OOAD第八章

1. 类图——类+关联

   面向对象设计的基础就是使用类

   类是用来代表现实事务或者功能的构造块。

   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。

   可视性（Visibility）标记表示：

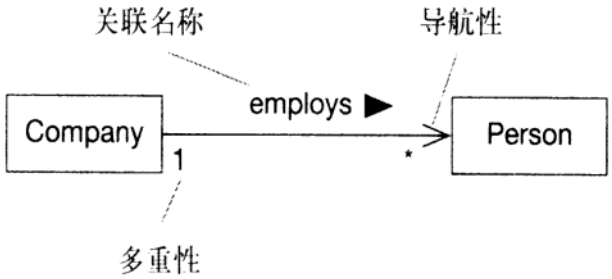
   + 公共

   # 保护

   - 私用

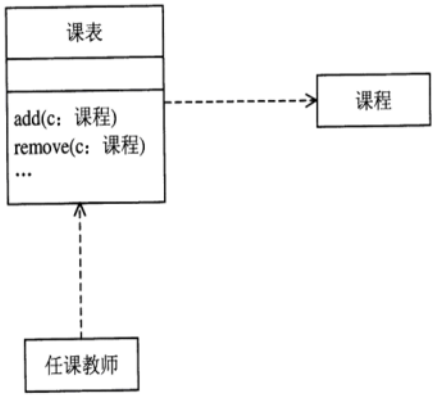
   关联关系

   使一个类知道另一个类的属性和方法



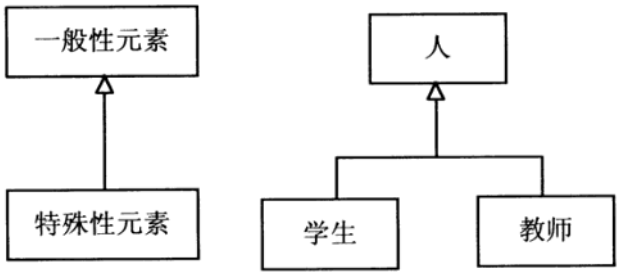
   依赖关系

   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 。



   聚合关系

   是整体和个体之间的关系

2. GRASP: 基于职责设计对象——基于OO设计的系统方法

OOD会被解释为：

把系统分成相对独立但又相互联系的对象组合的一种设计方法

步骤：

发现对象--对象的属性--对象的行为--对象的关系。

模式（pattern）是对软件设计问题的可重用的解决方案

设计模式就是在大量的实践中总结和理论化之后优选的代码结构、编程风格及解决问题的思考方式。

GRASP模式

GoF设计模式

GRASP模式——面向对象设计的角度

着重考虑设计类的原则以及如何分配类的功能

GOF模式——针对特定问题而提出的解决方法

着重考虑设计的实现、类的交互和软件质量。

GRASP可以说是GoF等设计模式的基础。

GRASP是General Responsibility Assignment Software Patterns（通用职责分配软件模式）的简称

核心思想“职责分配”,即Responsibility Assignment

GRASP的主要特征：

对象职责分配的基本原则。

主要应用在分析和建模上。

GRASP的核心思想的理解：

自己干自己的事（职责的分配）

自己干自己的能干的事（职责的分配）

自己只干自己的事（职责的内聚）

也就是说，如何把现实世界的业务功能抽象成对象，如何决定一个系统有多少对象，每个对象都包括什么职责， GRASP 模式给出了最基本的指导原则

GRASP基本原则

信息专家、创建者、高内聚、低耦合、控制者、多态、纯虚构、间接性 、变化预防

面向对象的过程就是将职责分配给对象的过程

职责不是类的方法

类的方法是用来实现职责的

职责的分配可以反应在协作图或顺序图中

GRASP是为对象分配职责的模式

交互图通常用作为对象分配职责的工具

创建者模式

如果以下条件之一为真时（越多越好），将创建类A实例的职责分配给类B

指导我们分配那些与创建对象有关的职责，这是很常见的**任务**

信息专家——“该干嘛干嘛去，别管别人的闲事”

把职责分配给具有完成该职责所需信息的那个类

信息专家模式对应于面向对象设计原则中的单一职责原则

低耦合

分配职责以使耦合保持在较低的水平

应该以降低类之间的耦 合关系作为职责分配的原则。

耦合往往能够减少修改软件所需的时间、工作量和缺陷。

控制器

在控制器模式中，要求系统事件的接收与处理通常由 一个高级类来代替；一个子系统需要定义多个控制器，分别对应不同的事务处理。

通常，一个控制器应当把要完成的功能委托给其他对象，它只负责协调和控制，本身不完成太多 的功能。

它可以将用户界面所提交的请求转发给其他类来处理，控制器可以重用，且不能包含太多业务逻 辑，一个系统通常也不能设计一个统一的控制器。

控制器模式（Controller）是GRASP模式中解决 事件处理职责问题的模式

分类：

外观控制器

提供了从UI层往其他层的服务调用的主要入口

代表了整个系统，设备或一个子系统

用例控制器

对每一个用例设置一个单独的控制器

当有太多的系统事件并设计不同的过程，用例控制器是一个好的选择

GRASP有以下共识：

系统事件的接收与处理通常由一个高级类来代替；

一个子系统会有很多控制器类，分别处理不同的事物。

高内聚

职责分配应保持高内聚

表现为“各司其职”，即：自己只干跟自己相关的工作，别人的工作让别人做。

内聚与耦合的辩证关系

高内聚要求把紧密关联的职责聚集在同一个类中，防止功能的扩散和类的无谓增加，从而减少类之间的关 联，降低类之间发生耦合的机率。

高内聚要求把不相关的功能分散到不同的类，势必造成相互关联类的增加，从而增大类之间发生耦合的机率。

面向对象设计，应该考虑效率，实现难度等因素，同时兼顾高内聚（High Cohesion）与低耦合（Low Coupling）性。