UML 建模学习

1 什么是UML

1. `UML`是`Unified Model Language`的缩写，中文是`统一建模语言`，是由一整套图表组成的标准化建模语言。
2. UML提供了极富表达能力的建模语言，可以让软件开发过程中的不同人员分别得到自己感兴趣的信息。

1.1 UML 目的

`Page-Jones`在《Fundamental Object-Oriented Design in UML》一书中总结了UML的主要目的，如下：

1. 为用户提供现成的、有表现力的可视化建模语言，以便他们开发和交换有意义的模型。
2. 为核心概念提供可扩展性 (Extensibility) 和特殊化 (Specialization) 机制。
3. 独立于特定的编程语言和开发过程。
4. 为了解建模语言提供一个正式的基础。
5. 鼓励面向对象工具市场的发展。
6. 支持更高层次的开发概念，如协作，框架，模式和组件。
7. 整合最佳的工作方法 (Best Practices)。

1.2 UML 分类

1) 用例图(use case)

2) 静态结构图：类图、对象图、包图、组件图、部署图

3) 动态行为图：交互图（时序图与协作图）、状态图、活动图

2 类图

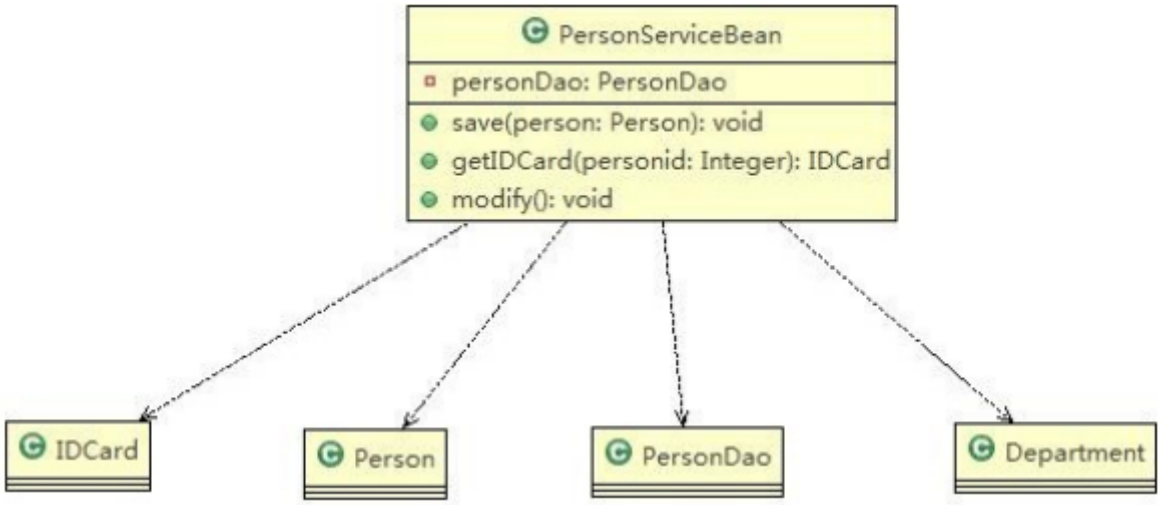
类之间的关系：依赖、泛化（继承）、实现、关联、聚合与组合

2.0 可见性

UML中，可见性分为4级

1. public 公用的： 用**+** 前缀表示，该属性对所有类可见
2. protected 受保护的：用 **#** 前缀表示，对该类的子孙可见
3. private 私有的：用 **-** 前缀表示，只对该类本身可见
4. package 包的：用 **~** 前缀表示，只对同一包声明的其他类可见

2.1 依赖关系



1) 类中用到了对方

2) 如果是类的成员属性

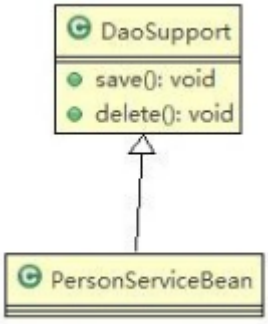
3) 如果是方法的返回类型

4) 是方法接收的参数类型

5) 方法中使用到

2.2 泛化关系

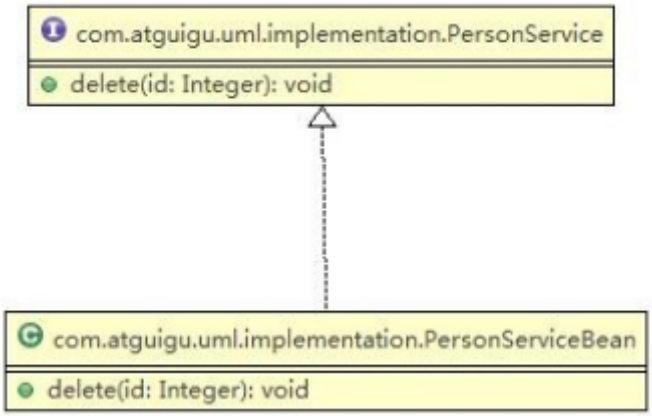
泛化关系实际上就是继承关系，他是依赖关系的特例。



1) 泛化关系实际上就是**继承关系**

2) 如果 A 类继承了 B 类，我们就说 A 和 B 存在泛化关系

2.3 实现关系

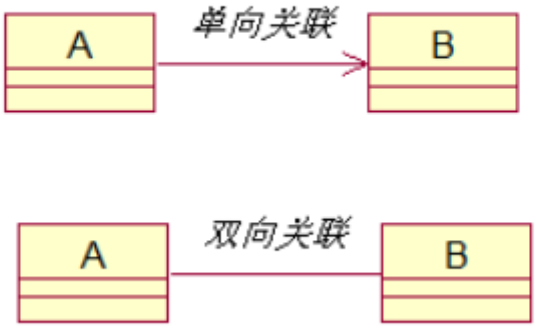


实现关系实际上就是 **A** 类实现 **B** 接口，他是依赖关系的特例

2.4 关联关系

关联（Association）关系是对象之间的一种引用关系，用于表示一类对象与另一类对象之间的联系。

关联具有导向性：即单向关系或者双向关系。



2.5 聚合关系

聚合关系（Aggregation）表示的是整体和部分的关系，**整体与部分可以分开。**聚合关系是关联关系的特例，所以他具有关联的导航性与多重性。

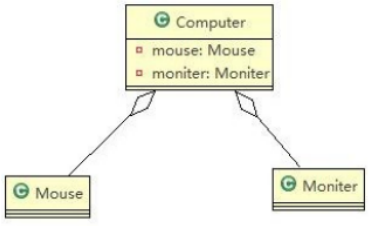
聚合的符号：空心菱形

如：一台电脑由键盘(keyboard)、显示器(monitor)，鼠标等组成；组成电脑的各个配件是可以从电脑上分离出来的，使用带空心菱形的实线来表示：

```
public class Computer{
   private Mouse mouse;
   private Monitor monitor;

   public void setMouse(Mouse mouse){
      this.mouse = mouse;
   }

   public void setMonitor(Monitor monitor){
      this.monitor = monitor;
   }
}
```



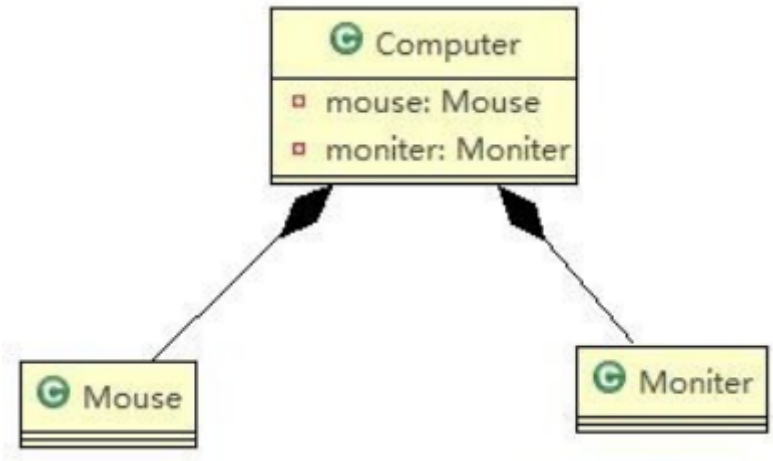
2.6 组合关系

组合关系：也是整体与部分的关系，但是整体与部分不可以分开。

再看一个案例：在程序中我们定义实体：Person 与 IDCard、Head, 那么 Head 和 Person 就是 组合，IDCard 和

Person 就是聚合。

但是如果在程序中 Person 实体中定义了对 IDCard 进行级联删除，即删除 Person 时连同 IDCard 一起删除，那么 IDCard 和 Person 就是组合了



聚合与组合的区别：

组合：整体类端的重数必须是1，部分类的重数是任意的。

聚合：整体类端的重数可以大于1，部分类的重数是任意的。