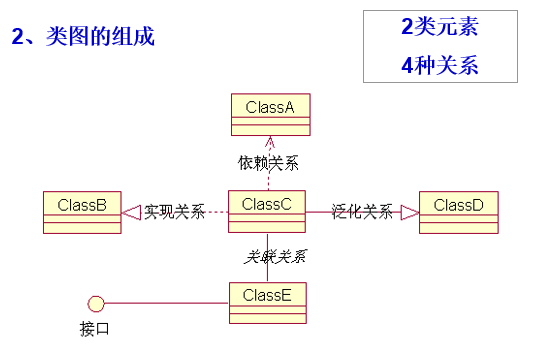
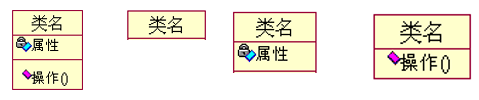
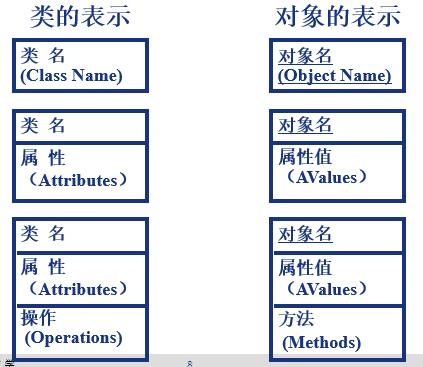
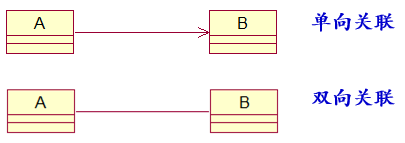
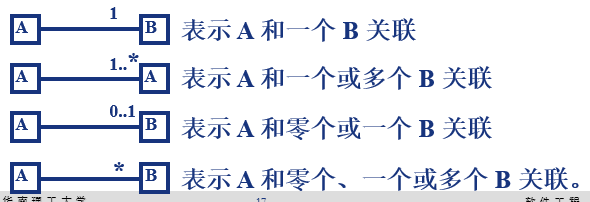
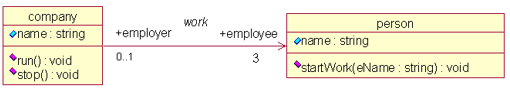
* **类图的概念：**
* **类图是描述类、接口以及它们之间关系的图，它显示了系统中各个类的静态结构，是一种静态模型。**
* **建立类模型的过程，是把现实世界中与问题有关的各种对象及其相互关系进行抽象和分类的过程。**
* **类图的组成**
* 
* （两种元素：类，接口，四种关系）
* **对象和类**
* **对象是指某个事物，大多对应于真实世界中的某个客观实体；但有些对象在真实世界中没有直接的对应物，是人们对某个事物的一种抽象描述。对象的基本特征可以归纳为对象的属性和行为两类。**
* **类是指对一组具有相同特征的对象的抽象描述；任何对象都是某个类的实例。**
* **类的符号**
* 
* **类图和对象图**
* 
* **基本要素**
* **关 联；**
* **关联的概念：**
  + - **关联用来表示来表示两个（或多个）类的对象之间的结构关系，它在代码中表现为一个类以属性的形式包含对另一个类的一个或多个对象的引用。**
* **关联的表示：**
* **用一条无向线段表示，是一种双向关系。**
* **用一条有向线段表示，是一种单向关系**
* **关联的命名：可以用动词词组或名词命名。**
* **二元关联**
* **1、概念**
* **两个类之间的关联称为二元关联。**
* **2、符号**
* 
* 角色及其命名
* **关联的两端与类之间(或与类的实例之间)的接口表示该类(或该类的实体)在这个关联中的行为，称之为角色。**
* **每个关联有两个角色（源和目标）。例如，对于客户和订单之间的关联是：客户和订单。**
  + **单向关联时，说明模型中的订单指出它是由哪个客户发出的；实现模型中的订单包含一个指向客户的指针。**
* **双向关联：相关联的两个类中都有指向对方的指针。**
* **角色的多元性(一个角色可以有多个对象来扮演)**
* 
  + **1 表示 1..1 ；\*代表零到无穷；0..1是选择符，表示没有或仅有1个；一个数；一个范围；数字和范围不连续的组合**
* 
  + **：\*可以换成n**
  + **1: 表示1个**
  + **0..\*： 表示任意多个（ ≥0）**
  + **\*： 表示任意多个（ ≥0）**
  + **1..\*： 表示1个或多个（≥1）**
  + **0..1： 表示0个或1个**
  + **5..11： 表示5-11个**
  + **1，3，8： 表示个1或3个或8个**
  + **0，3..8： 表示0个或3-8个**
  + 关联的元素
  + **（1）关联的名称**
  + **（2）端点**
  + **（3）多重性**
  + **（4）有序**
  + **（5）多元关联**
* **属 性；**
* 属性的语法
* **UML规定其语法为：**
* **可见性 名称[多重性]:类型 = 缺省值 {约束特性}**
  + **可见性：“+”表示公有操作，“#”表示受保护的操作，“-”表示私有操作。**
  + **客户属性的名称可以定义为一个单独的类：定义名字的属性及其相关的操作；然后在客户类和该属性名称类之间建立关联。**
* **操 作；**
* **描述类的行为的函数，有些地方也称为方法。**
* **操作的语法**
* **可见性 名称(参数表):返回类型表达式{约束特性}**
* **泛 化；**
* **在现实生活中，有时会出现一类事物具有另一类事物**
* **的全部特点，除此之外还有自身的特点。**
  + **泛化关系(继承关系)定义类和包之间的一般元素和特殊元素之间的分类关系。**
* 空心实线箭头指向超类（有子类的公共部分）
* **特化(Specialization): 实例化**
* **继承(Inheritance):**
* **泛化关系的一种实现机制**
* **并非所有的泛化关系都适合用继承关系实现**
  + **多态性：在泛化关系中，父类所定义的操作被子类继承之后，可以表现出不同的行为。（函数重载）**
* **（1）聚合**
  + **聚合也是表示类和类之间的“整体－部分”关系，整体类端用空心菱形表示，加箭头指向部分类。聚合意味着一个类拥有但共享另一个类的对象。**
  + **关联关系和聚合关系来语法上是没办法区分的，从语义上才能更好的区分两者的区别。**
* **（2）组合**
* **组合是聚合的一种特殊情形，用实心菱形表示。**
* **组合是一种特殊形式的聚合。与聚合相比，它有两个特点：**
* **一个部分类最多只能属于一个整体类**
  + **当整体类不存在时，部分类将同时被销毁**
  + **在整体类端加实心菱形**
  + **依赖**
  + **依赖表示一个类以某种形式依赖于其他类。当两个类处于依赖关系中时，其中一个类的改变可能会影响另一个类。**
  + **虚线箭头**
  + **A- - - >B，A依赖B，B是参数或者局部变量**
  + **B端有单词表示角色，是B在A中的体现形式。**
* **授 权；**
* **约束规则**
  + **约束规则的语法：将约束条件放在括号{ }中，用自然语言或其他常见的设计语言来描述，其描述要简洁准确。**
  + **类图的建模分析步骤**
  + **（1）寻找出需求中的名词(候选对象)。**
  + **（2）合并含义相同的名词，排除范围以外的名词，并寻找隐含的名词。**
  + **（3）去掉只能作为类属性的名词。**
  + **（4）剩下的名词就是要找的分析类（候选类)。**
  + **（5）根据常识、问题域、系统责任确定该类有那些属性。**
  + **（6）补充该类动态属性，如状态、对象间联系（如聚合、关联）等属性。**
  + **（7）从需求中的动词、功能或系统责任中寻找类的操作（候选操作）。**
  + **（8）从状态转换，流程跟踪、系统管理等方面补充类的操作。**
  + **（9）对所寻找的操作进行合并、筛选。**
  + **（10）对所寻找的操作在类间进行合理分配（职责分配），形成每个类候选操作。**
  + **（11）补充每个类的的分析文档，为类的进一步设计打下基础。**
  + **属性或类**
  + **（4）定义属性**
  + **寻找属性时，一般要从类的实例-对象来着手，并从以下几个方面进行考虑：**
  + **1、按一般常识这个对象有那些属性。**
  + **2、在当前问题域该对象有那些属性。**
  + **3、根据系统责任，这个对象应该有那些属性。**
  + **4、为了实现某些功能，对象需要增加那些属性。**
  + **5、对象有那些区别的状态。**
  + **6、对象整体和部分属性设。**
  + **（5）定义操作**
  + **找操作要从类的实例-对象来着手，并从以下几个方面进行考虑：**
  + **1、从需求中的功能，寻找对象的操作。**
  + **2、系统行为推迟到设计阶段。**
  + **3、暂不考虑对象中读、写属性这样的操作。**
  + **4、根据系统责任，这个对象应该有那些操作。**
  + **5、分析对象的状态转换，寻找操作。**
  + **6、追踪流程，寻找操作。**
    - **7、类本身要使用那些操作来维护信息更新以及信息的一致性和完整性。**
  + **系统实体识别法**
  + **系统需要存储、分析、处理的信息实体；**
  + **系统内部需要处理的设备；**
  + **与系统交互的外部系统；**
  + **系统相关人员；**
  + **系统的组织实体。**
  + **⑴ 分解技术　将整体类和组合类分解。可控制单个类的规模。**
  + **⑵ 抽象技术　根据一些类的相似性建立抽象类，并建立抽象类与这些类之间的继承关系。**