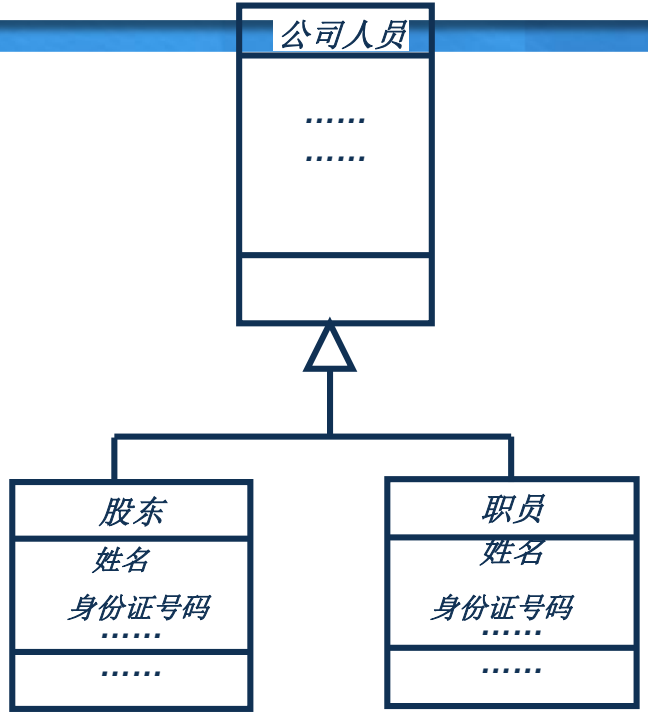

OOA
UML
OO
OOP
Process
OOD


人员
学生
教师
博士
博士生
博士后
在站博士后
出站博士后

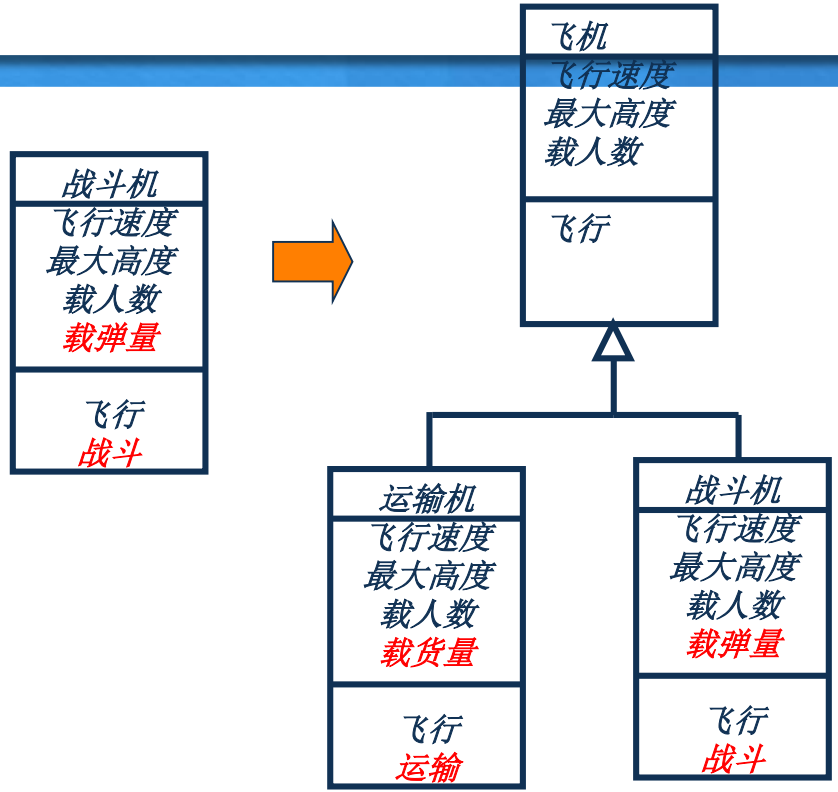
5）考察类的属性与服务




OOA
UML
OO
OOP
Process
OOD

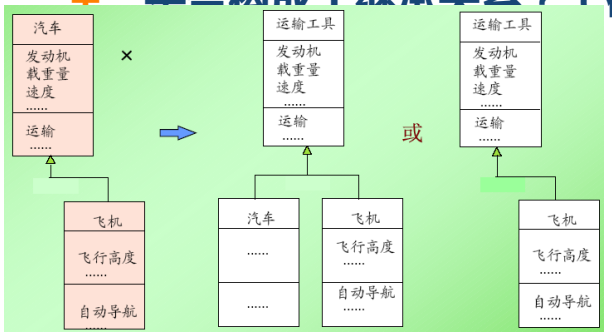

公司人员
.....
.....
股东
姓名
身份证号码
.....
.....
职员
姓名
身份证号码
.....
.....

6)考虑领域范围内的复用



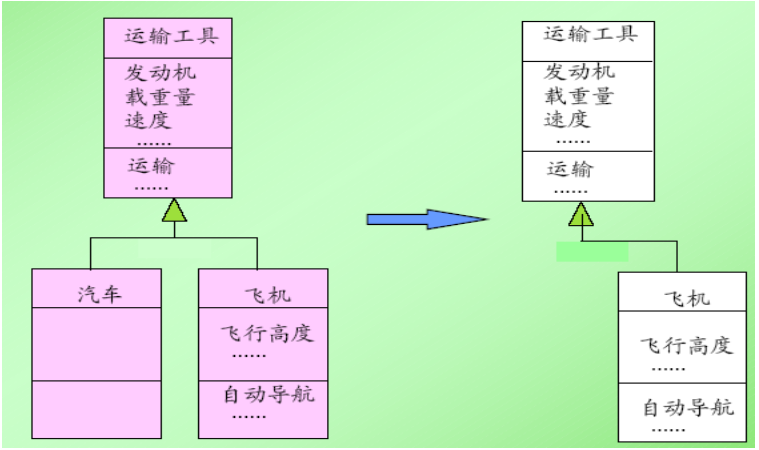
4、审查与调整

- 问题域与系统责任是否需要这样的分类？
  - （例：书—善本书（内容有用，流传稀少，校刻精良，具有文物、学术或艺术价值之本 ）
- 是否符合分类学的常识？
- 是否构成了继承关系？（确实继承了一些属性或服务）

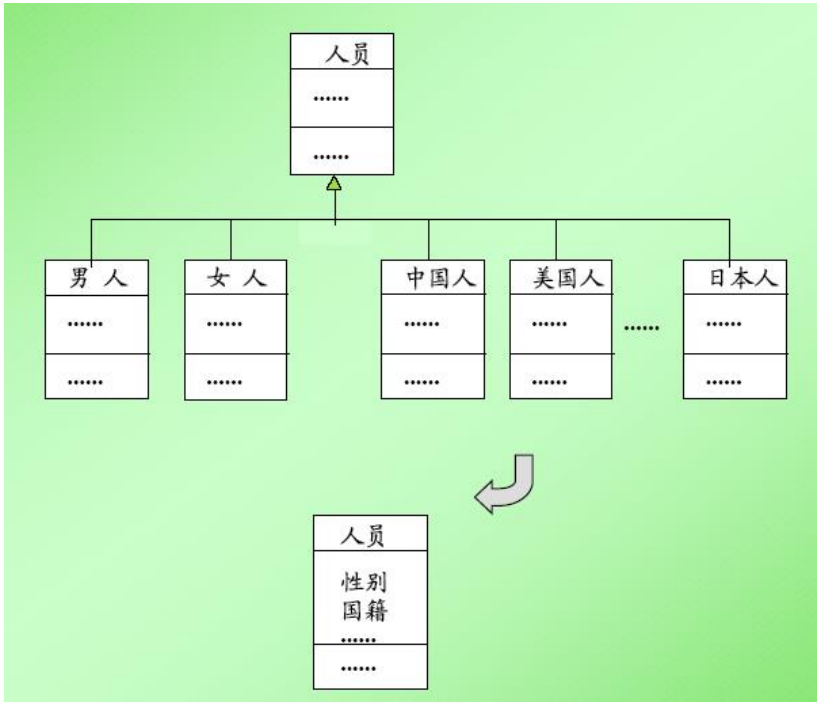


一般-特殊结构的简化

+ （1）取消没有特殊性的特殊类
  - 从一般类划分出太多的特殊类，使系统中类的设置太多，增加了系统的复杂性；
  - 建立过深的继承层次，增加了系统的理解难度和处理开销。

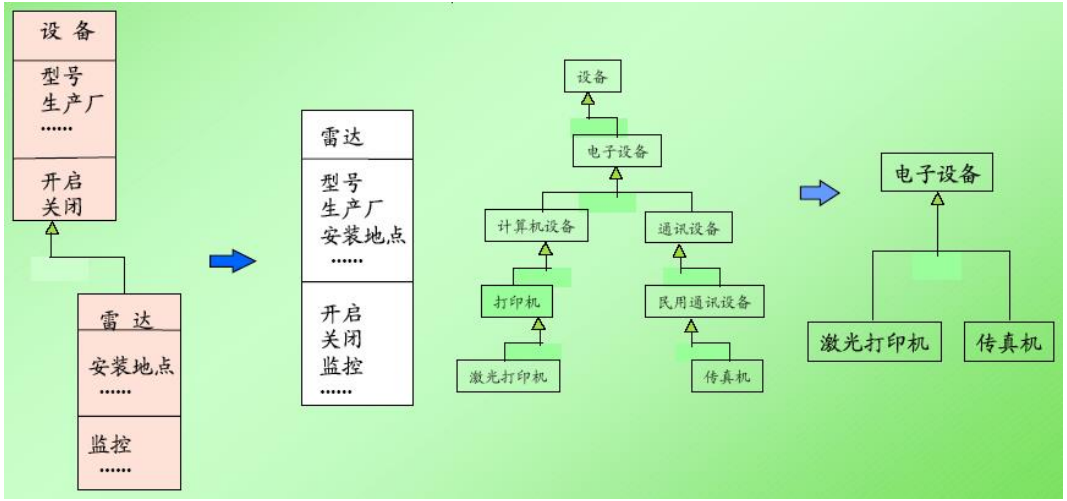


（2）增加属性简化一般 - 特殊结构

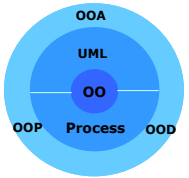


（3）取消用途单一的一般类，减少继承层次

- 一般类存在的理由
  - 有两个或两个上以上的特殊类
  - 需要用它创建对象实例
  - 有助于软件复用



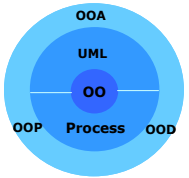
5、调整对象层和特征层



定义泛化的活动，将使分析员对系统中的对象类及其特征有更深入的认识。在很多情况下，随着泛化的建立，需要对类图的对象层和特征层作某些修改，包括增加、删除、合并或分开某些类，以及增、删某些属性与服务或把它们移到其它类。

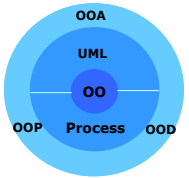
To be continued…

4.2识别关联

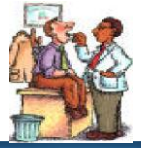


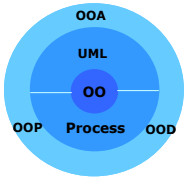
- 概念与表示法
- 关联的类型
- 识别关联
- 对象层、特征层的增补及关联说明

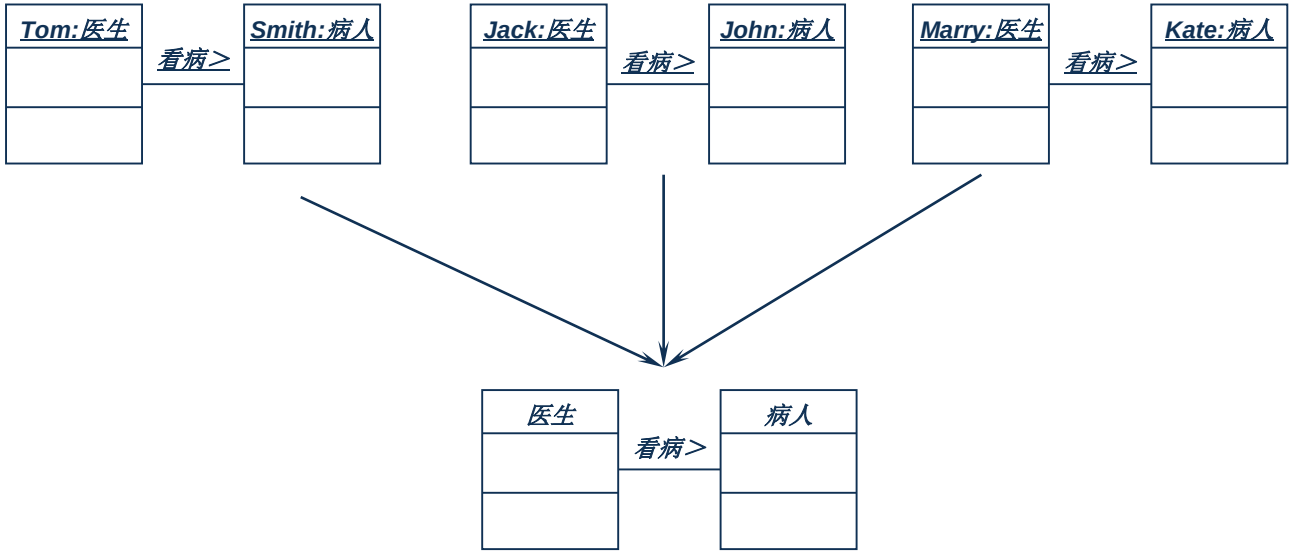
概念与表示法

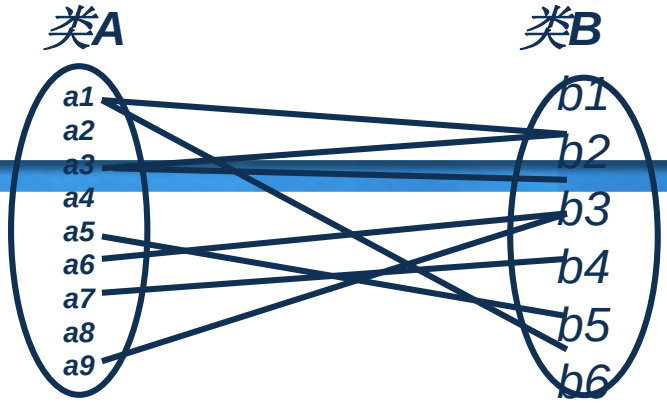


- **关联**
  - **关联描述了引用类的实例的元组的集合。**
  - **两个类之间可以存在多个关联**
- **链**
  - **链是对象引用的元组。在最常见的情况下，它是一对对象引用。它是关联的一个实例。**




OOA
UML
OO
OOP
Process
OOD

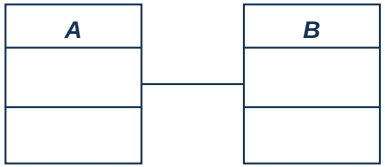

Tom:医生
看病>
Smith:病人
Jack:医生
看病>
John:病人
Marry:医生
看病>
Kate:病人
医生
看病>
病人

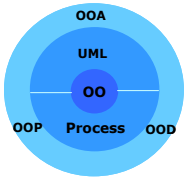


<a1,b1>,<a1,b6>,<a3,b1>,<a3,b2>

……..

A×B的子集

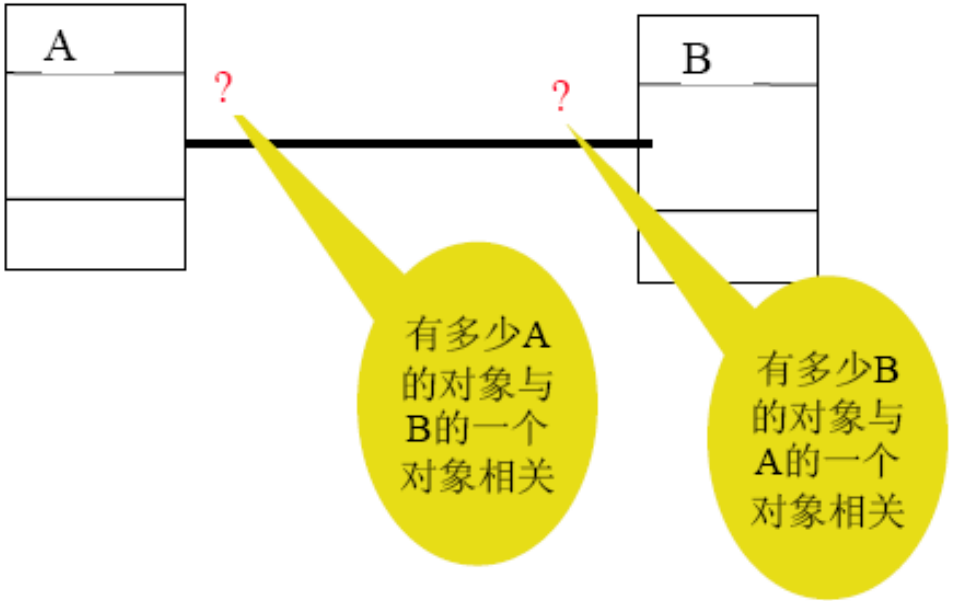




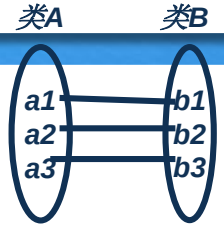
+ 多重性

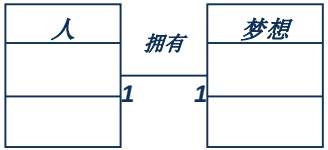
- 多重性是非负整数集的一个子集。
- 另一端上的多重性是指，对于本端的一个对象，需要另一端对象的个数。
- 其中的下限和上限都是文字整型值，说明从下限到上限的整数闭区间。此外星号（*）可以用于上限，表明不限制上限。
- 如果多重性规约由单个的（*）构成，那么它就表明了无穷的非负正整数的范围，也即它等价于0..*。

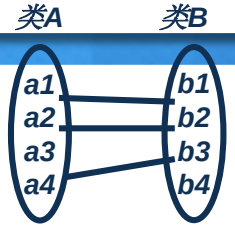

OOA
UML
OO
OOP
Process
OOD

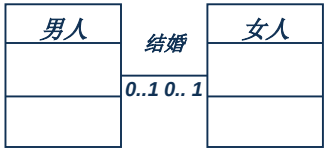

A
B
?
?
有多少A的对象与B的一个对象相关
有多少B的对象与A的一个对象相关

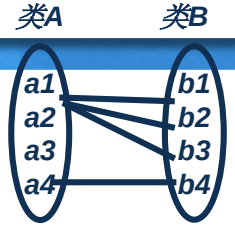

OOA
UML
OO
OOP
Process
OOD

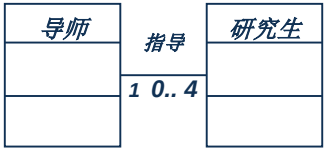

类A
类B
a1
a2
a3
b1
b2
b3

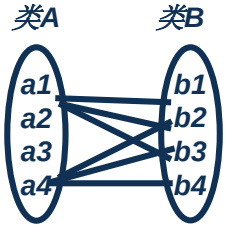

人
拥有
梦想
1
1

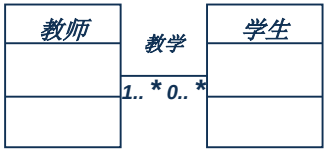

类A
类B
a1
a2
a3
a4
b1
b2
b3
b4

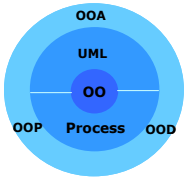

男人
结婚
女人
0..1
0.. 1


类A
类B
a1
a2
a3
a4
b1
b2
b3
b4


导师
指导
研究生
1
0.. 4


类A
类B
a1
a2
a3
a4
b1
b2
b3
b4

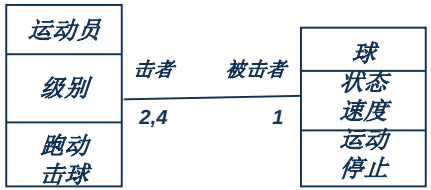

教师
教学
学生
1.. *
0.. *

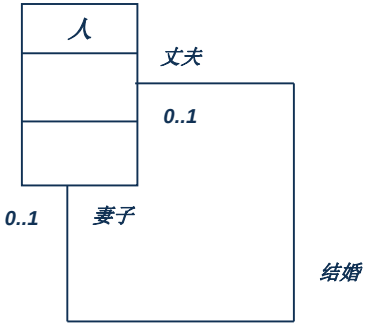


+ 角色

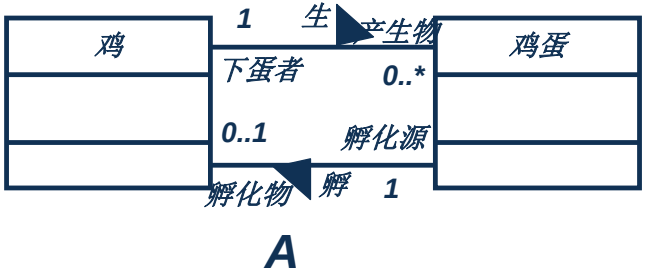
- 在关联的每一个端点上有一个角色。每一个角色具有一个角色名，用来描述其类被其他的类看作是什么。
- 当需要强调一个类在一个关联的确切含义时，使用关联角色名。
- 如果使用角色名，就可以省略关联名。
- 角色名也决定了其类的多重性；就是说，该类与其他类的一个实例相关联的实例的数量。

举例

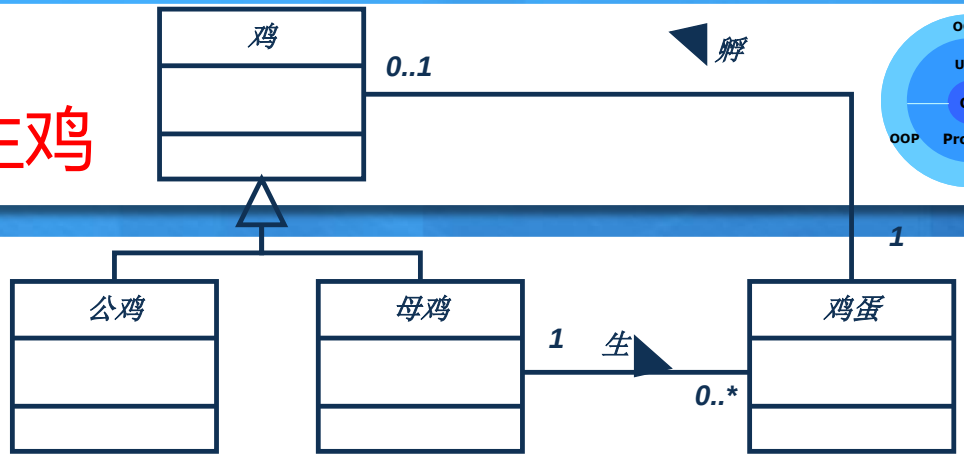




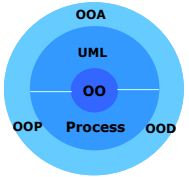
思考题
鸡生蛋,蛋生鸡



A

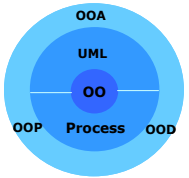


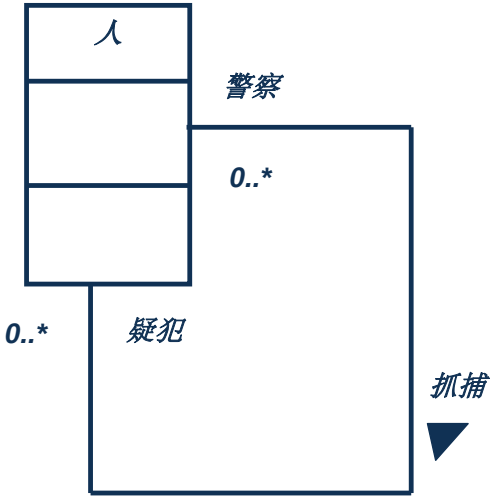
B



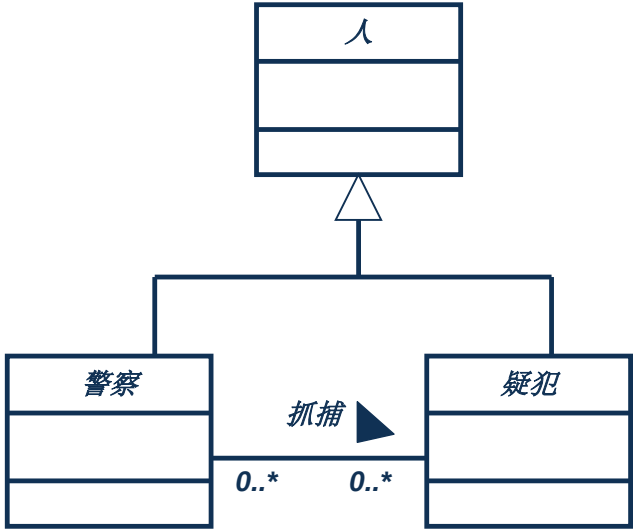
思考题

+ 下列哪幅图最能精确描述警察与疑犯的关系?





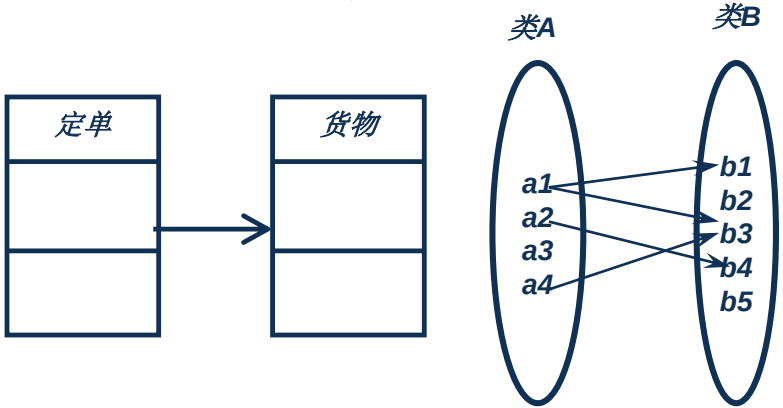
A



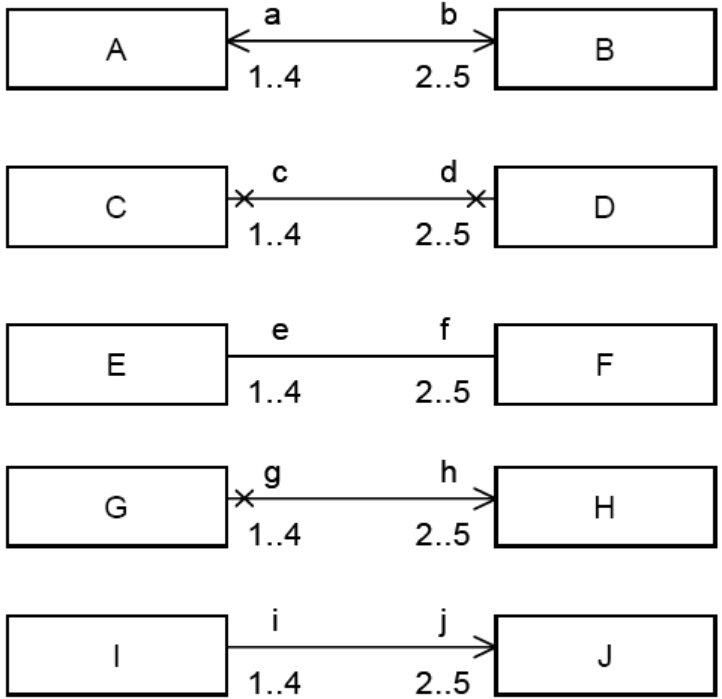
B

+ 导航性

- 导航性限制了关联上的访问方向
- 双向的导航不必画箭头



UML2.0中对导航性的表示



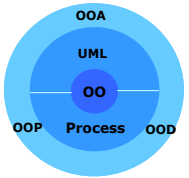
To be continued…

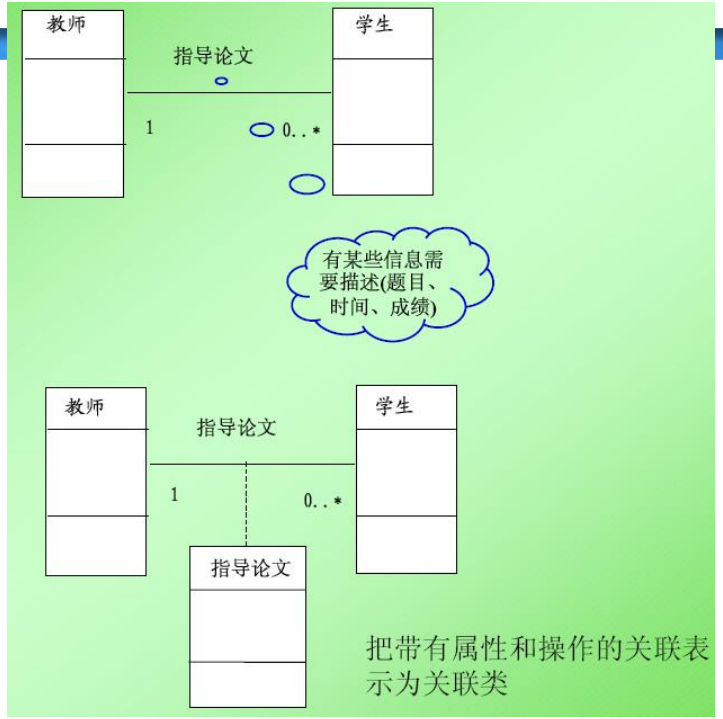
关联的类型

- 关联类
- 限定关联
- 异或关联
- 有序关联
- 多元关联
- 聚合

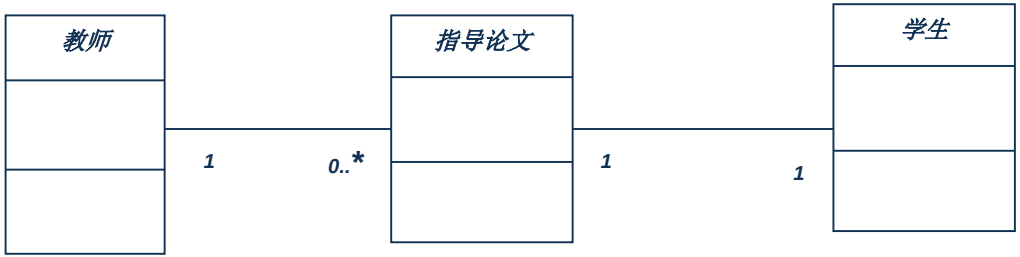
关联类

+ 具有关联和类的特征的建模元素。关联类既可以被看作是具有类的性质的关联，也可以被看作为具有关联性质的类。
+ 把关联类表示成一个用虚线连接到关联路径的类符号。
+ 关联和关联类符号表示同模型元素，它们的名字相同。名字可以放置在关联上或类符号中（但它们的名字必须相同）。
+ 如果一个类的一个属性可以同时有多个值，就考虑使用关联类。
  - 例如，一个人的薪水。

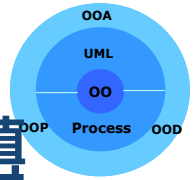

OOA
UML
OO
OOP
Process
OOD


教师
指导论文
学生
1
0..*
有某些信息需要描述(题目、时间、成绩)
教师
指导论文
学生
1
0..*
指导论文
把带有属性和操作的关联表示为关联类

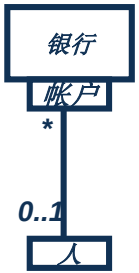
关联类到二元关联的转换

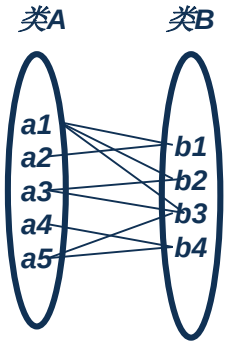


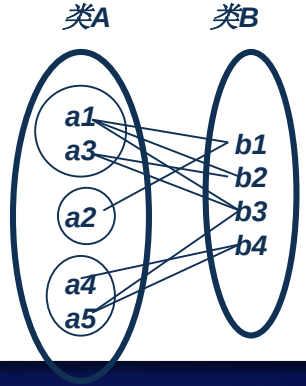
限定关联（Qualifier）



+ 二元关联的属性或者属性列表,在此关联中,属性的值从整个对象集合里选择一个唯一的关联对象或者关联对象集.



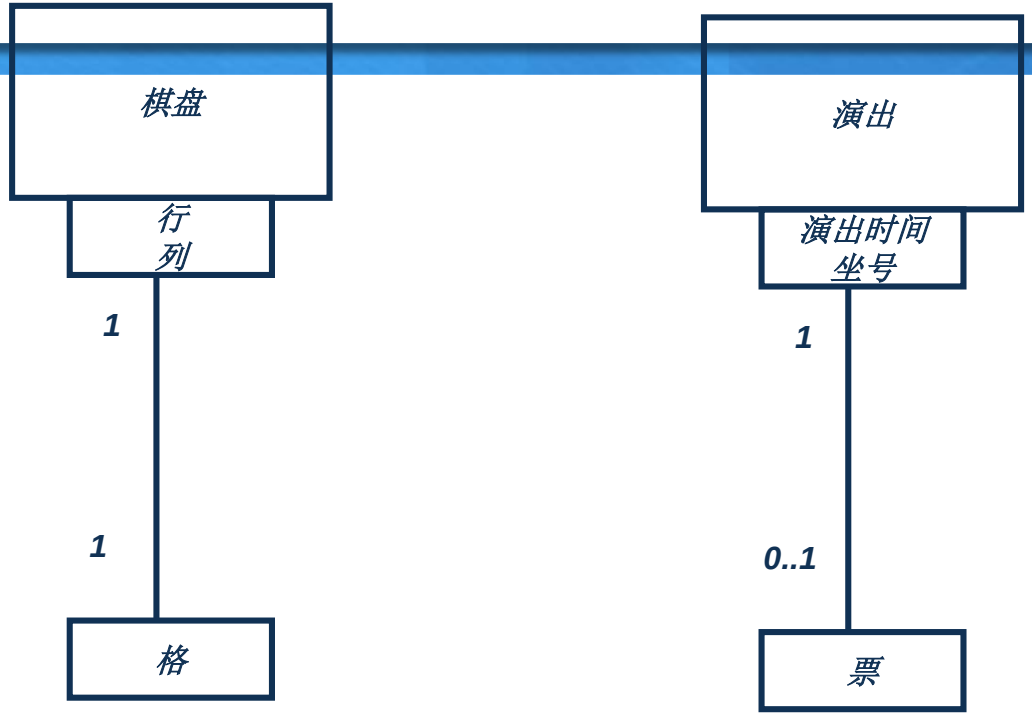




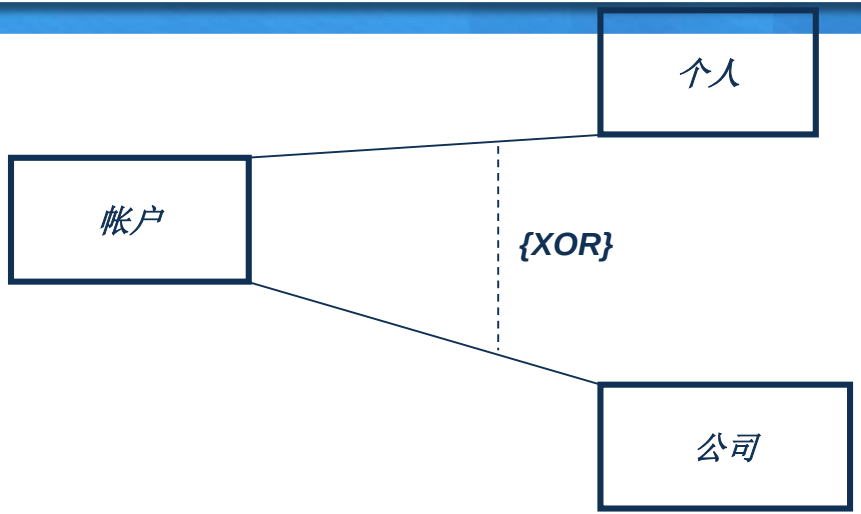
限定关联（Qualifier）

- 限定符可以是目标类的一个属性
- 但通常是关联的属性
- 该属性的值限定了关联


OOA
UML
OO
OOP
Process
OOD

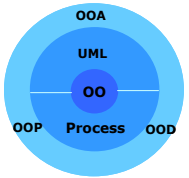

棋盘
行
列
1
1
格
演出
演出时间
坐号
1
0..1
票

异或关联



有序关联

建立关联的策略



- 认识对象之间的静态联系
  - 考虑问题域和系统责任——哪些类的对象实例之间的关系需要在系统中表达。
    - 物理位置或亲密度
    - 包含
    - 通信
    - 享受服务
    - 存储信息
- 认识关联的属性与操作
  - 对于考虑中的每一种关联，进一步分析它是否应该带有某些属性和操作。就是说，是否含有一些仅凭一个简单的关联不能充分表达的信息。
- 分析并表示实例连接的多重性
  - 从连接线的每一端，看本端的一个对象可能与另一端的几个对象发生连接，把结果标注到连接线的另一端。
- 对多对多的关联的处理

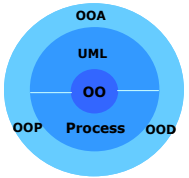
多对多关联的问题

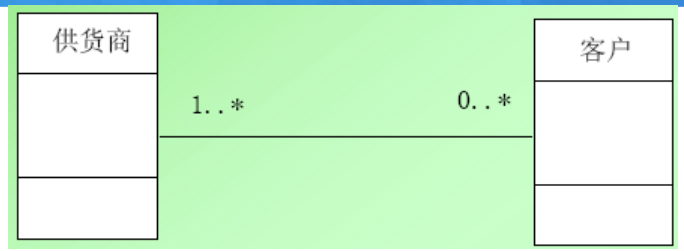
- 用面向对象语言实现为两个数组指针
- 用关系数据库实现为多对多的表
  - 多对多的表的信息是冗余的
  - 两个多对多关系转化为不符合第四范式的表

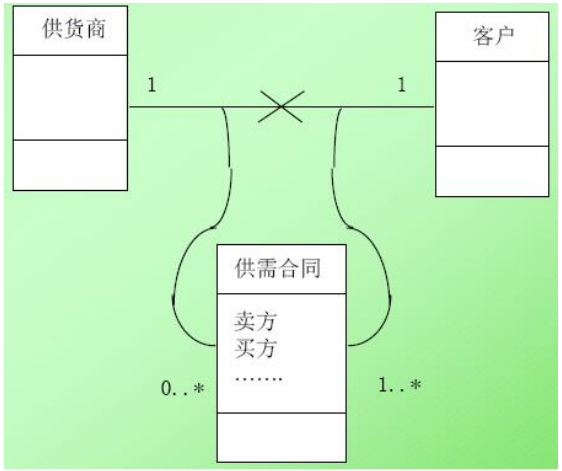
降低多对多关联的多重性

- 限定关联
  - 用面向对象语言实现为带有参数的筛选函数
  - 用关系数据库实现为先筛选后连接
- 引入新的中间类
  - 切断多对多类
  - 从问题域中寻找新的中间类，它与双方都构成1对多关系
  - 若问题域中找不到相应的类，可构想一个

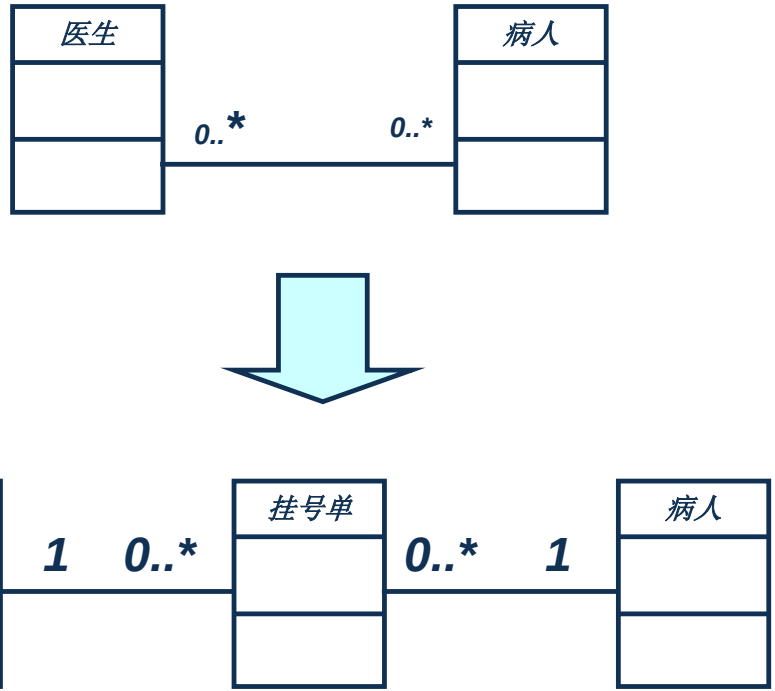
降低多对多关联的多重性



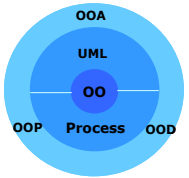


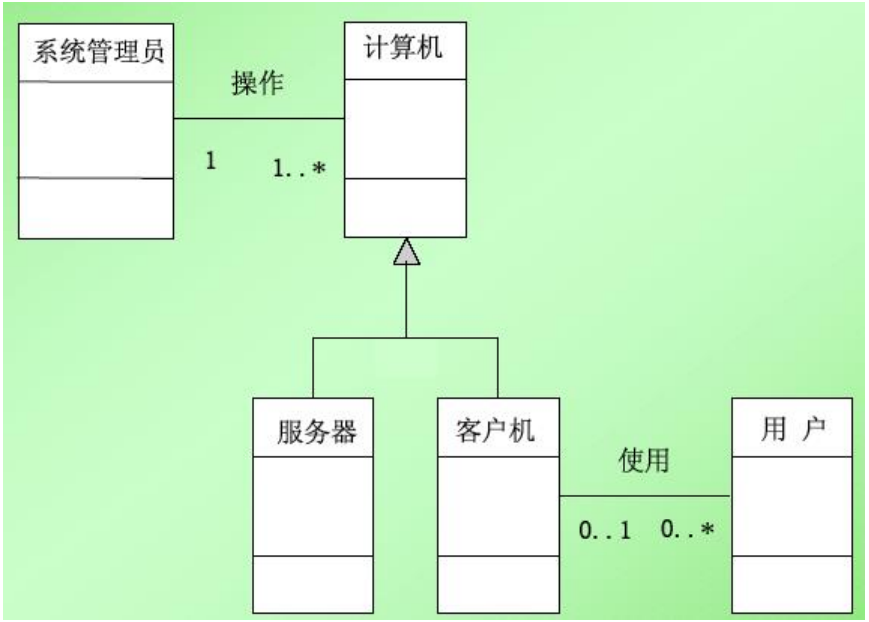



OOA
UML
OO
OOP
Process
OOD

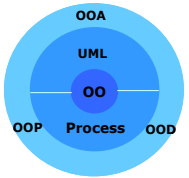

医生
0..*
0..*
病人
医生
1
0..*
挂号单
0..*
1
病人

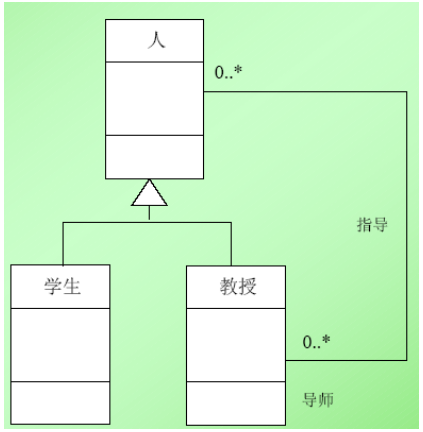
泛化与关联的综合运用
例题1

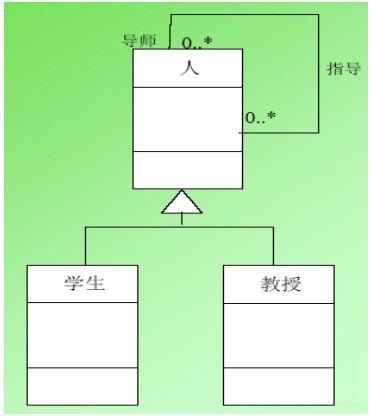


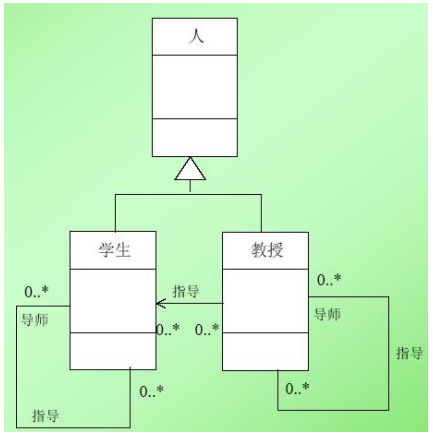


泛化与关联的综合运用
例题2:









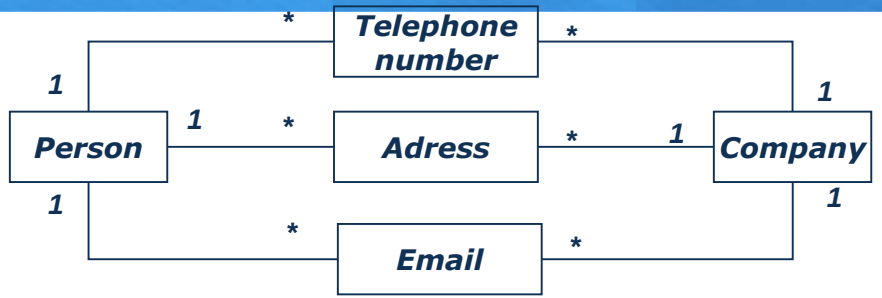
To be continued…

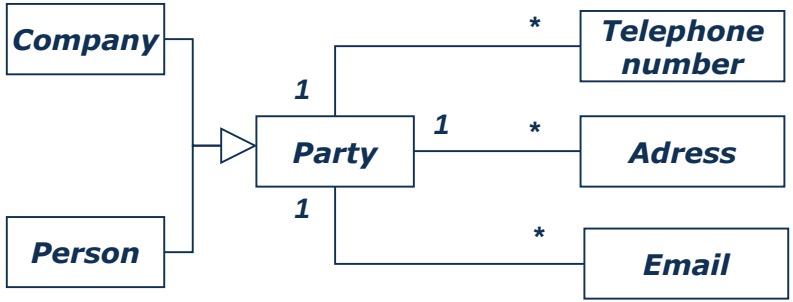
分析模式

- Martin Fowler
- 通用分析模式
  - Accountability责任(party Organization Hierarchies Organization structure ...)
  - Observation and Measurements观察与测量
  - Rederring to Objects对象的引用
- 领域分析模式
  - 财务领域
  - 规划
  - 商业
  - ...

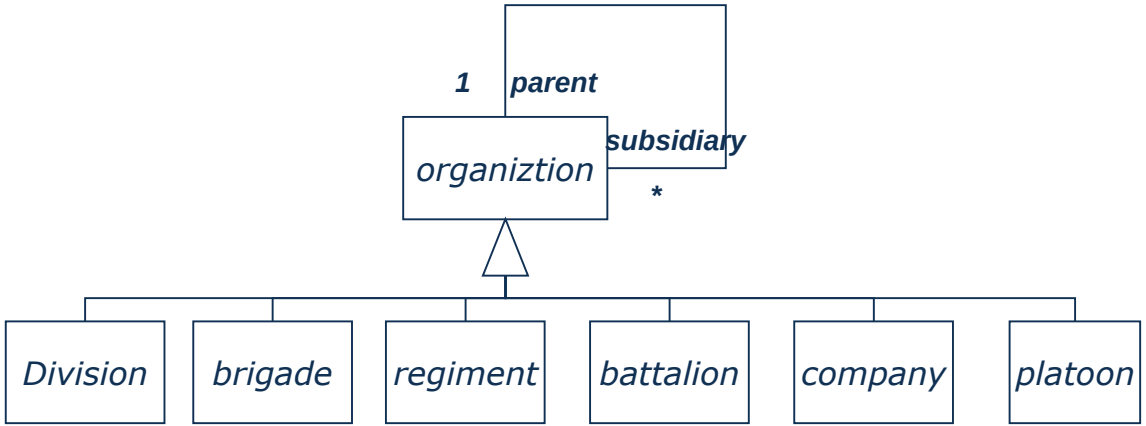
分析模式 Party


OOA
UML
OO
OOP
Process
OOD

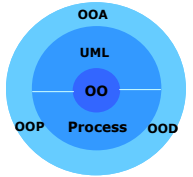

Telephone number
Person
Adress
Company
Email
*
1

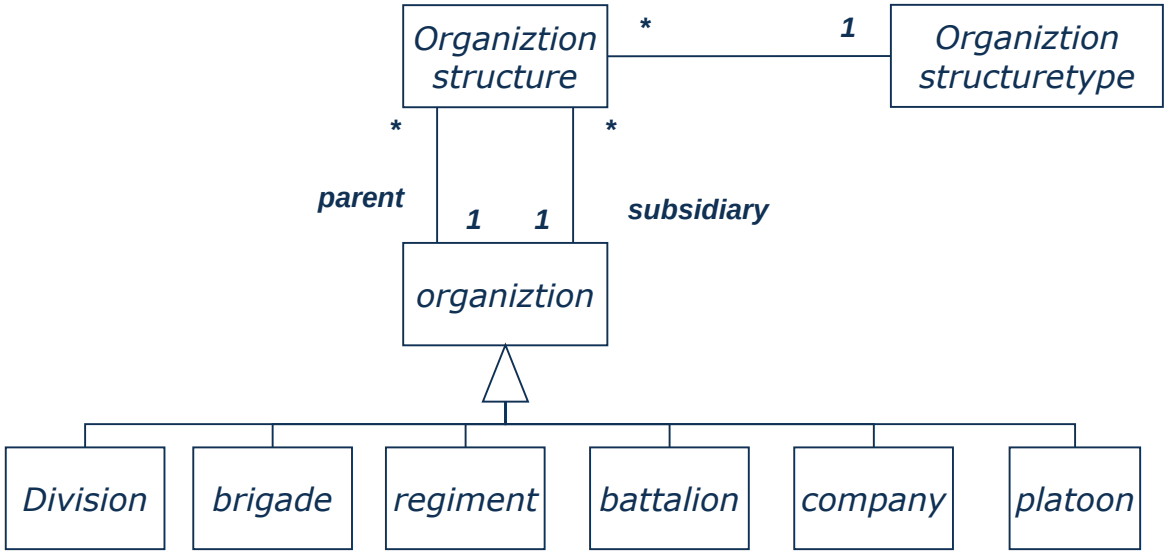

Company
Person
Party
Telephone number
Adress
Email
1
*

分析模式 Organization Hierarchies(组织层次)



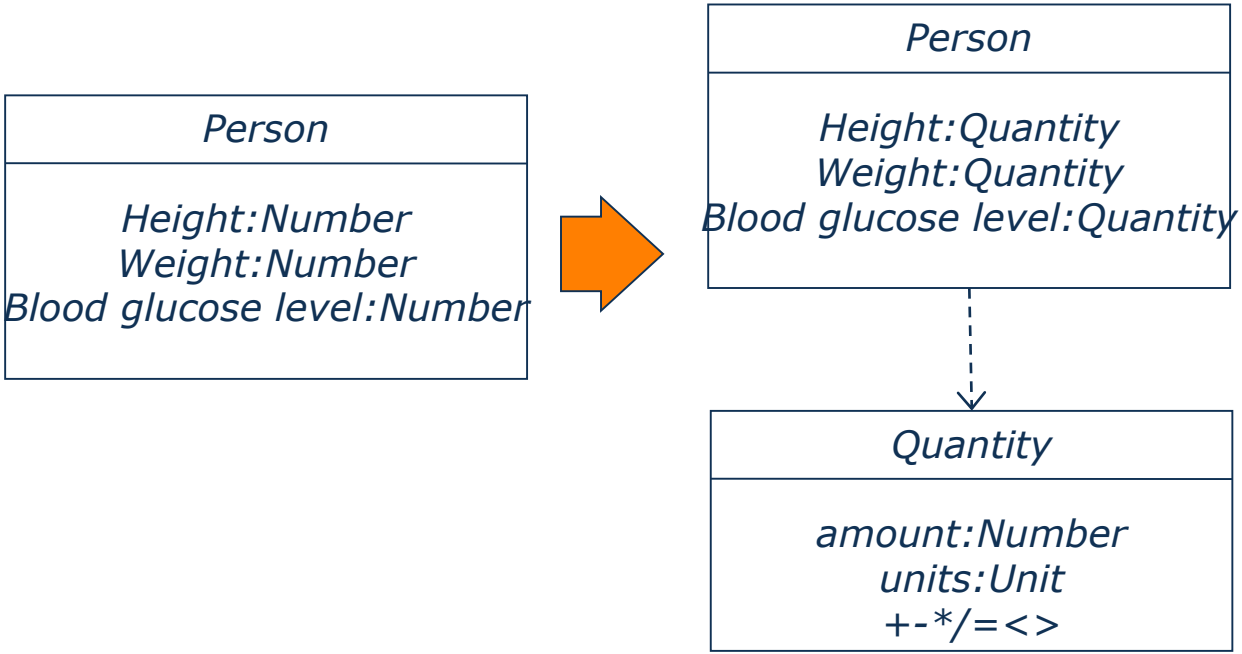
分析模式 Organization structure(组织结构)



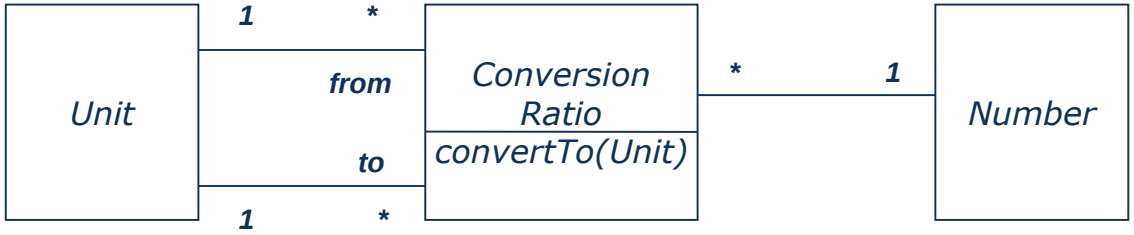


Quantity 数量

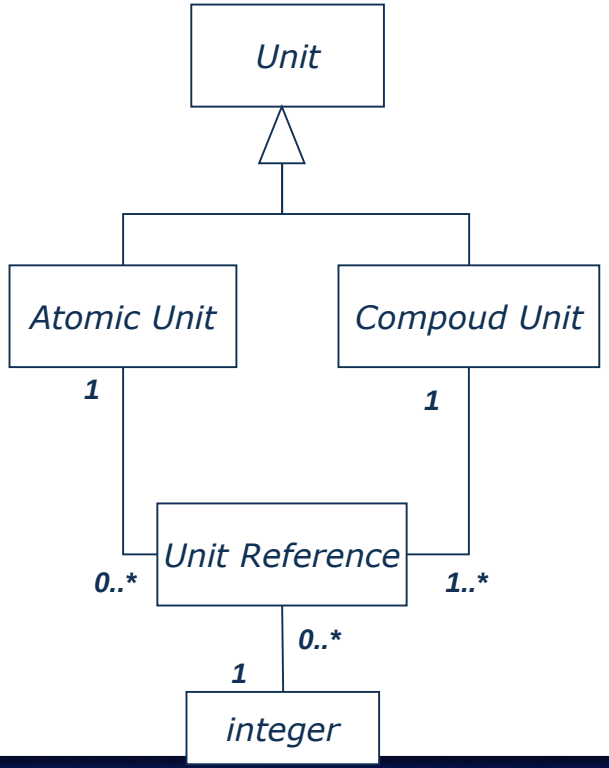

OOA
UML
OO
OOP
Process
OOD


Person
Height:Number
Weight:Number
Blood glucose level:Number
Person
Height:Quantity
Weight:Quantity
Blood glucose level:Quantity
Quantity
amount:Number
units:Unit
+-*/=<>

Conversion Ratio

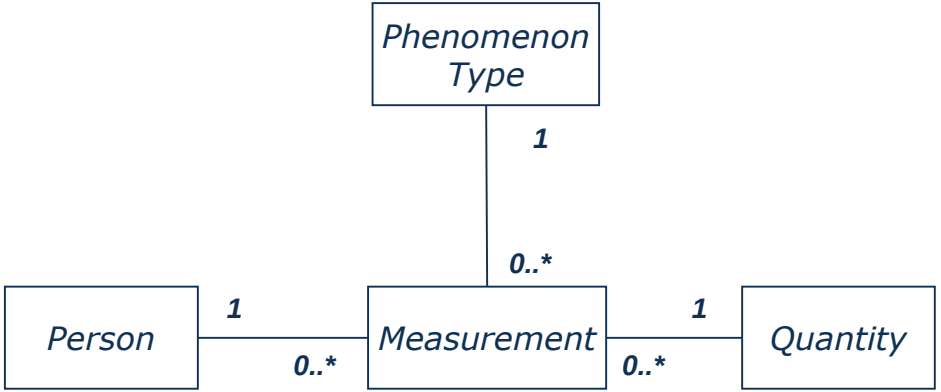


Compound Unit



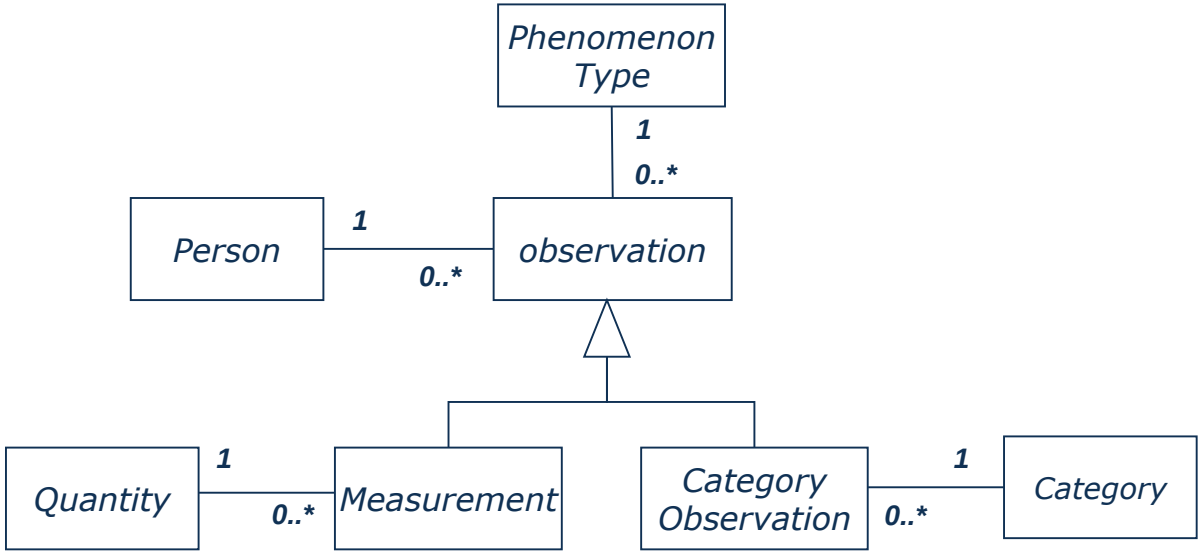
Measurement 测量

- 解决对象的大量变化的属性问题



Observation 观察

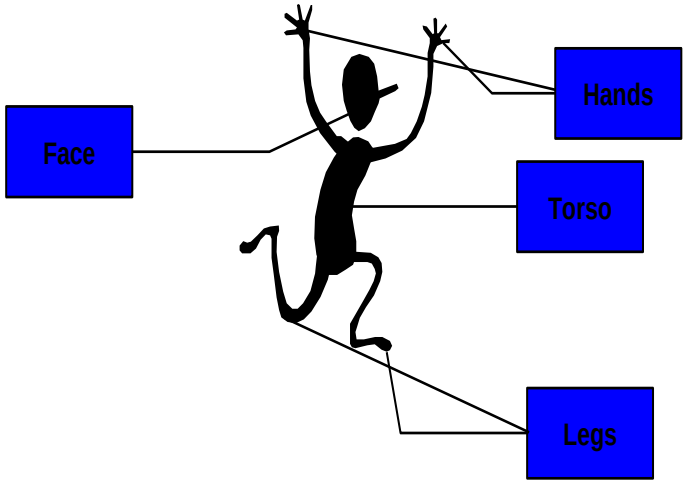
- 解决定量描述的问题



To be continued…

4.3 聚合

- 概念与表示法
- 性质
- 分类
- 识别策略

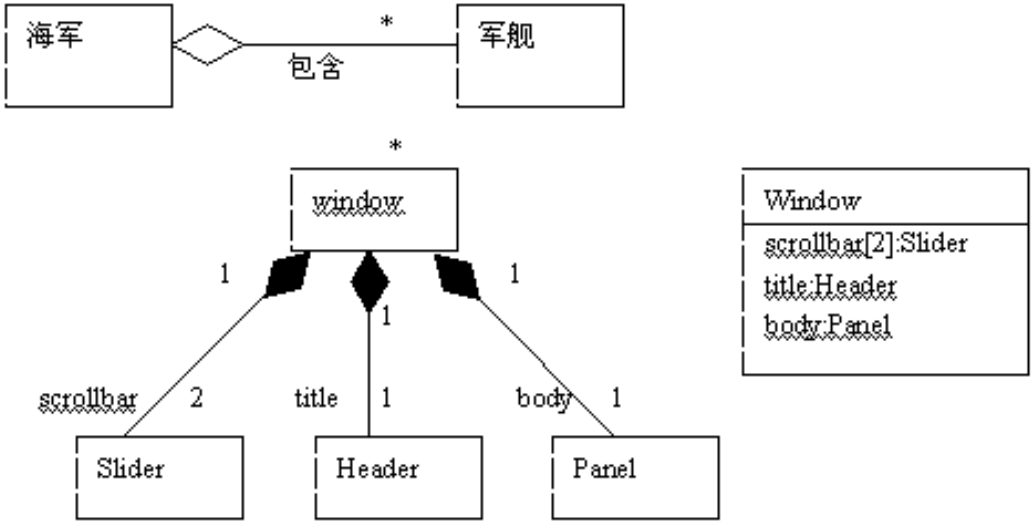


概念与表示法

+ 聚合(aggregation）是关联的一种特殊形式，表示整体和部分之间的“整体 - 部分”关系。
+ 聚集(aggregate)是聚合关系中作为“整体”的类，而把作为“部分”的类称为成分或部分。
+ 类与类之间的聚合关系指的是，一个类的对象实例，以另一个类的对象实例作为其组成部分，是种“a part of”或“has a”；也可理解为，一个类定义引用另一个类定义。
+ 组合(Composition)是聚合的一种形式，其中，其部分和整体之间具有很强的“属于”关系，并且它们的生存期是一致的。这种聚集末端的多重性不能超过1。
+ 组合对象是组合类的实例。

表示法

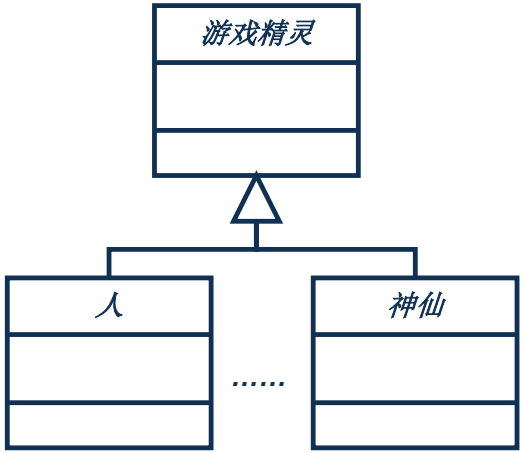

OOA
UML
OO
OOP
Process
OOD


海军
*
包含
军舰
*
window
1
1
1
scrollbar
2
title
1
body
1
Slider
Header
Panel
Window
scrollbar[2]:Slider
title:Header
body:Panel

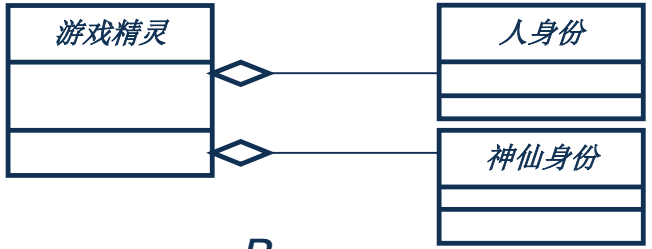
性质

- 结构性质
  - 部分必须与它们所构建的整体有某些结构上或功能上的关系。
- 数学性质
  - a．反对称性
    - 如果对象A是对象B的一部分，那么对象B就不能是对象A的一部分。
  - b．传递性
    - 如果对象A是对象B的一部分，对象B是对象C的一部分，那么对象A是对象C的一部分。

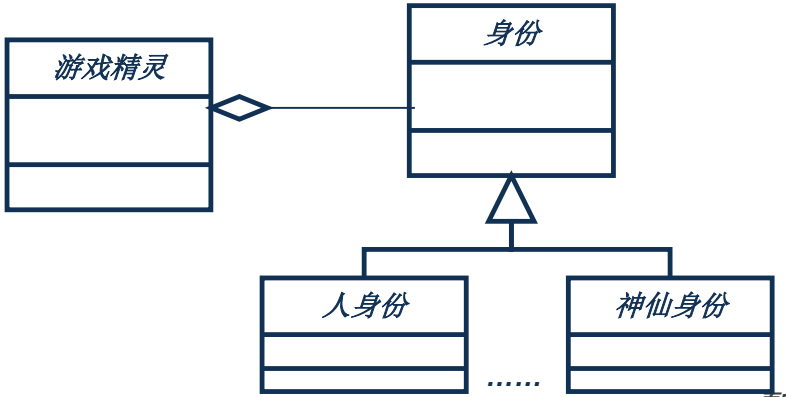
思考题
动态分类:人会变成神仙!



A



B



C

Martin Fowler: Aggregation And Composition

- http://martinfowler.com/bliki/AggregationAndComposition.html
- Few things in the UML cause more consternation than aggregation and composition, in particular how they vary from regular association.
- The full story is muddied by history. In the pre-UML methods there was a common notion of defining some form of part-whole relationships. The trouble was that each method defined different semantics for these relationships (although to be fair, some of these were pretty semantics free).
- So when the time came to standardize, lots of people wanted part-whole relationships, but they couldn't agree on what they meant. So the UML definers introduced two relationships.
- **aggregation** (*white diamond*) has no semantics beyond that of a regular association. It is, as Jim Rumbaugh puts it, a modeling placebo. People can, and do, use it - but there are no standard meanings for it. So if you see it, you should inquire as to what the author means by it. I would advise not using it yourself without some form of explanation.
- **composition** (*black diamond*) does carry semantics. The most particular is that an object can only be the part of one composition relationship. So even if both windows and panels can hold menu-bars, any instance of menu-bar must be only held by one whole. This isn't a constraint that you can easily express with the regular multiplicity markers.

分类

+ Morton Winston等人
+ 整体部分关系的性质
  - 构造性：整体部分之间存是否存在功能或结构上的关系
  - 同质性：整体部分之间的类型是否相同
  - 不变性：整体部分之间的是否不可分离性

根据性质的不同可将整体部分关系分为以下几类:

名称	构造性	同质性	不变性	定义	举例
组装-部分	√	×	×	部分保持它们的标示和功能	键盘-计算机,书-章,电话-部件
材料-对象	×	×	√	部分失去它们的标示	面包-面粉,糖,黄油;汽车-钢,塑料,玻璃;咖啡
划分-对象	×	√	×	整体与同质部分的组合	一片面包-一条面包;一秒-一小时;一升咖啡-一杯咖啡
地点-地区	×	√	√	整体与同质部分的组合,但部分构造是不变的	山峰-山;房间-旅馆;大兴-北京
集合-成员	×	√	√ 有序	成员间有序排列	电话本-人名,月计划表-日计划表
容器-容物	×	×	×	空间\时间\社会上的联系	包-包中的容物
成员-合作	√	×	√	具有不变性的容器-容物	舞伴,相声搭档,

识别策略

- 物理上的整体事物和它的组成部分
  - 例：机器、设备和它的零部件
- 组织机构和它的下级组织及部门
  - 例：公司与子公司、部门
- 团体（组织）与成员
  - 例：公司与职员
- 一种事物在空间上包容其它事物
  - 例：生产车间与机器
- 抽象事物的整体与部分
  - 例：学科与分支学科、法律与法律条款
- 具体事物和它的某个抽象方面
  - 例：人员与身份、履历
- 在材料上的组成关系
  - 例如，面包由面粉、糖和酵母组成，汽车是由钢、塑料和玻璃组成。

审查与筛选

- 是否属于问题域或系统责任？
  - 例：公司职员与家庭
- 部分对象是否有一个以上的属性？
  - 例：汽车与轮胎（规格）
- 是否有明显的整体-部分关系？
  - 例：学生与课程，谁是整体？部分？

To be continued…

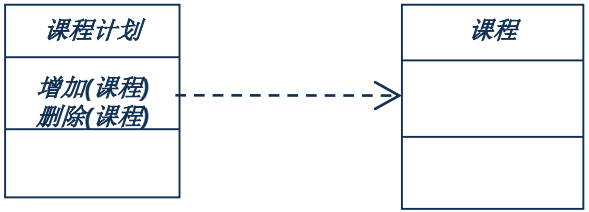
4.4 依赖

- 一个依赖规约了两个或多个模型元素（或两个模型元素集合）之间的一种语义关系，对目标元素的改变可能需要改变该依赖中的源元素。
- 下面种类的依赖是预定义的：
  - access --访问；derive --派生；import --移入；refine--精化
  - trace--跟踪；use--使用
  - include–包含； extend—扩展

表示法

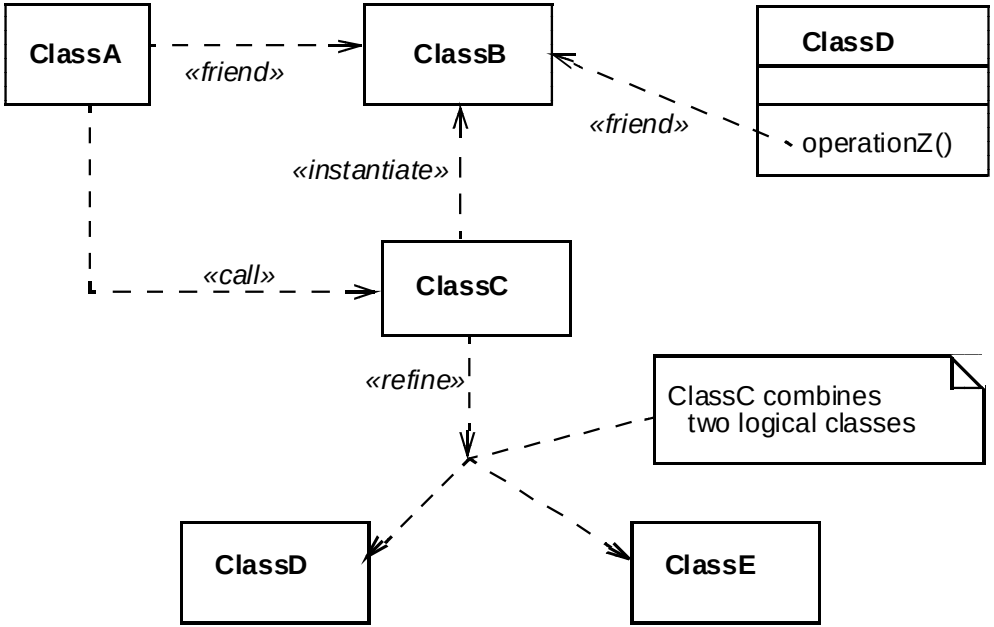
+ 把依赖表示为两个模型元素之间的虚线箭头。在箭头尾部的模型元素（客户）依赖箭头头部的模型元素（提供者）。箭头可以用放在书名号内的字符串标识。

+ 可能会有一组元素作为客户或提供者。在这种情况下，一个或多个尾部在客户端的箭头被连接到头部在提供者端的一个或多个箭头的尾部。如果需要，可以在连接处放置小的圆点。依赖的注释应该依附在连接点上。用没有箭头的虚线表示注释或约束与它应用到的元素之间的连接。这不是依赖。



不同类型的依赖


OOA
UML
OO
OOP
Process
OOD


ClassA
«friend»
ClassB
ClassD
operationZ()
«friend»
«instantiate»
«call»
ClassC
«refine»
ClassC combines
two logical classes
ClassD
ClassE

不同类型的依赖

+ Dependency relationship is used to show that some element or a set of elements depends on other model element(s), and on class diagrams is usually represented by usage dependency or abstraction.

+ A **usage** is a dependency relationship in which one element (**client**) requires another element (or set of elements) (**supplier**) for its full **implementation** or **operation**.

+ The usage dependency does not specify how the client uses the supplier other than the fact that the supplier is used by the definition or implementation of the client. For example, it could mean that some method(s) within a (**client**) class uses objects (e.g. parameters) of the another (**supplier**) class.

+ A usage dependency is shown as a dependency with a **«use»** keyword attached to it.

+ **Create** is a usage dependency denoting that the client classifier creates instances of the supplier classifier. It is denoted with the standard stereotype **«create»**.

+ **Call** is a usage dependency that specifies that the source operation invokes the target operation. This dependency may connect a source operation to any target operation that is within the scope including, but not limited to, operations of the enclosing classifier and operations of other visible classifiers.

+ **Send** is a usage dependency whose source is an **operation** and whose target is a **signal**, specifying that the source sends the target signal.

不同类型的依赖

+ Abstraction is a dependency relationship which relates two elements or sets of elements (called client and supplier) representing the same concept but at different levels of abstraction or from different viewpoints.

+ Realization is a specialized abstraction relationship between two sets of model elements, one representing a specification (the supplier) and the other represents an implementation of the latter (the client).

+ Interface realization is a specialized realization relationship between a Classifier and an Interface. This relationship signifies that the realizing classifier conforms to the contract specified by the interface.

4.5 接口与实现

+ **接口**
  - **接口定义为一个类的对外可见的一组操作的描述符,它定义了类对外提供的服务.**
  - **接口是一种协约,处于使用者和实现者之间.**

+ An interface is a classifier that declares of a set of coherent public features and obligations.

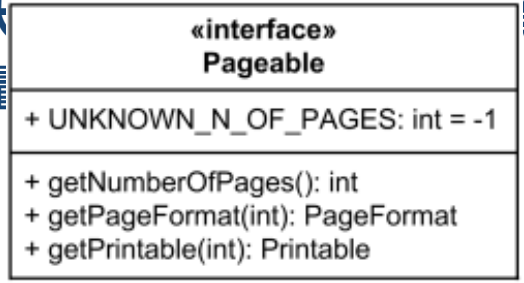
+ An interface specifies a contract.

+ Any instance of a classifier that realizes (implements) the interface must fulfill that contract and thus provides services described by contract.

+ The obligations that may be associated with an interface are in the form of various kinds of constraints (such as pre- and postconditions) or protocol specifications, which may impose ordering restrictions on interactions through the interface.

To be continued…

- 接口的作用
  - 是面向对象的一个重要机制
  - 实现多继承
  - 建立类和类之间的“协议”
    - 把类根据其实现的功能来分别代表，而不必顾虑它所在的类继承层次；这样可以最大[illegible]隐藏实现细节
    - 实现不同类之间的常量



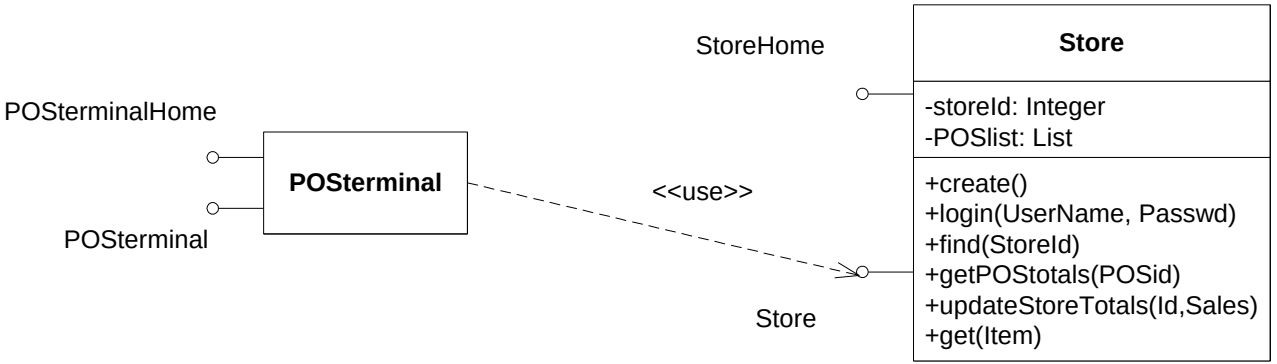
接口与类

+ In **UML 1.4** interface was formally equivalent to an abstract class with no attributes and no methods and only abstract operations.

+ Since interfaces are declarations, they are not instantiable. Instead, an interface specification is implemented by an instance of an instantiable classifier, which means that the instantiable classifier presents a public facade that conforms to the interface specification.
+ Any given classifier may implement more than one interface. Interface may be implemented by a number of different classifiers.

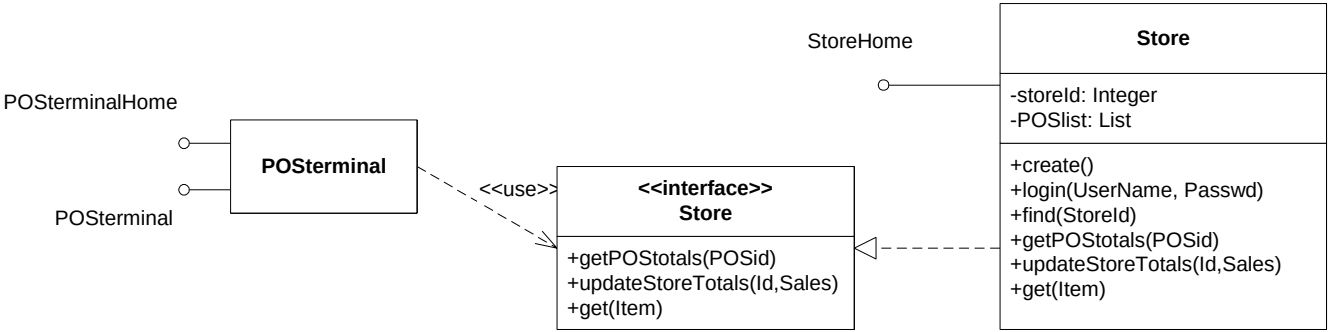
接口的表示(简单)


OOA
UML
OO
OOP
Process
OOD


StoreHome
Store
-storeId: Integer
-POSlist: List
POSterminalHome
POSterminal
POSterminal
<<use>>
+create()
+login(UserName, Passwd)
+find(StoreId)
+getPOStotals(POSid)
+updateStoreTotals(Id,Sales)
+get(Item)
Store

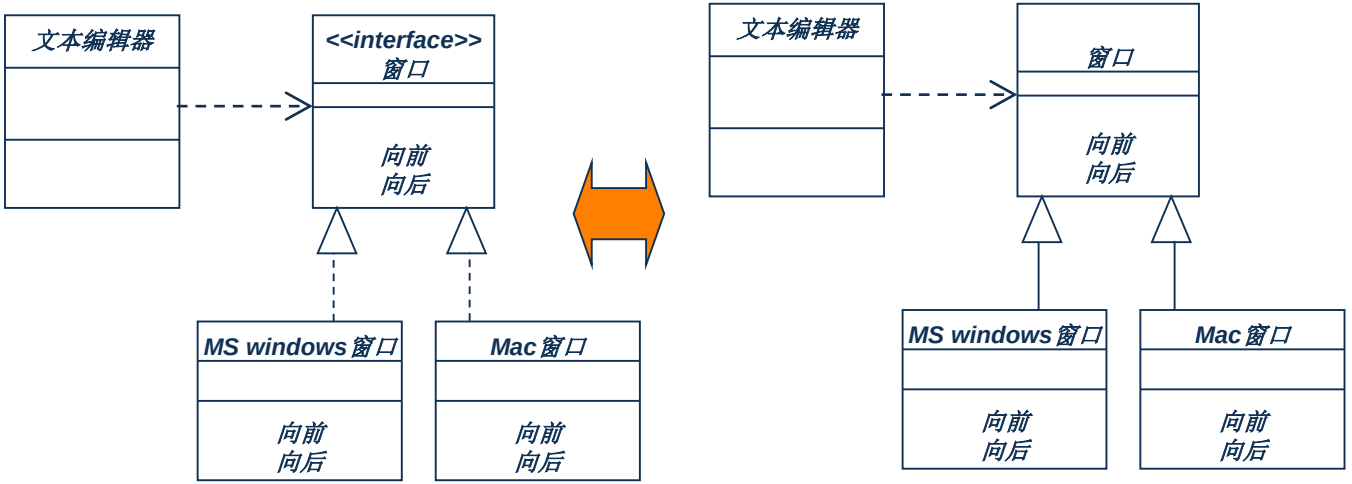
接口的表示(完全)


OOA
UML
OO
OOP
Process
OOD

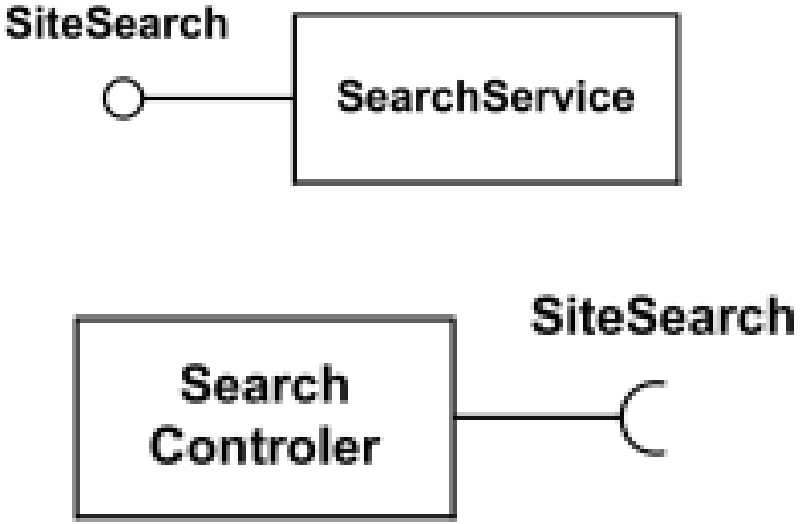

StoreHome
Store
-storeId: Integer
-POSlist: List
+create()
+login(UserName, Passwd)
+find(StoreId)
+getPOStotals(POSid)
+updateStoreTotals(Id,Sales)
+get(Item)
POSterminalHome
POSterminal
POSterminal
<<use>>
<<interface>>
Store
+getPOStotals(POSid)
+updateStoreTotals(Id,Sales)
+get(Item)

接口与泛化的关系

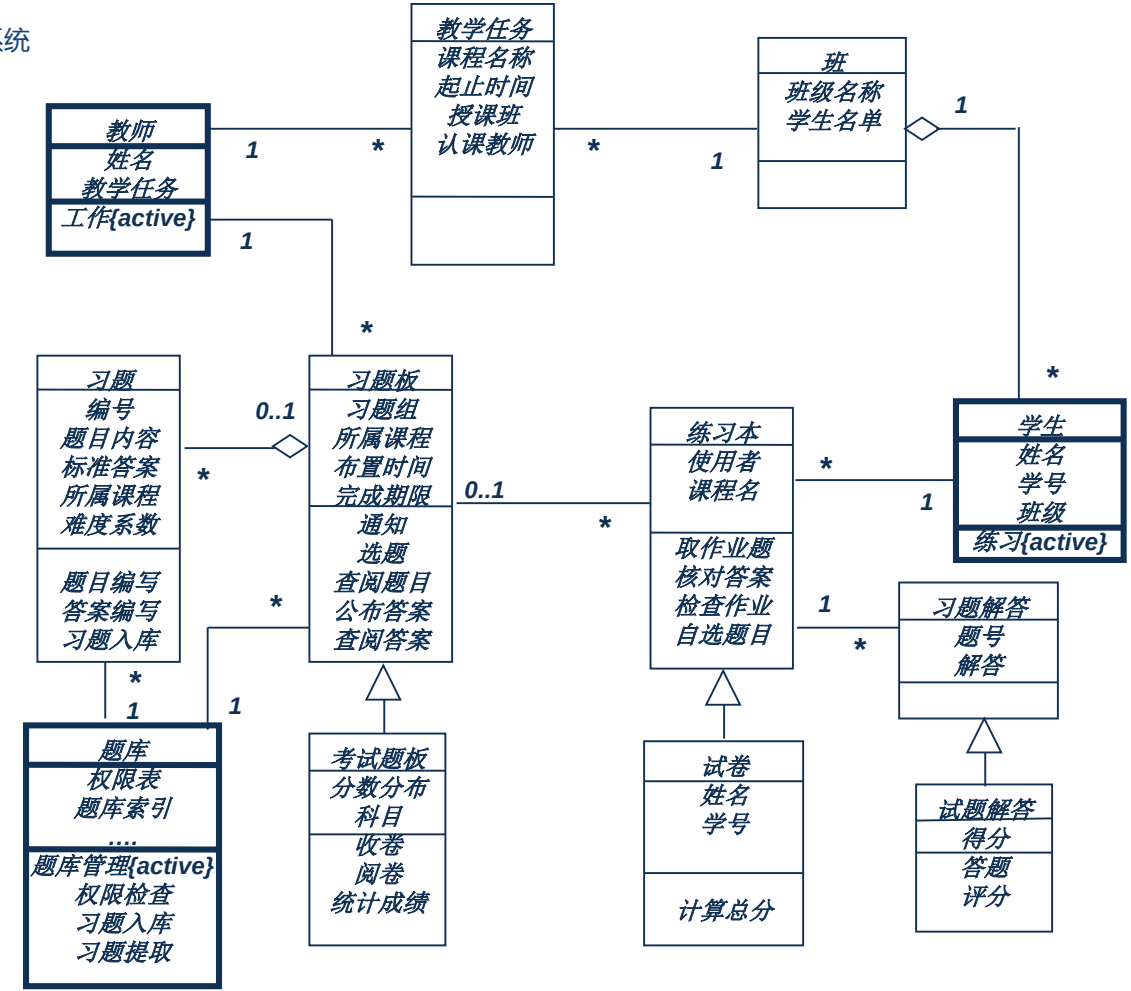

OOA
UML
OO
OOP
Process
OOD


文本编辑器
<<interface>>
窗口
向前
向后
MS windows窗口
向前
向后
Mac窗口
向前
向后
文本编辑器
窗口
向前
向后
MS windows窗口
向前
向后
Mac窗口
向前
向后

UML2.0需接口与供接口



例题:题库管理系统

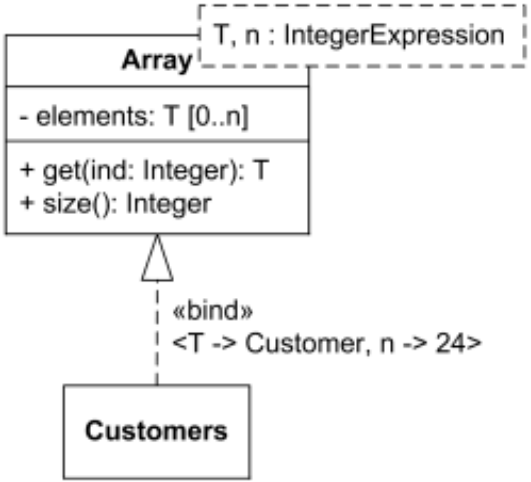

教学任务
课程名称
起止时间
授课班
认课教师
班
班级名称
学生名单
教师
姓名
教学任务
工作{active}
习题
编号
题目内容
标准答案
所属课程
难度系数
题目编写
答案编写
习题入库
习题板
习题组
所属课程
布置时间
完成期限
通知
选题
查阅题目
公布答案
查阅答案
练习本
使用者
课程名
取作业题
核对答案
检查作业
自选题目
学生
姓名
学号
班级
练习{active}
习题解答
题号
解答
试题解答
得分
答题
评分
题库
权限表
题库索引
....
题库管理{active}
权限检查
习题入库
习题提取
考试题板
分数分布
科目
收卷
阅卷
统计成绩
试卷
姓名
学号
计算总分
1
*
0..1

模板类与绑定

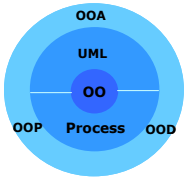
+ UML classes could be templated or bound.

+ The example below shows template class Array with two formal template parameters. The first template parameter T is an unconstrained class template parameter. The second template parameter n is an integer expression template parameter.

+ Template binding for the bound class Customers substitutes the unconstrained class T with class Customer and boundary n with integer value 24. Thus, the bound class Customers is an Array of 24 objects of Customer class.



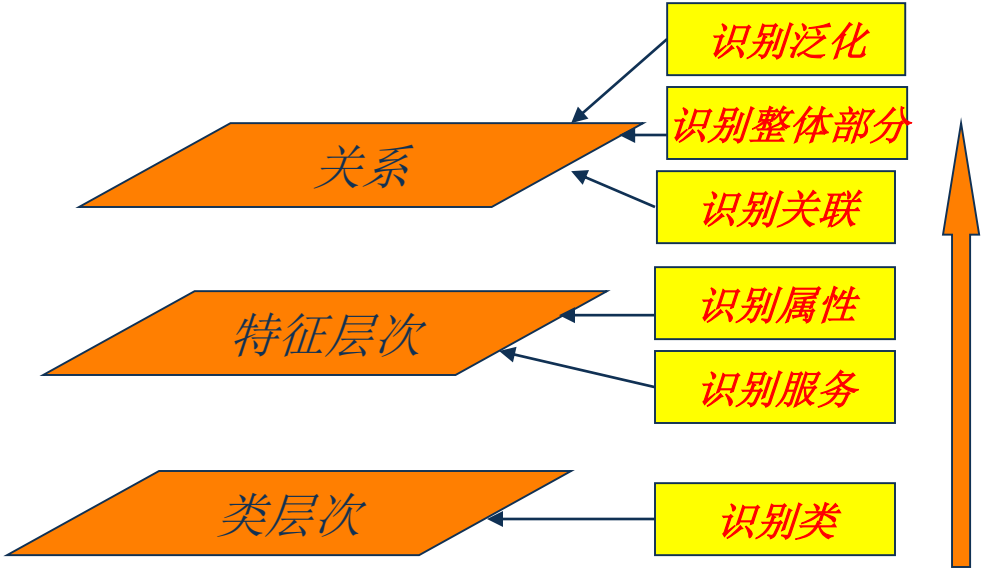
总结



- 对象与类
  - 对象、类、主动对象、主动类、主动服务、特征标记、类的角色
- 属性
  - 属性、实例属性、类范围属性
- 服务
  - 服务、抽象操作、类范围操作
- 关系
  - 分类关系、继承——泛化（一般-特殊）
    - 泛化、一般类、特殊类
  - 构成关系——聚合（整体-部分）
    - 聚合、聚集、成分、组合
  - 静态联系——关联(实例连接)
    - 关联、链、多重性、角色、多元关联、关联类
  - 使用关系（行为依赖）——依赖
    - 依赖
  - 接口与实现

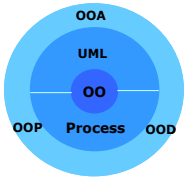
类图的生成过程


OOA
UML
OO
OOP
Process
OOD


识别泛化
识别整体部分
识别关联
关系
识别属性
识别服务
特征层次
识别类
类层次

To be continued…

思考题:



+ 某供电局准备开发线路监控软件系统，用于各条供电线路的情况。该系统由专职的管理员来操作。每条供电线路安装一个线路检测仪，每30秒采集1次该线路的信息（包括电压、电流）。每隔1小时，线路检测仪通过专线向线路监控软件系统传送该小时的数据，系统接受后，保存在系统中。

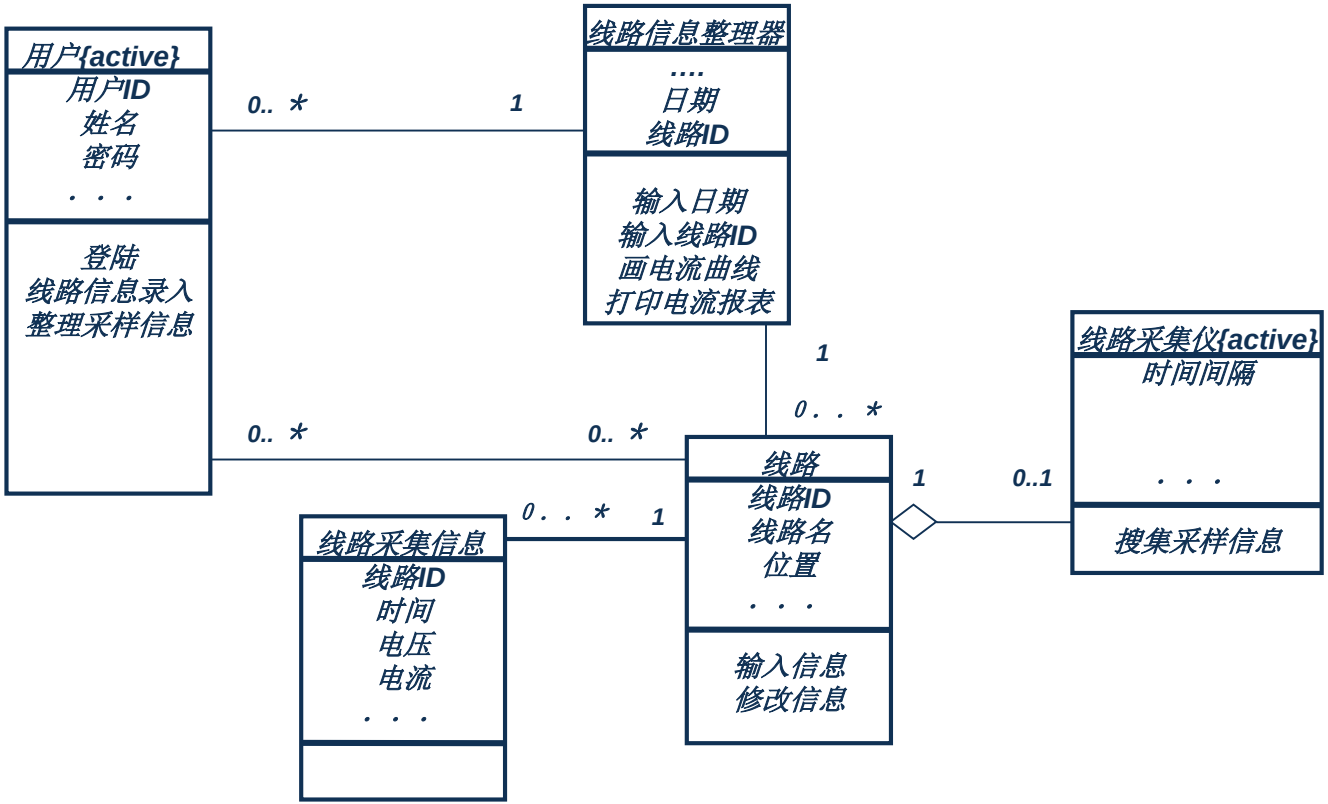
+ 管理员登陆系统后，可进行如下几项工作：

+ ①线路信息录入：录入每条线路的基本信息，包括线路号、位置等等。

+ ②搜集采样信息：当系统与某线路的检测仪通讯时，系统提示管理员正在与该线路的检测仪通讯，并将检测仪所收集的该线路该时段的采样信息保存在系统中。

+ ③采样信息整理：可以根据这些数据生成该线路该日的数据报表与曲线，并能打印出来。

- 请建立线路监控系统的OOA类图


用户{active}
用户ID
姓名
密码
. . .
登陆
线路信息录入
整理采样信息
0.. *
1
线路信息整理器
....
日期
线路ID
输入日期
输入线路ID
画电流曲线
打印电流报表
1
0 . . *
线路采集仪{active}
时间间隔
. . .
搜集采样信息
0..1
1
0.. *
0.. *
线路
线路ID
线路名
位置
. . .
输入信息
修改信息
0 . . *
1
线路采集信息
线路ID
时间
电压
电流
. . .

思考题:

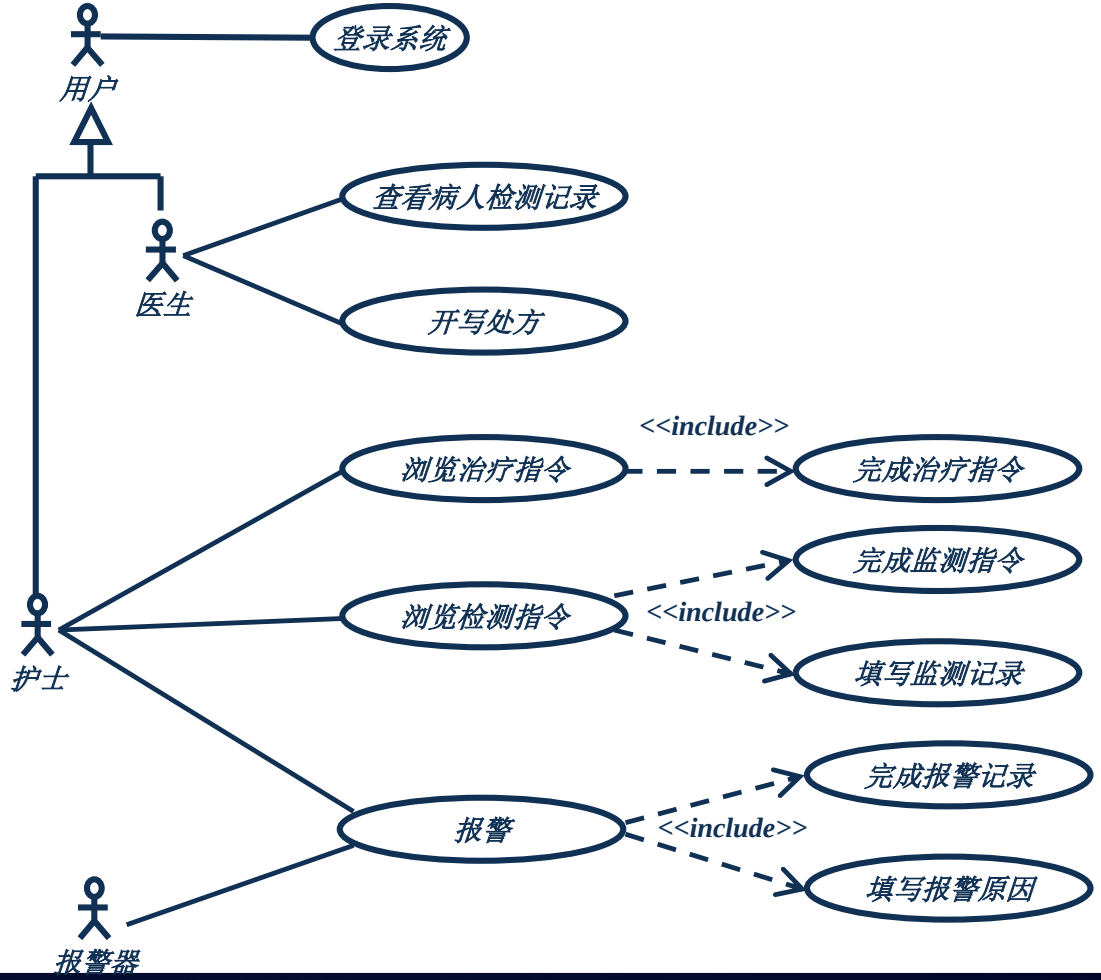
+ 某医院住院部拟开发医疗信息管理系统，用于管理病人住院期间的治疗、护理与健康情况。

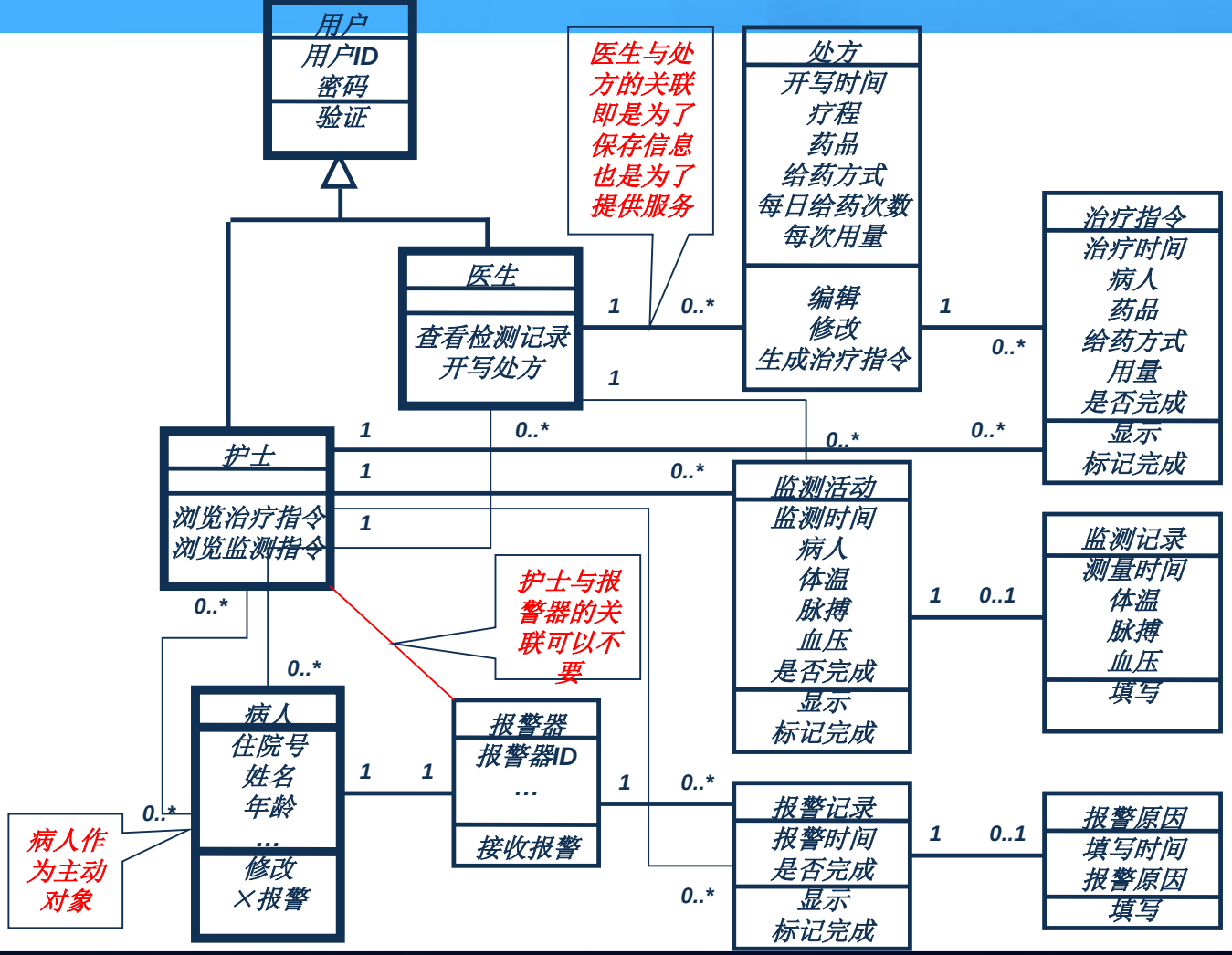
+ 该系统的使用者是医生和护士，医生登陆系统后，可以查看特定住院病人的监测记录，也可以开写处方，其中包括开写时间、疗程、药品、给药方式、每日给药次数、每次用量。系统将根据处方，生成一系列的治疗指令，主要包括治疗时间、病人、药品、给药方式、用量等。

+ 值班护士登陆系统后，可以看到她应该完成的一系列的治疗指令，提醒她在何时、为哪位病人进行治疗；如果她完成了一项治疗指令，她应将该治疗指令标记为完成。同时值班护士还可以看到她应该完成的一系列的监测指令，提醒她在何时、为哪位病人进行哪项监测活动（监测活动包括体温、脉搏、血压等）；如果她完成了一项监测，她应将该监测指令标记为完成，并填写监测记录，主要包括监测时间、病人、体温、脉搏、血压等。

+ 病人并不直接使用系统，但病人的基本信息应在系统中保留，包括住院号、姓名、年龄、性别、所属科室、护理级别等。病人可随时按报警器，报警器通知系统，系统产生报警信号通知值班护士立即探视，并产生一条报警记录，值班护士探视完毕后，她应将该报警记录标记为完成，并填写报警原因。

- 请根据医疗信息管理系统的文本需求描述，进行分析整理、建立该系统的用况图，并附用况的文本说明
- 请建立医疗信息管理系统的OOA类图


登录系统
用户
查看病人检测记录
医生
开写处方
<<include>>
浏览治疗指令
完成治疗指令
完成监测指令
浏览检测指令
<<include>>
填写监测记录
护士
完成报警记录
报警
<<include>>
填写报警原因
报警器


用户
用户ID
密码
验证
医生与处方的关联即是为了保存信息也是为了提供服务
处方
开写时间
疗程
药品
给药方式
每日给药次数
每次用量
编辑
修改
生成治疗指令
医生
查看检测记录
开写处方
治疗指令
治疗时间
病人
药品
给药方式
用量
是否完成
显示
标记完成
护士
浏览治疗指令
浏览监测指令
监测活动
监测时间
病人
体温
脉搏
血压
是否完成
显示
标记完成
监测记录
测量时间
体温
脉搏
血压
填写
护士与报警器的关联可以不要
病人
住院号
姓名
年龄
…
修改
×报警
报警器
报警器ID
…
接收报警
报警记录
报警时间
是否完成
显示
标记完成
报警原因
填写时间
报警原因
填写
病人作为主动对象
1
0..*
0..1

To be continued…

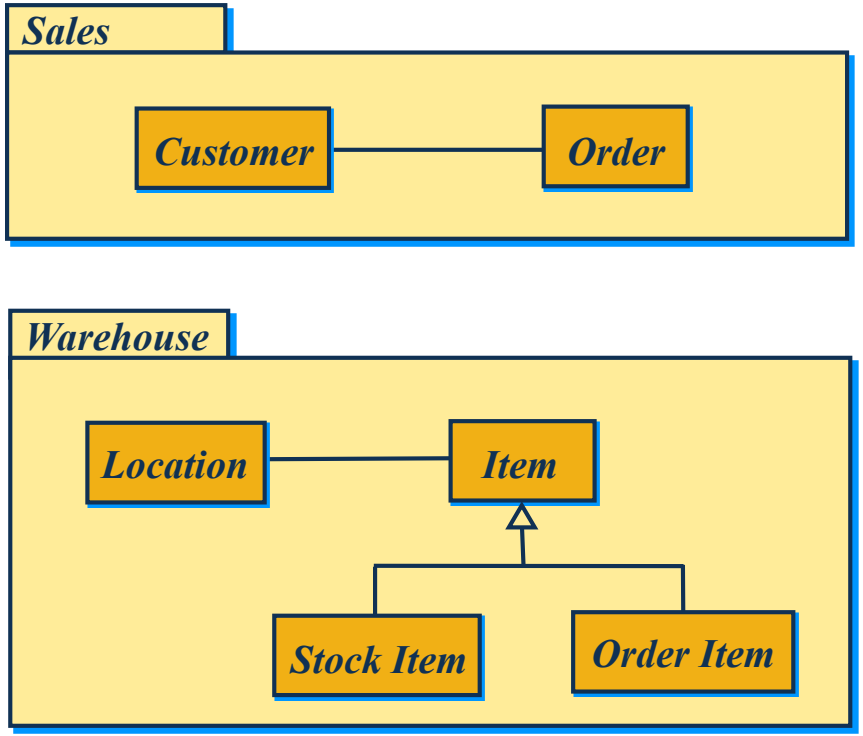
包图

- 概念与表示法
- 如何分包

概念与表示法

- 定义包是对模型元素分组的机制。
- 使用包的最常见目的是把建模元素组织成为组，作为一个集合进行命名和处理。
- 包可以拥有类、接口、构件、节点、协作、用况和图，甚至可以是其它包。拥有是一种组成关系，这意味着被拥有的元素被声明在包中。如果包被撤消了，元素也要被撤消。
- 一个元素只能被一个包所拥有。
- 设计良好的包，把在语义上接近并倾向于一起变化的元素组织在一起。因此结构良好的包是松耦合、高内聚的，而且对其内容的访问具有严密的控制。

例子


Sales
Customer
Order
Warehouse
Location
Item
Stock Item
Order Item

包

- 包可以包含不同类型的建模元素
  - 包括其他的包
- 包为其内容物定义了名域
- 包可以用于不同的目的

核心概念

元素	描述	表示
包	建模元素的分组.	
引入	是两个包之间的一种许可依赖关系，一个包中的可见性为公有的模型元素，可以在指定的包（包括嵌套在该包中的子包）中被引用，相当于把提供者包的内容附加到客户包的命名空间中，而不必对名称进行限制	«import» 
访问	是两个包之间的一种许可依赖关系，它允许客户包使用提供者包中的可见性为公有的元素，当客户包引用这些元素时，需要使用元素的全路径名。	«access» 

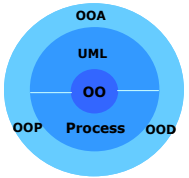
可见性

+ 包中的每一个包含元素都有一个相对于包含它的包的可见性
  - 公共元素对外可见,表示为 ‘+’
  - 受保护元素仅对继承包可见, 表示为 ‘#’
  - 私有元素对外不可见,表示为 ‘-’
  - 包内元素可见对外不可见,表示为 ‘~’
+ 与类中的属性与服务相似

包继承

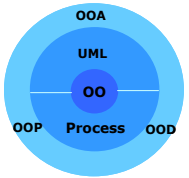
+ 若包A继承包B,则包A 继承了包B所有的和其引入的公共的和受保护的元素

什么时候使用包?



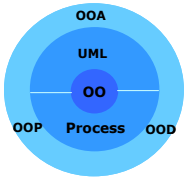
- 产生庞大的建模元素的概况
- 组织庞大的模型
- 组织相关的建模元素
- 分割命名空间

如何划分包



- 识别低层包
  - 每个泛化结构和每个聚合结构作为一个包
  - 关联密集的类划分到一个包
  - 独立的类暂时作为一个包
- 包的合并
  - 如果低层包数量过多，则合并为高层包
  - 依据：低层包之间——
    - 从概念考虑：接近，或具有较强的相关性
    - 从作用考虑：属于某项大的功能
    - 观察类图：偶合紧密
    - 分布情况：分布在同一台处理机

如何划分包(续)



- 组织包的层次
  - 包合并之后，决定低层包是保留还是取消
  - 每个包7±2个内层成分
  - 层次不宜太多
  - 包的划分不是唯一的，有一定的随意性
- 标识可见性
  - 对每一个包，区别哪些元素可以在包外访问。把它们标记为公共的，把所有其它的元素标记为受保护的或私有的。
- 建立包间的依赖关系
  - 用引入依赖或访问依赖显式地连接包。

To be continued…