

# Lecture 4：软件设计与实现

## 模块结构以及依赖设计

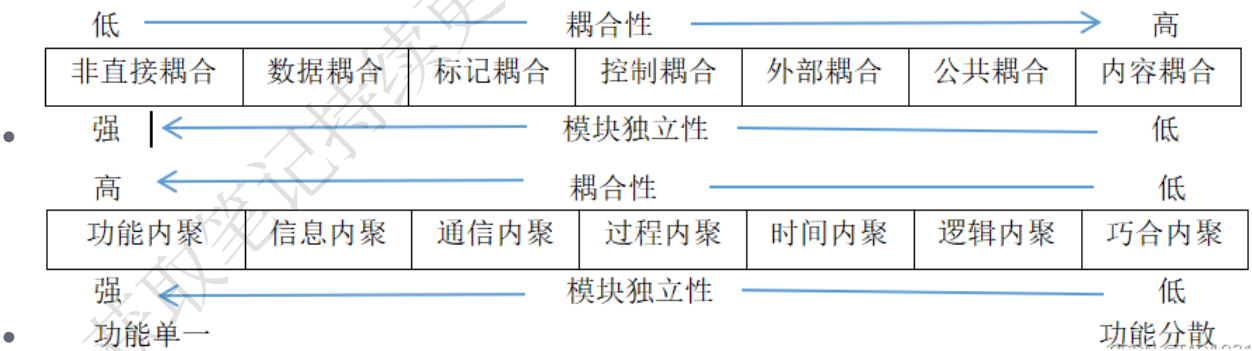
概念：模块结构以及依赖设计

- 概念：什么是模块
  - 独立命名具有输入输出和特性的程序实体；
- 概念：设计原则
  - 软件模块化；
  - 抽象；
  - 信息屏蔽；
  - 模块独立性；

概念：模块独立性的衡量指标

- 概念：
  - 耦合：是模块之间的互相连接的紧密程度的度量。模块之间的连接越紧密联系越多，耦合性就越高，而其模块独立性越弱；
  - 内聚：是模块功能强度，一个模块内部各个元素彼此结合的紧密程度的度量。一个模块内部各个元素之间的联系越紧密，则它的内聚性就越高，相对地，它与其他模块之间的耦合性就越低，而模块独立性就越强；
  - 模块独立性比较强的模块应是高内聚低耦合的模块

图示：



## 具体设计

### 顺序图

概念：顺序图

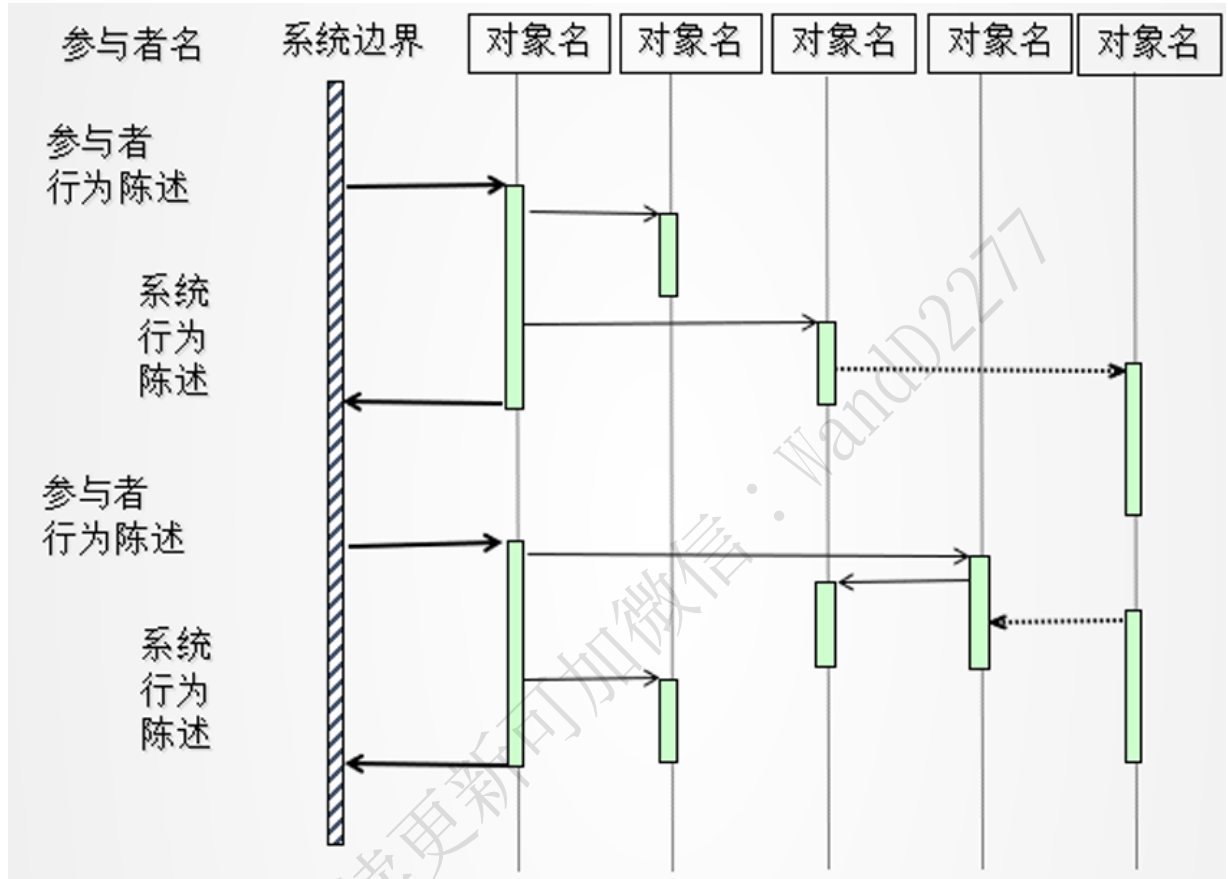
- 概念：
  - 是一种详细表示对象之间以及对象与参与者实例之间交互的图，它由一组协作的对象（或参与者实例）以及它们之间可发送的消息组成，强调消息之间的顺序；
- 概念：对象生命线
  - 把对象的生命周期表示成称之为“生命线”的垂直虚线；

- 生命线代表一个对象在特定时间内的存在。对象不一定位于水平的一排，应该错落有致。根据对象的创建时间；

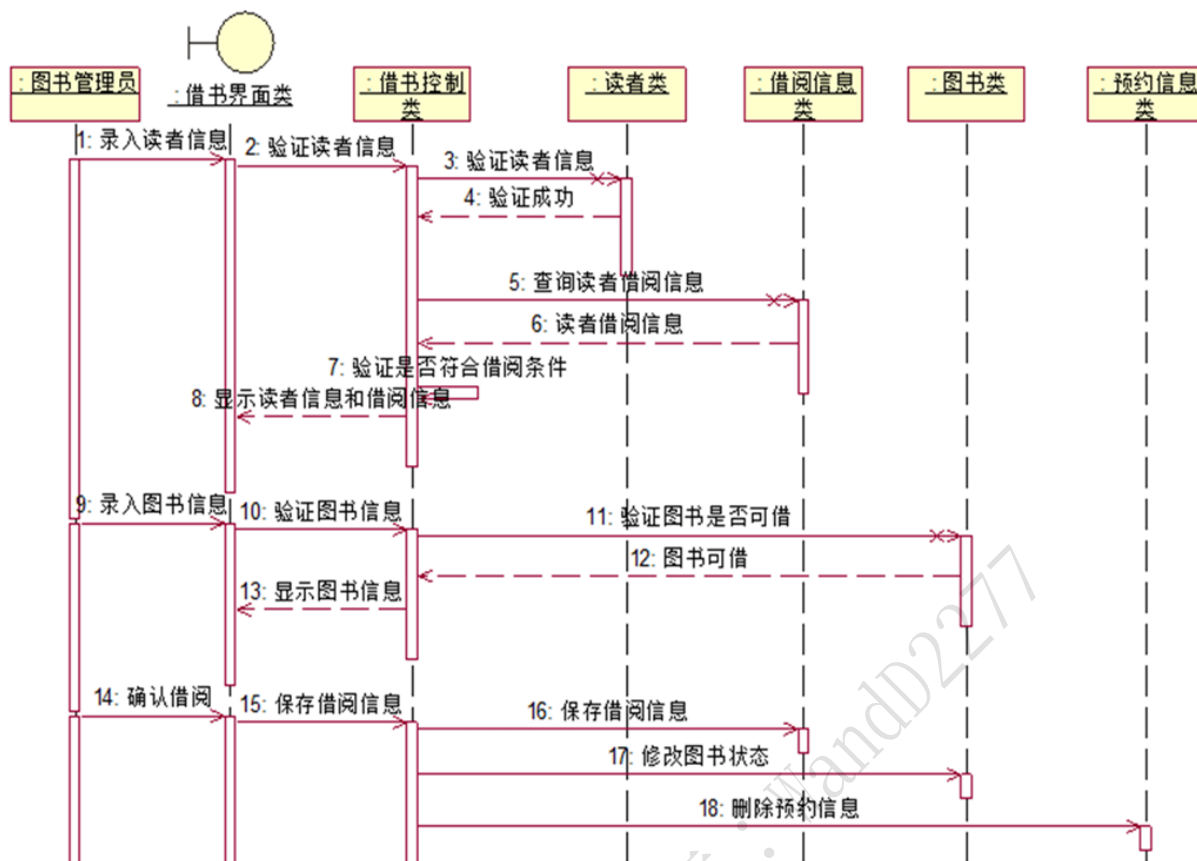
- 概念：消息

- 消息是对象之间的通讯的规格说明，类之间通过发送和接收消息进行通信。这样的通讯用于传输将发生的活动所需要的信息。它即包含了控制信息（如调用），也包含了所使用的数据的规格说明；

- 结构：



- 图示：



## 类图设计

### 概念：类图设计

#### 概念：UML

- 统一建模语言（英语：Unified Modeling Language，缩写 UML）是非专利的第三代建模和规约语言；
- UML 是一种开放的方法，用于说明、可视化、构建和编写一个正在开发的、面向对象的、软件密集系统的制品的开放方法；
- 区分 UML 模型和 UML 图是非常重要的，UML 图，包括用例图、协作图、活动图、序列图、部署图、构件图、类图、状态图，是模型中信息的图形表达方式，但是 UML 模型独立于 UML 图存在；

#### 概念：类图

- 类图是描述类、协作（类或对象间的协作）、接口及其关系的图；
- 类图是逻辑视图的重要组成部分，用于对系统的静态结构建模，涉及到具体的实现细节；

#### 步骤：

- （1）发现领域对象，由对象抽象得到概念类；
- （2）确定了系统的所有用例之后，就可以开始识别目标系统中的对象和类了。把具有相似属性和操作的对象定义为一个类。目标系统的类可以划分为边界类、控制类和实体类；