Object Oriented Analysis & Design
面向对象分析与设计

Lecture_06 从分析到设计

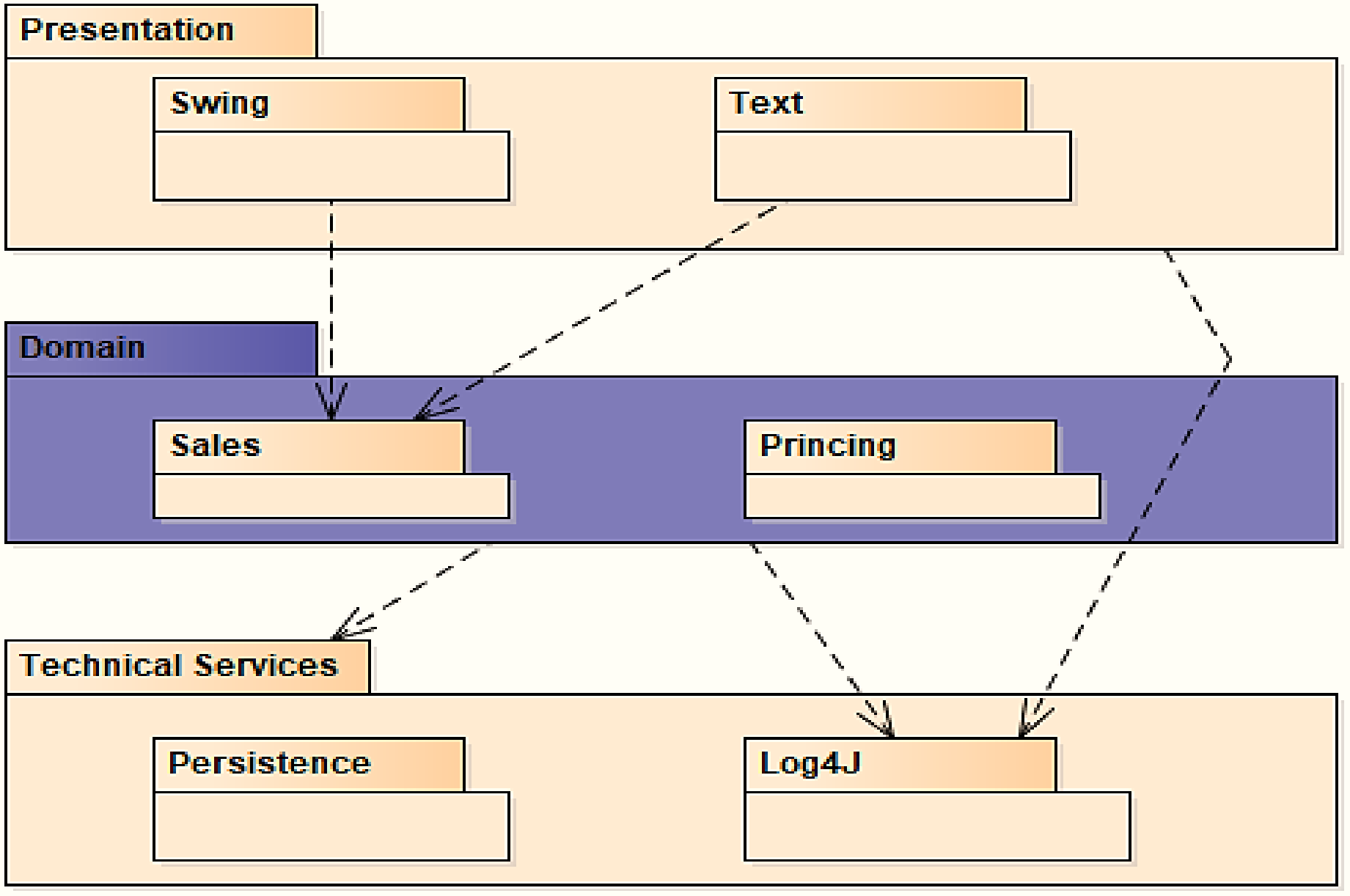
主讲: 姜宁康 博士

6、面向对象设计 … Object-oriented Design

- 面向对象设计在系统开发中的位置
- Contrast the importance of object design skill versus UML notation knowledge

6.1 面向对象设计的关注点

- 面向对象设计
  - 主要工作在领域相关domain
  - 也称为应用逻辑、业务逻辑
- 面向对象设计不直接关注
  - 用户界面
  - 数据储存



6.1 面向对象设计的关注点

- 面向对象设计
  - 领域层与 UI 层、数据层通过特定的接口，进行通信
- 例如，Model-View 划分原则
  - View与领域层有不同的关注点
    - 例如
    - GUI 不应该计算税费(这是 domain层应该做的)，但是可以对数据进行验证
    - Domain 对象不应当去探知外部事件的发生，但应当被通知到
- 例如，观察者模式 Observer pattern
    - GUI 对象通过专门的接口，到 Domain对象中注册
    - Domain 对象并不知道是谁来注册的，仅仅知道通信接口

6.2 领域对象与设计对象的一致性

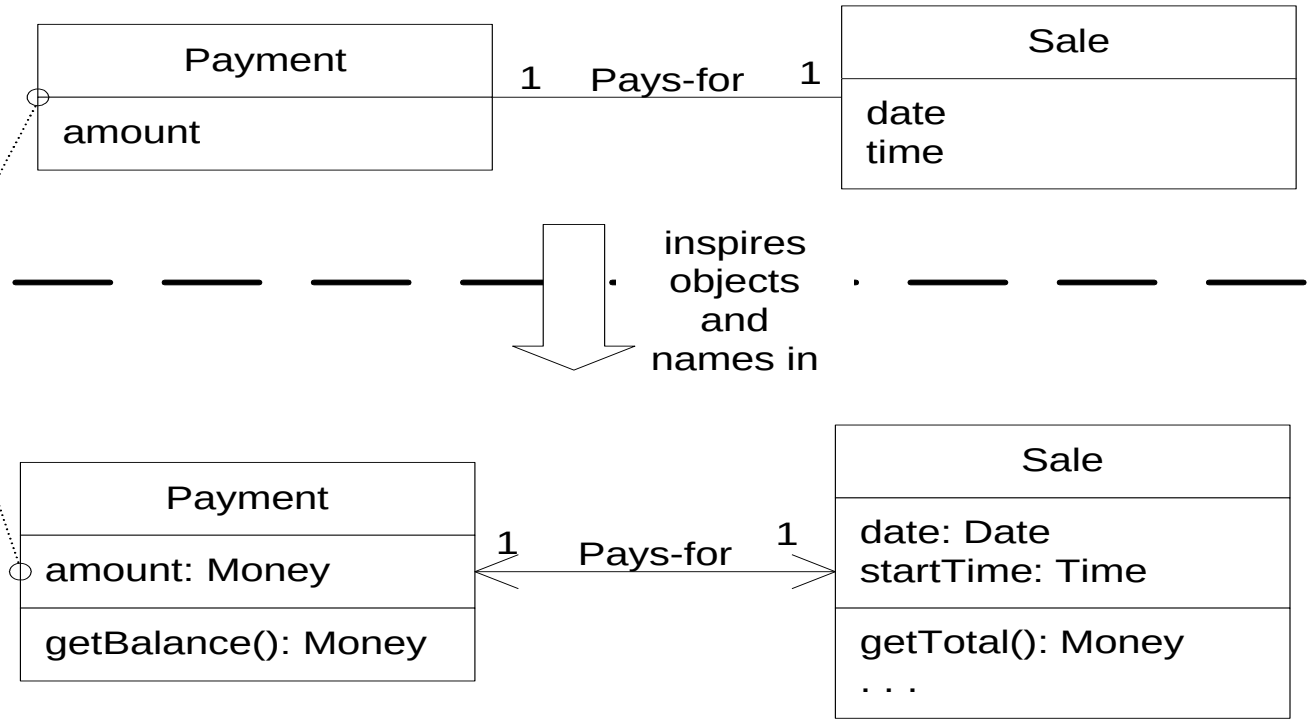
- 领域对象对应真实世界的对象 Domain objects should correspond with real-world objects
- 设计活动从领域模型开始

领域模型中的 Payment 是一个概念，在设计模型中是一个软件类。它们不是同一件事，但前者启发了后者的命名/定义

这样就降低了表示差异
这是面向对象技术的一个重要思路

UP Domain Model

Stakeholder's view of the noteworthy concepts in the domain.



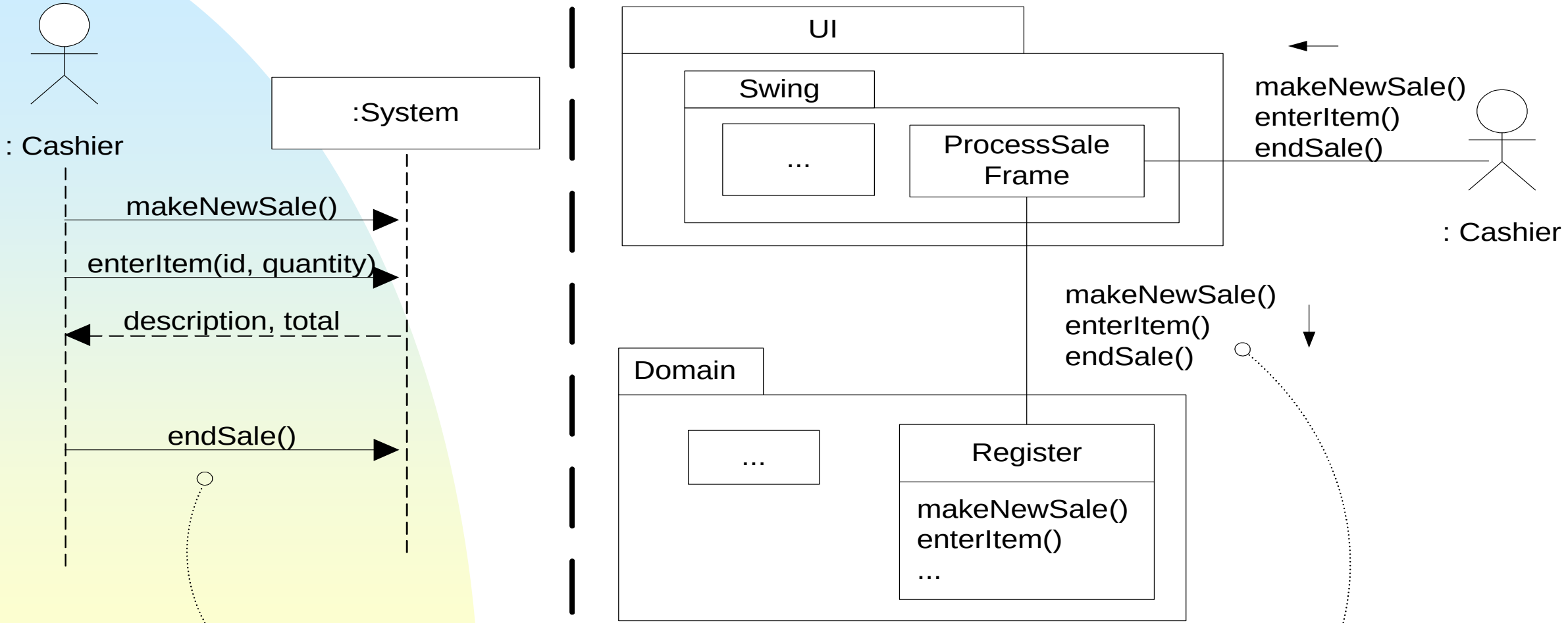
Domain layer of the architecture in the UP Design Model

The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

6.3 从SSD到设计

- GUI产生、或者检测到系统消息，发送到领域层的某个对象（术语：控制器，Controller）



SSD上的每个系统操作，都将有系统进行处理。它们表示UI层对应用层或者领域层的调用

6.4 设计思想的来源

- 模式 Pattern (及其变种...)
  - 软件设计最新的概念 Important concept in software design today
    - 同学们要经常去关注，比如 www.csdn.net，一些开源网站等
  - 研习针对大规模/小规模问题的“最佳实践”解决方案
  - 教材、论文记录的一些方法
  - 在不同上下文中重用——不要从头开始设计解决方案
- 设计，还需要一点点灵感！

6.5 大规模系统设计遇到的问题

- 如何定义领域层对象与子系统之间的协作 How to structure collaborations between domain objects and subsystems
  - e.g 永久存储子系统 the Persistence subsystem
- 如何定义UI对象到领域层对象之间的协作？ How to structure collaborations between the UI objects and the domain layer objects
- 如何设计与实现“向上”的协作？ How to design and implement “upward” collaborations
  - e.g 领域层对象到 UI 对象

6.6 Responsibility-Driven Design (RDD)

- 设计的总体思路
  - 标识职责 responsibilities，并把它们分配给不同的类
- 职责
  - 行为职责 Doing
    - create an object, perform calculations
    - initiate operations on other objects
    - control and coordinate activities
    - ...
  - 认知职责 knowing
    - about private encapsulated data
    - about related objects
    - about things it can derive or calculate
    - ...

6.6 Responsibility-Driven Design (RDD)

- 职责描述是一种抽象，粒度大小不一
  - 软件对象只有方法 methods，没有职责
  - 从职责到对象方法的转换
    - 比如，“负责永久存储”，粒度太大
    - 比如“负责计算税费”，粒度要好一些

- 职责描述是一种很好的隐喻
  - 职责驱动的设计 RDD，对象设计时可以问这样的问题
    - 这个对象有哪些职责?
    - 这个对象与哪个对象协作?

6.7 对象设计的基本原则 GRASP 原则

- 该如何分配职责呢? What are the guiding principles to help assign responsibilities
- 有没有什么原则可以给予指导?
- 它们是
  - 通用的职责分配软件原则(模式) GRASP General Responsibility Assignment Software Principles (patterns)
    - 这是一组非常基本和通用的对象设计原则

6.7 对象设计的基本原则 GRASP 原则

- The 9 GRASP Principles
  - Creator 创建者
  - Information Expert 信息专家
  - Controller 控制器
  - Low Coupling 低耦合
  - High Cohesion 高内聚
  - Polymorphism 多态
  - Pure Fabrication 纯虚构
  - Indirection 间接
  - Protected Variations 隔离变化

记住每条原则的名字、含义、如何使用，它们是本课程最重要、最有用的内容

6.7 案例

- enterItem(itemID: ItemID, quantity: Integer)
- Cross References: Use cases: Process Sale
- Postconditions:
  - A SalesLineItem instance sli was created
  - sli was associated with the current Sale
  - sli.quantity became quantity
  - sli was associated with a ProductSpecification based on itemID

- 问题：
  - 由谁来创建 SalesLineItem 实例呢?

6.7 案例

- 答案
  - 应用“创建者Creator”原则
  - What object should create an X ?
    - Ignores special-case patterns such as Factory
  - Choose an object C, such that :
    - C contains X
    - C closely uses X
    - C has the initializing data for X
  - The more, the better
- 因此，应该有“Sale”对象来创建 SalesLineItem 对象

小结

- 大型软件系统的设计
  - 软件架构
  - 分层、子系统
- 面向对象设计主要关注领域层对象、与 UI 层、数据存储层之间有接口
- 领域层需要完成的功能，以职责来描述
- 面向对象设计的主要任务: 把职责分配类和对象
- 如何分配呢?
  - GRASP是很好的指导原则
- 让我们期待课程的下一阶段
  - 面向对象设计 (OOD, Object-Oriented Design)

- 本讲结束