**UML**

## 一、基本概念

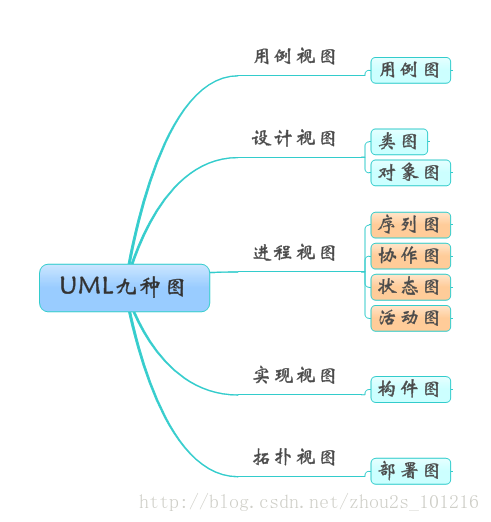
**UML（Unified Modeling Language）**是一种统一建模语言，为面向对象开发系统的产品进行说明、可视化、和编制文档的一种标准语言。下面将对UML的九种图+包图的基本概念进行介绍以及各个图的使用场景。

二、静/动态图

如下图所示，UML图分为用例视图、设计视图、进程视图、实现视图和拓扑视图，又可以静动分为静态视图和动态视图。

（1）静态图分为：用例图，类图，对象图，包图，构件图，部署图。

（2）动态图分为：状态图，活动图，协作图，序列图。



**1、用例图（UseCase Diagrams）**

用例图主要回答了两个问题：

1. 是谁用软件。
2. 软件的功能。

从用户的角度描述了系统的功能，并指出各个功能的执行者，强调用户的使用者，系统为执行者完成哪些功能。



**2、类图（Class Diagrams）**

用户根据用例图抽象成类，描述类的内部结构和类与类之间的关系，是一种静态结构图。 在UML类图中，常见的有以下几种关系: 泛化（Generalization）,  实现（Realization），关联（Association)，聚合（Aggregation），组合(Composition)，依赖(Dependency)。

**各种关系的强弱顺序： 泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖**

**2.1.泛化**

【泛化关系】：是一种继承关系，表示一般与特殊的关系，它指定了子类如何继承父类的所有特征和行为。例如：老虎是动物的一种，即有老虎的特性也有动物的共性。



**2.2.实现**

【实现关系】：是一种类与接口的关系，表示类是接口所有特征和行为的实现。



**2.3.关联**

【关联关系】：是一种拥有的关系，它使一个类知道另一个类的属性和方法；如：老师与学生，丈夫与妻子关联可以是双向的，也可以是单向的。双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头。

【代码体现】：成员变量



**2.4.聚合**

【聚合关系】：是整体与部分的关系，且部分可以离开整体而单独存在。如车和轮胎是整体和部分的关系，轮胎离开车仍然可以存在。

聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系。

【代码体现】：成员变量



**2.5.组合**

【组合关系】：是整体与部分的关系，但部分不能离开整体而单独存在。如公司和部门是整体和部分的关系，没有公司就不存在部门。组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

【代码体现】：成员变量

【箭头及指向】：**带实心菱形的实线，菱形指向整体**



**2.6.依赖**

【依赖关系】：是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖.

【代码表现】：局部变量、方法的参数或者对静态方法的调用

【箭头及指向】：带箭头的虚线，指向被使用者



**关联和依赖的区别——**  
发生依赖关系的两个类都不会增加属性。发生关联关系的两个类，其中的一个类成为另一个类的属性，而属性是一种更为紧密的耦合，更为长久的持有关系。

依赖(dependency)

而这种使用关系是具有偶然性的、临时性的、非常弱的，但是B类的变化会影响到A；比如某人要过河，需要借用一条船，此时人与船之间的关系就是依赖；

表现在代码层面，为类B作为参数被类A在某个method方法中使用。用带虚线的箭头。

class A

{

public void f(B b)

{

}

}

class B

{

}

关联(association)

他体现的是两个类、或者类与接口之间语义级别的一种强依赖关系，比如我和我的朋友；这种关系比依赖更强、不存在依赖关系的偶然性、关系也不是临时性的，一般是长期性的，

而且双方的关系一般是平等的、关联可以是单向、双向的；表现在代码层面，为被关联类B以类属性的形式出现在关联类A中，也可能是关联类A引用了一个类型为被关联类B的全局变量；

class A{

private B b;

}

class B

{

}

依赖和关联区别：我用锤子修了一下桌子，我和锤子之间就是一种依赖，我和我的同事就是一种关联。依赖是一种弱关联，只要一个类用到另一个类，但是和另一个类的关系不是太明显的时候（可以说是“uses”了那个类），就可以把这种关系看成是依赖，依赖也可说是一种偶然的关系，而不是必然的关系。关联是类之间的一种关系，例如老师教学生，老公和老婆这种关系是非常明显的。依赖是比较陌生，关联是我们已经认识熟悉了。

**2.7 各种类图关系**



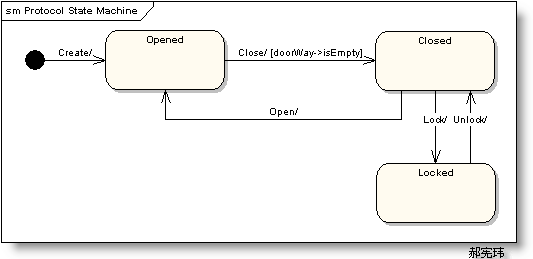
**3、对象图（Object Diagrams）**

描述的是参与交互的各个对象在交互过程中某一时刻的状态。对象图可以被看作是类图在某一时刻的实例。



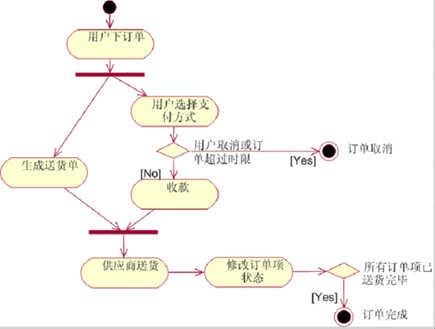
**4、状态图（Statechart Diagrams）**

是一种由状态、变迁、事件和活动组成的状态机，用来描述类的对象所有可能的状态以及时间发生时状态的转移条件。



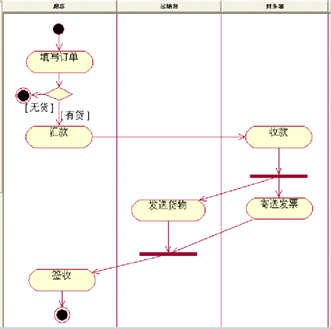
**5、活动图（Activity Diagrams）**

是状态图的一种特殊情况，这些状态大都处于活动状态。本质是一种流程图，它描述了活动到活动的控制流。交互图强调的是对象到对象的控制流，而活动图则强调的是从活动到活动的控制流。活动图是一种表述过程基理、业务过程以及工作流的技术。它可以用来对业务过程、工作流建模，也可以对用例实现甚至是程序实现来建模。



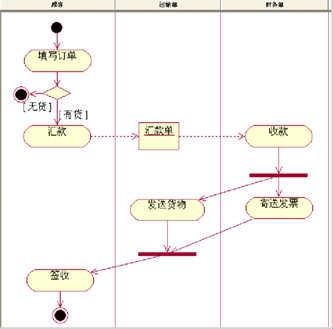
**5.1 带泳道的活动图**

泳道表明每个活动是由哪些人或哪些部门负责完成。



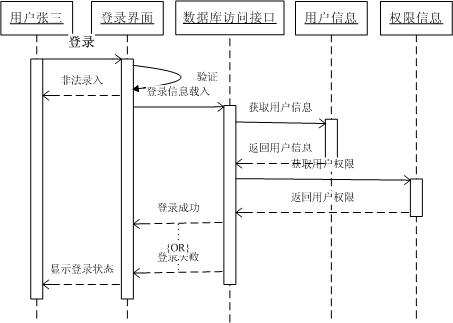
**5.2 带对象流的活动图**

用活动图描述某个对象时，可以把涉及到的对象放置在活动图中，并用一个依赖将其连接到进行创建、修改和撤销的动作状态或者活动状态上，对象的这种使用方法就构成了对象流。对象流用带有箭头的虚线表示。



**6、序列图-时序图（Sequence Diagrams）**

交互图的一种，描述了对象之间消息发送的先后顺序，强调时间顺序。序列图的主要用途是把用例表达的需求，转化为进一步、更加正式层次的精细表达。用例常常被细化为一个或者更多的序列图。同时序列图更有效地描述如何分配各个类的职责以及各类具有相应职责的原因。

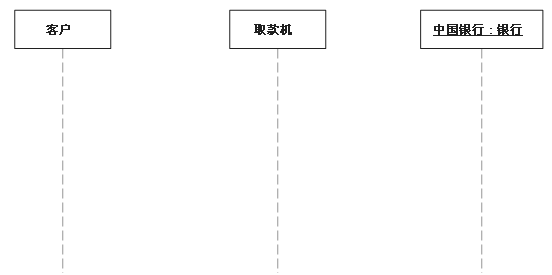


消息用从一个对象的生命线到另一个对象生命线的箭头表示。箭头以时间顺序在图中从上到下排列。

　　序列图中涉及的元素：

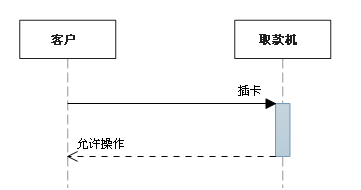
**6.1 生命线**

生命线名称可带下划线。当使用下划线时，意味着序列图中的生命线代表一个类的特定实例。



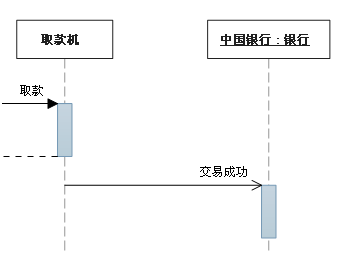
**6.2 同步消息**

同步等待消息

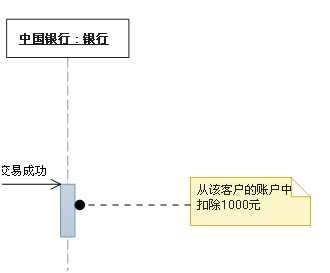


**6.3 异步消息**

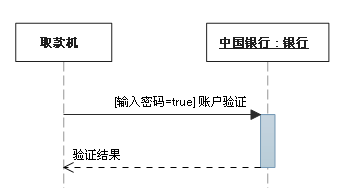
异步发送消息，不需等待



**6.4 注释**



**6.5 约束**

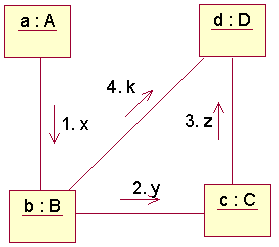


**6.6 组合**

组合片段用来解决交互执行的条件及方式。它允许在序列图中直接表示逻辑组件，用于通过指定条件或子进程的应用区域，为任何生命线的任何部分定义特殊条件和子进程。常用的组合片段有：抉择、选项、循环、并行。

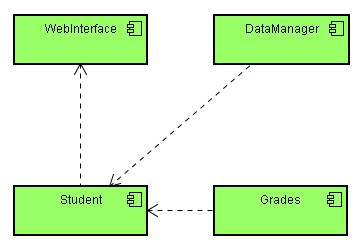
**7、协作图（Collaboration Diagrams）**

交互图的一种，描述了收发消息的对象的组织关系，强调对象之间的合作关系。时序图按照时间顺序布图，而写作图按照空间结构布图



**8、构件图（Component Diagrams）**

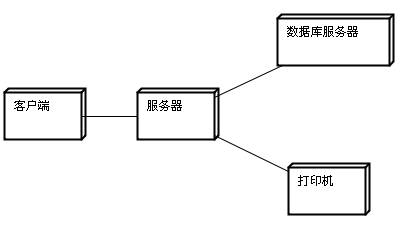
构件图是用来表示系统中构件与构件之间，类或接口与构件之间的关系图。其中，构建图之间的关系表现为依赖关系，定义的类或接口与类之间的关系表现为依赖关系或实现关系。



**9、部署图（Deployment Diagrams）**

描述了系统运行时进行处理的结点以及在结点上活动的构件的配置。强调了物理设备以及之间的连接关系。

部署模型的目的：描述一个具体应用的主要部署结构，通过对各种硬件，在硬件中的软件以及各种连接协议的显示，可以很好的描述系统是如何部署的；平衡系统运行时的计算资源分布；可以通过连接描述组织的硬件网络结构或者是嵌入式系统等具有多种硬件和软件相关的系统运行模型。



## 图的差异比较

**1.序列图(时序图)VS协作图**

序列图和协作图都是交互图。二者在语义上等价，可以相互转化。但是侧重点不同：序列图侧重时间顺序，协作图侧重对象间的关系。

共同点：时序图与协作图均显示了对象间的交互。

不同点：时序图强调交互的时间次序。

           协作图强调交互的空间结构。

**2.状态图VS活动图**

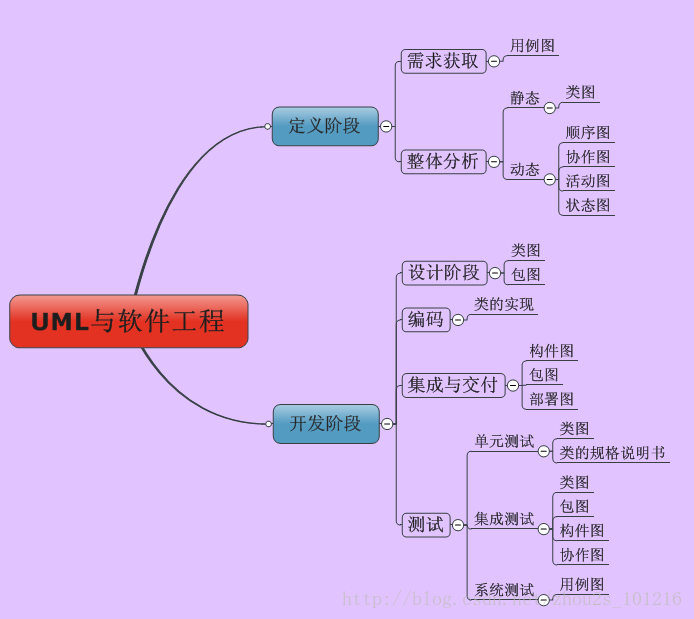
状态图和活动图都是行为图。状态图侧重从行为的结果来描述，活动图侧重从行为的动作来描述。状态图描述了一个具体对象的可能状态以及他们之间的转换。在实际的项目中，活动图并不是必须的，需要满足以下条件：1、出现并行过程&行为；2、描述算法；3、跨越多个用例的活动图。

**3.活动图VS交互图**

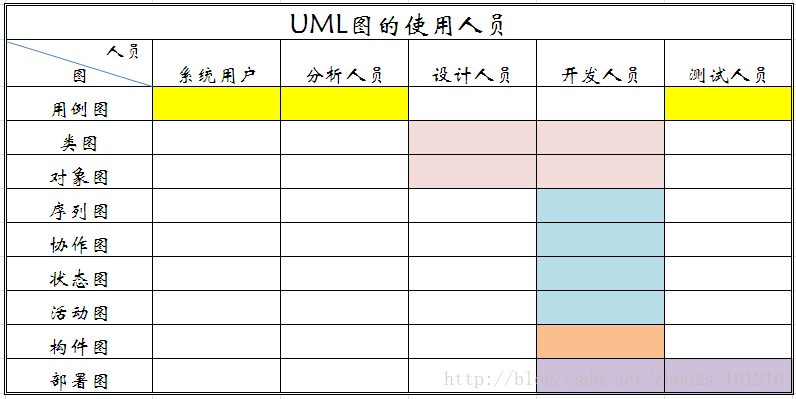
二者都涉及到对象和他们之间传递的关系。区别在于交互图观察的是传送消息的对象，而活动图观察的是对象之间传递的消息。看似语义相同，但是他们是从不同的角度来观察整个系统的。

## 三、UML与软件工程

UML图是软件工程的组成部分，软件工程从宏观的角度保证了软件开发的各个过程的质量。而UML作为一种建模语言，更加有效的实现了软件工程的要求。如下图，在软件的各个开发阶段需要的UML图。



**下表是UML使用人员图示**



**PowerDesigner**

## PD概念

**1 PowerDesigner 是一个集所有现代建模技术于一身的完整工具，它集成了强有力的业务建模技术、传统的数据库分析和实现，以及UML对象建模。**通过了元数据的管理、冲突分析和真正的企业知识库等功能。利用它可以制作数据流程图、概念数据模型、物理数据模型.帮助企业快速高效地进行企业应用系统构建及再工程(Re-engineer)。 IT专业人员可以利用它来有效开发各种解决方案，从定义业务需求到分析和设计，以至集成所有现代 RDBMS 和Java、.NET、PowerBuilder和 Web Services的开发等。PDM定义了模型的物理实现细节。例如，所选RDBMS的数据类型特征、索引定义、视图定义、存储过程定义、触发器定义等。

2 PowerDesigner最初由Xiao-Yun Wang（王晓昀）在SDP Technologies公司开发完成。PowerDesigner是Sybase的企业建模和设计解决方案，采用模型驱动方法，将业务与IT结合起来，可帮助部署有效的企业体系架构，并为研发生命周期管理提供强大的分析与设计技术。PowerDesigner独具匠心地将多种标准数据建模技术（UML、业务流程建模以及市场领先的数据建模）集成一体，并与 .NET、WorkSpace、PowerBuilder、Java™、Eclipse 等主流开发平台集成起来，从而为传统的软件开发周期管理提供业务分析和规范的数据库设计解决方案。此外，它支持60多种关系数据库管理系统（RDBMS）/版本。PowerDesigner运行在Microsoft Windows平台上，并提供了Eclipse插件。

3 Power Designer 是Sybase公司的CASE工具集，使用它可以方便地对管理信息系统进行分析设计，他几乎包括了数据库模型设计的全过程。利用Power Designer可以制作数据流程图、概念数据模型、物理数据模型，还可以为数据仓库制作结构模型，也能对团队设计模型进行控制。他可以与许多流行的软件开发工具，例如PowerBuilder、Delphi、VB等相配合使开发时间缩短和使系统设计更优化。

4 power designer是能进行数据库设计的强大的软件，是一款开发人员常用的数据库建模工具。使用它可以分别从概念数据模型(Conceptual Data Model)和物理数据模型(Physical Data Model)两个层次对数据库进行设计。在这里，概念数据模型描述的是独立于数据库管理系统(DBMS)的实体定义和实体关系定义；物理数据模型是在概念数据模型的基础上针对目标数据库管理系统的具体化

## PD的4种模型文件

**1 概念数据模型（CDM）**

   CDM表现数据库的全部逻辑结构，与具体的数据结构无关，以ER图理论为基础，反映客观存在的事物。

**2 物理数据模型（PDM）**

   PDM与具体的物理实现有关，如具体的DB，反映在软件设计上即是数据库物理设计。

**3 面向对象模型（OOM）**

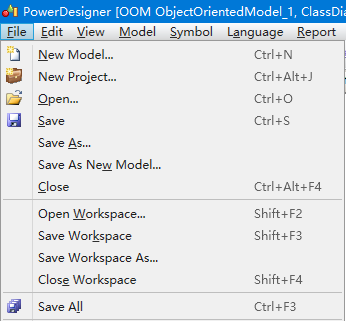
一个OOM包含一系列的包，类，接口，和他们的关系。这些对象一起形成所有的一个软件系统

**4 业务程序模型 (BPM)**

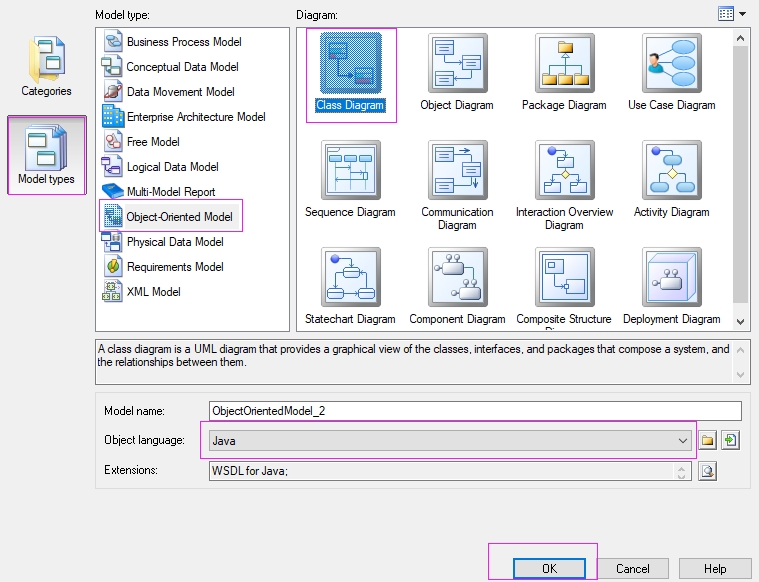
BPM 描述业务的各种不同内在任务和内在流程，且客户如何以这些任务和流程互相影响。

## 三、实战操作

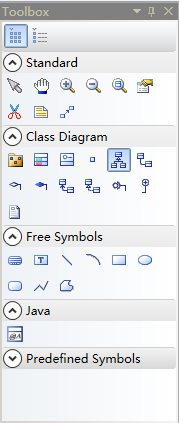
因为后面要理清楚spring中类与类之间关系，我们来看看如何使用powerDesigner绘制类图，下载powerDesiger，一路下一步安装好，打开，点【file】-【New Model】



**选择【Model types】-【Object-Oriented Model】-【Class Diagram】-【java】-【ok】**



利用右侧的toolbox构建



**泛化 generalization**

**实现 Realization**

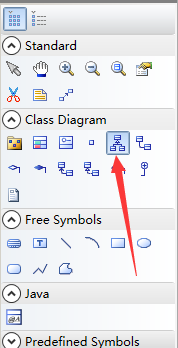
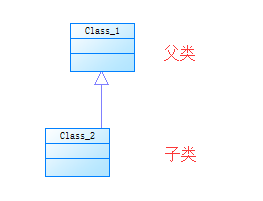
**关联 Association**

**聚合 Aggregation**

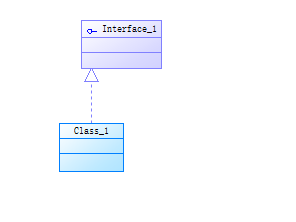
**组合 Composition**

**依赖 Dependency**

**1 泛化，子类继承父类**



**2实现，实现类继承接口**

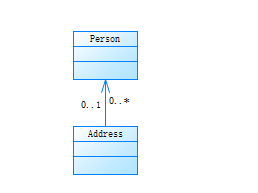


**3 关联**

是一种拥有的关系, 它使一个类知道另一个类的属性和方法,这个概念听起来感觉有点蒙,其实吧,就是一个对象A中持有对象B的引用,这样对象B中的属性和方法是不是A也知道了,就是这个意思,在实际的代码中体现在一个类上声明一个成员变量

**类A中有类B的引用**

**类Person 中有 类 Address**

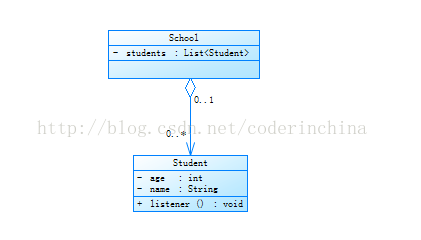


**4 聚合**

是整体与部分的关系, 且部分可以离开整体而单独存在,如车和轮胎是整体和部分的关系,轮胎离开车仍然可以存在聚合关系是关联关系的一种，是强的关联关系；关联和聚合在语法上无法区分，必须考察具体的逻辑关系

比如一个学校有高中部和初中部,这些学生加起来都是属于这个学校的,那么从聚合的概念上讲整体就是这个学校,部分就是学生了,且部分可以离开整体而单独存在,这意思是说学生可以转校去别的学校,但是这个学校还是存在的,不会因为一个学生转校了找个学校就没了,

箭头指向:带空心菱形的实心线，菱形指向整体



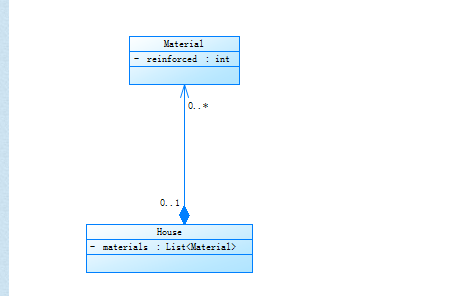
**5 组合**

是整体与部分的关系, 但部分不能离开整体而单独存在. 如公司和部门是整体和部分的关系, 没有公司就不存在部门。组合关系是关联关系的一种，是比聚合关系还要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。

代码体现:成员变量

箭头指向:带实心菱形的实线,菱形指向整体

比如一个大厦肯定是由很多种材料构成的,如果没有这些材料,这大厦就不存在了,意思是这个大厦生命周期over了

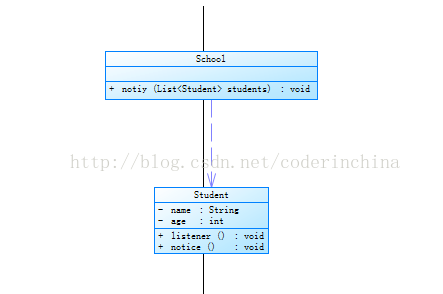


**6 依赖**

***是一种使用的关系,  一个类的实现需要另一个类的协助,所以要尽量不使用双向的互相依赖***

代码表现:局部变量、方法的参数或者对静态方法的调用

箭头指向：带箭头的虚线，指向被使用者



**7 各种关系的强弱顺序**

**泛化 >实现>组合> 聚合> 关联> 依赖**

最后把这些关系在java代码中的体现总结下

(1) 依赖关系：关系对象出现在局部变量或者方法的参数里，或者关系类的静态方法被调用   
(2) 关联关系：关系对象出现在实例变量中   
(3) 聚合关系:关系对象出现在实例变量中   
(4) 组合关系：关系对象出现在实例变量中   
(5) 泛化关系: extends   
(6) 实现： implements

**StarUml**

## 理论知识

**1 StarUML介绍**

一款开放源码的UML开发工具，是由韩国公司主导开发出来的产品，可以直接到StarUML网站下载。可以用来创建UML类图。StarUML(简称SU)，是一种创建UML类图，生成类图和其他类型的统一建模语言(UML)图表的工具。**StarUML是一个开源项目之一发展快、灵活、可扩展性强。**

**2 StarUML特点**

● 可绘制9款UML图：用例图、类图、序列图、状态图、活动图、通信图、构件图、部署图以及复合结构图等。

● 完全免费：StarUML是一套开放源码的软件，不仅免费自由下载，连代码都免费开放。

● 多种格式影像文件：可导出JPG、JPEG、BMP、EMF和WMF等格式的影像文件。

● 语法检验：StarUML遵守UML的语法规则，不支持违反语法的动作。

● 正反向工程：StarUML可以依据类图的内容生成Java、C++、C#代码，也能够读取Java、C++、C#代码反向生成类图。反向工程有两个主要用途，其一是旧有的源码反转成图之后，可以构建UML模型的方式继续将新的设计添加上去；另一项用途是想要解析源码时，可以通过反转的类图来理解，不再需要查看一行又一行的代码，这将节省大量的时间和精力。

● 支持XMI：StarUML接受XMI 1.1、1.2和1.3版的导入导出。XMI是一种以XML为基础的交换格式，用以交换不同开发工具所生成的UML模型。

● 导入Rose文件：StarUML可以读取Rational Rose生成的文件，让原先Rose的用户可以转而使用免费的StarUML。早期，Rational Rose是市场占有率最高的UML开发工具，同时也是相当昂贵的工具。由于Rational Rose非常闻名，后来让IBM给收购了。

● 支持模式：支持23种GoF模式(Pattern)，以及3种EJB模式。GoF模式出自于Erich Gamma等4人合著的Design Patterns：Elements of Reusable Object-Oriented Software一书，其内列出了23种软件模式，可解决软件设计上的特定问题。StarUML也支持3种常用的EJB模式，分别为EntityEJB、MessageDrivenEJB、SessionEJB。

StarUML也结合了模式和自动生成代码的功能，方便落实设计。

## 操作实战

**Visio**

## 理论知识

Office Visio 是office软件系列中的负责绘制流程图和示意图的软件，是一款便于IT和商务人员就复杂信息、系统和流程进行可视化处理、分析和交流的软件。使用具有专业外观的 Office Visio 图表，可以促进对系统和流程的了解，深入了解复杂信息并利用这些知识做出更好的业务决策。

Microsoft Office Visio帮助您创建具有专业外观的图表，以便理解、记录和分析信息、数据、系统和过程。大多数图形软件程序依赖于艺术技能。然而，在您使用 Visio 时，以可视方式传递重要信息就像打开模板、将形状拖放到绘图中以及对即将完成的工作应用主题一样轻松。Office Visio 2010中的新增功能和增强功能使得创建 Visio 图表更为简单、快捷，令人印象更加深刻。

## 操作实战