表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

②泛化（generalization）

定义：

泛化是一般性事物（称为超类或父类）和它的较为特殊种类（称为子类）之间的一种关系，有时称为“is-a-kind-of”关系。

4点说明：

❶子类可继承父类的属性和操作，并可有更多的属性和操作；

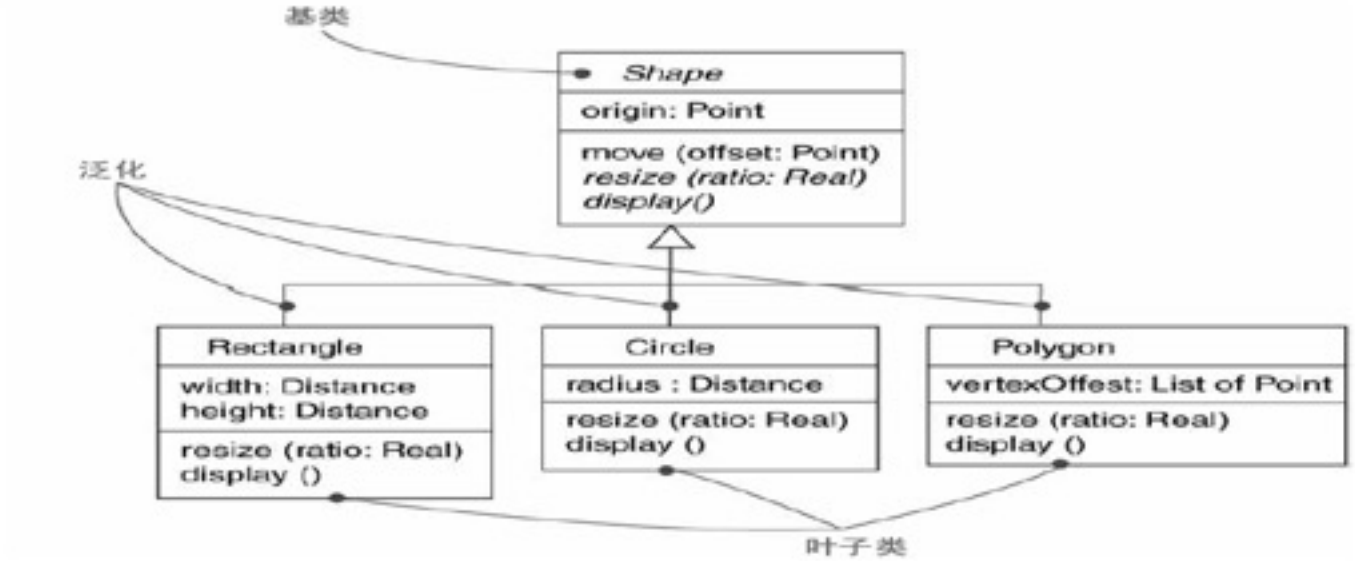
❷子类可以替换父类的声明；

❸若子类的一个操作的实现覆盖了父类同一个操作的实现，这种情况被成为多态性，但两个操作必须具有相同的名字和参数。

表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

❹一个类可以有0个、1个或多个父类。没有父类且最少有一个子类的类被称为根类或基类；没有子类的类称为叶子类。如果一个类只有一个父类，则说它使用了单继承；如果一个类有多个父类，则说它使用了多继承。



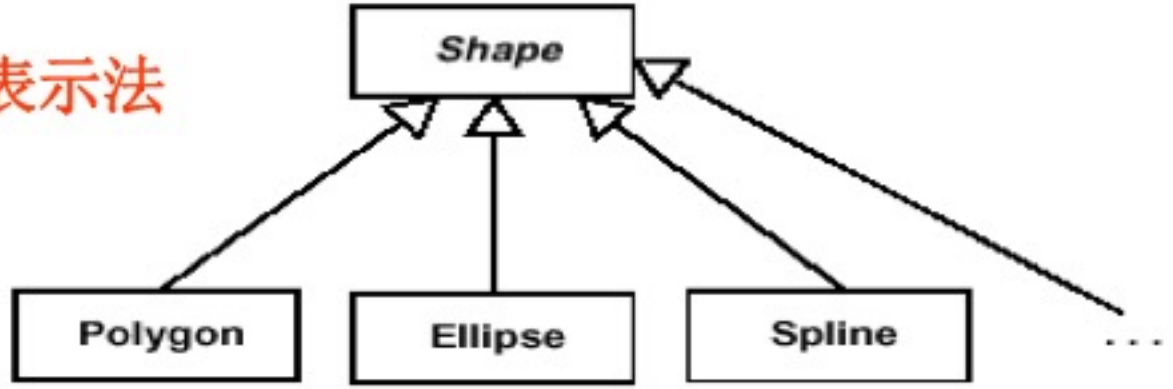
注：在大多数情况中，用类和接口之间的泛化来表明继承关系。在UML中，也可在其他类目之间创建泛化，例如在结点之间。

表达关系的术语

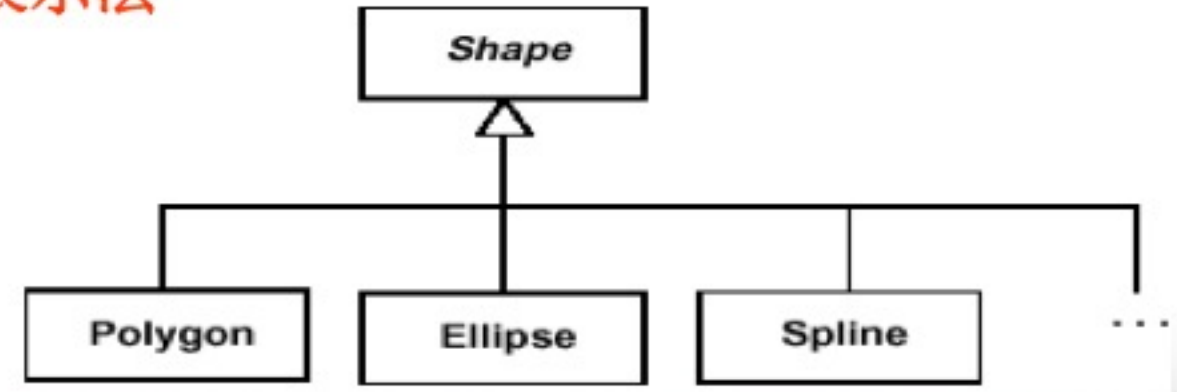
- 关联
- 泛化
- 实现
- 依赖

表示：

分离表示法



共享表示法



表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

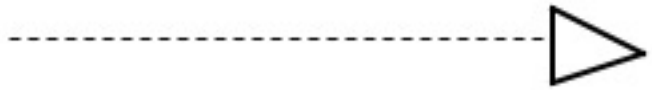
③细化（也称为实现）（realization）

定义：细化是类目之间的一种语义关系，其中一个类目规约了保证另一个类目执行的契约。

说明：在以下2个地方会使用细化关系：

- 接口与实现它们的类和构件之间；
- 用况与实现它们的协作之间。

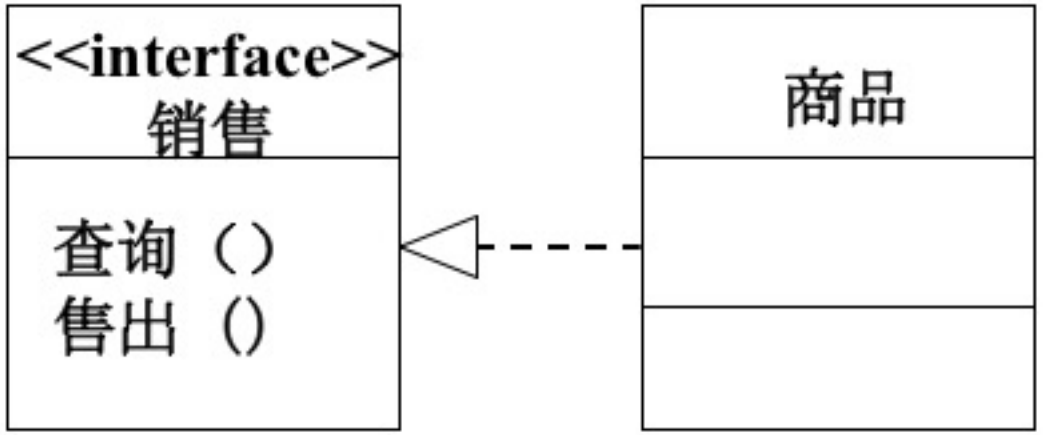
表示：



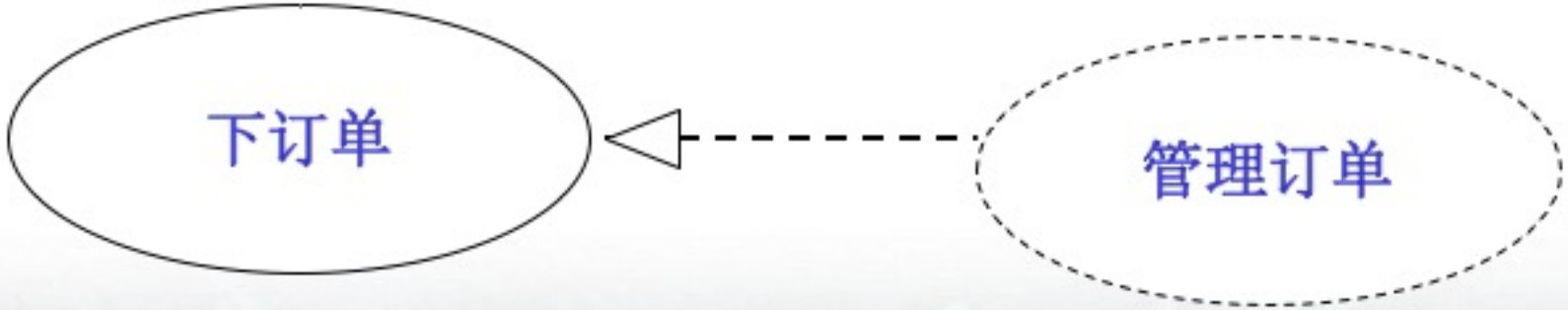
表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

例1：接口和实现它们的类之间的关系



例2：用况和实现它们的协作之间的关系



表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

④依赖

定义：依赖是一种使用关系，用于描述一个事物（如类Window）使用另一事物（如类Event）的信息和服务。

3点说明：

❶在大多数情况里，使用依赖来描述一个类使用另一个的操作；

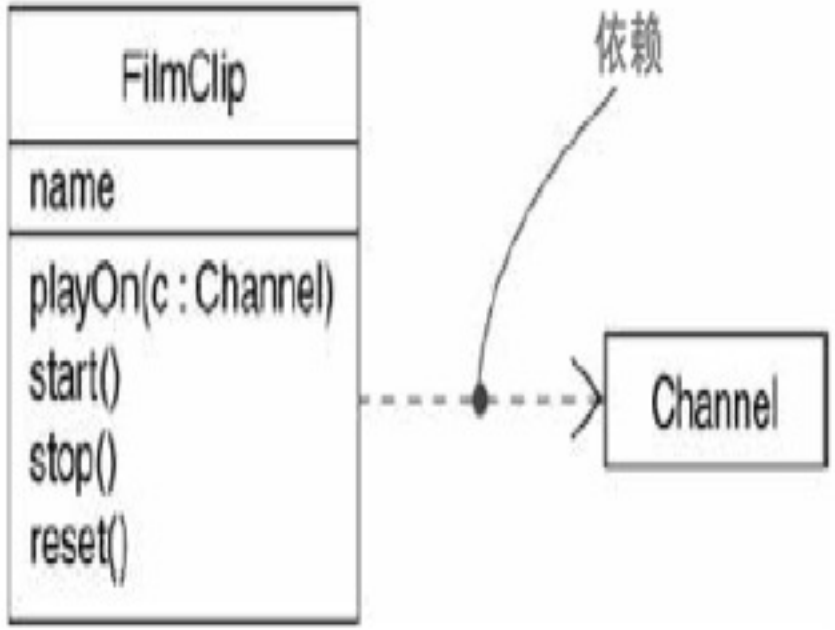
❷如果被使用的类发生变化，那么另一个类的操作也会受到影响；

❸依赖可用于其它事物之间，例如注解之间和包之间。

表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

表示：一条有向虚线。例如：



表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

为了进一步表达依赖的语义，UML对依赖进行了分类，并给出了相应的标记。

❶绑定（bind）：表明源的实例化是使用目标给定的实际参数来达到的。例如，可以把模板容器类（目标）和这个类实例（源）之间的关系模型化为绑定。其中绑定涉及到一个映射，即实参到形参的映射。

❷导出（derive）:表明可以从目标推导出源。例如类Person有属性“生日”和“年龄”，假定属性“生日”是具体的，而“年龄”是抽象的，由于“年龄”可以从“生日”导出，因此可以把这两个属性之间的这一关系模型化为导出。

表达关系的术语

- 关联
- 泛化
- 实现
- 依赖

❸允许（permit）：表明目标对源而言是可见的。一般情况下，当许可一个类访问另一个类的私有特征时，往往把这种使用关系模型化为允许。

❹实例（instanceOf）：表明源的对象是目标的一个实例。

❺实例化（instantiate）：表明源的实例是由目标创建的。

❻幂类型（powertype）：表明源是目标的幂类型。幂类型是一个类目，其对象都是一个给定父类的子类。

❼精化（refine）：表明源比目标更精细。例如在分析时存在一个类A，而在设计时的A所包含的信息要比分析时更多。

❽使用（use）：表明源的公共部分的语义依赖于目标的语义.

UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

以上谈到的4个术语，是UML模型中可以包含的基本关系。它们也有一些变体，例如精化、跟踪、包含和扩展等。

四种关系的一般用法：

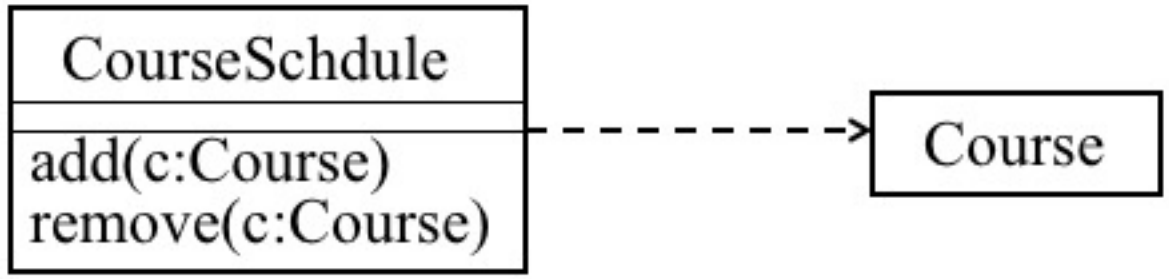
① 模型化简单依赖

例如，一种常见的依赖关系是：一个类只是使用另一个类作为它的操作参数。

对此,可从含有操作的类到被该操作用做参数的类创建一个依赖。即:

UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系



注：如果操作**add**和**remove**给出了明显的操作标记（**c:Course**，如上所示），则一般就不需要给出这个依赖；但当省略操作标记时或一个模型还描述了被使用类的其它关系时，就应显示这一依赖。

UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

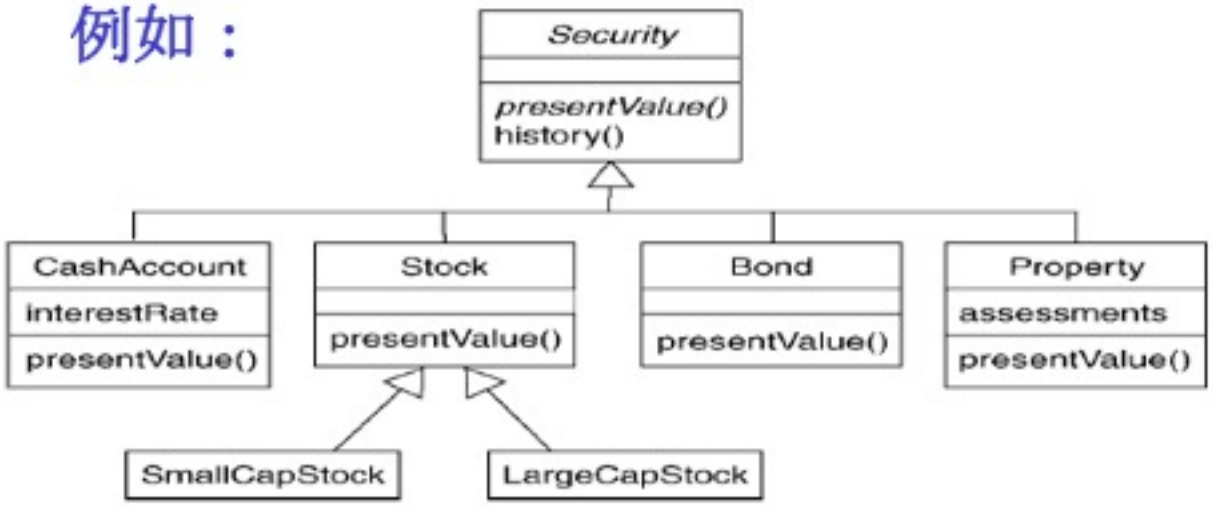
②模型化单继承

第一步:对于给定的一组类，发现2个或2个以上类的共同责任、属性和操作。

第二步:把发现的共同责任、属性和操作放到一个一般类中其中要注意，不要引入过多的层次。

第三步:画出从每个特殊类到一般类（父类）的泛化关系。

例如：



注：
- 斜体字表明是一个抽象类或抽象操作；
- 子类中给出的操作为非斜体字，表明给出了操作的实现。

UML基本关系的一般用法

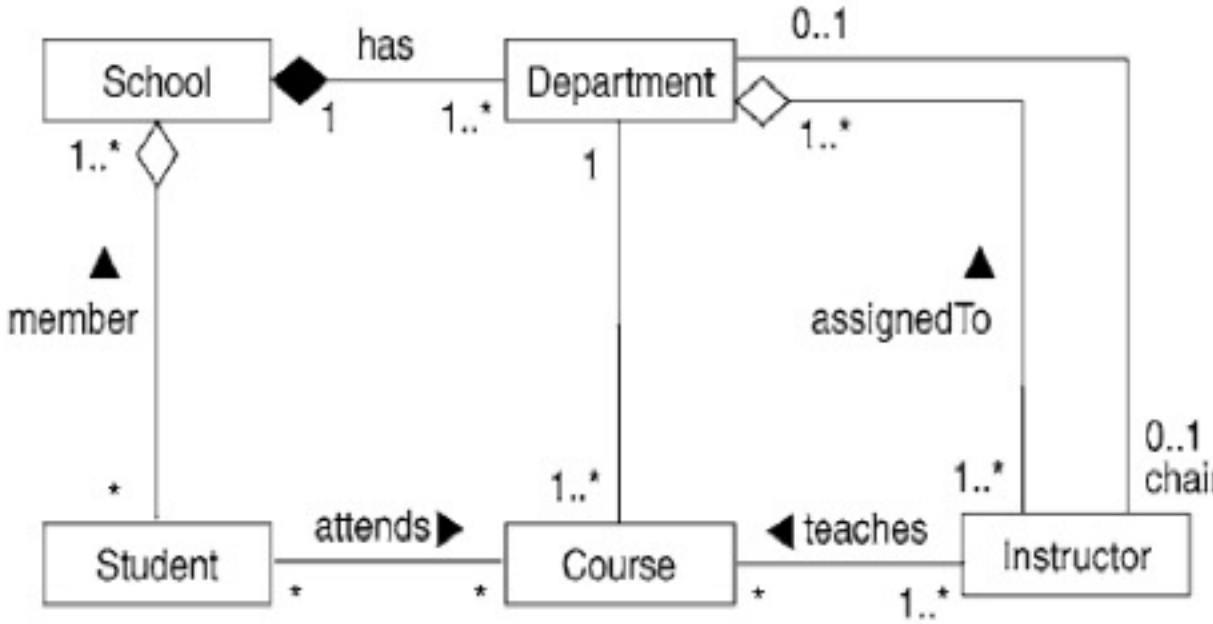
- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

③模型化结构关系

第一步：标识关联

若对于每一个类，需要导航到另一个类的对象，那么就要在这2个类之间给出一个关联。

——这是关联的数据驱动观点。



例如：
要了解Student所要参与的课程，因此就应在Student和Course之间给出一个关联，用于描述学生参与的课程；

. . . . . .

UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

若对于每一个类的对象需要与另一个类的对象进行交互，并且后一个对象不作为前一个对象的局部变量或操作参数，那么就要在这2个类之间给出一个关联。

——这是关联的行为驱动观点。

UML基本关系的一般用法

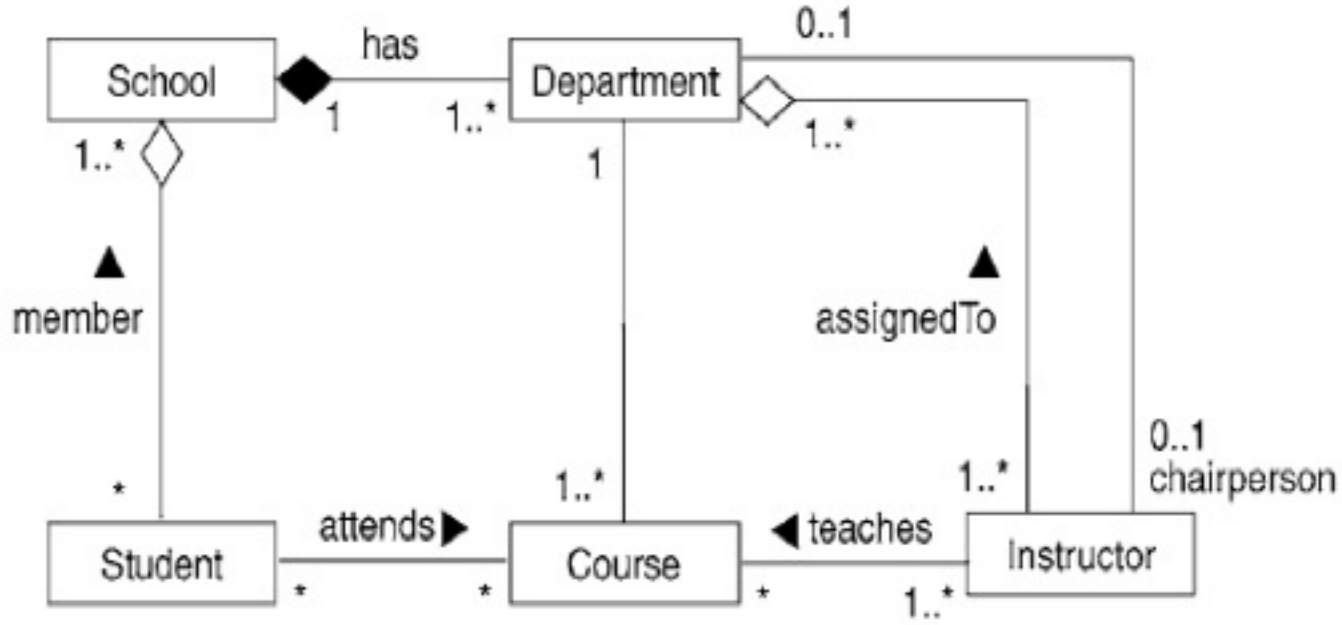
- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

第二步：对于标识的每一个关联，添加语义描述

例如，就下图而言，给出关联的多重性：

——每门课程至少有一名教师，而一名教师可以教多门课程。

——每门课程是精确地属于一个系的。

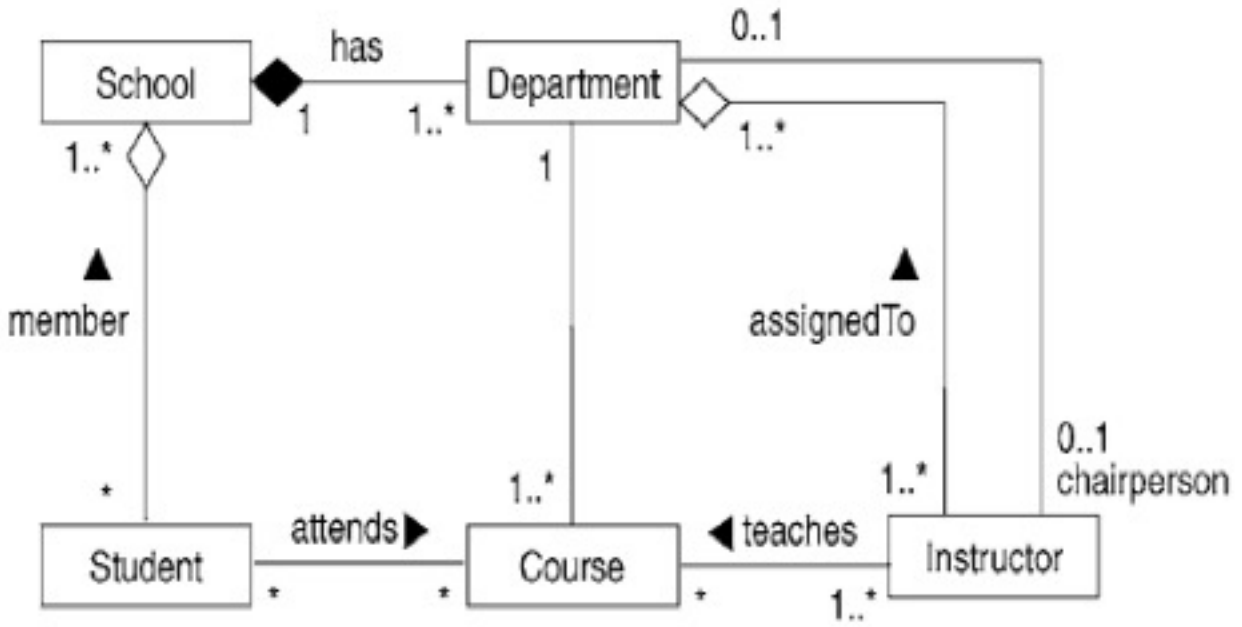


UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

第三步：标识“整体／部分”

如果关联中的一个类与另一端的类相比，前者在结构上或组织上是一个整体，而后者似乎是它们的一部分，那么就要把它们标识为聚合，例如，见下图：



聚合：一所学校可以有0到多名学生，一个学生可以注册在一所或多所学校学习；

聚合：一所学校可以有一个或多个系，而每个系只能属于一所学校；

注意：在该例中，Department和Instructor之间有两个关联，其中：一个关联（聚合）说明可以指派一名教师到一个或多个系中工作，而一个系可以有一名或多名教师；另一关联表明一个系只能有一名教师作系主任，而某些教师不是系主任。

UML基本关系的一般用法

- 模型化简单依赖
- 模型化单继承
- 模型化结构关系

基本策略

在用UML对关系建模时，要遵循以下策略：

- 仅当要建模的关系不是结构关系时，才使用依赖。

  这条策略意味着什么？

- 仅当关系是“is-a-kind-of”关系时，才使用泛化。

  聚合可否替代多继承？

- 一般不要引入循环的泛化关系。
- 应保持泛化关系的平衡：继承的层次不要多深，不要过宽（如果出现这种情况，就要寻找可能的中间抽象类）。