UML建模基础
—类图

东软IT人才实训中心

Neusoft

第三章：类、关系、类图

目标：

本章旨在向学员简要介绍类、类关系 类图,通过本课的学习，学员应该掌握如下知识：

1）类的关系

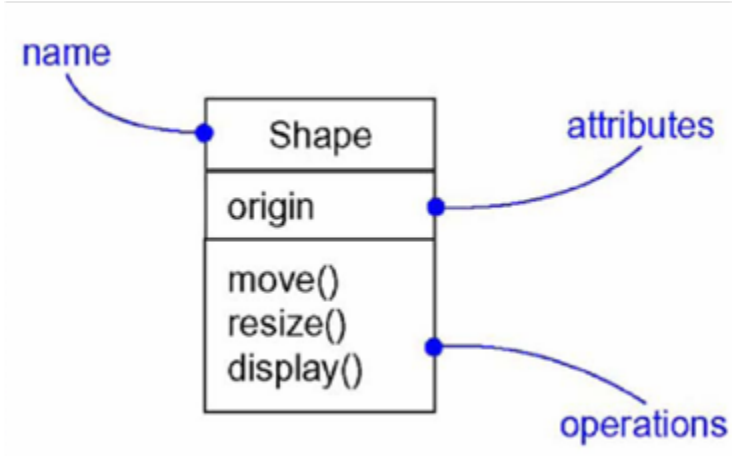
2）类图的画法

学时：2学时

教学方法：讲授ppt＋上机练习+案例分析

3.1 类

- 概念
  - 对一组具有相同属性、操作、关系和语义的对象的描述
  - 一个对象是一个类的实例
  - 包括：
    - 名称(name)
    - 属性(attributes)
    - 操作(operations)



3.1 类（续）

- 如何设计类
  - 对系统词汇建模
    - 分析人员根据需求陈述与用例描述中出现的名词和名称短语来提取实体对象。
    - 对于每一个抽象，识别一个职责集。清楚地定义每个类，职责要在所有类之间很好地均衡。
    - 提供为实现每个类的职责所需要的属性和操作

3.1 类（续）

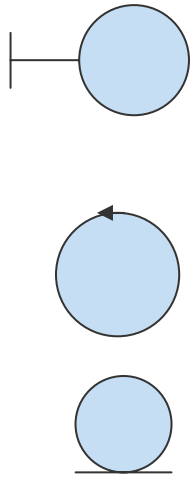
- 其他类概念
  - 通过构造型（Stereotype）扩展的类
    - 接口interface
    - 边界类boundary
    - 实体类entity
    - 控制类control
    - 其他自定义构造型
  - 抽象类

3.1 类（续）

- 属性
  - 静态属性
  - 派生属性
- 属性、操作的隐藏与显示

3.1 类（续）

- 分类
  - 边界类（Boundary class）
    - 接口与系统外部某些事物的媒介
  - 控制类（Control Class）
    - 负责协调用例的行为
  - 实体类（Entity Class）
    - 封装了数据以及数据相关的操作



3.2 边界类

- 边界类帮助系统接口与系统外部进行交互。边界对象将系统与其外部环境的变更（与其他系统的接口的变更、用户需求的变更等）分隔开，使这些变更不会对系统的其他部分造成影响
- 分类
  - 用户接口类
    - 帮助与用户进行通信的类，通过标准的I/O设备提供人机界面
  - 系统接口类
    - 帮助与其他系统进行通信的类，系统接口对象隐藏如何与外部接口通信的细节
  - 设备接口类或Timer
    - 提供对硬件设备的软件接口

3.3 控制类

- 本身不完成任何功能，负责协调其他类的工作
  - 控制类并不能处理用例需要执行的一切事务。相反，它协调其他用例实施此功能的对象的活动。控制类将工作委派给已被指定负责此项功能的对象。控制类通常被看成一个乐队的指挥，它指挥（控制）参与use case的其它对象的行为，通知对象什么时候执行以及执行什么。
  - 通常一个use case会对应一个（或多个）控制类，来控制use case的任务流程

3.4 实体类

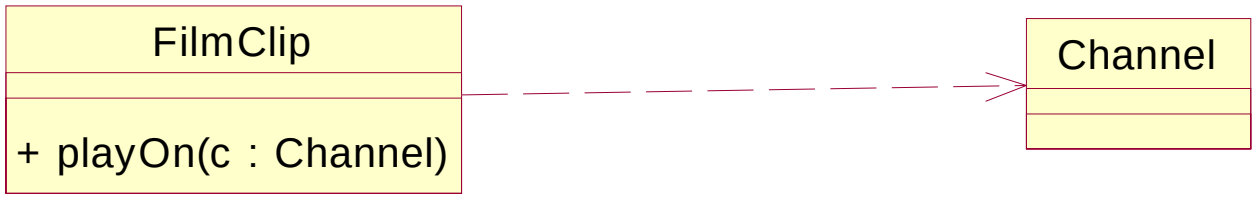
- 是系统中要永久保存的逻辑数据结构。
- 通常代表系统中信息的存储
- 对应于系统中的名词
  - 有时Actor也可作为实体类

3.5 类之间的关系

- 四种重要的关系
  - 依赖(dependency)
  - 关联(association)
    - 聚合(aggregation)
    - 组成(composite)
  - 泛化(generalization)
  - 实现(realization)

3.5 类之间的关系（续）

- Dependency（依赖）
  - 是一种使用关系，它说明一个事物规格说明的变化可能影响到使用它的另一个事物，但反之未必
  - 通常是单向的，带箭头的虚线，指向被依赖的模型元素。
  - 使用情景：
    - 一个类中某操作(operation)的参数是另一个类的对象
- 带箭头的虚线，指向被依赖的模型元素。

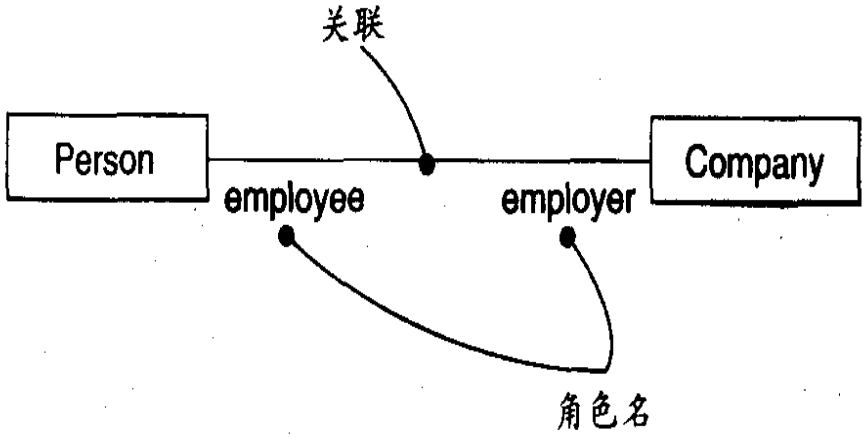


3.5 类之间的关系（续）

- 关联(association)
  - 指明一个事物的对象与另一个事物的对象间的联系。
  - 关联可以是双向的，单向的，自关联的
  - 把关联画成一条连接相同类或不同类的实线
  - 可以为相关联的两个类设置特定的角色
  - 多重性：为相关联的两个类设置基数关系
  - 关联名：为关联关系起个名字，进行标识。关联名用动词或动词短语来命名。

3.5 类之间的关系（续）

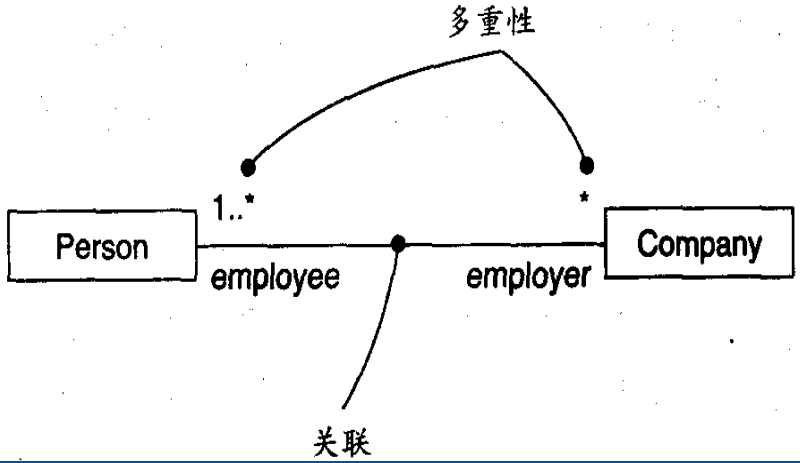
- 关联的角色
  - 角色是关联中靠近它的一端的类对另外一端的类呈现的职责。
  - 当一个类处于关联的某一端时，该类就在这个关系中扮演了一个特定的角色。
  - 相同类可以在其他关联中扮演相同或不同的角色。



例：
扮演employee角色的类person与扮演employer角色的类Company相关联。

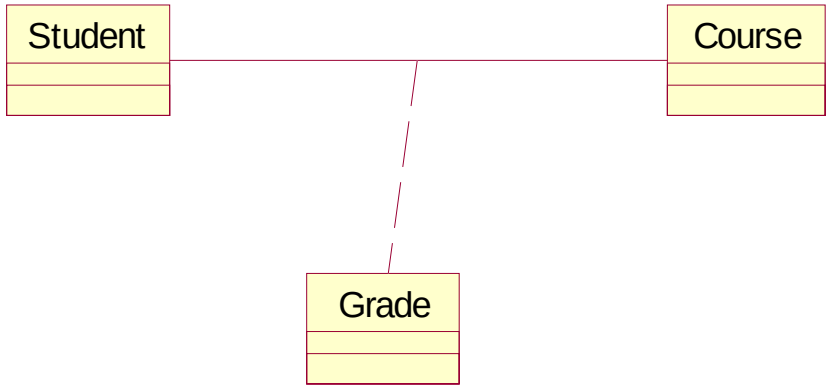
3.5 类之间的关系（续）

- 关联的多重性
  - 指定关联的一端的多重性，就是说明：在关联另一端的类的每个对象要求在本端的类必须有多少个对象。这个“多少”被称为关联角色的多重性。
  - 把它写成一个表示取值范围的表达式或写成一个具体值。
  - 可以精确地表示多重性为 1 （ 1 ）、 0 或 1 （ 0 ‥ 1 ）、很多（ 0 ‥ * ）、 1 个或很多（ 1 ‥ * ）。



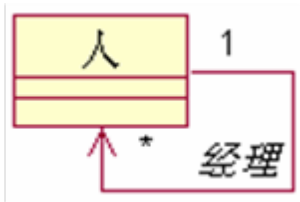
3.5 类之间的关系（续）

- 关联类（Association Class ）
  - 两个类之间的关联是通过某个类完成的
  - 把关联类画成一个类符号，并把它用一条虚线连接到相应的关联上。
  - 分别建立Student，Course类, Grade类
  - 建立Student与Course的关联
  - 将Grade 用Association Class连接到该关联



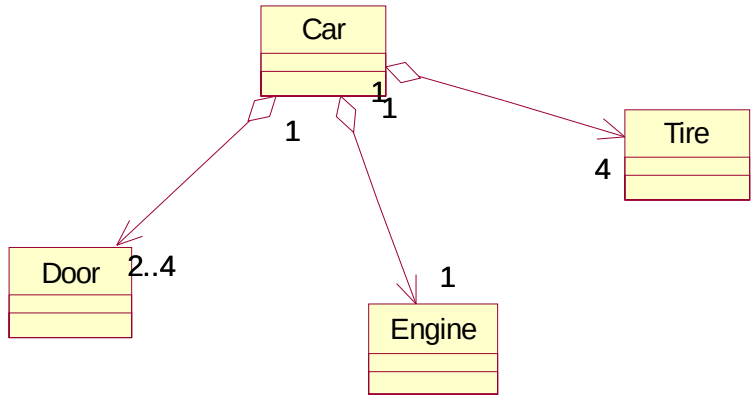
3.5 类之间的关系（续）

- 递归关联
  - 如果一个类与本身有关联关系，此关联称为递归关联。
  - 递归关联指的是同类的对象之间语义上的连接。



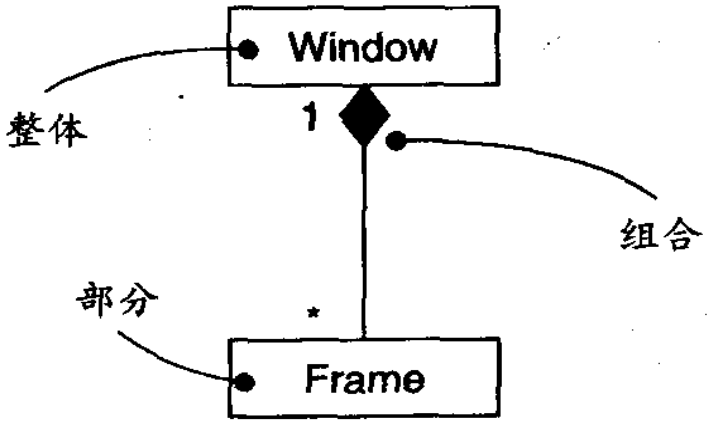
3.5 类之间的关系（续）

- 聚合（aggregation）
  - 表示“整体/部分”的关联关系。“has a ”
  - 也是一种关联，也被称作“聚集”
  - 被表示为在整体的一端用一个空心菱形修饰的简单关联。



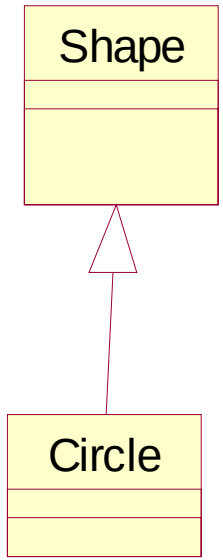
3.5 类之间的关系（续）

- 组成（composite）
  - 是强聚合
  - 表示聚合中的每一个部分只能属于一个整体
  - 组合确实只是一种特殊的关联，用整体端有实心菱形箭头的简单关联修饰它。



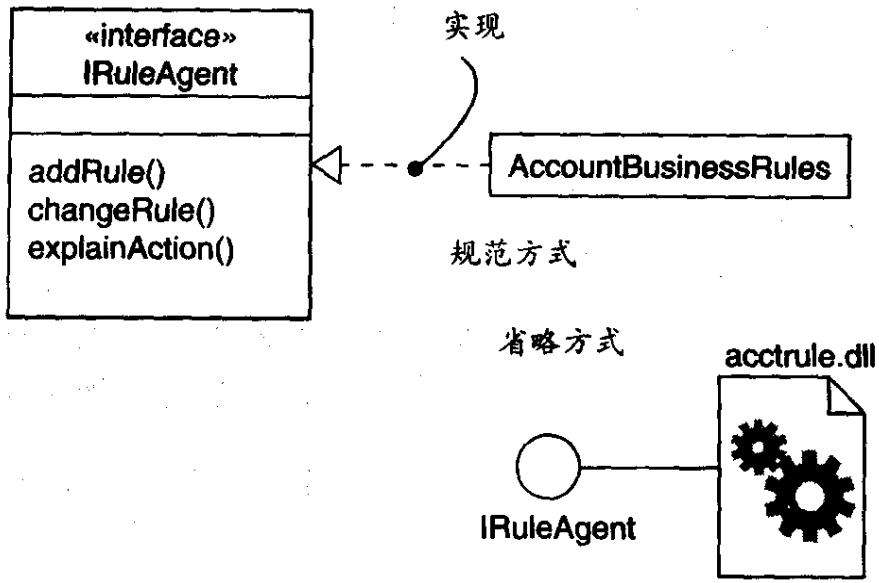
3.5 类之间的关系（续）

- 泛化(generalization)
  - 表示“一般/特殊”关系。“is-a-kind-of”
  - 也就是继承关系
  - 带空心箭头的实线表示，箭头指向父元素。



3.5 类之间的关系（续）

- 实现(realization)
  - 表示类和接口之间的关系
  - 实现是类元之间的语义关系，在该关系中一个类元描述了另一个类元保证实现的契约。



3.6 对关系建模遵循策略

- 仅当被建模的关系不是结构关系时，才使用依赖。
- 仅当关系是Is-a-kind-of关系时，才使用泛化。往往可以用聚合代替多继承。
- 小心不要引入循环的泛化关系。
- 一般要保持泛化关系的平衡；继承的层次不要太深（5层），也不要太宽（可能的中间抽象类）。
- 在对象间有结构关系的地方，要以使用关联为主。
- 要一致地使用直线或斜线。
- 避免线段交叉
- 仅显示对理解特定的成组事物必不可少的关系。

3.7 类图

- 类图
  - 表示一系列类、接口和它们的关系
  - 表示系统的静态视图，在Logical View中
- 用例实现的类图又叫VOPC（View Of Participated Classes）
  - VOPC类图表示用例实现的参与类以及这些类之间的关系
  - 确保跨越子系统的用例实现的一致性
- 类图的元素
  - 类
  - 关系

3.8 练习

3.9 小结

- 类的关系
- 类图的画法

Neusoft

Beyond Technology