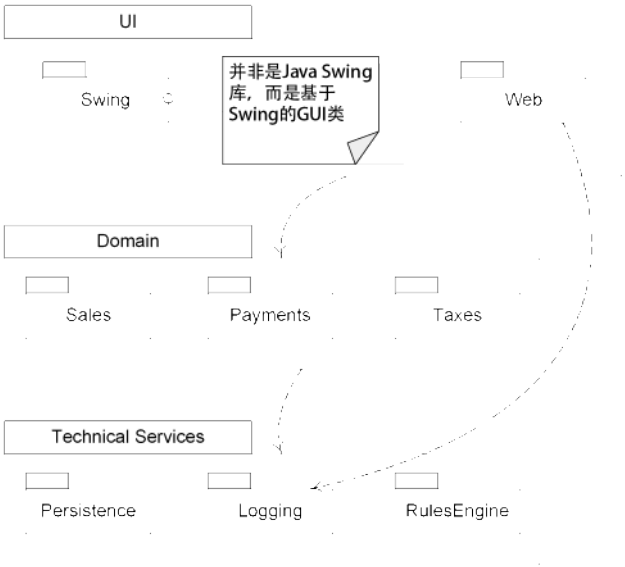
OOAD第七章

1. 逻辑架构和层

   逻辑架构（logical architecture）是软件类的宏观组织结构，它将软件类组织为包（或命名空间）、子系统和层等

   层（layer）是较为粗粒度的对类、包或子系统分组，具有对系统主要方面加以内聚的职责



   OO系统中通常包括的层有：

   用户界面（UI）

   应用逻辑和领域对象（domain）

   技术服务（technical services）

   严格的分层架构中，层只能调用与其相邻的下层的服务

   宽松的分层架构中，较高层可以调用其下任何层的服务

2. 软件架构

   是一个系统的草图

   描述的对象是直接构成系统的抽象组件

   各个组件之间的连接则明确和相对细致地描述组件之间的通信

   在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象

   在面向对象领域中，组件之间的连接通常用接口来实现

   使用层进行设计——减少耦合和依赖性、增强内聚性、提高潜在的复用性且使概念更清晰

   将系统的大型逻辑结构组织为独立的、职责相关的离散层，低层高层各司其职

   协作和耦合是从较高层到较低层进行的，要避免从较低层到较高层的耦合

   内聚职责；使关系分离

   同一层内的对象在职责上应该具有紧密关联，不同层中对象的职责则不应该混淆

   领域层和领域模型之间的关系

领域层是软件的一部分，领域模型是概念角度分析的一部分，它们是不同的。
利用来自领域模型的灵感创建领域层，可以获得在实现世界和软件设计之间的低表示差异

层和分区
架构中的层表示对系统的垂直方向的划分。
分区 - 表示对层在水平方向进行划分，形成相对平行的子系统。例如，技术服务层可以划分为安全和统计等分区。

不要将外部资源表示为最底层
外部资源（如某个数据库）表示为“低于”（例如）基础层（Foundation）的层，是对逻辑视图和架构部署视图的混淆
对某个持久数据集合（例如库存数据）的访问可以视为领域层中的子领域—库存子领域。
提供数据库访问的一般性服务则可以视为技术服务分区（Partition）——持久性服务

模型对象不应该直接与视图对象连接，对于视图对象也是如此
模型←→领域层对象（如Sale、Payment）
视图←→UI对象（如：窗口、web页、applet和reports报表）

3. UML交互图——描述对象间通过消息的交互
   包括：顺序图（sequence diagram）和通信图（communication diagram）