

第五章 定义对象间的关系 II

曹东刚

caodg@pku.edu.cn

北京大学信息学院研究生课程 - 面向对象的分析与设计
<http://sei.pku.edu.cn/~caodg/course/oo>



内容提要

1 关联关系

- 概念与表示法
- 复杂关联问题
- 如何建立关联

2 交互关系（消息）

3 依赖关系

关联 (association)

是两个或者多个类上的一个关系（即这些类的对象实例集合的笛卡儿积的一个子集合），其中的元素提供了被开发系统的应用领域中一组有意义的信息。

关联是对象实例之间的关系，但定义在类层次给出

n 元关联

n 元关联 (n -ary association): 多个类之间的关联

二元关联 (binary association): 两个类之间的关联

问题: n 元关联中涉及的类的数量和 n 的关系?

关联的实例

关联的每一个元素称为关联的一个实例，又称链（link）
一个链是一个有序对或 n 元组，关联是这些有序对或 n 元组的集合
关联位于类的抽象层次，链位于对象的抽象层次

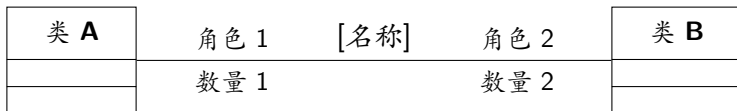
关联 vs 聚合

在概念上，都是对象实例间的一种静态关系
都是在类的抽象层次上定义
最终都通过对象的属性来实现

关联没有聚合具有的 is-a-part-of 语义
模型表示法不同
关联不能用聚合所用的嵌套对象实现

UML 将聚合视为关联的一种特殊情况

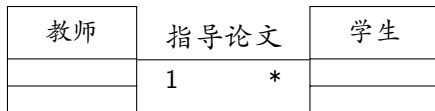
二元关联表示法



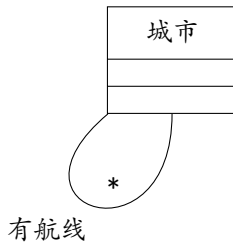
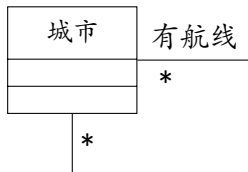
数量约束: 0, 1, 1..4, 1..*, *

多重性类型: 1 对 1, 1 对多, 多对多

二元关联表示法示例



二元关联表示法示例



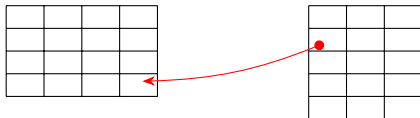
二元关联的实现: 1 对 1 和 1 对多

编程语言: 对象指针



二元关联的实现: 1 对 1 和 1 对多

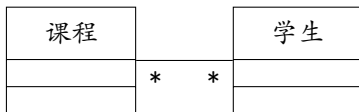
关系数据库: 外键



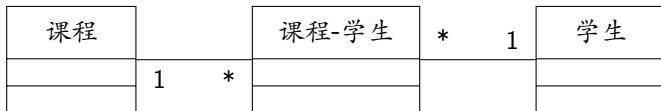
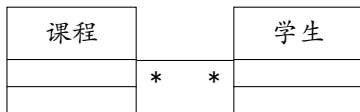
教师

课程

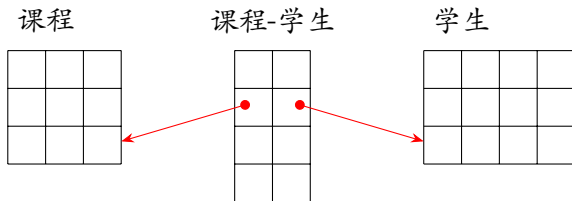
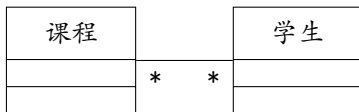
二元关联的实现：多对多



二元关联的实现：多对多



二元关联的实现：多对多



内容提要

1 关联关系

- 概念与表示法
- 复杂关联问题
- 如何建立关联

2 交互关系（消息）

3 依赖关系

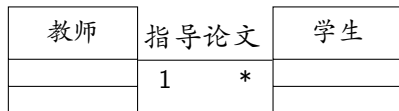
带有属性和操作的关联

问题：关联附加信息如何表示

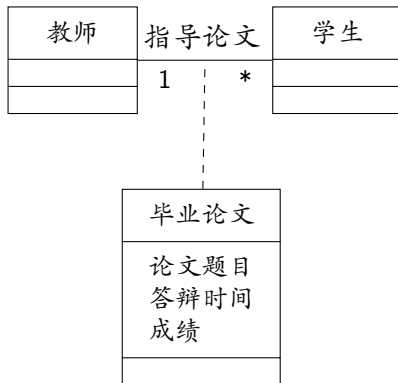
| 教师 | 指导论文 | 学生 |
|----|------|----|
| | 1 * | |
| | | |

带有属性和操作的关联

问题：关联附加信息如何表示

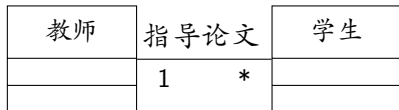


UML: 引入关联类

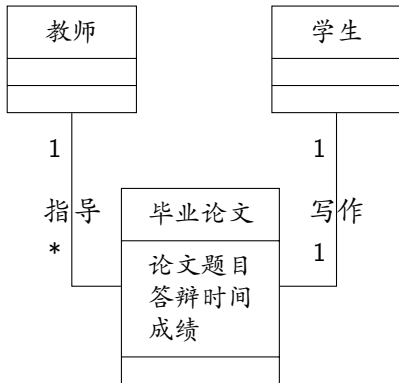


带有属性和操作的关联

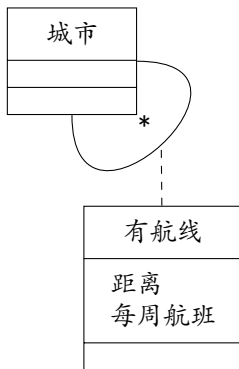
问题：关联附加信息如何表示



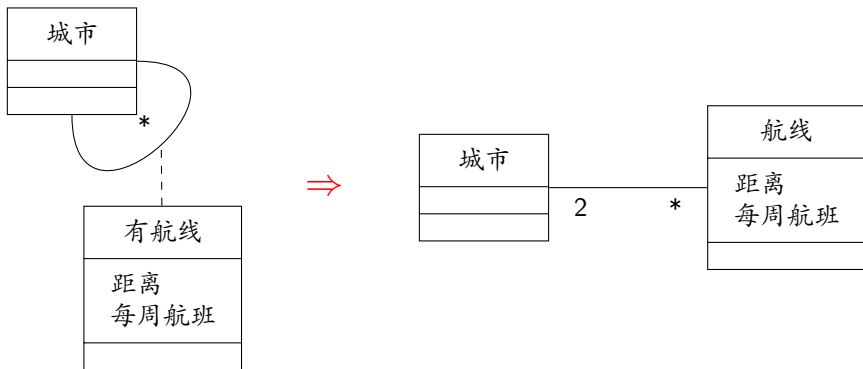
本书：用基本对象概念



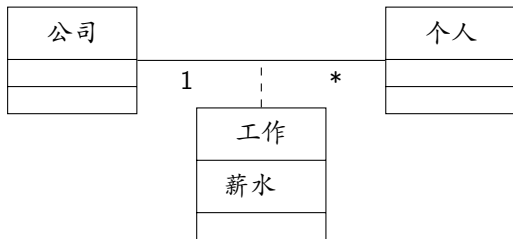
其他例子



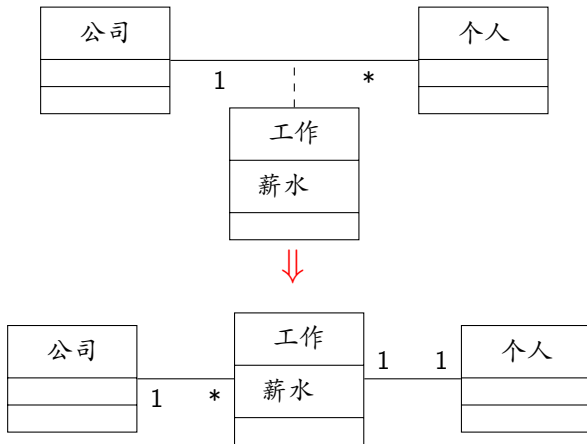
其他例子



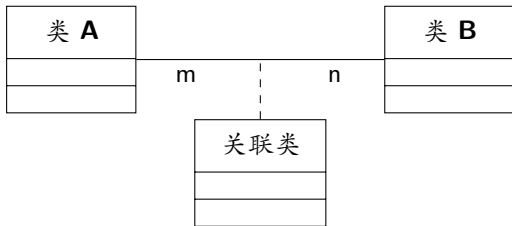
其他例子



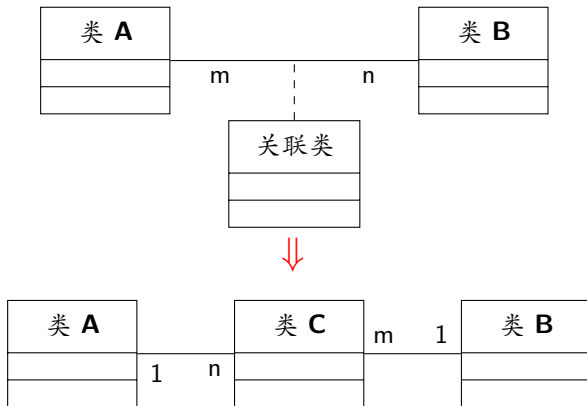
其他例子



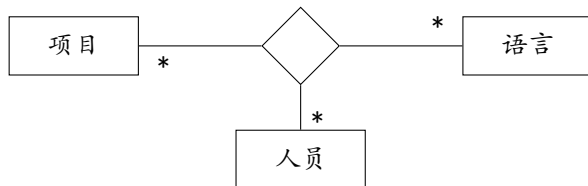
复杂关联表示法的转换



复杂关联表示法的转换

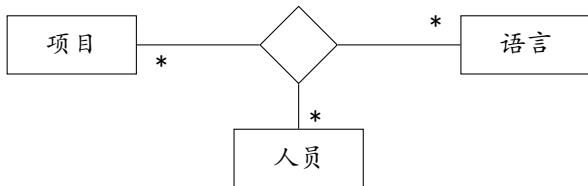


UML 和 OMT 的三元关联及其表示法



n 元关联

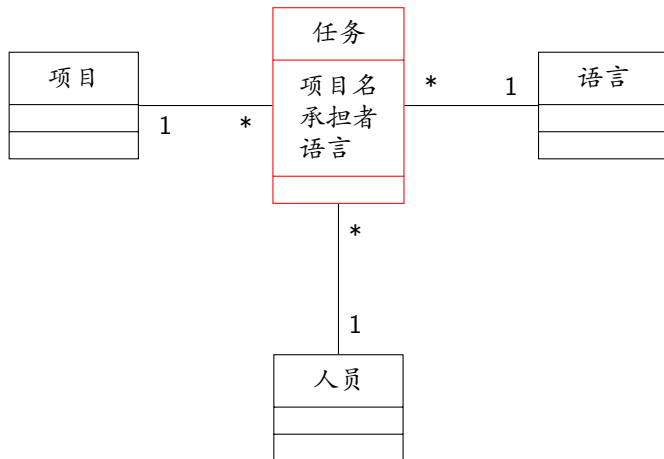
UML 和 OMT 的三元关联及其表示法



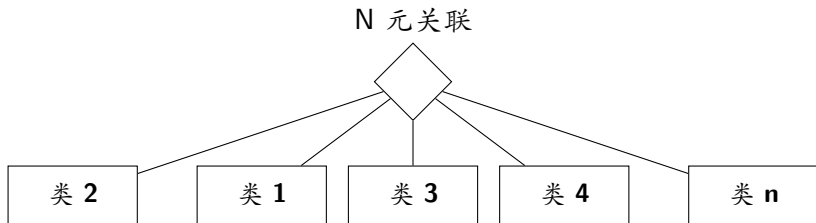
问题

- 编程语言不能直接支持
- 推广到 n 元关联用什么表示
- 多重性表示困难

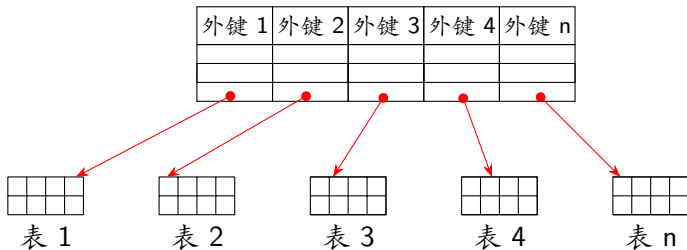
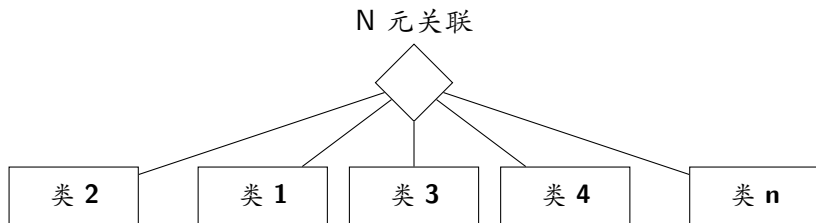
新增类表示 3 元关联



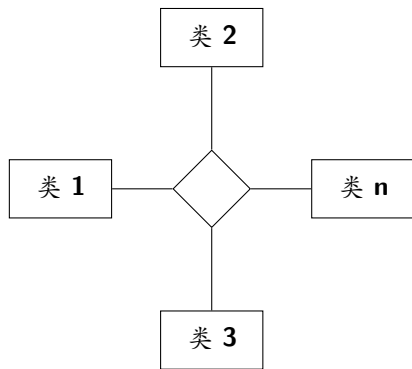
考虑 n 元关联的实现



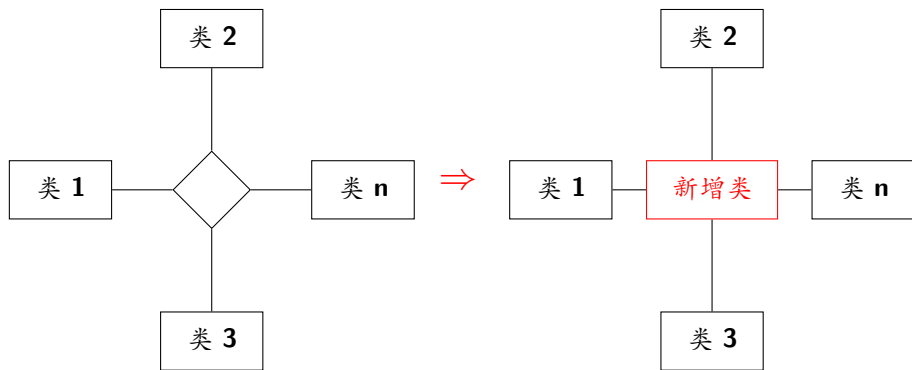
考虑 n 元关联的实现



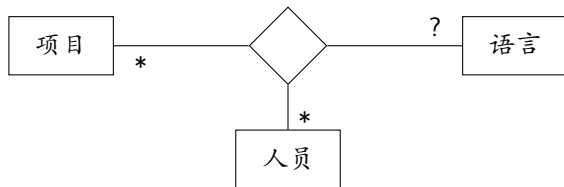
n 元关联的通用处理方法



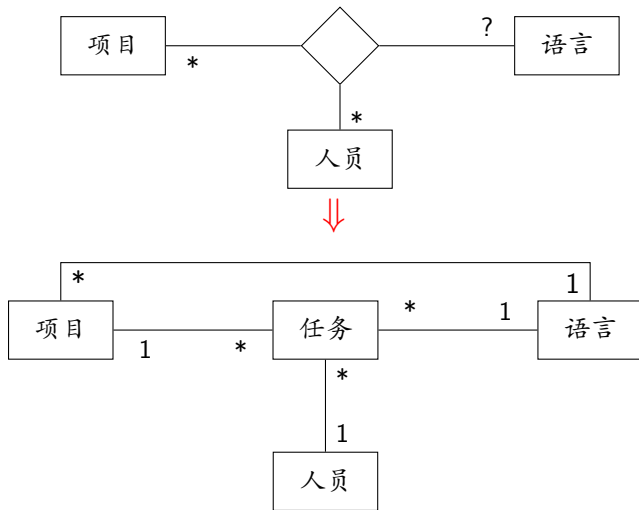
n 元关联的通用处理方法



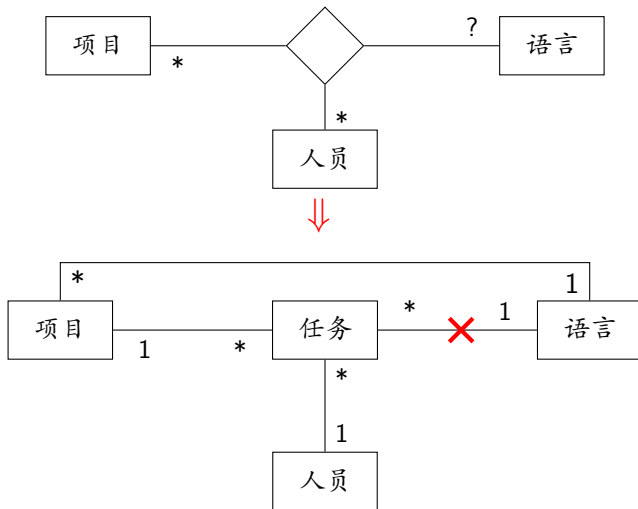
n 元关联的多重性问题



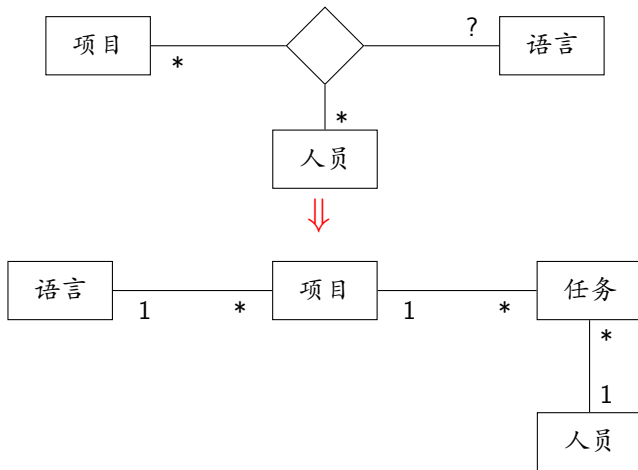
n 元关联的多重性问题



n 元关联的多重性问题



n 元关联的多重性问题



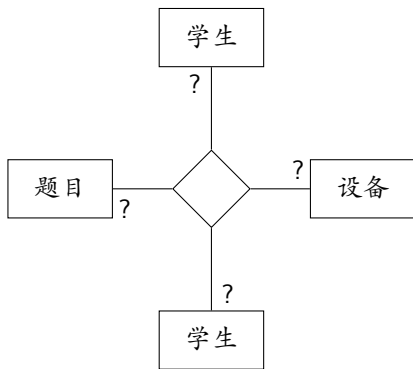
一个类在关联中多次出现

例

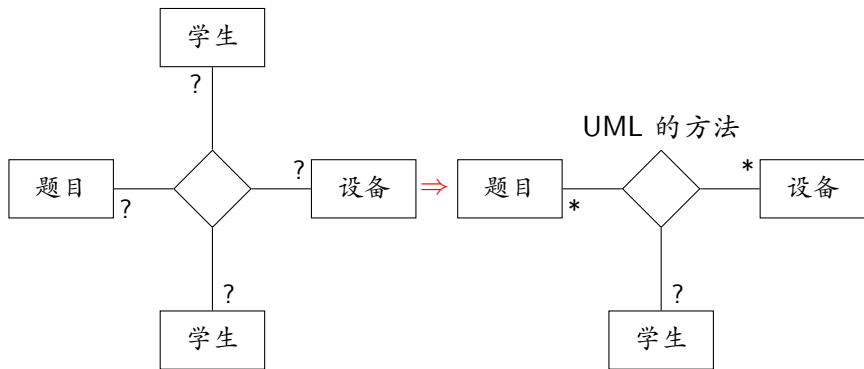
课程实习中每两名学生在一台设备上合作完成一个题目

- 系统要求记录和查阅哪两个学生是合作者
- 记录每组学生的实习题目和使用的设备
- 一个题目可以供多组学生实习，可以在不同的设备上完成
- 一台设备可以供多组学生使用，可以做不同的题目

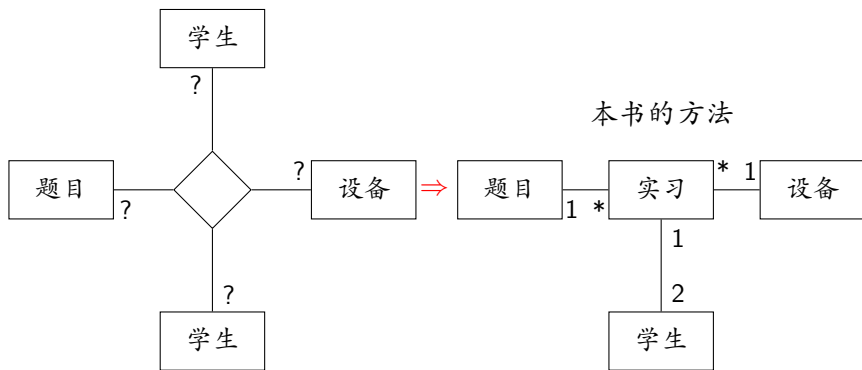
一个类在关联中多次出现



一个类在关联中多次出现



一个类在关联中多次出现



关联端点的复杂情况

关联端点 (association end)

关联端点就是关联与类目相连接的一个末端，它是关联的一部分，而不是类目的一部分

在表示法里，其实就是关联的连接线与类符号相衔接的点，用于修饰关联

修饰：在端点附近标注符号或者文字，或者将端点画成不同的形状。各种修饰使得关联概念变得复杂臃肿

UML1 的关联端点修饰

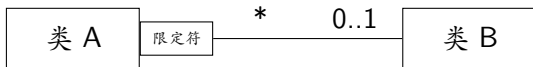
| | | |
|------------------------------|-----------------|---|
| 多重性 (multiplicity) | 1, *, 1..* | ✓ |
| 有序 (ordered) | {ordered} | x |
| 限定符 (qualifier) | | ? |
| 导航性 (navigability) | ————→ | x |
| 聚合标志 (aggregation indicator) | ————◇ | ✓ |
| 角色名 (rolename) | player, manager | ✓ |
| 接口说明 (interface specifier) | 角色名: 接口说明 | x |
| 可变性 (changeability) | {frozen} | x |
| 可见性 (visibility) | + # - | x |

限定符

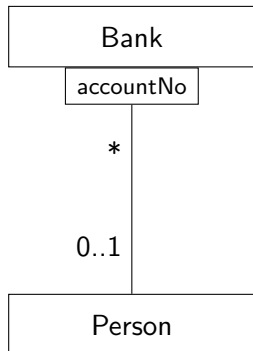
限定符

是关联的一种属性，它的值划定了跨过一个关联与一个对象相关的对象集合

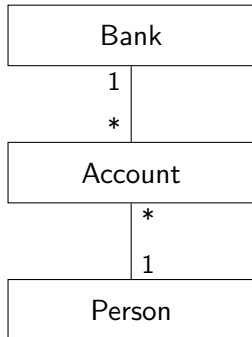
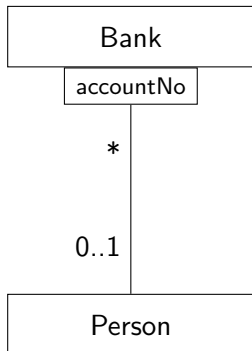
用限定符修饰的关联称为受限关联 (qualified association)



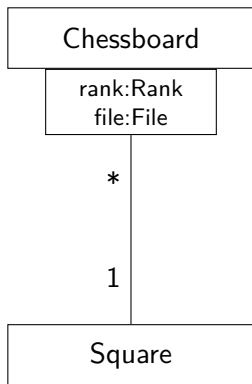
限定符的例子及其简单解决方案



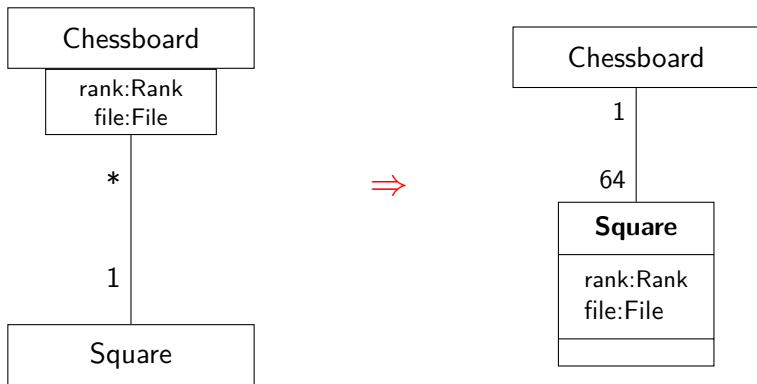
限定符的例子及其简单解决方案



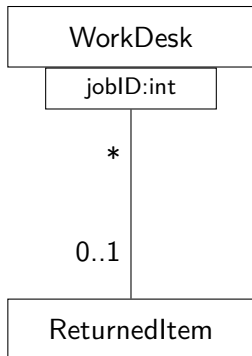
限定符的例子及其简单解决方案



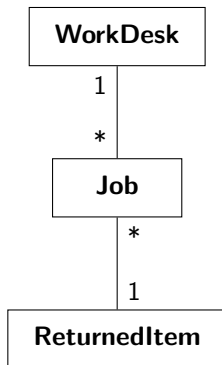
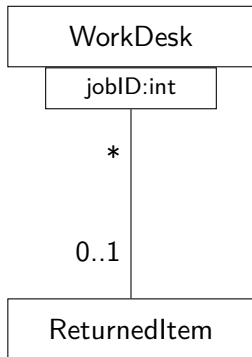
限定符的例子及其简单解决方案



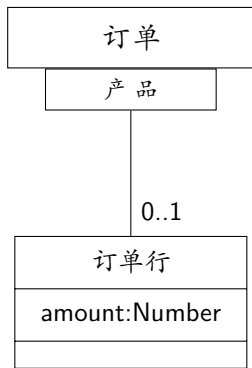
限定符的例子及其简单解决方案



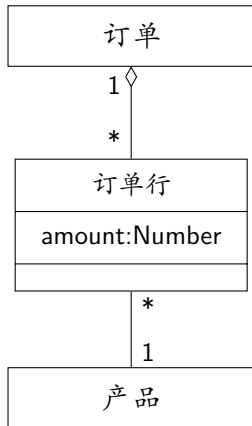
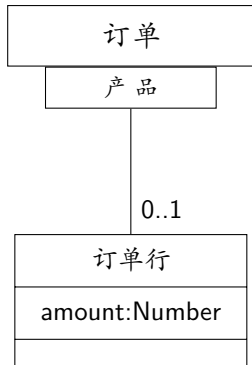
限定符的例子及其简单解决方案



限定符的例子及其简单解决方案



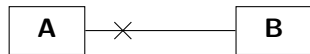
限定符的例子及其简单解决方案



UML2 新增两种图形修饰



聚合种类



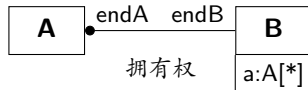
不可导航



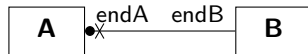
多种修饰



可导航



拥有权



出现悖论

UML2 新增了更多含在花括号内的特性串

| | |
|--|--------------|
| <code>{subsets <property-name>}</code> | 子集合 |
| <code>{redefines <end-name>}</code> | 重定义 |
| <code>{union}</code> | 由它子集合的合并而派生 |
| <code>{ordered}</code> | 有序集合 |
| <code>{nonunique}</code> | 允许同一元素出现一次以上 |
| <code>{sequence}, {seq}</code> | 序列 |

注意 UML2 规范的定义说明

定义：（特性串）缺省表明关联端点代表了一个集合

解读：关联另一端的一个实例与本端的一组实例相关联

正确对待各种复杂概念

- 不鼓励全面使用规范中定义的各种复杂概念
- 重点掌握基本概念，提高用基本概念解决问题的能力 and 技巧
- 尽可能用类、二元关联以及少数几种修饰建立清晰的模型

正确对待各种复杂概念

- 不鼓励全面使用规范中定义的各种复杂概念
- 重点掌握基本概念，提高用基本概念解决问题的能力 and 技巧
- 尽可能用类、二元关联以及少数几种修饰建立清晰的模型

简单为美

内容提要

1 关联关系

- 概念与表示法
- 复杂关联问题
- 如何建立关联

2 交互关系（消息）

3 依赖关系

如何建立关联

- 1 根据问题域和系统责任发现所需要的关联，哪些类的对象实例之间存在着对用户业务有意义的关系？
 - 问题域中实际事物之间有哪些值得注意的关系？
 - 这种信息是否需要通过有序对（或者 n 元组）来体现？
 - 这些信息是否需要在系统中进行保存、管理或维护？
 - 系统是否需要查阅和使用由这种关系所体现的信息？

准则：关联确实为系统提供了有用的信息

如何建立关联

2 简化复杂关联

- 对关联属性和操作的处理
- 对 n 元关联的处理
- 避免一个类在关联中多次出现
- 多对多关联的处理

如何建立关联

2 简化复杂关联

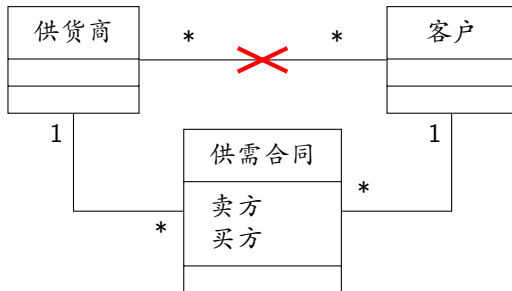
- 对关联属性和操作的处理
- 对 n 元关联的处理
- 避免一个类在关联中多次出现
- 多对多关联的处理



如何建立关联

2 简化复杂关联

- 对关联属性和操作的处理
- 对 n 元关联的处理
- 避免一个类在关联中多次出现
- 多对多关联的处理



如何建立关联

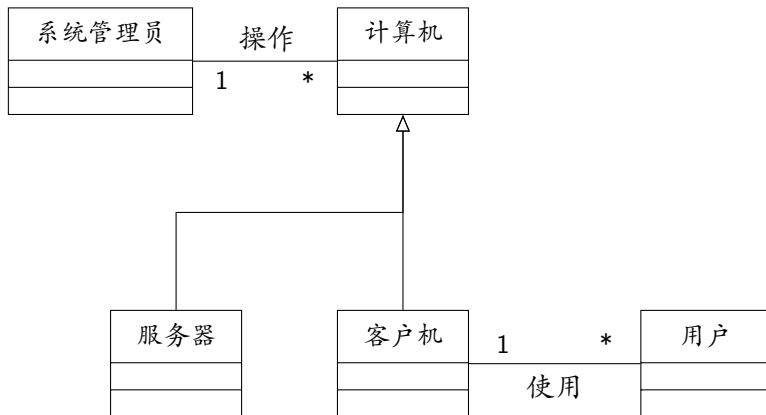
3 为关联端点添加修饰

- 分析关联的多重性
- 给出关联名或角色名
- 识别聚合种类
- 其他修饰
 - 导航性、特性串等——根据实际情况决定是否采用
 - 限定符——用简单的类和关联的概念解决

4 在类中设立实现关联的属性

如何建立关联

5 关联定位



内容提要

1 关联关系

2 交互关系（消息）

- 概念与表示法
- 如何建立消息

3 依赖关系

什么是消息

现实生活中

人或其他事物之间传递的信息，例如：

- 人与人之间的对话、通信、发通知、留言
- 交通信号灯对车辆和行人发出的信号
- 人发给设备的遥控信号等 ...

什么是消息

软件系统中

进程或软件成分之间传送的信息

- **控制信息**: 例如一次函数调用, 或唤醒一个进程
- **数据信息**: 例如传送一个数据文件

什么是消息

面向对象的系统中

(按严格封装的要求) 消息是对象之间在行为上的唯一联系方式

- 狭义: 消息是向对象发出的服务请求
- 广义: 消息是对象之间在一次交互中所传送的信息

消息有发送者和接收者, 遵守共同约定的语法和语义

顺序系统中的消息

- 每个消息都是向对象发出的服务请求, 最常见的是函数调用
- 消息都是同步的
- 接收者执行消息所请求的服务
- 发送者等待消息处理完毕再继续执行
- 每个消息只有唯一的接收者

并发系统中的消息

并发系统

是由多个任务并发执行的系统，系统中含有多个主动对象（或有一个主动对象，但它含有多个主动操作）和若干被动对象

控制流

主动操作实现上对应并发执行的处理机调度单位[进程](#)或[线程](#)，本书统称为控制流

每个控制流是由一系列顺序执行的操作所构成的活动序列
控制流内部的消息和控制流之间的消息

控制流间的消息

■ 消息的多种用途

- 服务请求
- 传送数据
- 发送通知
- 传递控制信号, ...

定义

消息是对象之间一次交互中所传送的信息

控制流间的消息

■ 消息的同步与异步

同步消息 (synchronous)

仅当发送者要发送一个消息而且接收者已做好接收消息的准备时才能传送的消息

异步消息 (asynchronous)

发送者不管接收者是否做好接收准备都可以发送的消息

限时消息 (timeout)

当接收者未准备好时发送者只等待一个限定的时间

- 接收者对消息有不同响应方式
 - 创建控制流响应消息
 - 控制流已存在，立即响应
 - 延迟响应
 - 不响应

- 发送者对消息处理结果有不同期待方式
 - 等待回应
 - 事后查看结果
 - 不等待不查看

- 消息的接收者可能不唯一
 - 定向消息 (directed message)
 - 广播消息 (broadcast message)

- 消息的接收者可能不唯一
 - 定向消息 (directed message)
 - 广播消息 (broadcast message)

OOA 阶段不必考虑所有细节

并发系统消息的多样一方面来源于需求, 一方面来源于实现系统。
OOA 阶段应主要考虑需求问题

消息对面向对象建模的意义

消息体现了对象之间的行为依赖关系，是实现对象之间的动态联系，使系统成为一个能运行的整体，并使各个部分能够协调工作的关键因素

- 在顺序系统中，消息体现了过程抽象的原则
- 在并发系统中
 - 控制流内部的消息，使控制流内部呈现出清晰的脉络
 - 控制流之间的消息，体现了控制流之间的通信关系

OO 模型需要表示消息的哪些信息

重要性递降

- 1 对象之间是否存在着某种消息?
- 2 这种消息是控制流内部的还是控制流之间的?
- 3 每一种消息是从发送者的哪个操作发出的? 是由接收者的哪个操作响应和处理的?
- 4 消息是同步的还是异步的?
- 5 发送者是否等待消息的处理结果?

要不要在类图中表示消息

UML 的处理方式:

不在类图中表示消息，只在协作图和顺序图中表示

理由：把类图定义为静态结构图，不表示动态信息




问题：

类图中的“操作”本身就是动态信息

交互图是实例级的，抽象级别是个别实例

交互图的局部有限性

各种箭头

| 箭头种类 | 图形符号 | 用途 |
|--------|---|------------------------------------|
| 实线开放箭头 |  | 关联的导航性（类图），异步消息（顺序图） |
| 虚线开放箭头 |  | 依赖（类图、包图、用况图、构件图），从消息接收者的操作返回（顺序图） |
| 实线封闭箭头 |  | 同步消息（顺序图、协作图） |

借用依赖关系表示类图中的消息

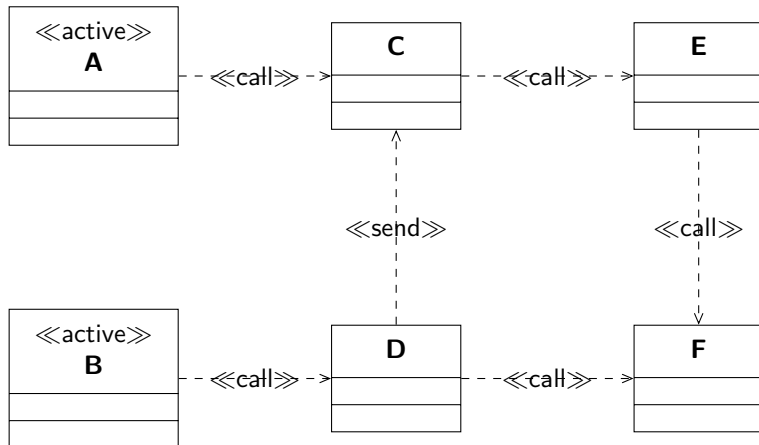
控制流内部消息:



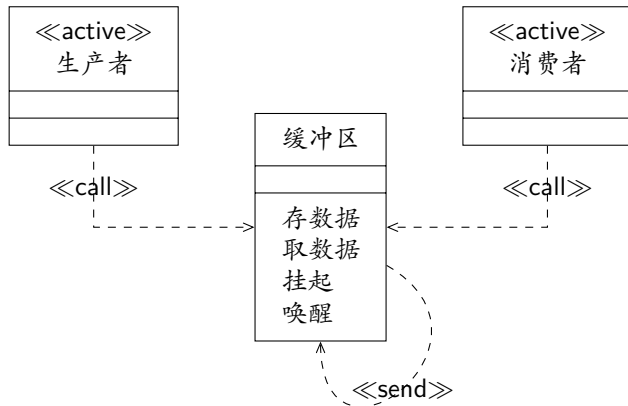
控制流之间消息:



表示法示例



表示法示例



内容提要

1 关联关系

2 交互关系（消息）

- 概念与表示法
- 如何建立消息

3 依赖关系

建立控制流内部的消息

策略——“操作模拟”和“执行路线追踪”

- 1 模拟当前对象操作的执行，考虑需要其它对象提供什么服务
- 2 判断该消息是否属于同一个控制流：
 - 二者应该顺序地执行还是并发地执行？
 - 是否引起控制流的切换？
 - 接收者是否只有通过当前消息的触发才能执行？
- 3 向接收者画出消息连接线，填写模型规约，重复上述工作到当前操作模拟完毕
- 4 对控制流内部的每一种消息，重复上述工作
- 5 针对每个主动类的每个主动操作重复上述工作，检查疏漏

建立控制流之间的消息

对每个控制流考虑以下问题：

- 1 执行时是否需要请求其他控制流中的对象提供服务？
- 2 执行时是否要和其他控制流中的对象交互数据？
- 3 执行时是否将产生某些可影响其他控制流执行的事件？
- 4 各控制流之间是否需要相互传递一些同步控制信号？
- 5 一个控制流将在何种条件下中止执行？在它中止之后将在何种条件下被唤醒？由哪个控制流唤醒？

从上述各个角度发现控制流之间的消息，在相应的类之间画出消息连接线

内容提要

1 关联关系

2 交互关系（消息）

3 依赖关系

■ 依赖关系

什么是依赖（dependency）

UML1.4 对依赖的定义和解释

“依赖：两个建模元素之间的一种关系，其中一个建模元素（独立元素）的一个改变将影响到另一个建模元素（依赖元素）。”

Booch 等《UML 用户指南》的解释

“两个事物之间的语义关系，其中一个事物（独立事物）的改变将影响到另一个事物（依赖事物）。”

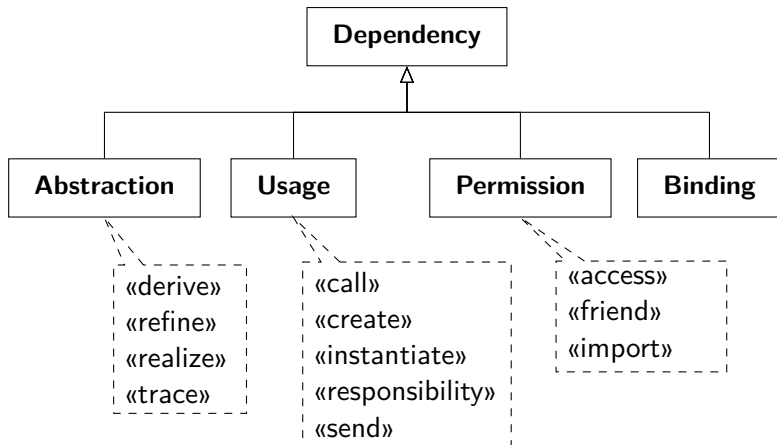
UML2 对依赖关系的新阐述

“依赖表明模型元素之间的供方/客方 (*supplier /client*) 关系，其中供方的修改可能影响到客方元素。依赖意味着，如果没有供方，客方的语义就是不完整的。依赖在一个模型中出现并不含有任何运行时的语义，它完全是以参与这种关系的模型元素的名义而不是以其实例的名义给出的。”

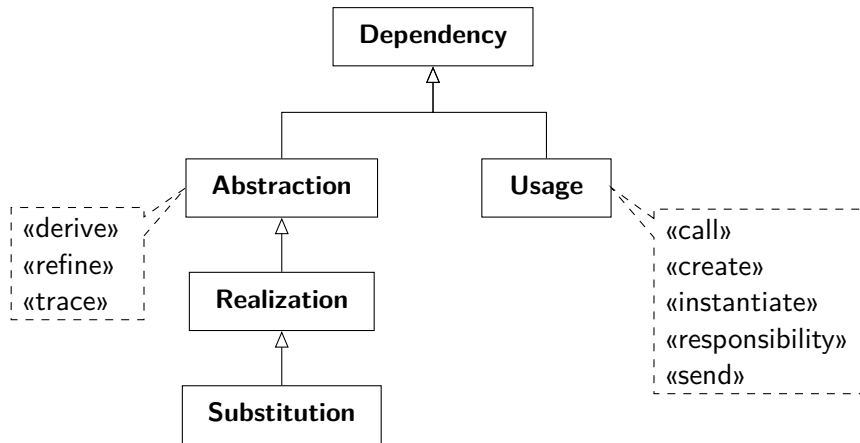
认识：

- 依赖着眼于表达纸面上的模型元素之间的关系，而不是这些元素所描述的客观事物在问题域中的固有关系
- 依赖基本上不能视为一种面向对象的概念

依赖的多种情况: UML1



依赖的多种情况: UML2



其他外观类似的表示

« 关键词 »

----->

1 作为依赖关系标准衍型的关键词

«instantiate»

«create»

«call»

«derive»

«refine»

«responsibility»

«send»

«trace»

其他外观类似的表示

« 关键词 »

----->

2 用于依赖但未明确称为标准衍型的关键词

«abstraction»

«realize»

«substitute»

«permit»

«use»

其他外观类似的表示

« 关键词 »

----->

3 附加于虚线开放箭头但不称为依赖关系的关键词

«access»

«apply»

«bind»

«extend»

«import»

«include»

«manifest»

«occurrence»

«represents»

依赖关系对面向对象建模的作用

- 继承、聚合、关联、消息这四种关系中都已经蕴涵了依赖的含义，不需要再用依赖关系再做重复的表示
- 除了上述关系之外，在 OO 建模中没有多少重要的信息必须用依赖关系表达
- 由于 UML 没有在类图中提供消息的表示法，可以借用依赖关系来表示类图中的消息

注意

避免建立语义含糊不清的依赖关系，更要避免用这些含糊不清的依赖关系代替含义明确的 OO 关系