上海电力学院

Shanghai University of Electric Power

面向对象分析与设计

第九章 问题域部分的设计

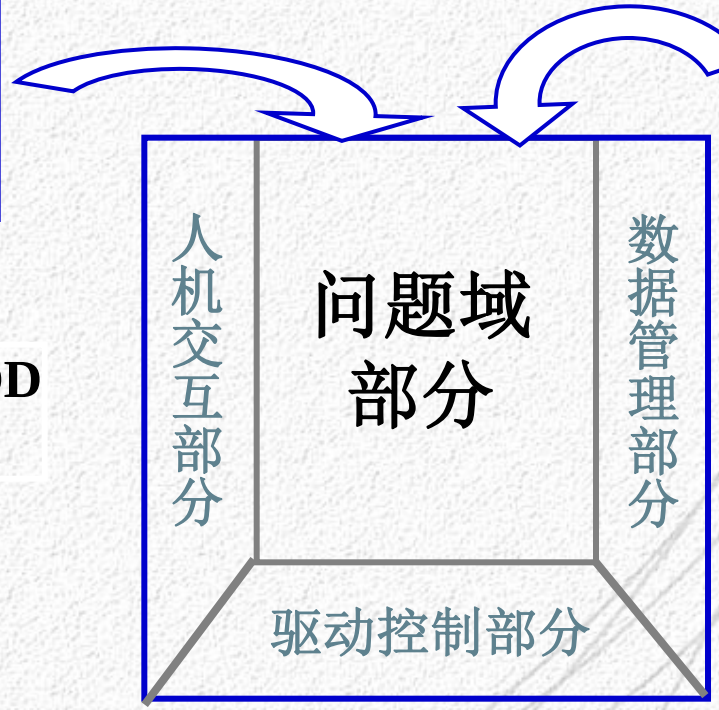
www.shiep.edu.cn

- 对OOA结果按实现条件进行补充与调整就是问题域部分。
- 进行问题域部分设计，要继续运用OOA的方法，包括概念、表示法及一部分策略。不但 要根据实现条件进行OOD设计，而且由于需求变化或新发现了错误，也要对OOA的结果进行修改。本章的重点是对OOA结果进行补充与调整，要强调的是这部分工作主要不是细化，但OOA未完成的细节定义要在OOD完成。

E-mail: zqbi@shiep.edu.cn

OOA模型

将OOA模型搬到OOD作为OOD的基础



按实现条件（编程语言，网络与操作系统，复用支持等）进行必要的调整。

E-mail: zqbi@shiep.edu.cn

（1）为复用设计与编程的类而增加结构
（2）增加一般类以建立共同协议
（3）按编程语言调整继承
（4）提高性能
（5）为数据存储管理增补属性与服务
（6）为编程方便增加底层成分
（7） 决定关系的实现方式

（8）对例外的处理

（9）编程语言限制了可用的属性类型

（10）构造或优化算法

（11）调整服务

（12）决定对象间的可访问性

（13) 考虑采用设计模式

（14）其它

E-mail: zqbi@shiep.edu.cn

（1）为复用设计与编程的类而增加结构

OOA识别和定义的类是本次开发中新定义的，需要进行编程。

如果已存在一些可复用的类，而且这些类既有分析、设计时的定义，又有源程序，那么，复用这些类即可提高开发效率与质量。

可复用的类可能只是与OOA模型中的类相似，而不是完全相同，因此需对二者进行修改。

目标：尽可能使复用成分增多，新开发的成分减少

E-mail: zqbi@shiep.edu.cn

不同程度的复用

当前所需的类的信息 比 可复用类定义的信息

= 直接复用

< 通过继承复用

> 删除可复用类的多余信息

≈ 删除多余信息，通过继承而复用

E-mail: zqbi@shiep.edu.cn

第四种情况的做法：

把要复用的类加到问题域，
　　标以“复用”
　　划掉（或标出）不用的属性与服务

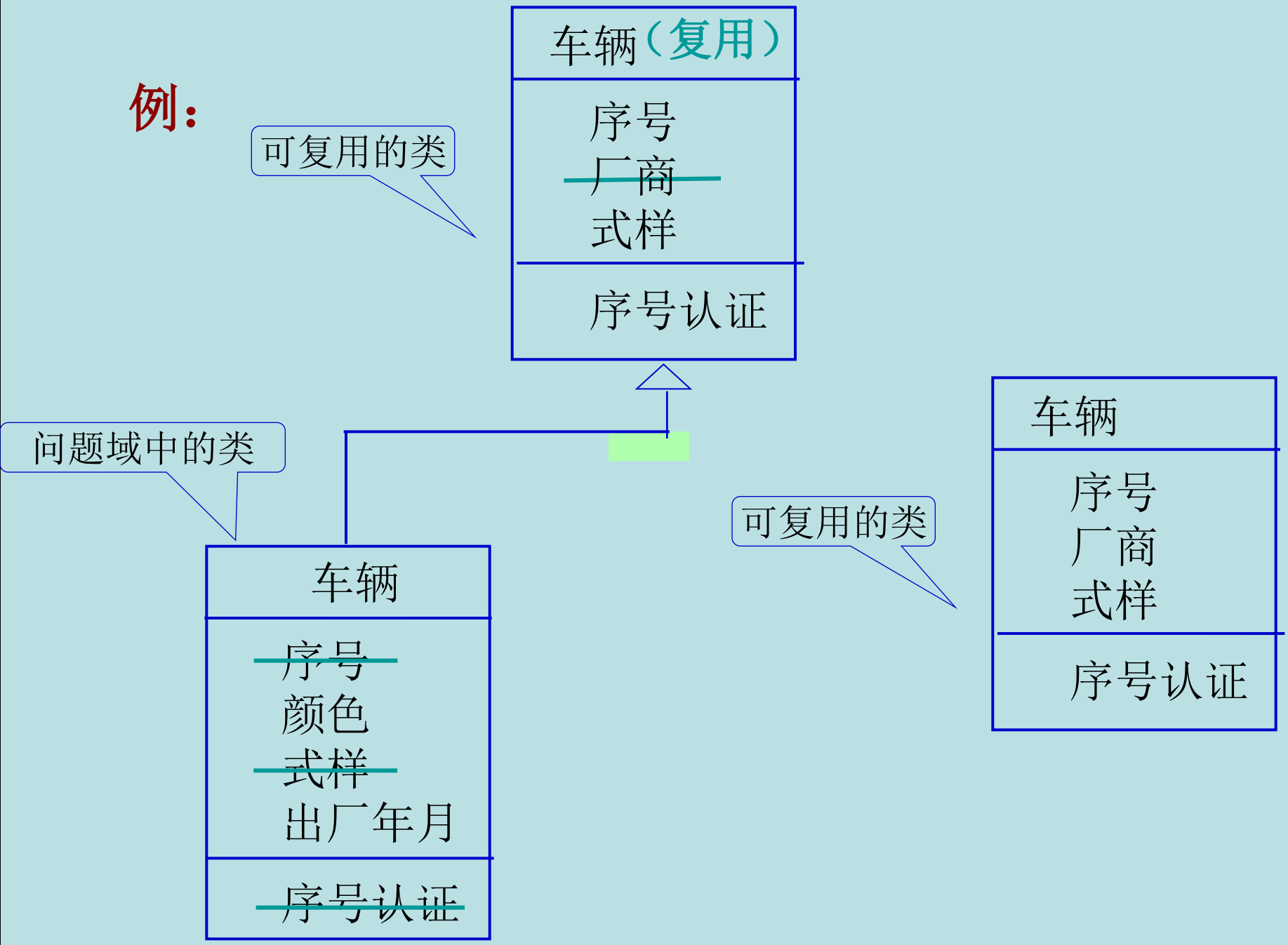
建立从复用类到问题域原有的类之间的泛化关系

由于问题域的类继承了“复用”类的特征，
　　所以有些属性和服务不需要了——划掉。

修改问题域原有类的结构和连接，必要时移到“复用”类

E-mail: zqbi@shiep.edu.cn

例：


车辆（复用）
序号
厂商
式样
序号认证
可复用的类
问题域中的类
车辆
序号
颜色
式样
出厂年月
序号认证
可复用的类
车辆
序号
厂商
式样
序号认证

（2）增加一般类以建立共同协议

增加根类：将所有的具有相似协议的类组织在一起

提供通用的协议

例：提供创建、删除、复制等服务

增加其他一般类：提供局部通用的协议

例：提供永久存储及恢复功能

E-mail: zqbi@shiep.edu.cn

对相似的操作的处理

有时在若干类中定义了相同的操作,就能够从共同的超类中继承它们。

然而，在不同类中的操作经常是相似的，而不是相同的。这时需要对操作原型做小的修改，使得操作就可以相互相匹配，即不仅操作的名字和特征标记必须相匹配，而且它们应该都具有相同的语义含义。

调整继承的策略

1．若有些操作比其他操作具有更少的参数，要加入所没有的参数，但在这些操作的方法中忽略新加入的参数。---缺点：对消息规约，有些麻烦。 (慎用)

2．若在一个类中不需要父类中定义的操作，就在该类中把它声明为一个无操作的方法。 (慎用)

E-mail: zqbi@shiep.edu.cn

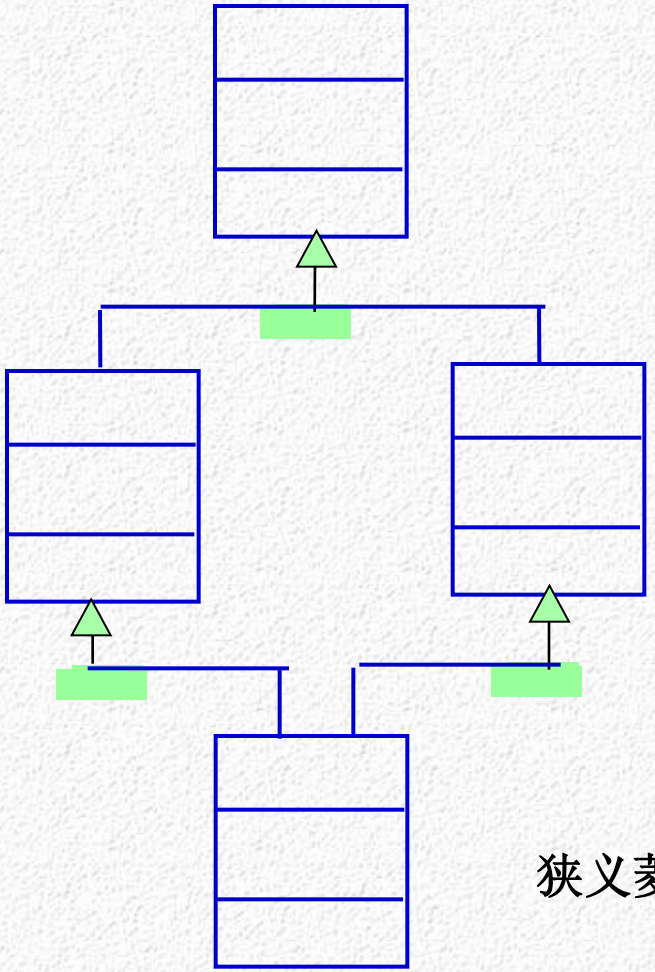
（3）按编程语言调整继承

起因：OOA强调如实地反映问题域，OOD考虑实现问题，

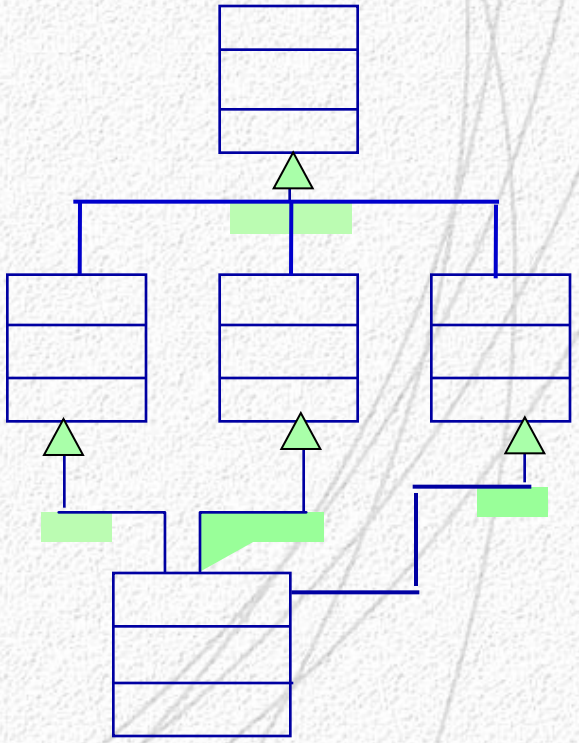
所用语言不支持多继承，甚至不支持继承

E-mail: zqbi@shiep.edu.cn

多继承模式



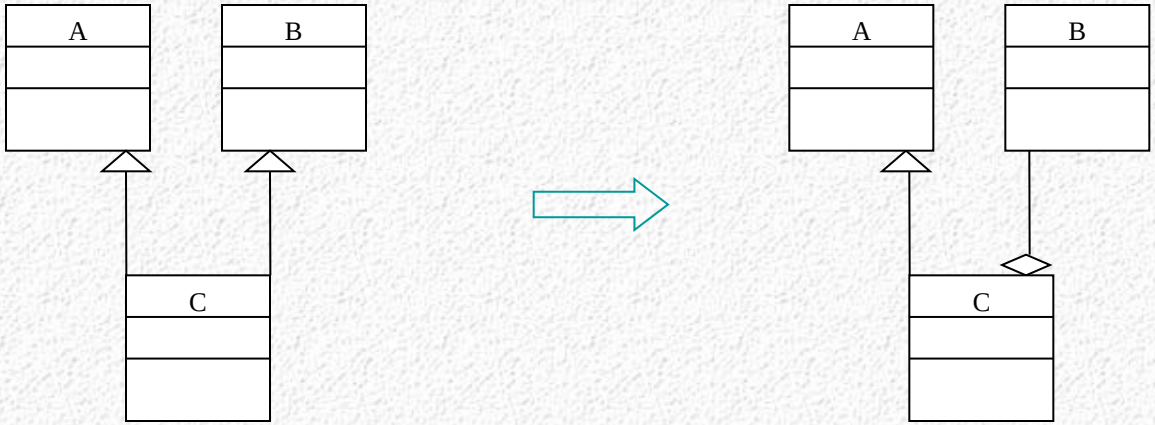
狭义菱形



广义菱形

E-mail: zqbi@shiep.edu.cn

把多继承调整为单继承

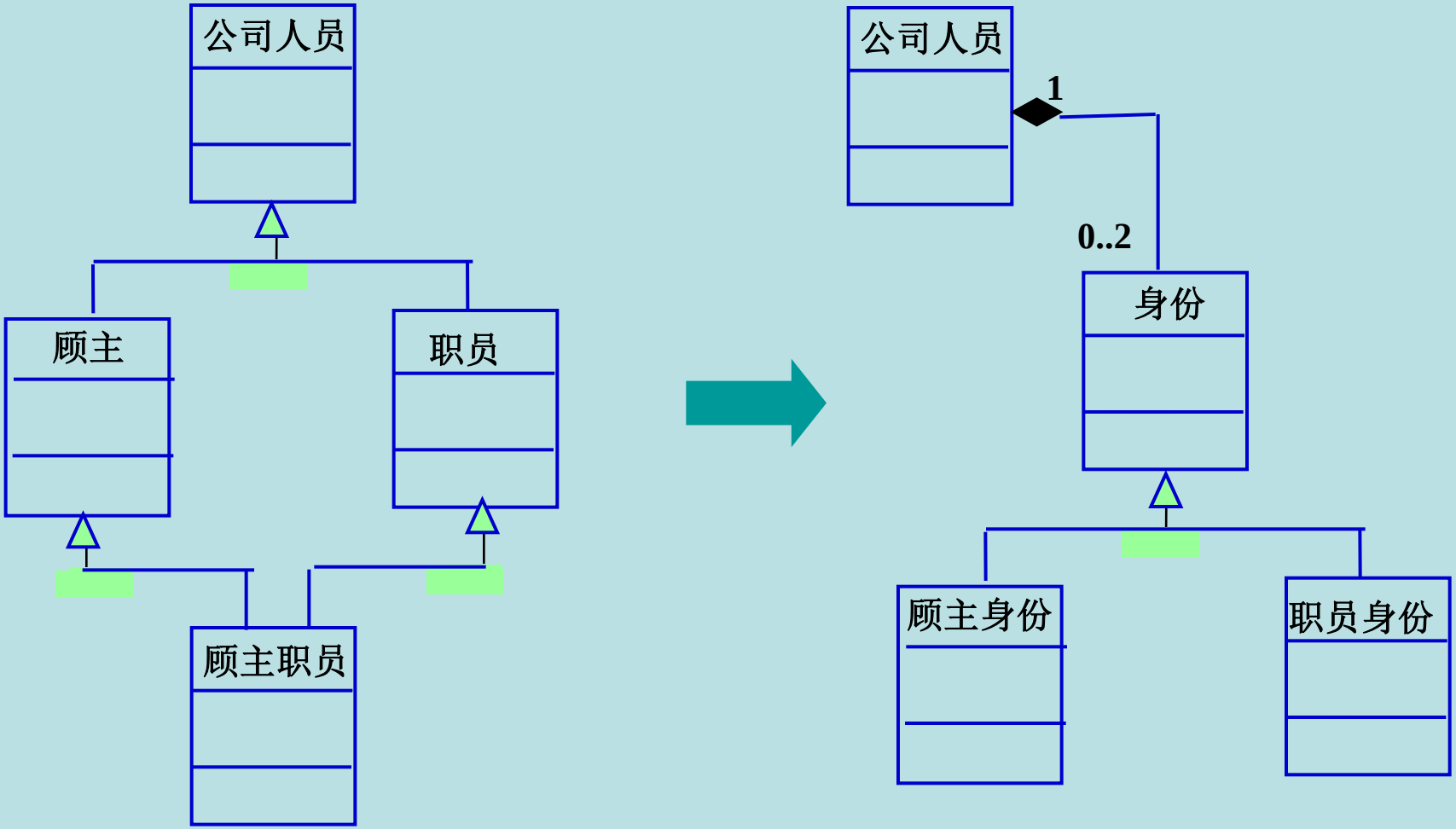


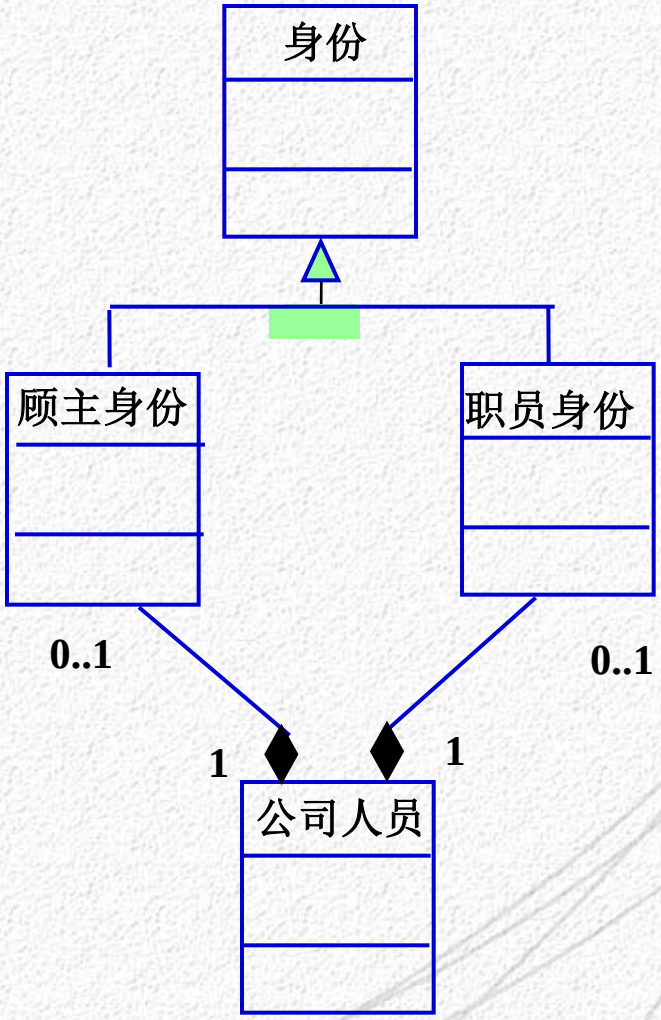
因为聚合和泛化是不同的概念，这种方法并不是通用的(按定义)。

在大多数情况下，需要考虑形成多继承的原因，将本来在特殊类中显式定义的信息离出来，作为部分对象，以原来的一般类作为整体对象。

E-mail: zqbi@shiep.edu.cn

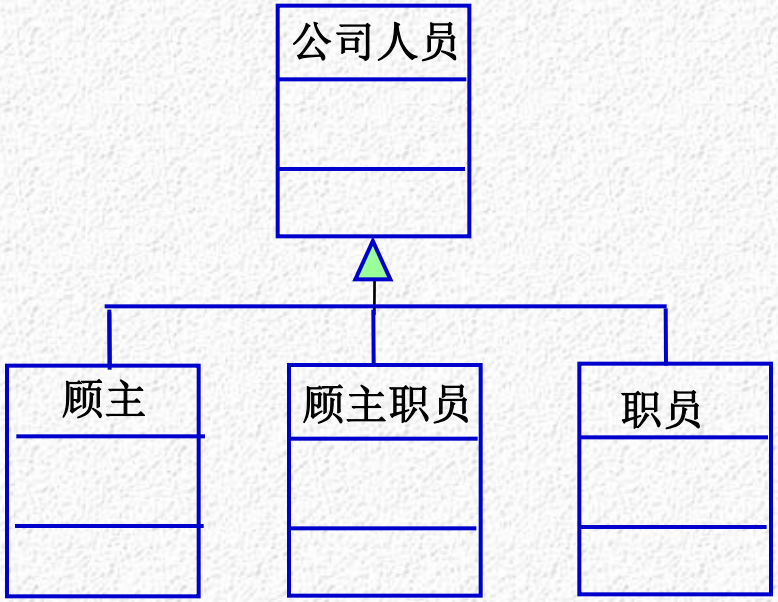
方法1：采用关联





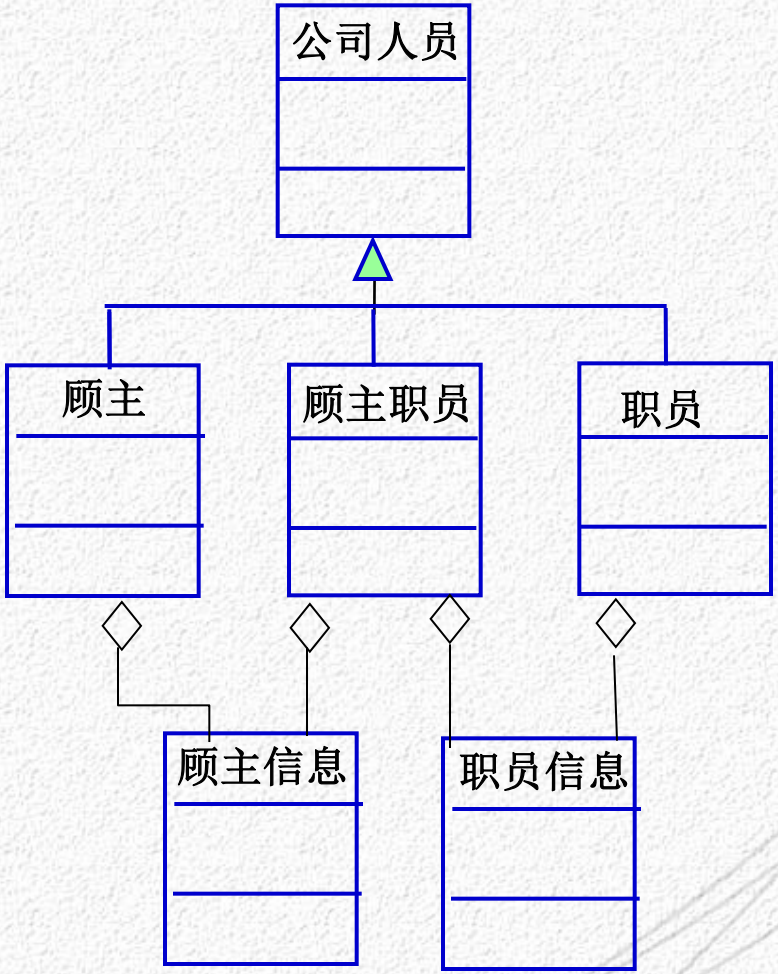
E-mail: zqbi@shiep.edu.cn

方法2：压平



问题：有什么缺点？

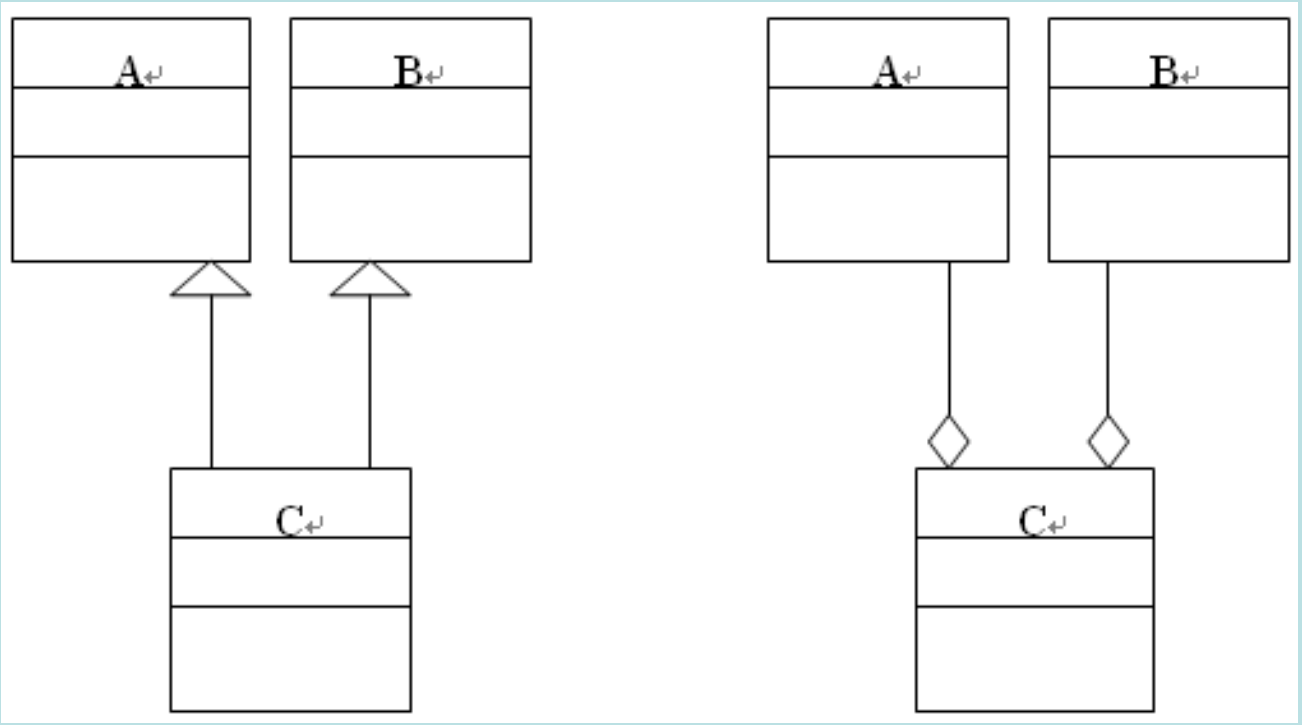
E-mail: zqbi@shiep.edu.cn



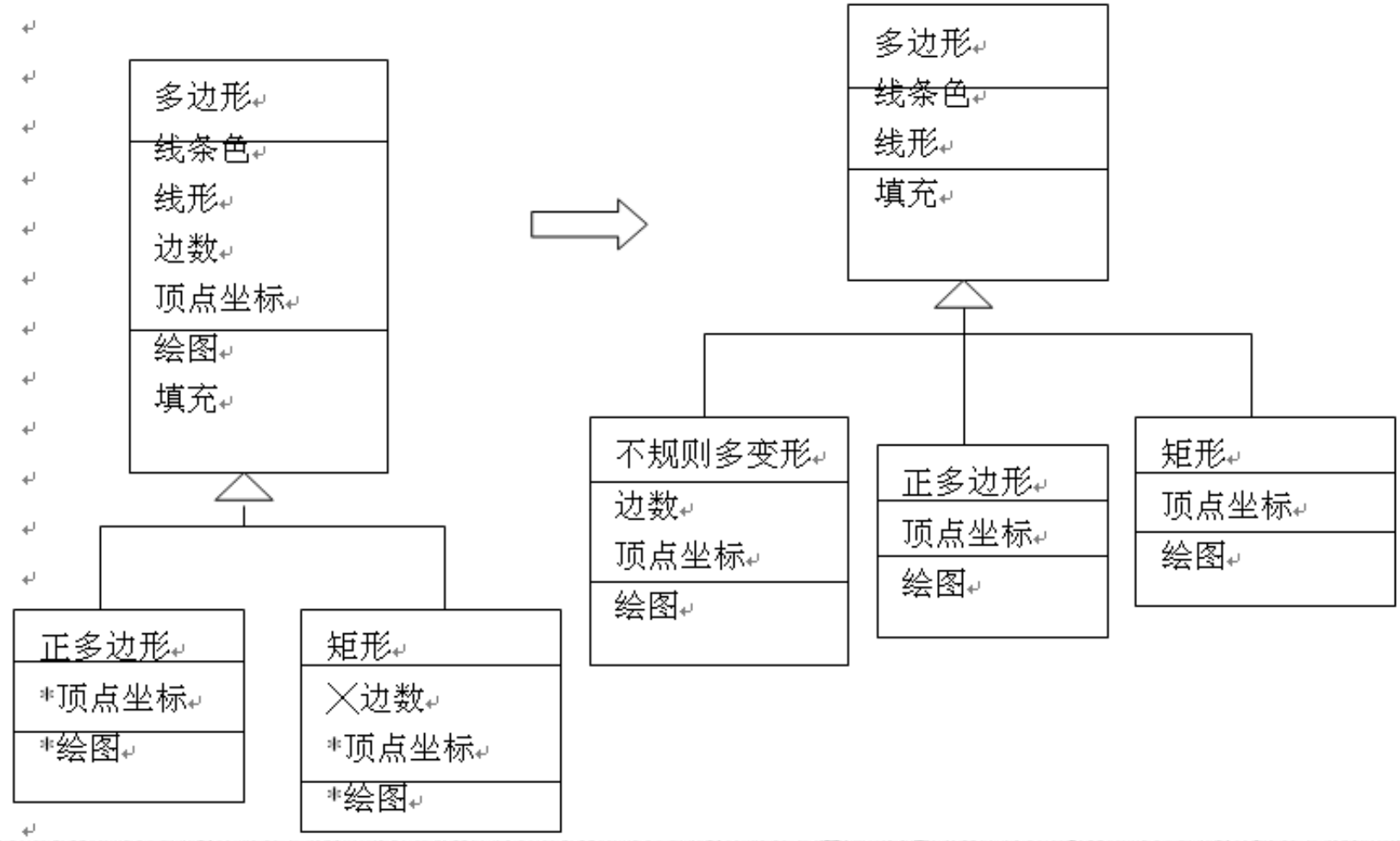
E-mail: zqbi@shiep.edu.cn

方法3：取消泛化

问题：有什么缺点？



对多态性的调整

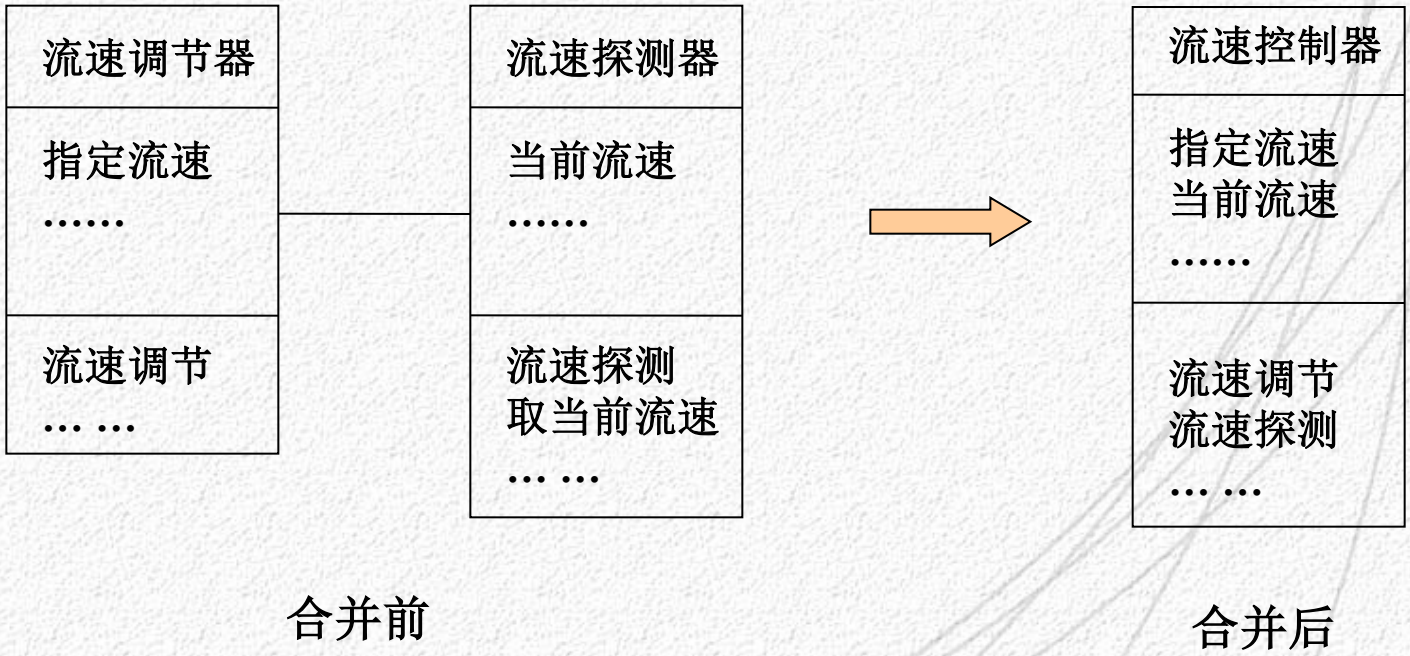


E-mail: zqbi@shiep.edu.cn

（4）提高性能

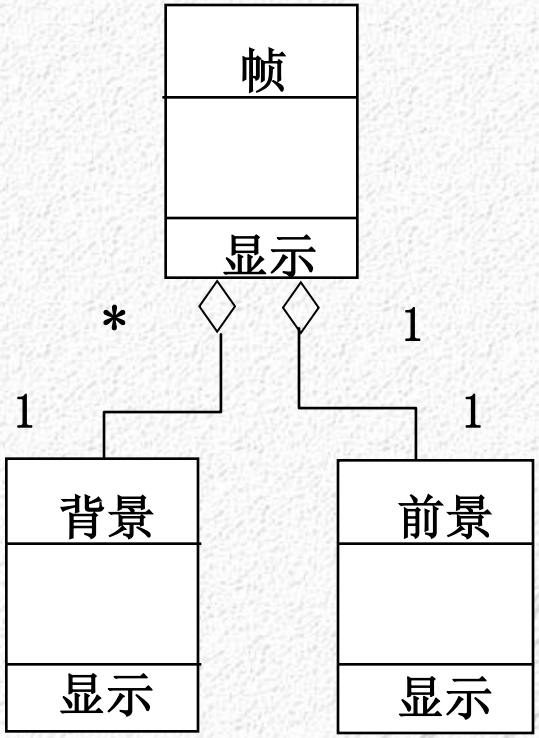
- 把需要频繁交换信息的对象，尽量地放在一台处理机上。
- 增加属性或类，以保存中间结果
- 提高或降低系统的并发度，可能要人为地增加或减少主动对象。
- 合并通讯频繁的类

E-mail: zqbi@shiep.edu.cn



E-mail: zqbi@shiep.edu.cn

•用聚合关系描述复杂类

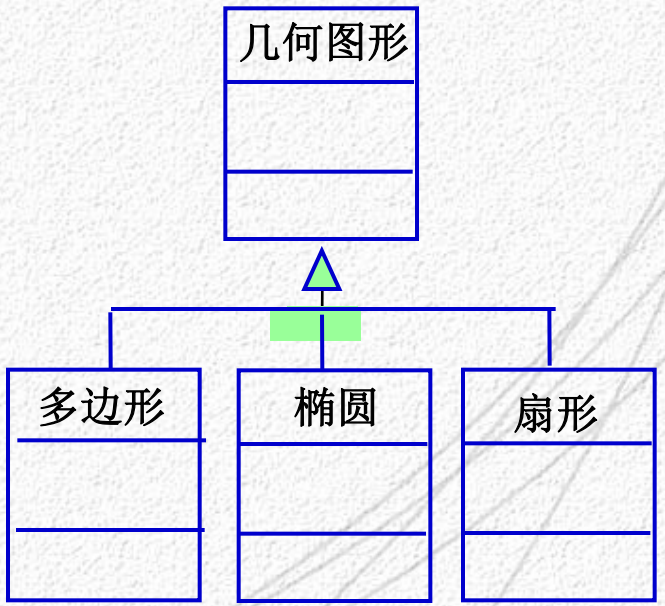


E-mail: zqbi@shiep.edu.cn

- 为编程方便增加底层成分

——细化对象的分类

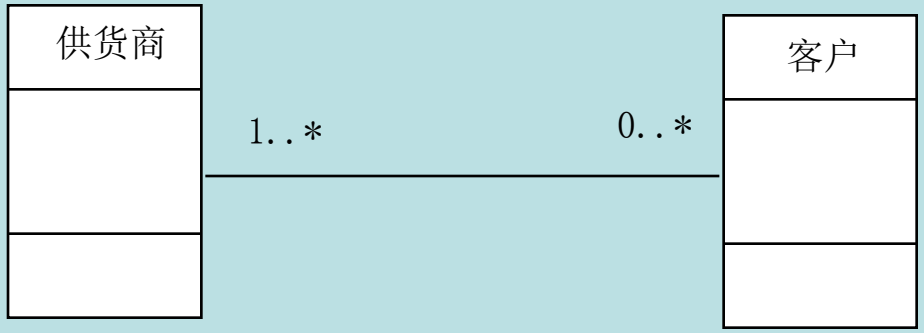
例：将几何图形分成多边形、椭圆、扇形等特殊类

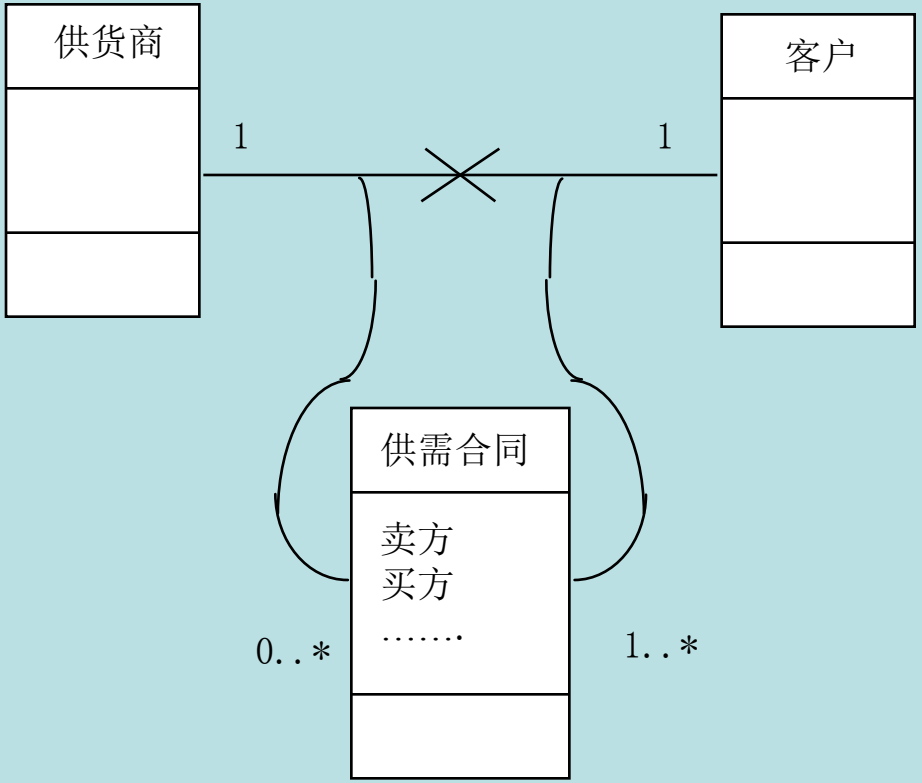


E-mail: zqbi@shiep.edu.cn

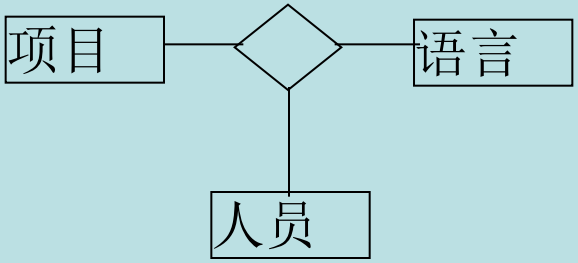
(5)对复杂关联的转化并决定关联的实现方式

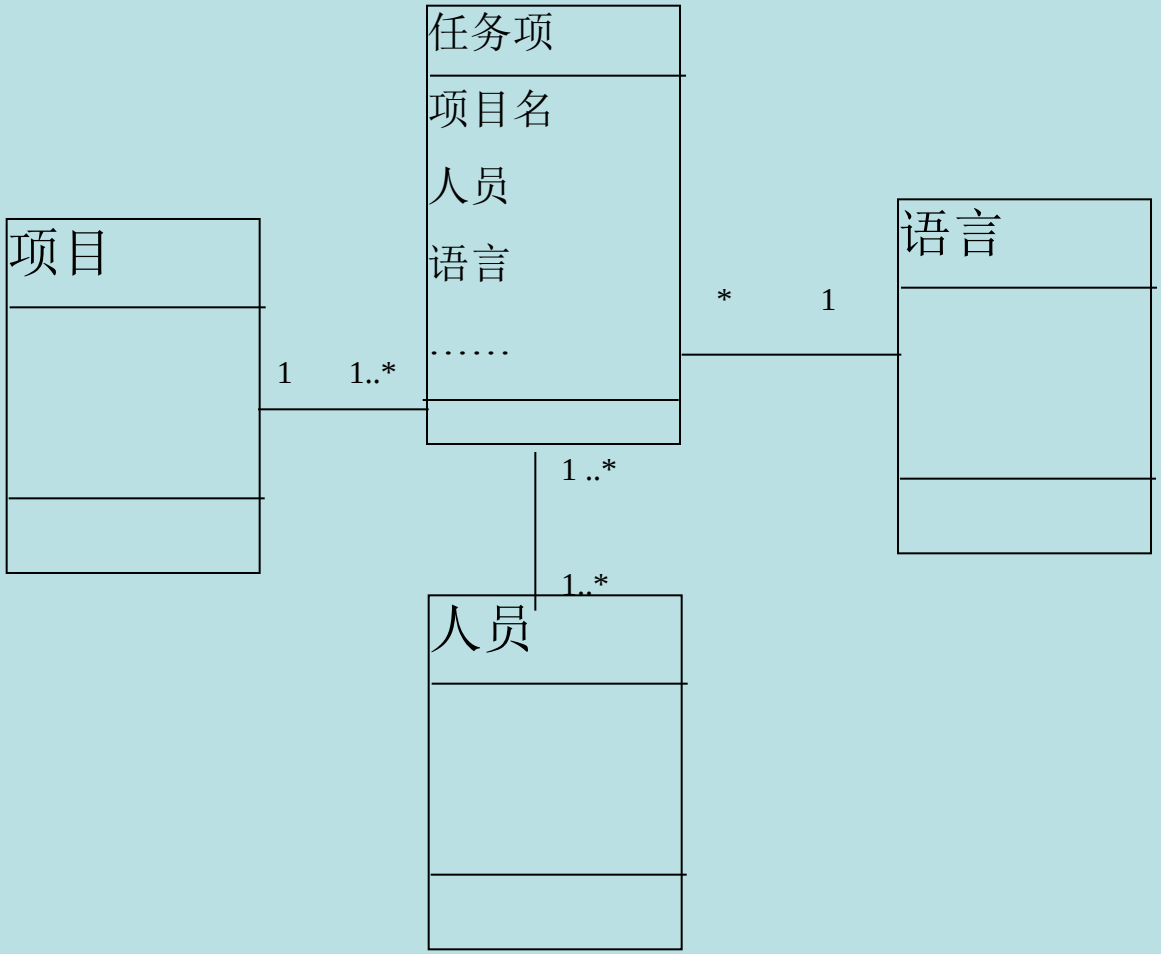
把多对多关联转化为一对多关联



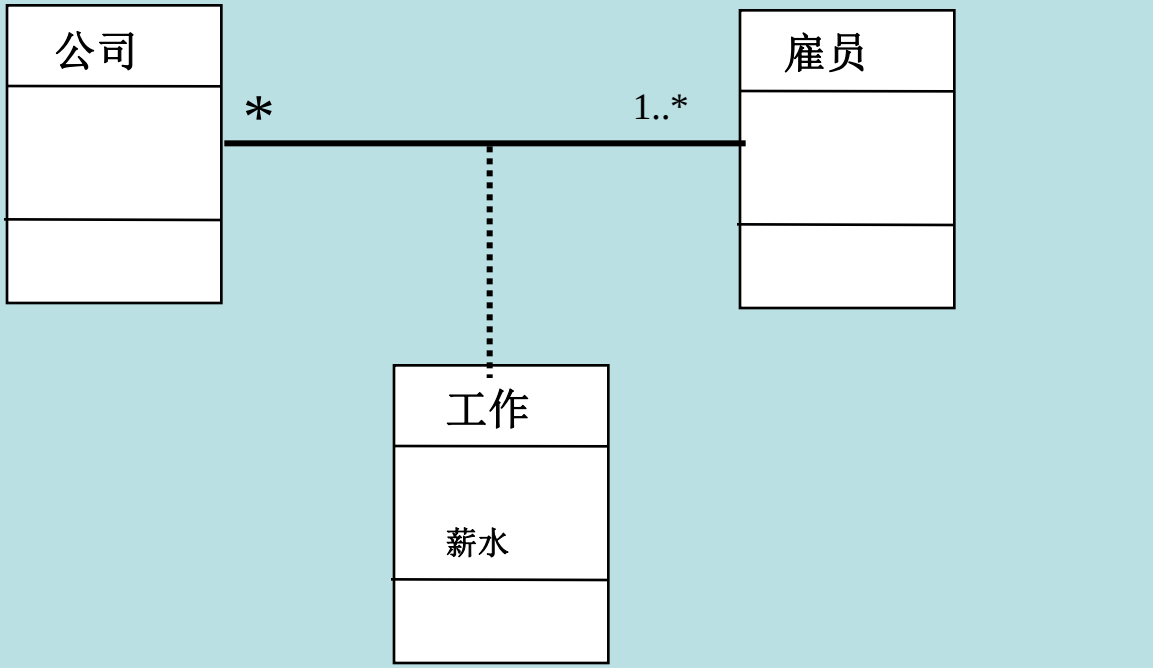


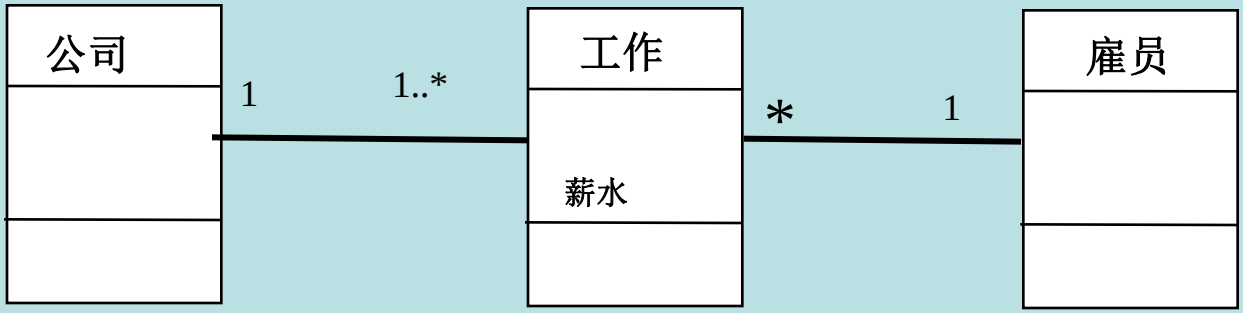
把多元关联转化为二元关联





把关联类转化为二元关联





■ 对关联进行调整后，要考虑关联的实现方式。

（1）聚合

- 决定在整体类中指出部分类时，是用部分类直接作为整体类中的属性的数据类型，还是把部分类用作指针或对象标识的基类型，再用这样的指针或对象标示定义整体类的属性。
- 如果是组合，最好用第1种方式，否则就需要在程序中保证整体对象与部分对象的生命周期的一致性。

E-mail: zqbi@shiep.edu.cn

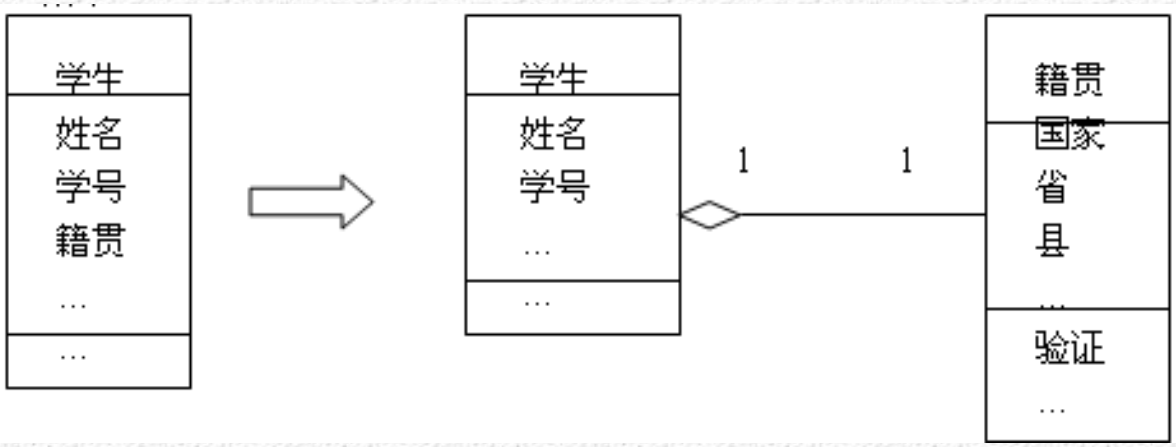
（2）关联

- 通常，通过在对象中设立指针或对象标识以指向或记录另一端的对象的方法，来实现关联。
- 如果是单向关联，就在源端的类中设立属性，用来标记另一端的类将来创建的对象
- 如果是双向关联，就在两端类中各设立属性，用来标记对方将来创建的对象。
- 如果关联中对方类的多重性是1，那么可在本方设立一个指向对方对象的指针，或设立一个记录对方对象引用的属性。
- 如果对方类的多重性大于1，那么可在本方设立一个指向对方对象的指针集合或引用集合。
- 若关联的某端有角色名，最好把其作为另一端类的属性名，以访问与角色名相邻的类。

E-mail: zqbi@shiep.edu.cn

（6）调整与完善属性

按照语法：[可见性] 属性名[‘：’ 类型][‘=’ 初始值] 对属性的定义进行完善。
每一个属性或者包含单个值，或者包含作为一个整体的密切相关的一组值



若要给出对属性的性质的约束，如“工龄<60”或“0≤英语成绩≤100”等，
也要看语言是否对其直接支持，否则要在算法上考虑如何实现。

为了维护数据的完整性，必须要考虑需要一起更新的多个相关联的数据值。特别是，当基本的数据发生变化时，必须更新导出的属性。通过下列方法可以做到这一点：

1)显式的代码

因为每一个导出属性是根据一个或多个基本对象属性定义的，更新导出属性的一种方法是，在更新基本对象属性的操作中插入更新导出属性的代码。这种附加的代码将明确地更新依赖基本对象属性的导出属性，使得基本属性与导出属性的值同步。

2)批处理性的重计算

当基本数据以批处理的方式改变时，可能在所有的基本数值改变之后，再重新计算所有的导出属性的值。

3)触发器

凡是依赖基本属性的属性，都必须将它自己向基本属性注册。当基本属性的值被更新时，由专门设置的触发器更新导出属性的值。

E-mail: zqbi@shiep.edu.cn

（7）构造和优化算法

对于需要设计的操作，要从如下几方面进行详细地定义：

1） 按照定义操作的格式： [可见性] 操作名[‘(’参数列表 ‘)’][‘:’返回类型] 完善操作的定义。

2）从问题域的角度，根据其责任，考虑实现操作的算法，即对象是怎样提供操作的。

3）若操作有前后置条件或不变式，考虑编程语言是否予以支持。若不支持，在操作的方法中要予以实现。

4）建议进一步地分析特定类的对象相关的所有交互图，找出其所有与之相关的消息。一个对象所要响应的每个消息都要由该对象的操作处理，其中的一个操作也可能要使用其他操作。如果类拥有状态图，还可根据内部转换以及外部转换上的动作，设计算法的详细逻辑。

E-mail: zqbi@shiep.edu.cn

可用自然语言或进行了一定结构化的自然语言描述算法，也可以使用程序框图或活动图描述算法。

在算法中还要考虑对例外和特殊情况的处理。如考虑对输入错误、来自中间件或其它软硬件的错误的消息以及其它例外情况的处理。

在系统较为复杂或需要处理大批量的数据的情况下，若系统在性能上有要求，就要对系统的体系结构和算法进行优化。

E-mail: zqbi@shiep.edu.cn

（8）决定对象间的可访问性(M)

从类A的对象到类B的对象有4种访问性

属性可见性：B是A的一个属性（关联、聚合）

```
class A
{  … ;B b;…}
```

参数可见性：B的对象是A的一个方法的参数（依赖）

```
A.amethod(B b)  //间接地找到一个对象，并赋给b。
```

局部声明可见性：B的对象是在A的一个方法中声明的一个局部变量（依赖）

```
class A::amethod
{  …;  B b;…}
```

全局可见性：B的对象在某种程度上全局可见（依赖）

声明B的全局实例变量

E-mail: zqbi@shiep.edu.cn

对于后三种情况而言，从类A到类B间存在着依赖关系，在程序运行期间A的对象与B的对象存在着临时性的连接（临时链），而第一种情况中的链是由从类A到类B间的关联实例化而来的。

E-mail: zqbi@shiep.edu.cn

（9） 定义对象实例

- 在逻辑上，一个类是对一组对象的抽象描述。在物理上，一个类所创建的各对象，要么在内存中，要么在外存中。在内存中创建的一个对象，用一个变量记录它的标识。在外存中的对象，可能保存在一个文件中，也可能保存在一个数据库表中。

E-mail: zqbi@shiep.edu.cn

在OOD中，根据不同的实现条件和实现策略，可以按如下的方式定义对象：

（1）用相应的类定义内存中的全局性对象，包括静态声明和动态创建两种方式。可以一次针对一个对象定义一个变量，也可以成批地定义对象。例如，可以定义一个数组，它的每个元素是一个对象变量，以此来成批地定义对象。

（2）当系统需要通过从外存读取数据来创建一个对象时，先创建该对象，再从外存中读取这个对象数据，把数据赋值给对象的相应属性。按照一定的策略，内存中的永久对象要保存到外存中，请参看数据管理部分。

无论那种方式，都需要在在OOD文档中加以说明。按如下格式在类描述模板的定义对象部分进行描述：

处理机：<节点名>{，<节点名>}；
内存对象：{<名称>[（n元数组）][<文字描述>]}；
外存对象：{<名称>[<文字描述>]}；

E-mail: zqbi@shiep.edu.cn

(10)其它

在OOD的问题域部分应该根据具体问题考虑使用设计模式。

在OOD的问题域部分，根据情况，还有一些其它需要考虑的问题。例如，考虑加入进行输入数据验证这样的类；考虑对来自中间件或其它软硬件的错误进行处理的类，以及对其它例外情况进行处理的类。

有些作法是在OOD阶段不把这样的读写属性的操作放在类中，而认为这是一种约定，编程人员能理解。

有些作法也不把诸如创建和复制对象这样的操作放在OOD模型中。

注意：

包容器/集合类（如JAVA的Vector,Hash table），是已经预定义的类库的一部分，一般不在类图中重新定义，可进行引用或继承。

不要错把包容器/集合类中方法（如JAVA的find）,在其他类中重新定义，可进行引用或继承。

E-mail: zqbi@shiep.edu.cn