

类图

主讲：华俊昌

目录

CONTENT

01 类图的基本概念

02 类图中的基本符号

03 类之间的关系

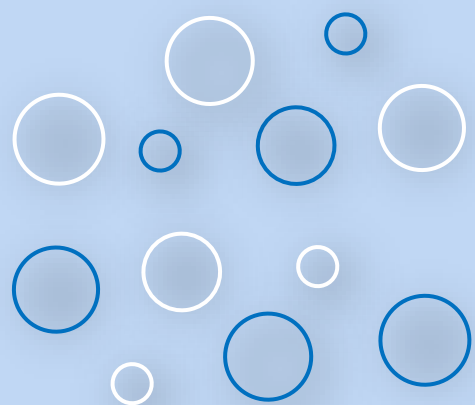
04 StarUML创建类图

05 课堂练习

01

Part one

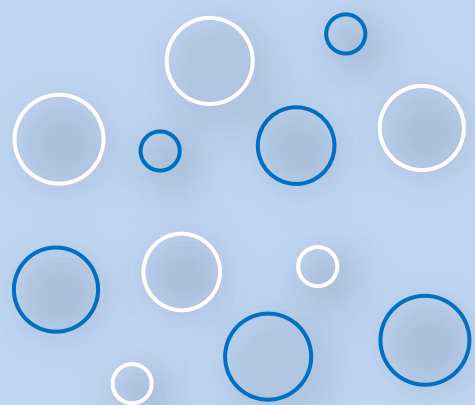
类图的基本概念



什么是类?



- 面向对象设计的基础就是使用类。类是用来代表现实事物或者功能的构造块。
- 类是具有相同属性、操作、关系和语义的对象的描述。
- 类的组成
 - 属性
 - 操作



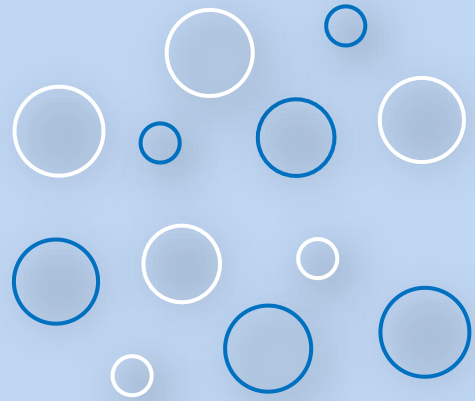
什么是类图?



类图的定义



类图是由若干类关联在一起，反映系统或者子系统组成结构的静态图。类图的建模贯穿工程的分析和设计阶段的始终，通常从商务伙伴能够理解的类开始建模，最终往往成为只有开发小组才能够完全理解的类。



类图的组成?



类图的组成

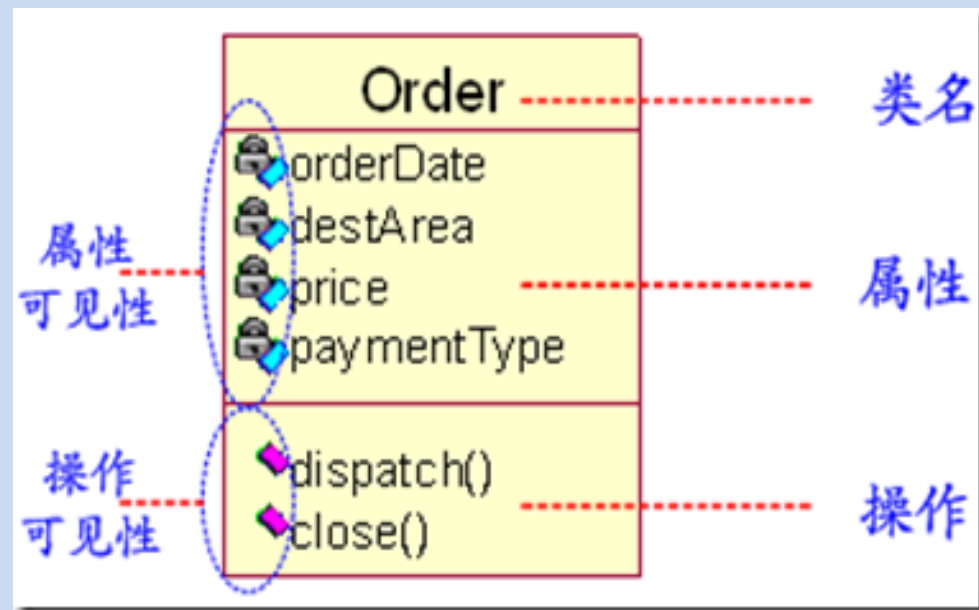


类图由如下元素组成：

- 类(Class)：是具有共同结构特征、行为特征、联系和语义的对象集合的抽象形式。

其中类包括包括：

- ✓ 名称（类名）：每个类都有一个唯一的名称。
 - ✓ 属性：是已被命名的类的特性，它描述该类实例中包含的信息
 - ✓ 操作（方法）：是类所提供的服务，它可以由类的任何对象请求以影响其行为
- 属性名和操作名首字母通常为小写。
- 关联（Association）：它表示类与类之间的关系。



02

Part two

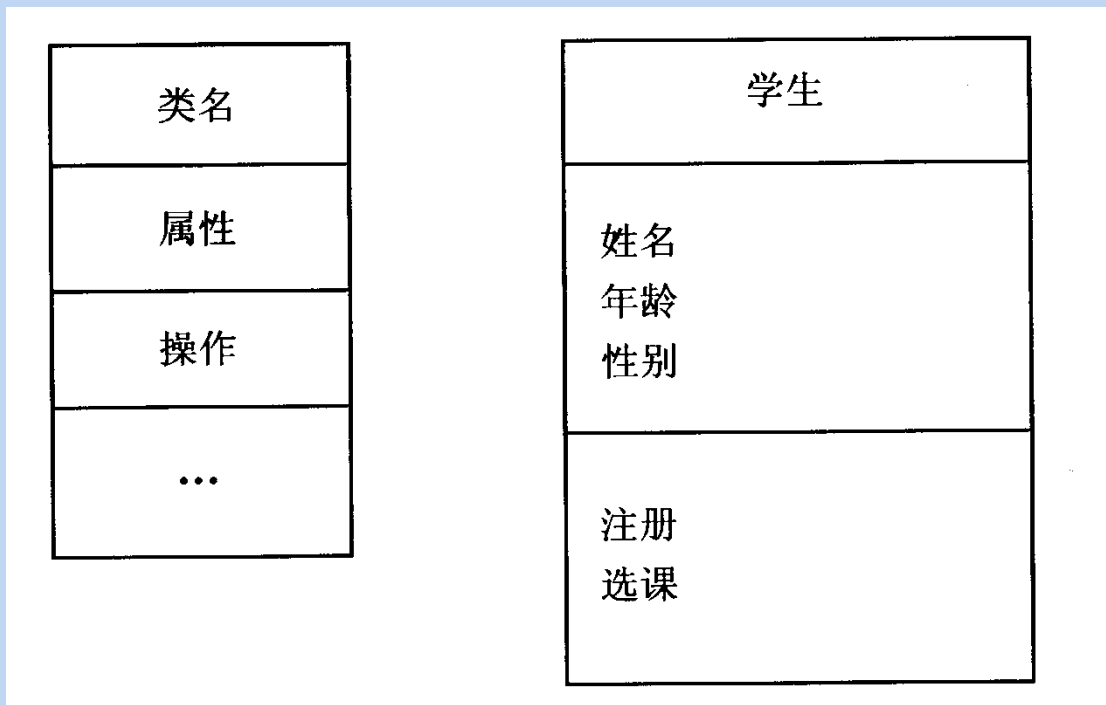
类图中的基本符号



类



类 (Class) 在UML中通常以实线矩形框表示，矩形框中含有若干分隔框，分别包含类的名字、属性、操作、约束以及其他成分等，如下图所示。

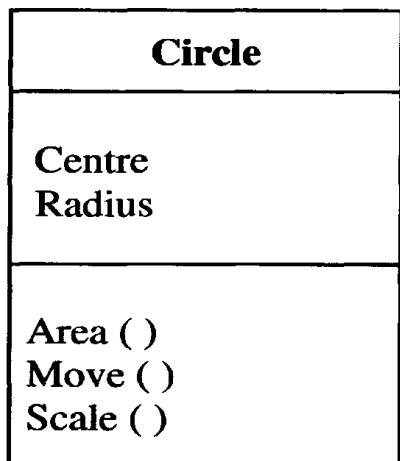


类的图形表示和示例

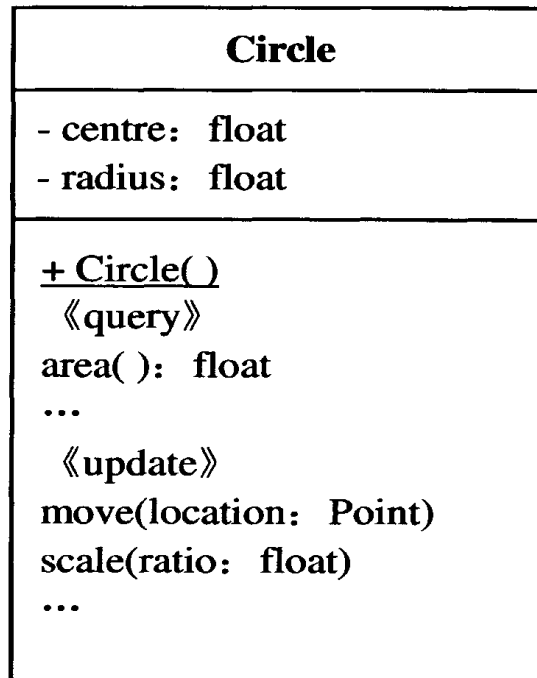
类图中，根据建模的不同景象，类图标中不一定列出全部的内容。在如在建立分析模型或设计模型时，甚至可以只列出类名，在图中着重表达的是类与类之间的联系；在建立实现模型时，则应当在类图标中详细给出类的属性和方法等细节。



(a)



(b)

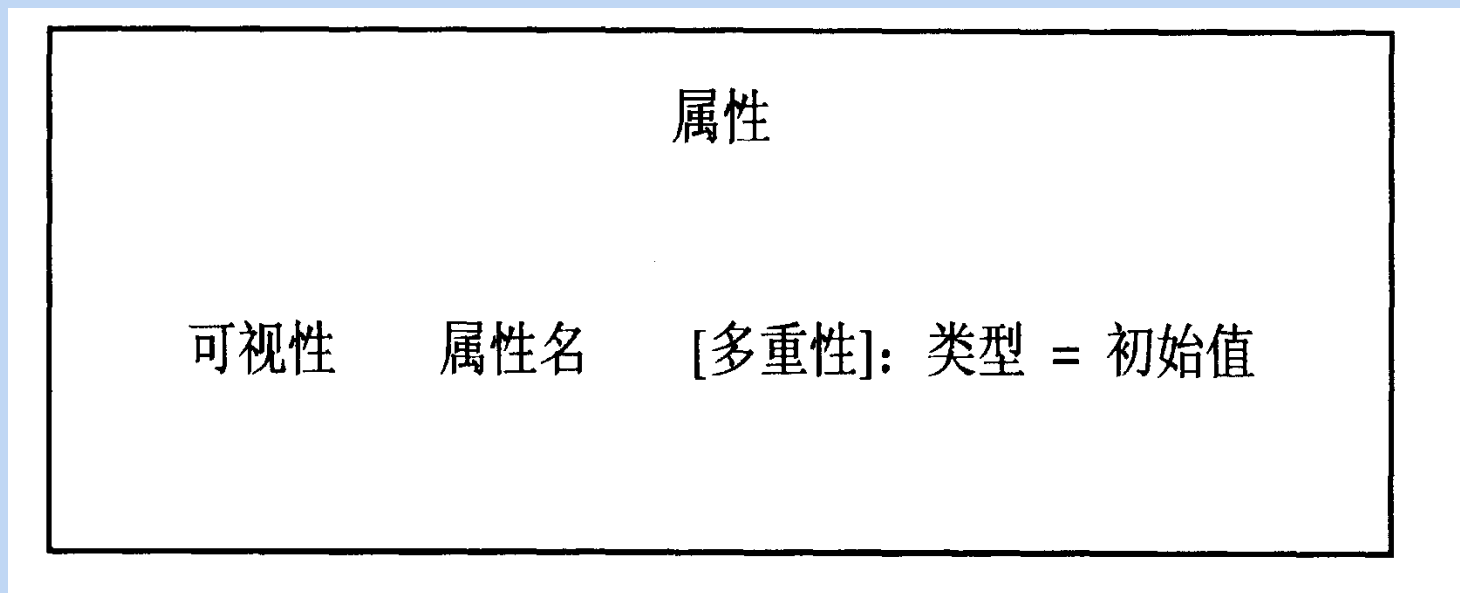


(c)



属性

属性 (Attribute) 在UML类图标矩形框中用文字串说明，如下图所示。





■ 可视性 (Visibility) 标记表示:

+ 公共

保护

- 私用

➤ 可视性也可以用以下关键字表示: public (公共)、protected (保护)、private (私用)。

➤ 若可视性标记为 “+” 或 “public”，则为公共属性，可以被外部对象访问。

➤ 若可视性标记为 “#” 或 “protected”，则为保护属性，可以被本类或子类的对象访问。

➤ 若可视性标记为 “-” 或 “private”，则为私用属性，不可以被外部对象访问，只能为本类的对象使用。

➤ 可视性可以缺省，表示该属性不可视。



Student
+Name[1] : String = NewStudent -Password[1] : String +GradeLevel[1] : Integer = 1 #Grades[0..*] : Integer + /GPA[1] : Double = Null

Student类



■ 属性类型表示:

冒号 “:” 后跟属性值的数据类型。数据类型的表示依赖于实现语言，如有的程序设计语言规定浮点数用保留字 “Float” 表示，有的则规定用保留字 “Real” 表示。

Student
+Name[1] : String = NewStudent
-Password[1] : String
+GradeLevel[1] : Integer = 1
#Grades[0..*] : Integer
+GPA[1] : Double = Null



- 数据类型可以是任何用户需要的内容，包括：
 - 来自程序设计语言如 Visual Basic、C++、C# 和 Java 的任何标准数据类型。
 - 一个已经定义的类。
 - 接口定义语言（Interface Definition Language, IDL）中的数据类型列表中的数据类型。
 - 读者在自己的系统建模中能够使用的其他类型。



属性初始值设置：

- 可以通过在属性名称和数据类型之后添加等于号（=）来为属性指定默认值，如下图所示。

Student
+Name : String = NewStudent
-Password : String
+GradeLevel : Integer = 1



属性多重性:

- 多重性为可选项，它表达该类的每个实例的属性值的个数。可以像应用于类之间的关系中那样把多重性应用于属性。例如，Student类具有属性Grades。不希望该属性只包含单个值，而是希望它包含该学生的所有成绩，可以是任意多个。

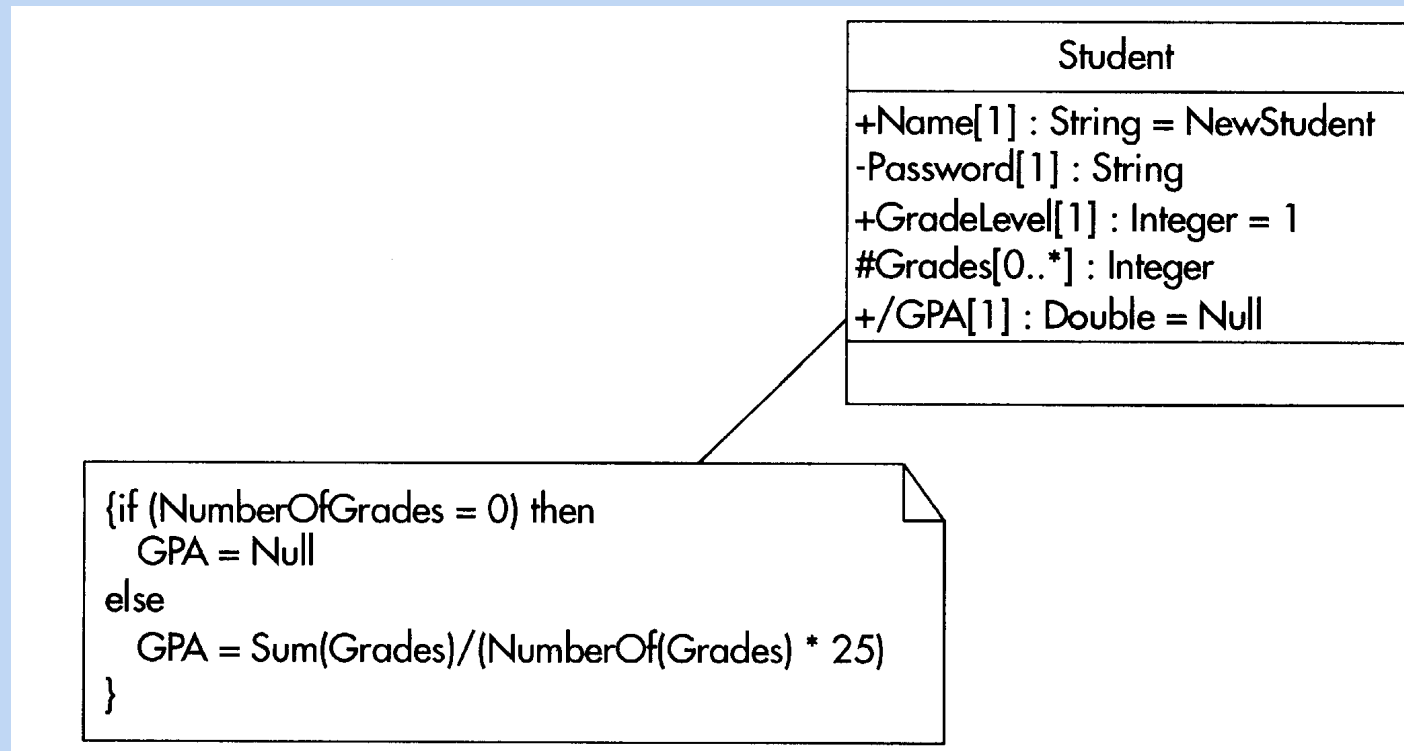
Student
+Name[1] : String = NewStudent
-Password[1] : String
+GradeLevel[1] : Integer = 1
+Grades[0..*] : Integer

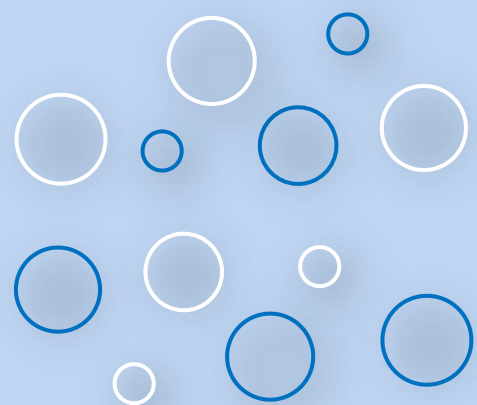
<u>CurrentStudent : Student</u>
Name : String = Zachary
Password : String = zachrocks
GradeLevel : Integer = 1
Grades : Integer = {90, 95, 87, 45, 100, 99}



派生的属性：

- 另一种可以为属性提供的信息是派生值，它可以使用数学函数、字符串函数或者将要在应用程序中实现的其他商务逻辑。要想指出一个属性是派生的，需要在属性名之前添加一个前斜线（/），并且要附加一个注释，其中包含了派生属性值的指令，如下图所示。

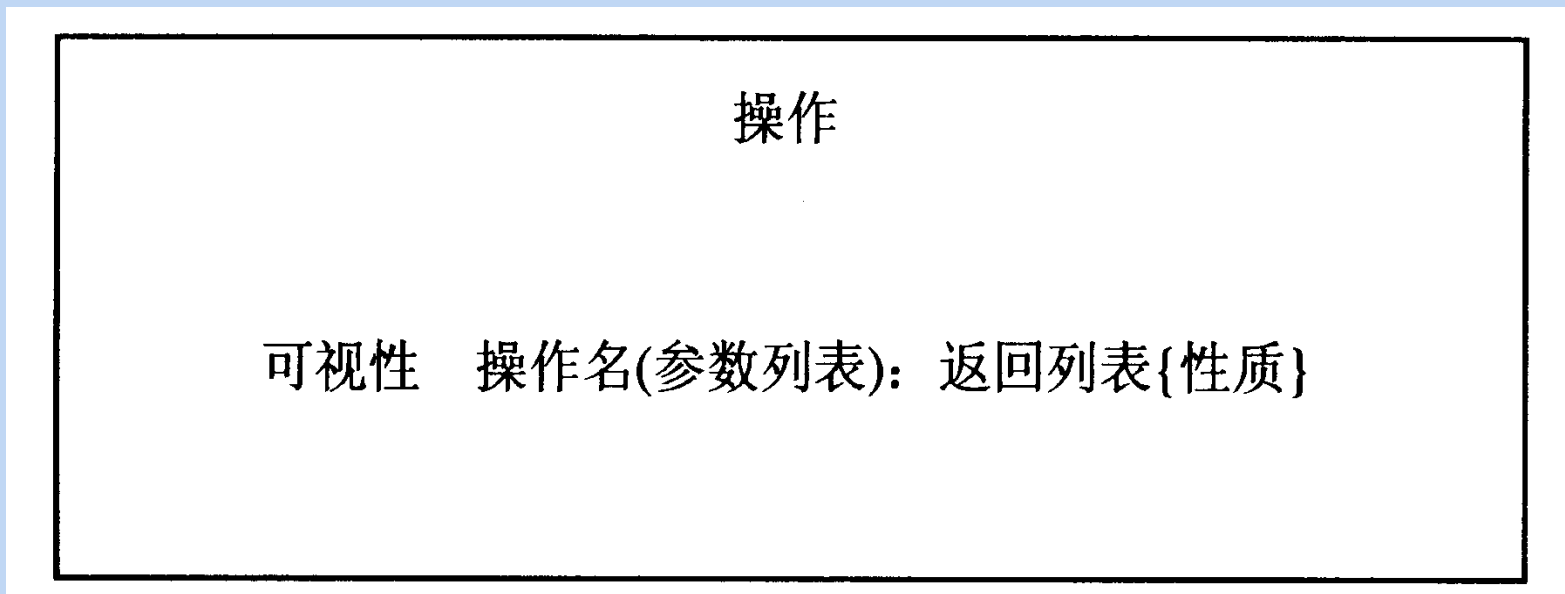




操作



操作 (Operation) 表示类能够提供的功能服务。它在UML类矩形框中用文字串说明，如下图所示。





操作名指示类可提供的功能服务，它后跟圆括号中的参数列表是可选项，即一个操作可以有参数，也可以没有参数。

参数列表由逗号分隔的操作的形式参数组成，**其格式为**：

参数名：类型=缺省值，...

返回列表是返回给调用者的单个变量值，它可以表示该操作程序运行的一个成功标志或者计算的值。

Student
+Name[1] : String = NewStudent -Password[1] : String +GradeLevel[1] : Integer = 1 #Grades[0..*] : Integer +GPA[1] : Double = Null
+AddGrade(in Grade : Integer = 100) +ClearGrades() +ChangePassword(in OldPassword : String, in NewPassword : String) : Boolean

03

Part three

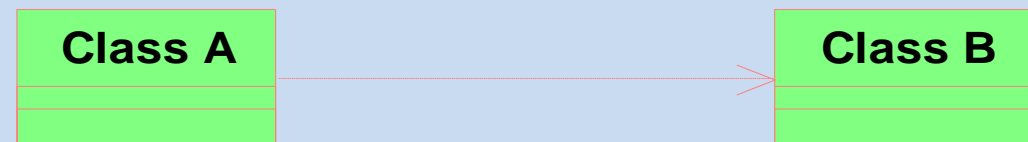
类之间的关系

类之间的关系

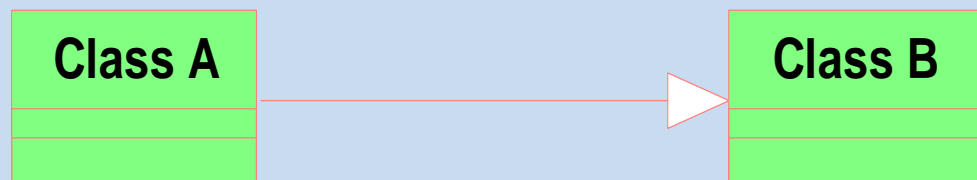
类之间可以建立四种关系：**关联**、**依赖**、**泛化**和**实现**。其标记如下图所示。



关联关系



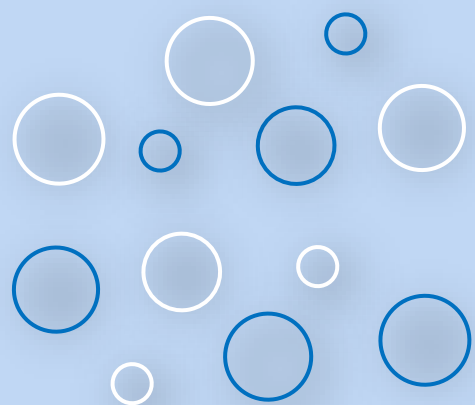
依赖关系



泛化关系



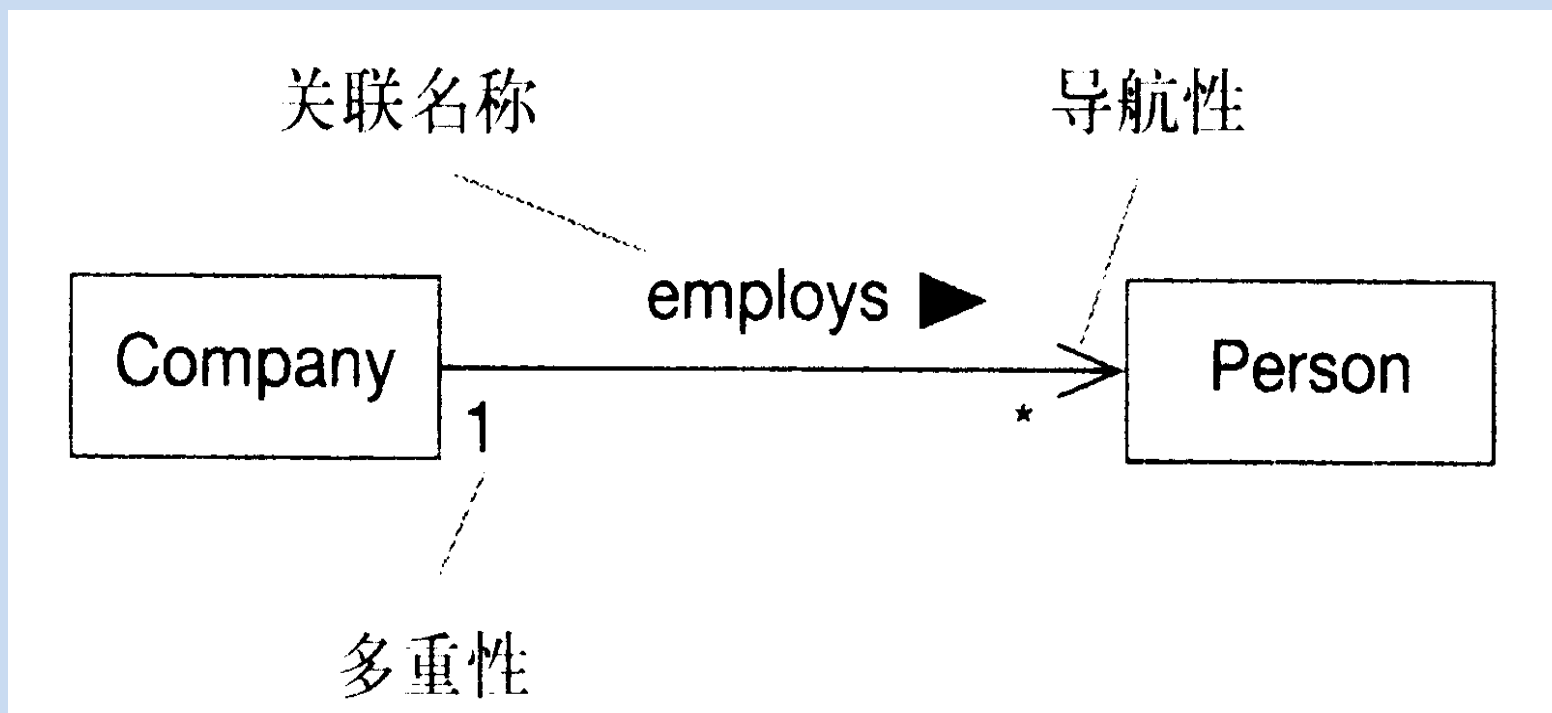
实现关系



关联关系

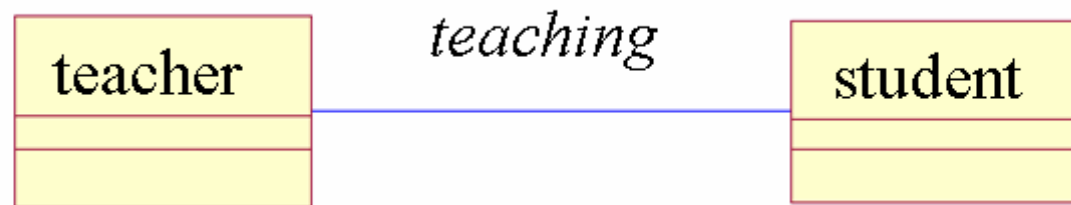
关联关系是指类之间的语义联系。关联可以具有如下特性：

- 关联名称
- 角色名称
- 多重性
- 导航性



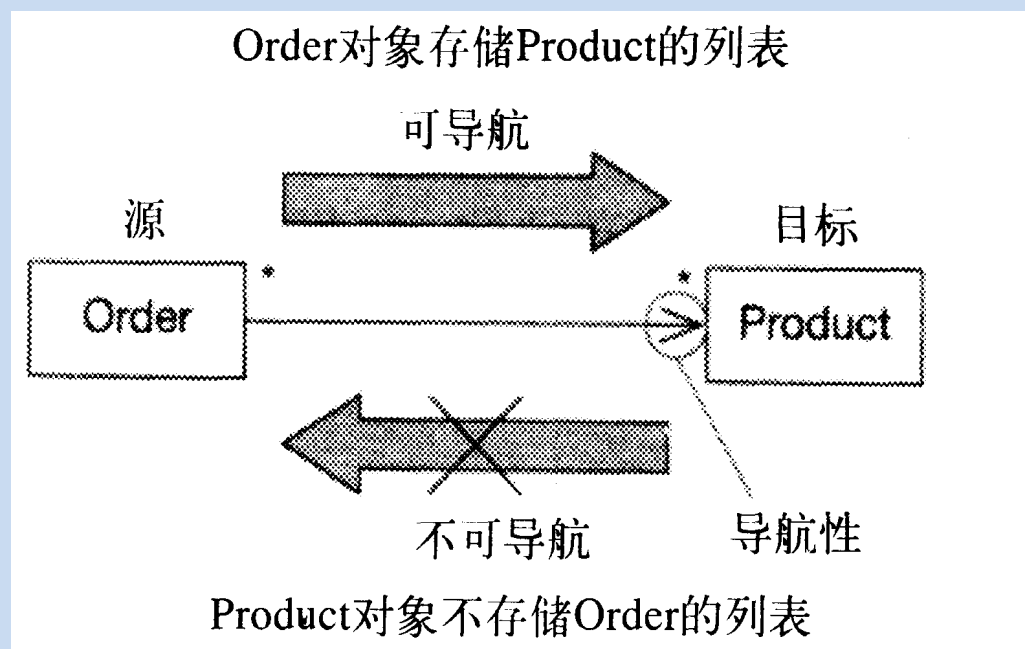


■ 关联名



■ 导航性 (navigability)

- 从一个类（对象）可以找到另外一个，反过来却不可。



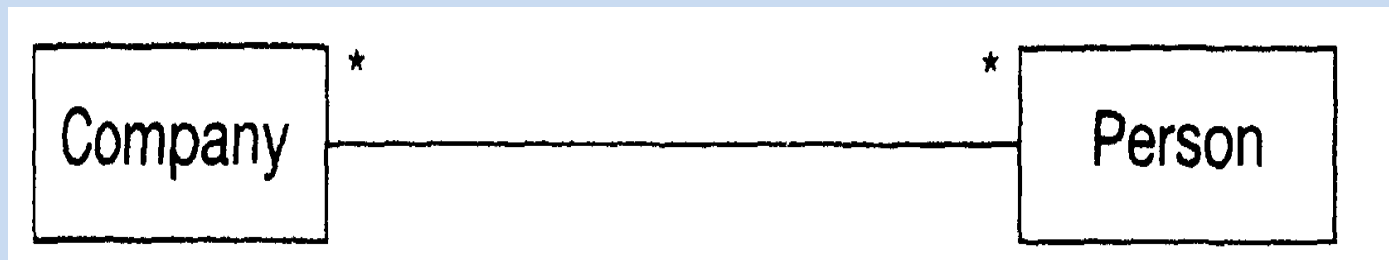


■ 关联角色



■ 关联的多重性 (multiplicity)

- 0 表没有实例的关联
- 0..1
- 0..n
- 1
- 1..n
- n



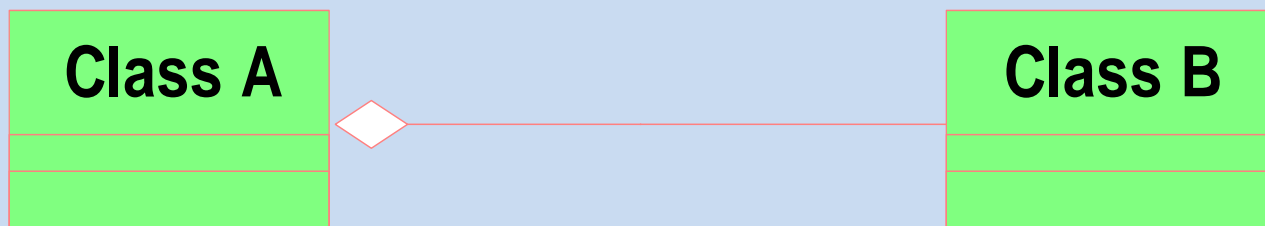
模型表达意义:

- 每个Person对象能够为很多Company对象工作。
- 每个Company对象能够雇佣很多Person对象



■ 聚合关系

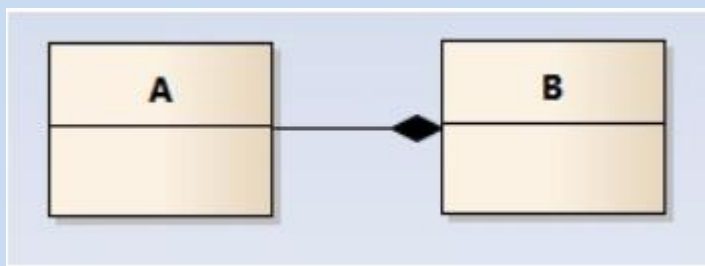
- 聚合是关联关系的一种特例，他体现的是整体与部分、拥有的关系，即 has-a 的关系，此时整体与部分之间是可分离的，他们可以具有各自的生命周期，部分可以属于多个整体对象，也可以为多个整体对象共享；比如计算机与CPU、公司与员工的关系等；即使整体不存在了，部分仍然存在；例如， 部门撤销了，人员不会消失，他们依然存在；



聚合关系

■ 组合关系

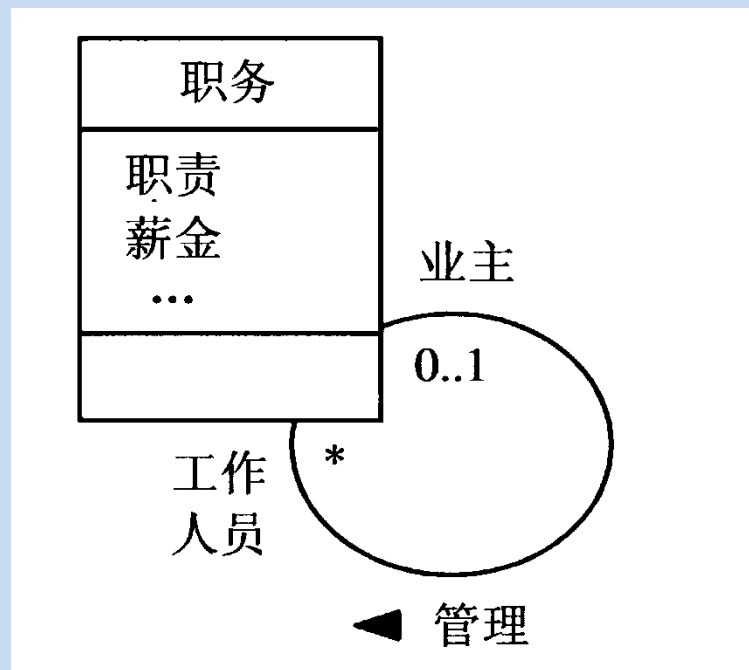
- 组合也是关联关系的一种特例，他体现的是一种contains-a的关系，这种关系比聚合更强，也称为强聚合；他同样体现整体与部分间的关系，但此时整体与部分是不可分的，整体的生命周期结束也就意味着部分的生命周期结束；比如你和你的大脑；表现在代码层面，和关联关系是一致的，只能从语义级别来区分；
- 组合关系用一条带实心菱形箭头直线表示，如下图表示A组成B，或者B由A组成；



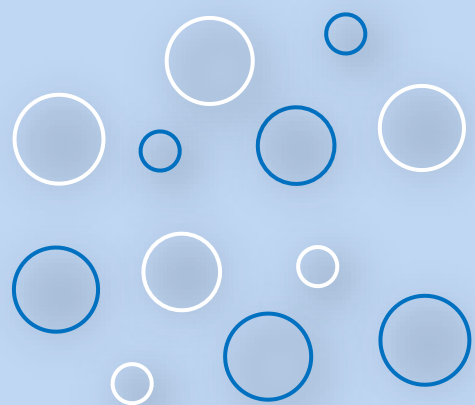


■ 自反关联

- 类具有到自身的关联，称为自反关联。



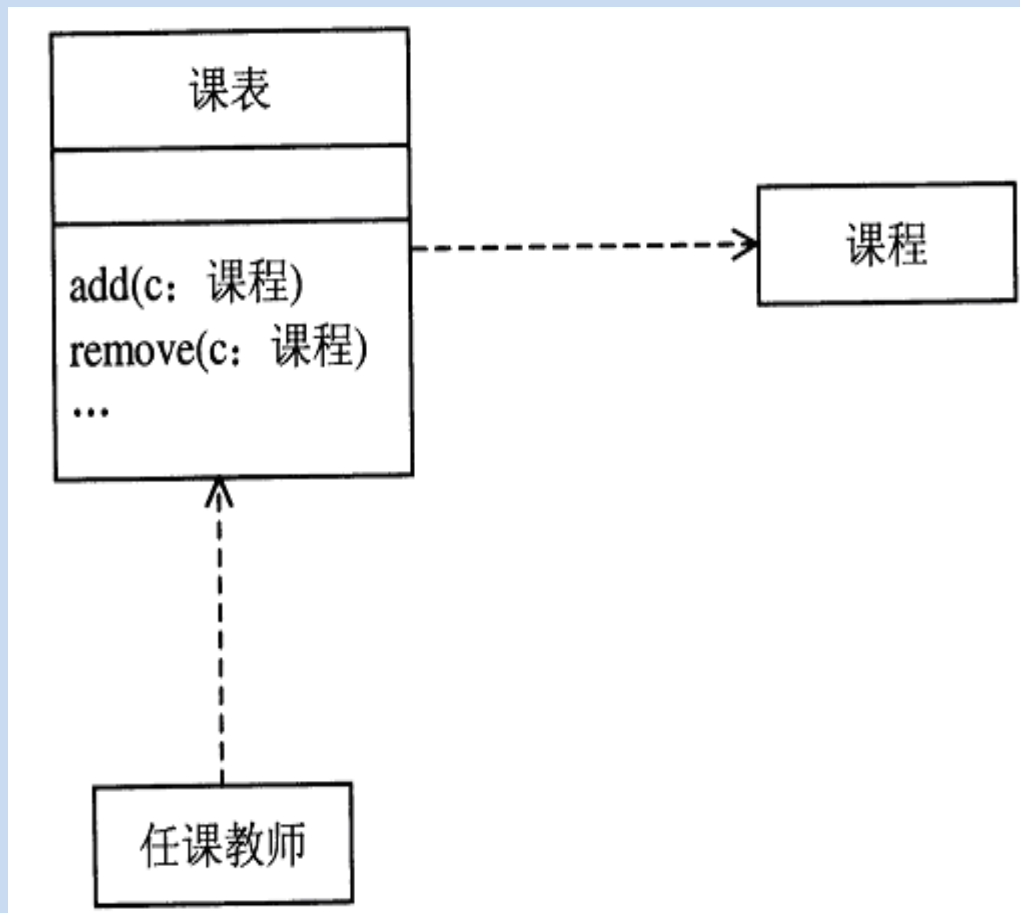
类的自反关联

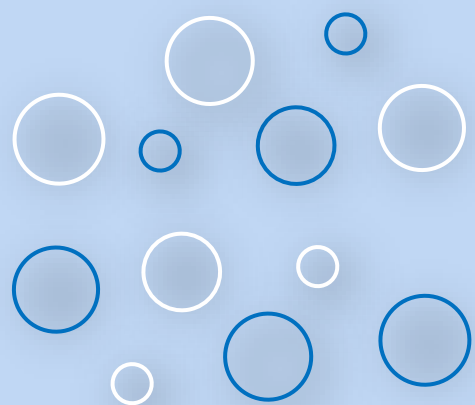


依赖关系



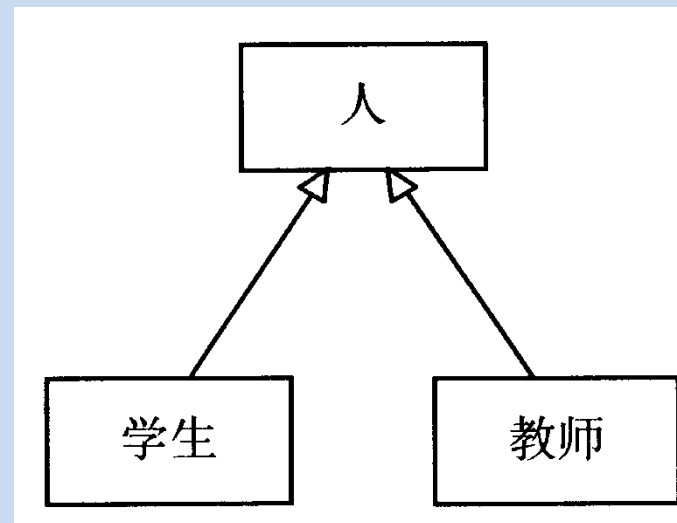
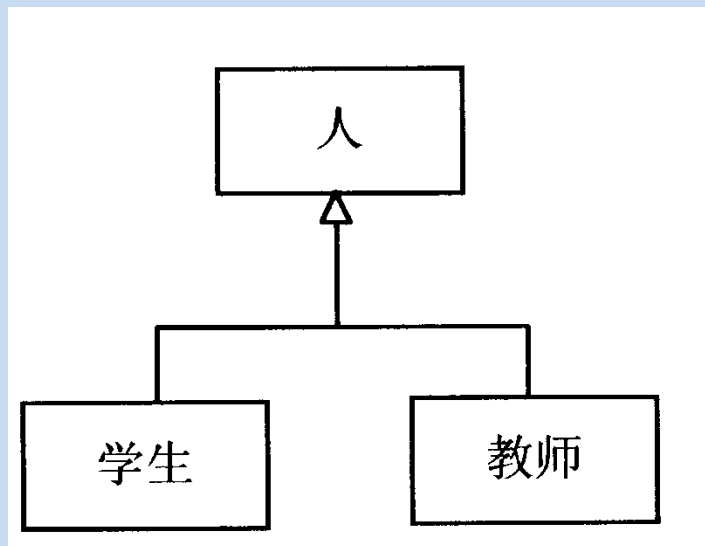
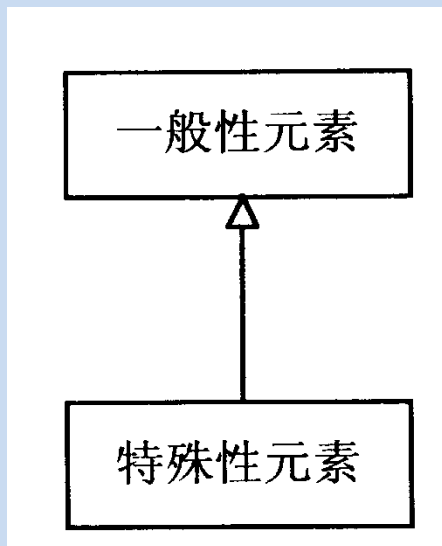
依赖关系是指一个类的元素使用了另一个类。依赖关系描述类之间的引用关系。

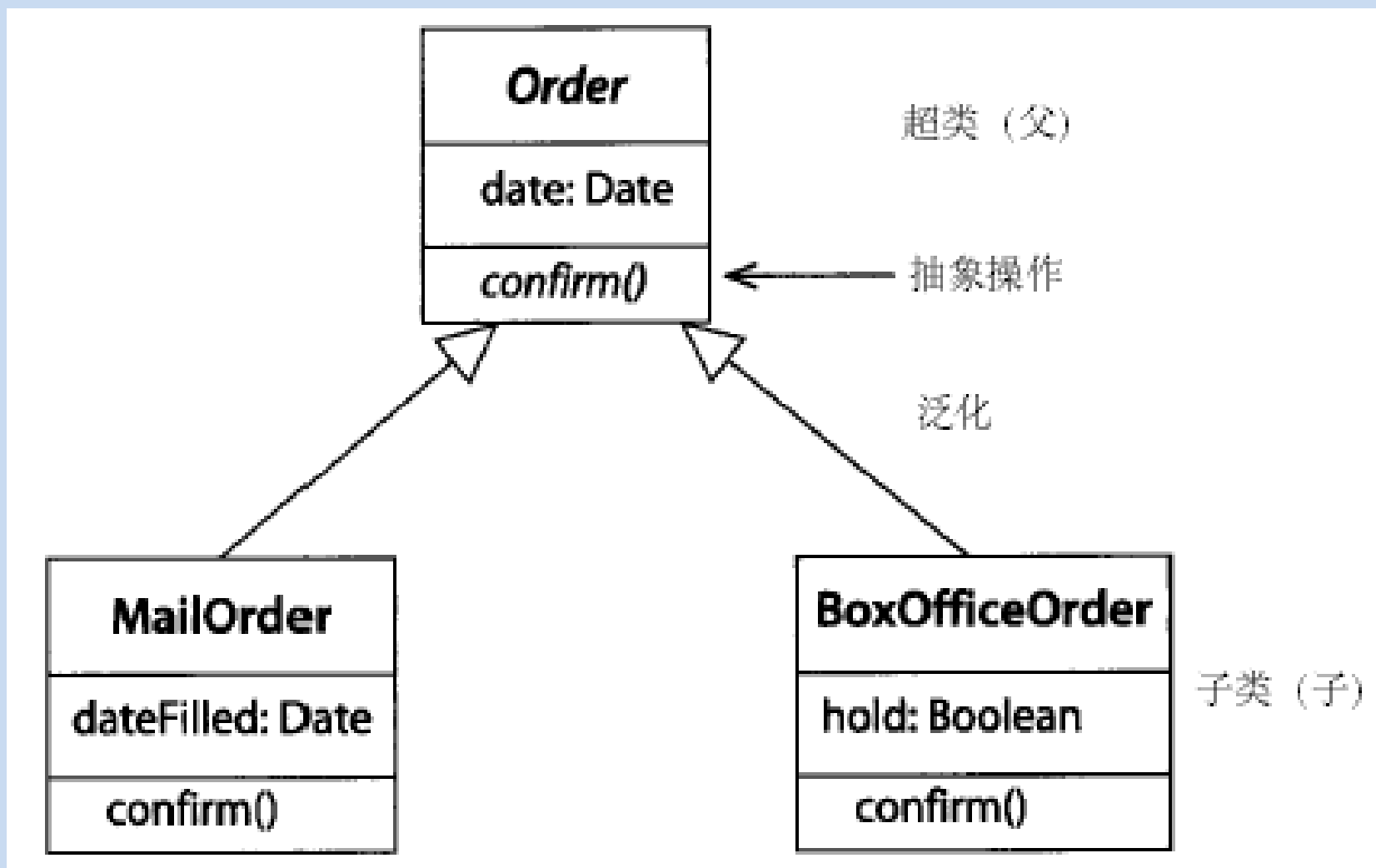


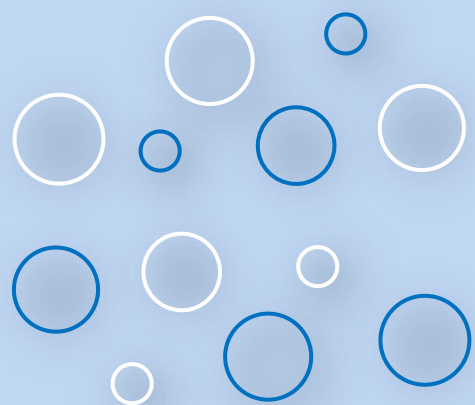


泛化关系

泛化关系是描述类之间的继承关系。利用泛化来表达类之间的相似性。



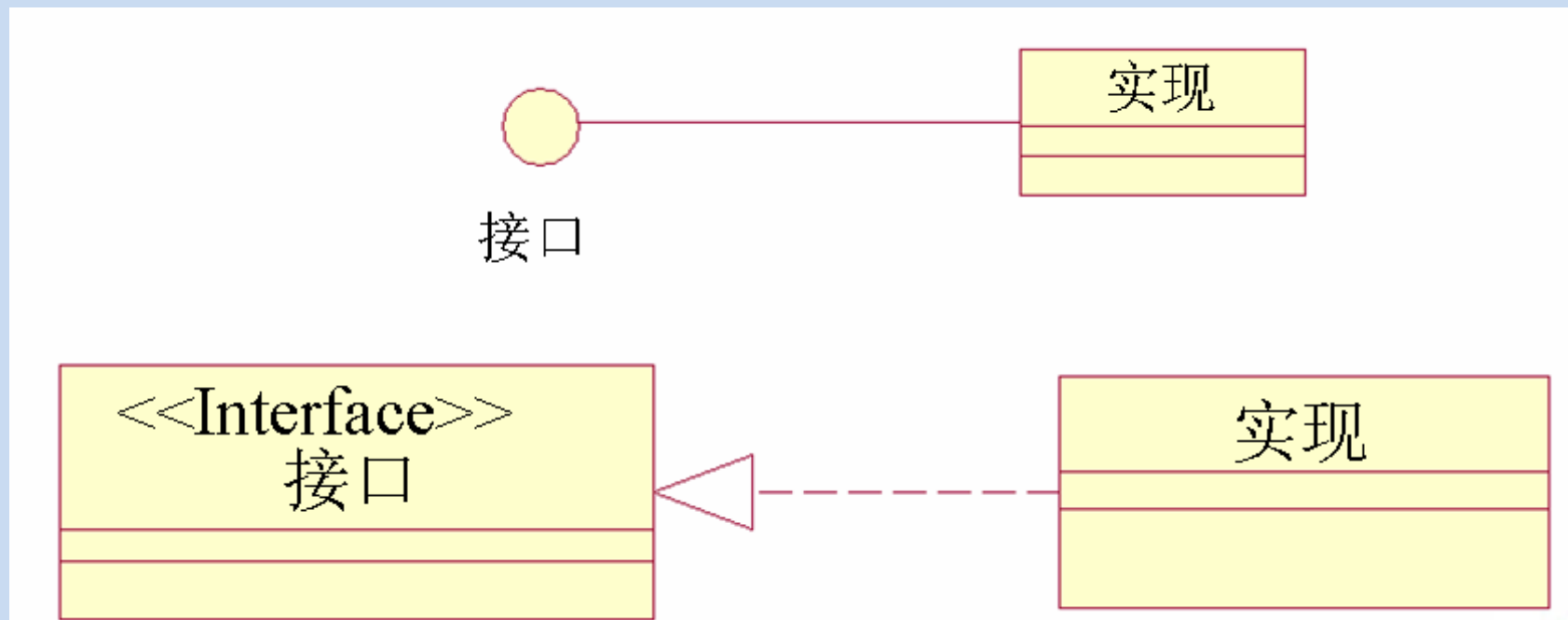




实现关系



- 一个元素完成另外一个元素的操作功能
- 如接口类及其实现；接口没有属性，只有声明的操作方法（对方法没有实现部分），而由实现类具体定义实现部分。



04

Part four

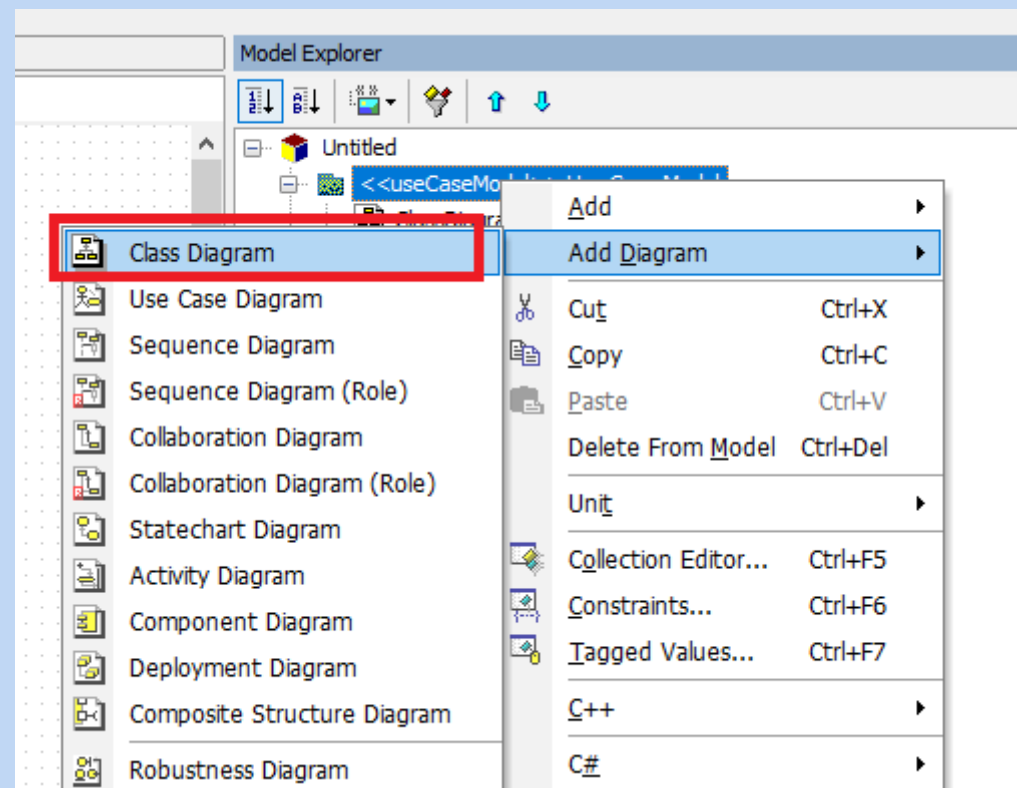
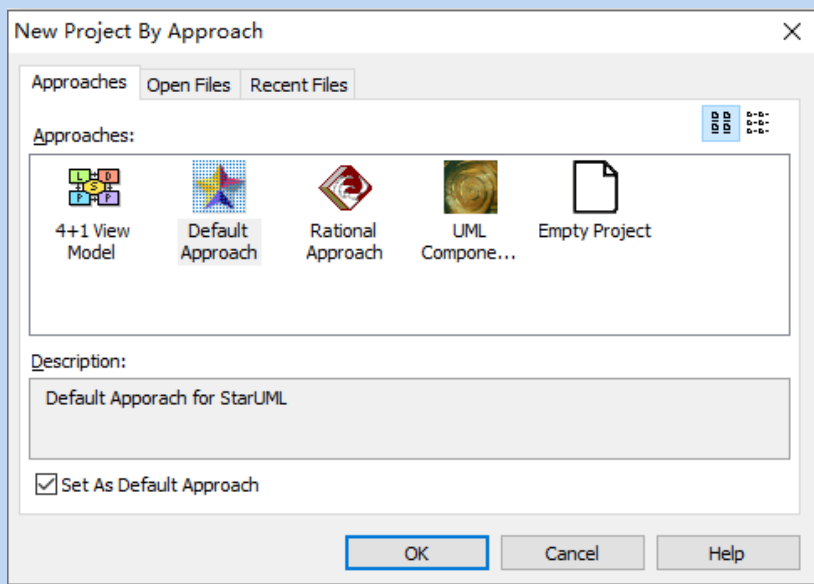
StarUML创建类图



StarUML是一种面向对象的统一建模语言的可视化建模工具，下面介绍如何使用StarUML绘制类图。

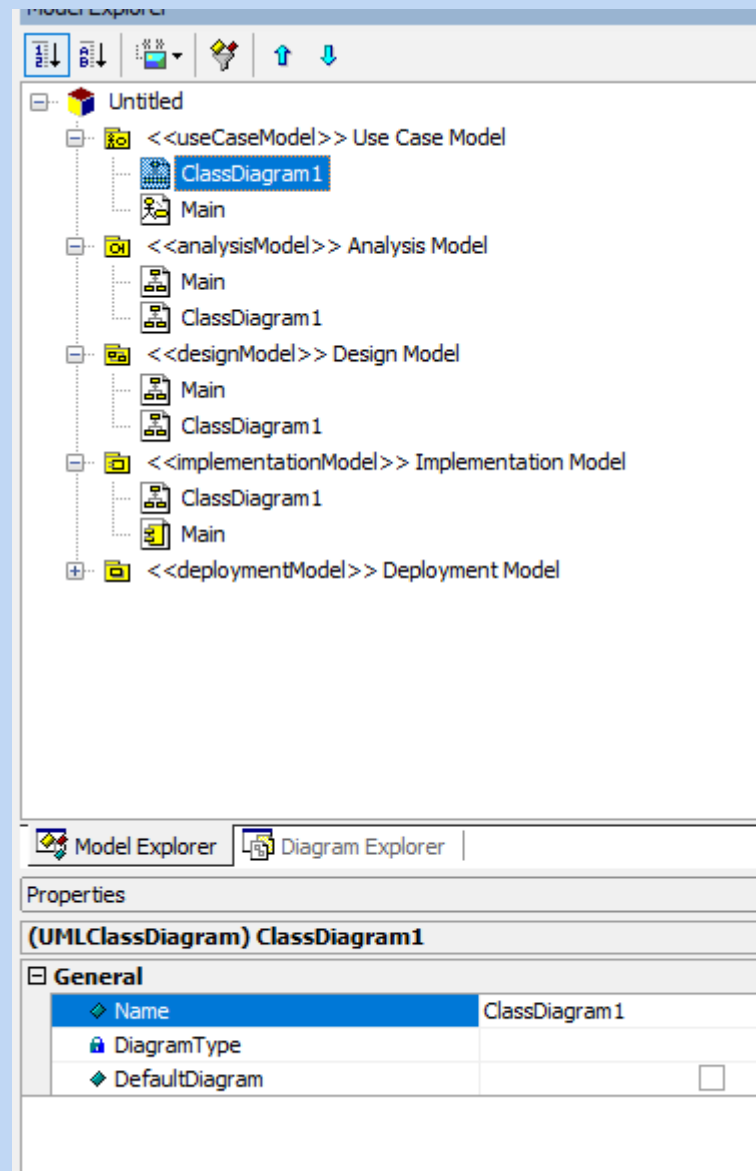
- 1 创建类
- 2 创建类之间的关联

- 打开StarUML后，创建一个DefaultApproach，在Use Case Model树型结构下可以使用默认名为Main的用例图，也可以在Use Case Model树节点右击，在弹出的快捷菜单中选择Add Diagram|Class Diagram命令建立新的类图。



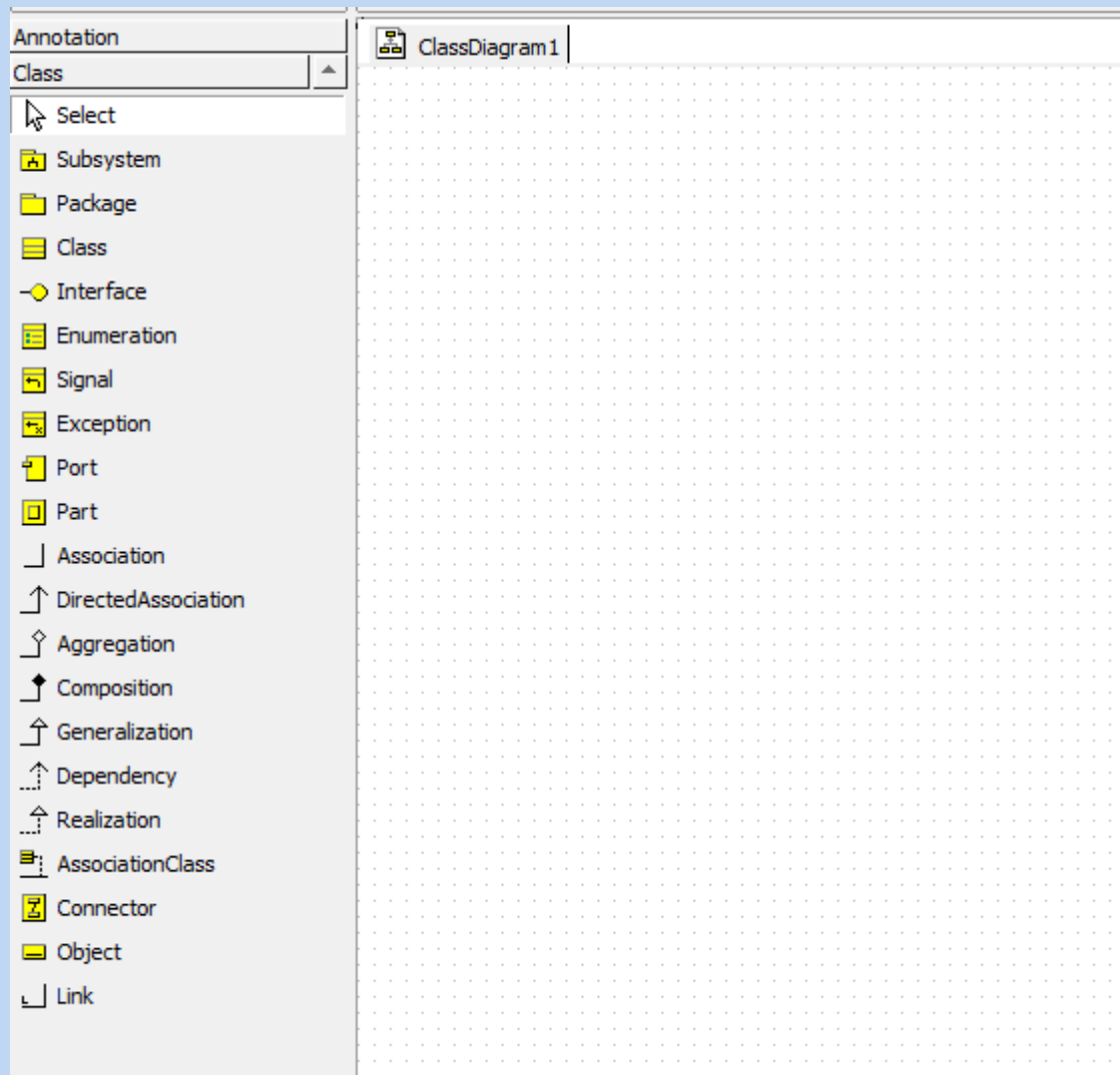


- 创建新的用例图，在Use Case Model树型下就多了一个名为ClassDiagram1的图标，这个图标就是新建的类图图标。选中此图标，用快捷键F2为新创建的用例图重命名，或者下方Properties属性窗口中修改Name。



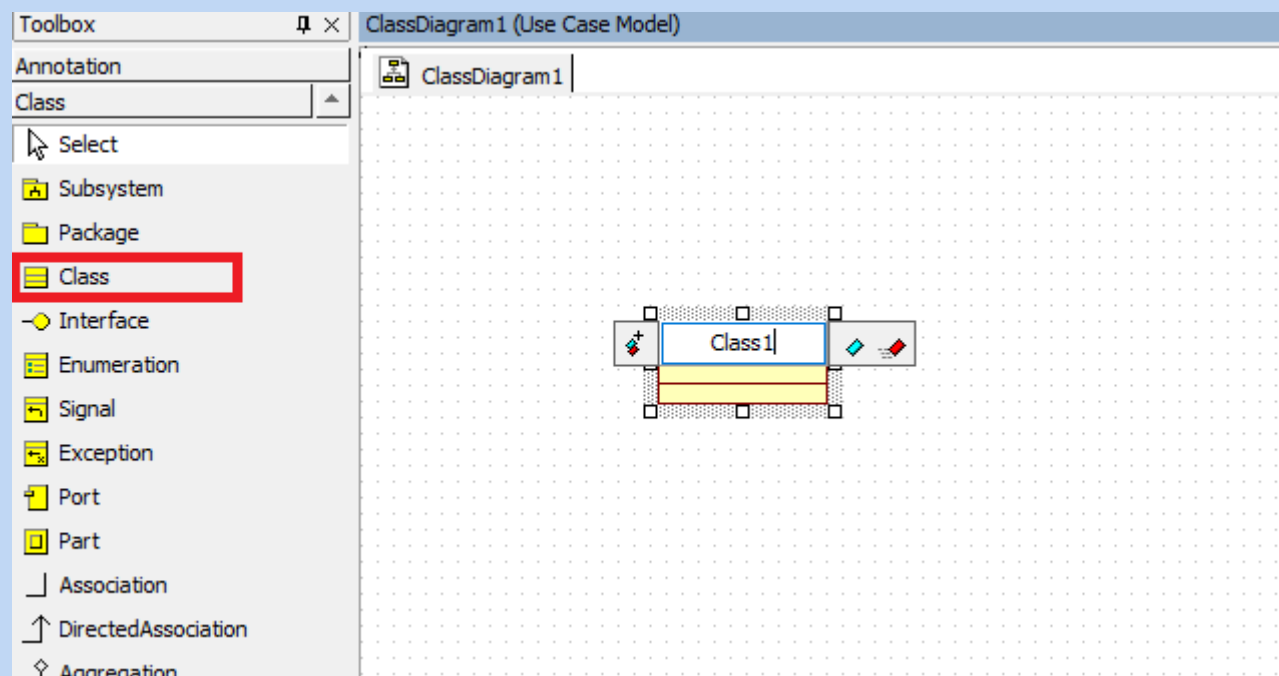


- 双击名为ClassDiagram1的类图图标，会出现类图图的编辑工具栏和编辑区。




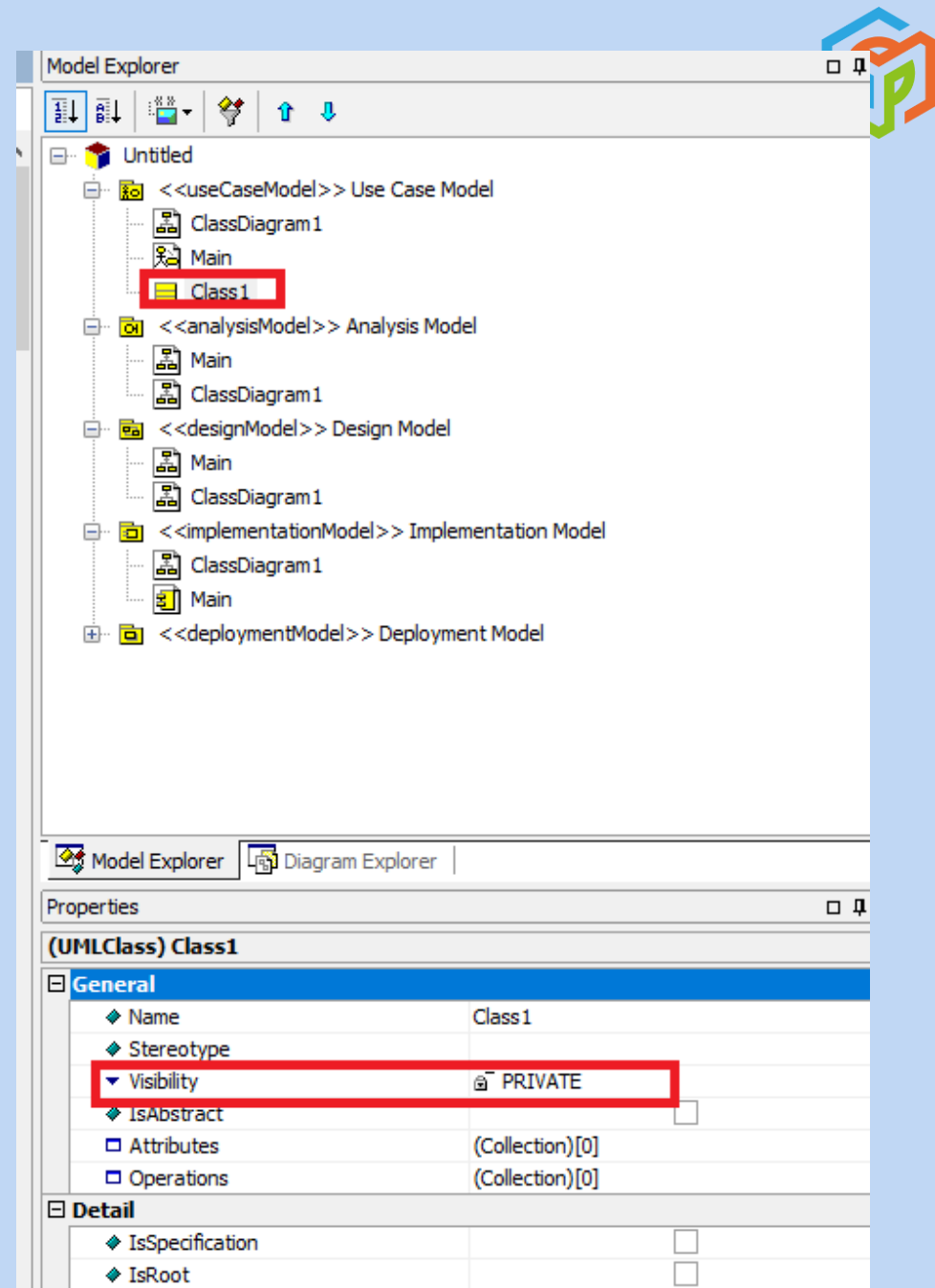
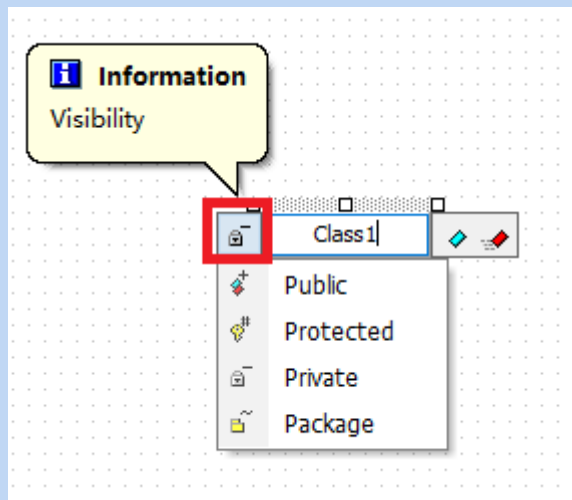


- 单击左侧工具栏上的Class, 然后在类图编辑区内单击。可修改类名



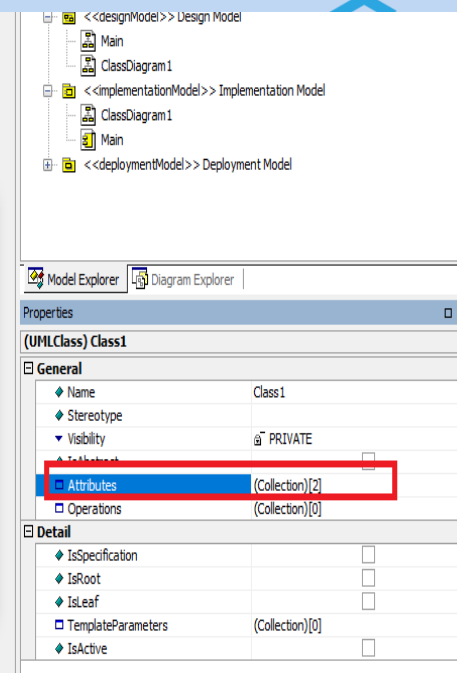
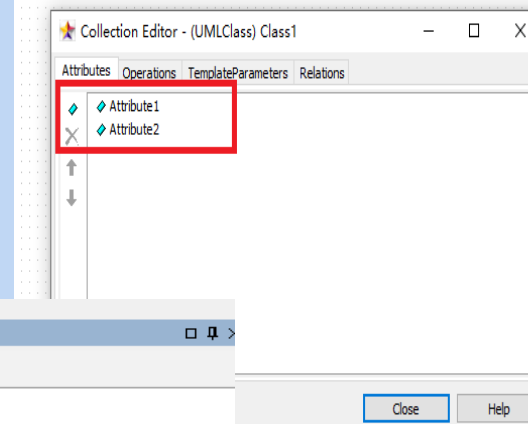
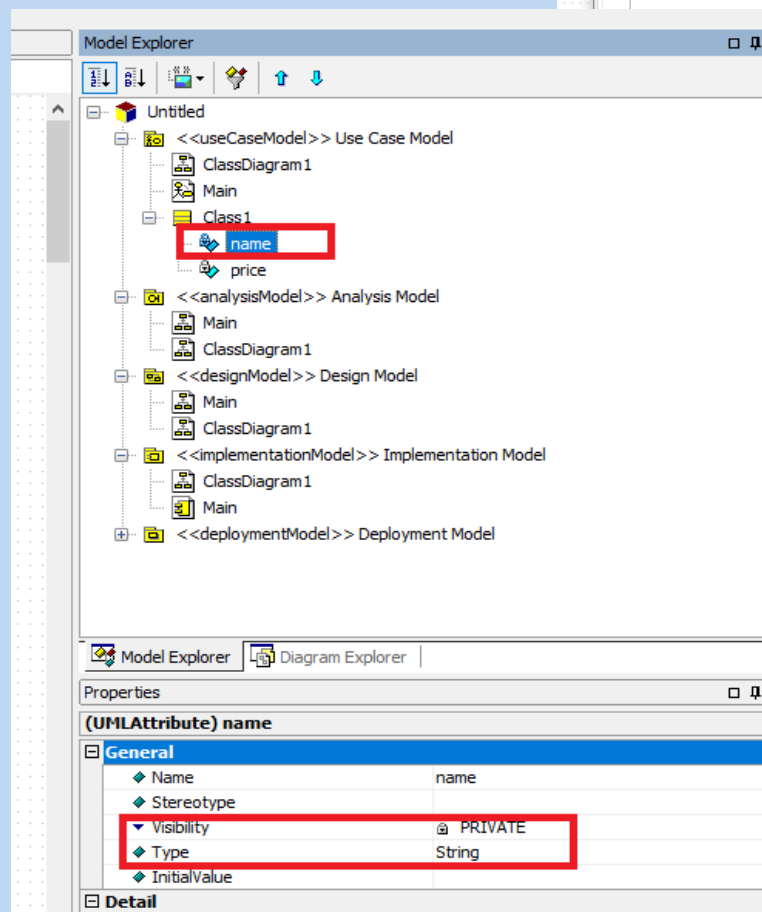
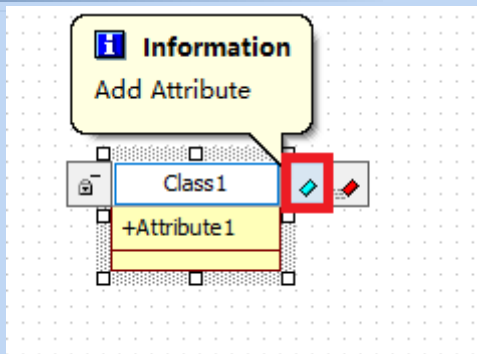
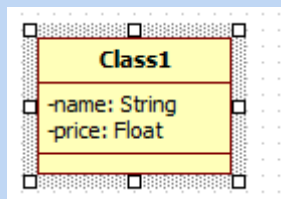
创建类1-设置可见域

- 双击Class，在左侧的  图标下拉选中可见域，或者在右侧模型树上Properties里修改Visibility属性



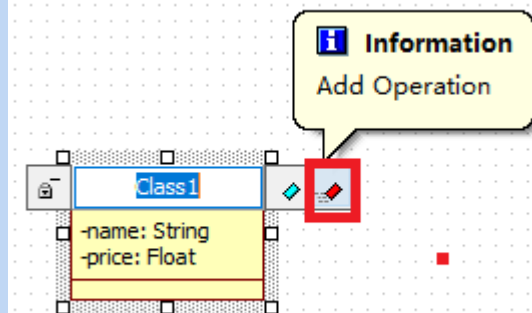
创建类1-新建属性

- 双击Class，在右侧的蓝色图标可以新增属性，或者右侧模型树的Properties的Attributes属性进行新增，修改，删除调整顺序等操作
- 设置属性的可见域
- 设置属性名称
- 设置属性类型，格式为可见域+属性名称：+属性类型

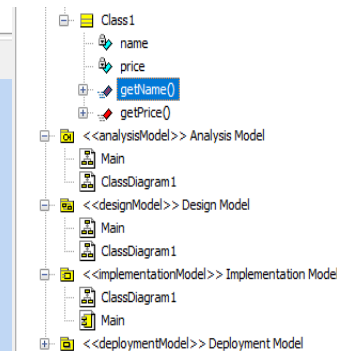
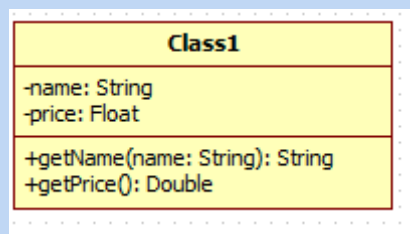


创建类1-新建方法

- 双击Class，在右侧的红色图标可以新增方法，或者右侧模型树的Properties的Operators属性进行新增，修改，删除调整顺序等操作
- 设置方法的可见域
- 设置方法名称
- 设置方法的参数和返回值类型。格式为可见域+方法名称(参数1名称:参数1类型, 参数2名称:参数2类型): +返回值类型



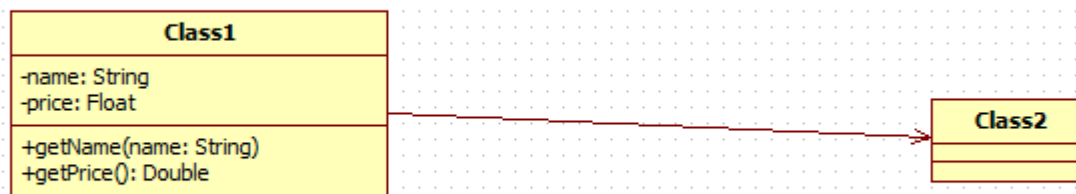
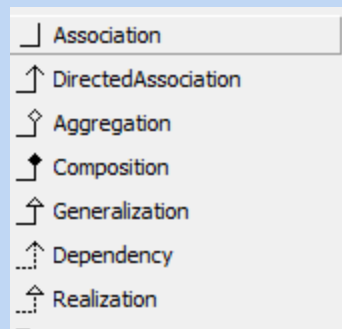
The screenshot shows the UML modeling software interface. The 'Collection Editor - (UMLClass) Class1' window is open, displaying a list of operations: 'Operation1()' and 'Operation2()'. The 'Properties' window is also open, showing the 'General' tab for '(UMLClass) Class1'. The 'Operations' property is highlighted, showing a collection of operations. The 'Model Explorer' and 'Diagram Explorer' panels are visible on the right side of the interface.



The 'Properties' window shows the 'General' tab for the '(UMLOperation) getName' method. The 'Visibility' property is set to 'PRIVATE'. The 'Parameters' property is set to '(Collection)[1]'. The 'Name' property is 'getName'.



- 选中左侧的几种关系，如（关联、聚合、组合、依赖、泛化、实现），选中Class1和Class2



05

Part five

课堂练习



画出类图

1.新建名为Account的账户类，该类包含的属性：accountNum(账户，String类型)、balance（余额，double类型），该类的构造函数Account(String accountNum, double balance)

该类包含的方法：访问器方法（所有属性的getter和setter方法），取款方法withdraw(double amount)和存款方法deposit(double amount)

2.创建Account类的一个子类CheckAccount代表可透支的账户,该CheckAccount中定义一个属性overdraft代表可透支限额。在CheckAccount类中重写withdraw方法



感谢您的观看
