# 面向对象的设计方法

## 面向对象设计综述

* 1. 面向对象的设计：以需求分析阶段的**用例模型和领域模型为基础**，运用UML构建软件系统结构，通过一系列设计模型**说明用例的实现过程**。
  2. 软件概要设计步骤
     1. 选择合适的软件架构
     2. 动态结构设计
     3. 静态结构设计
  3. 软件详细设计
  4. 面向对象设计的关键步骤
     1. 发现对象（发现软件类）：根据需求和选择的架构和模式确定系统由哪些对象构成；
     2. 确定对象属性：明确该对象应该具有的特征属性；
     3. 确定对象行为：明确对象应具有的功能和职责；
     4. 确定对象之间的关系：根据系统顺序图及操作契约以及选择的架构和模式明确系统是如何相互协作完成功能需求的交互过程；
     5. 前提：在确定软件框架结构的基础上

## 模型层次化（基于B/S结构）

* 1. **用户界面层**
     1. 系统功能的各种界面表现形式
     2. 尽量将用户界面层与系统的业务逻辑分离，专门处理系统与用户的交互
  2. **控制器层**
     1. 角色发给对象的请求，由控制器进行处理
     2. 处理不同用例的系统事件的对象，称为控制器
     3. 协调控制其他类共同完成用例规定的功能或者行为
  3. **业务层/应用层**
     1. 尽量参考领域模型中的**概念类**对软件对象进行定义和命名
     2. 【概念类和软件对象存在对应关系，实现用例对应的核心功能】
     3. 业务层主要用来实现用例要求的各种系统级功能
  4. **持久化层**
     1. 对于数据进行存储管理
     2. 降低耦合性
     3. 保存需要持久化存储的数据对象
  5. **系统层**
     1. 将操作系统提供的系统调用封装起来，生成系统访问类
     2. 使软件与操作系统分离
     3. 当程序需要在不同操作系统平台上进行移植时，只需要修改少数系统类

## 面向对象设计原则

见2

## 设计用例实现方案

* 1. 面向对象设计---主要考虑用例如何实现--**用例实现**设计
  2. 用例实现：在设计模型中描述分层结构中相互协作的软件对象如何实现用例的各个特定场景（用例），包括所有的成功和失败场景
  3. 用例实现的设计方案：UML 的sequence/collaboration diagram 进行绘制。【**顺序图**/协作图】
  4. 面向对象的设计模式
     1. 面向对象设计最关键的活动是正确地给对象分配职责
     2. 对象的职责通过调用对象的方法来实现
     3. **模式**
        1. 是面向对象软件的设计经验，是可重用的设计思想
        2. 模式定义了一组相互协作的类，包括类的职责和类之间的交互方式
        3. 组成：
           1. 名称
           2. 问题：问题域，何时解决
           3. 解决方案：设计的组成部分、组成部分的相互关系、各自的职责和协作方式
           4. 效果：模式应用的效果和使用模式应该权衡的问题
     4. **类职责分配模式**（GRASP）
        1. 设计类的来源：
           1. 领域模型中的概念类
           2. 新增类：负责对象持久化的类、负责通信的类
        2. 设计类的职责类型
           1. 了解型：【自己干自己的事】

对象要了解自己私有的封装数据；

了解相关联的对象；

了解能够派生或者计算的事物。

* + - * 1. 行为型：【自己干自己能干的事】

对象自身要能执行一些行为，如创建一个对象或者进行计算；

对象要能启动其他对象中的动作；

对象要能控制或协调其他对象中的活动。

* + - 1. 职责内聚：职责的内聚（自己只干自己的事）：目的是提高内聚降低耦合，减少不必要的关联关系
    1. 控制器模式（controller）
       1. 把接收和处理系统事件的职责分配给位于控制器层的对象
       2. 设计方案：
          1. 它代表整个系统（系统简单且不复杂），称为**外观（facade）控制器**；
          2. 它代表一个发生系统事件的用例场景，这个类通常命名为**“<用例名>控制器”，**称为用例控制器或者会话控制器。
       3. 在相同的用例场景中使用同一个控制器类处理所有的系统事件；
       4. 使用控制器的指导原则
          1. 外观控制器的应用场景
          2. 用例控制器的应用场景
          3. 控制器只是接收系统事件消息，并没有实现系统操作的职责，系统操作应该委托给领域对象处理
    2. **创建者模式（creator）**
       1. 如果符合下面的一个或者多个条件，则可将创建类A实例的职责分配给类B(B创建A)。
          1. B聚合（aggregate）或包含（contain）对象A；
          2. B记录（record）对象A；
          3. B密切使用对象A；
          4. B拥有创建对象A所需要的初始化数据（B是创建对象A的信息专家）。
       2. 创建者模式体现了低耦合的设计思想，是对迪米特法则的具体运用。
    3. **信息专家模式 (Information Expert)**
       1. 给对象分配职责的通用原则：将职责分配给拥有履行职责所必需信息的类，即信息专家。换言之，对象具有处理自己拥有信息的职责或能力。
       2. 根据信息专家模式，应该找到拥有履行职责所必须的信息的类，
       3. **选取类的方法**：
          1. 如果在设计模型中存在相关的类，先到设计模型中查看；
          2. 如果在设计模型中不存在相关的类，则到领域模型中查看，试着应用或扩展领域模型，得出相应的设计类。
       4. 职责的实现（即功能）需要信息，而信息往往分布在不同的对象中，**一个任务可能需要多个对象（信息专家）协作来完成。**