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

Lecture_07 通用的职责分配软件原则 GRASP

主讲: 姜宁康 博士

4、GRASP原则四：控制器 Controller

- What first object beyond the UI layer receives and co-ordinates (controls) a system operation?

  在领域层，由谁负责首先接收并协调来自UI层的系统操作?

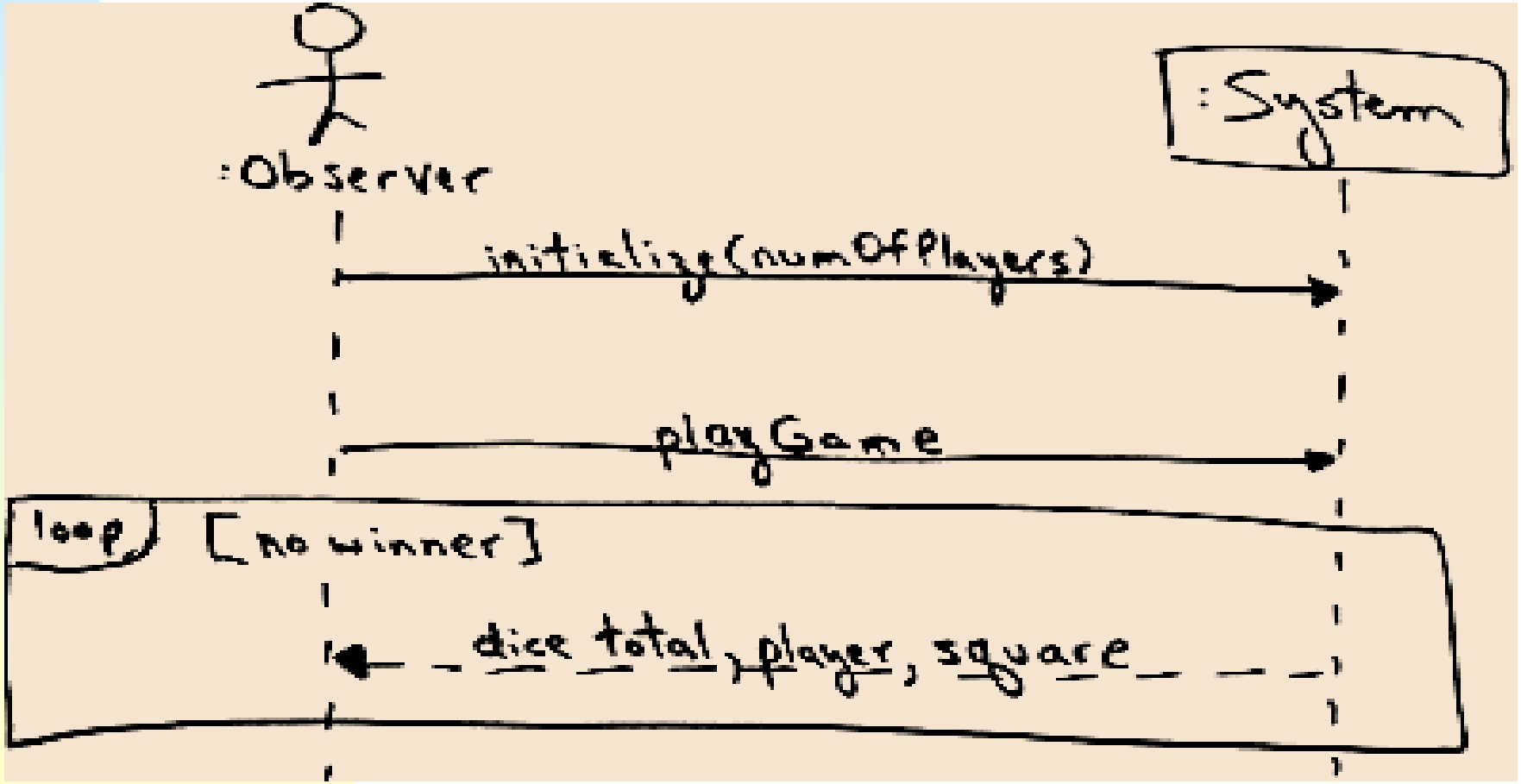
4.1 Mini Exercise 4

- For Monopoly game, Which object starts the game?

- Understanding the Problem
  - SSD — boundary between the User and SUD (system under development)
  - UI layer "catches" the request
  - The request is a system operation — public interface
  - Model-View Separation principle says UI must not contain business logic
  - Problem: UI应该把捕捉到的系统操作，发给领域层的哪个对象呢?

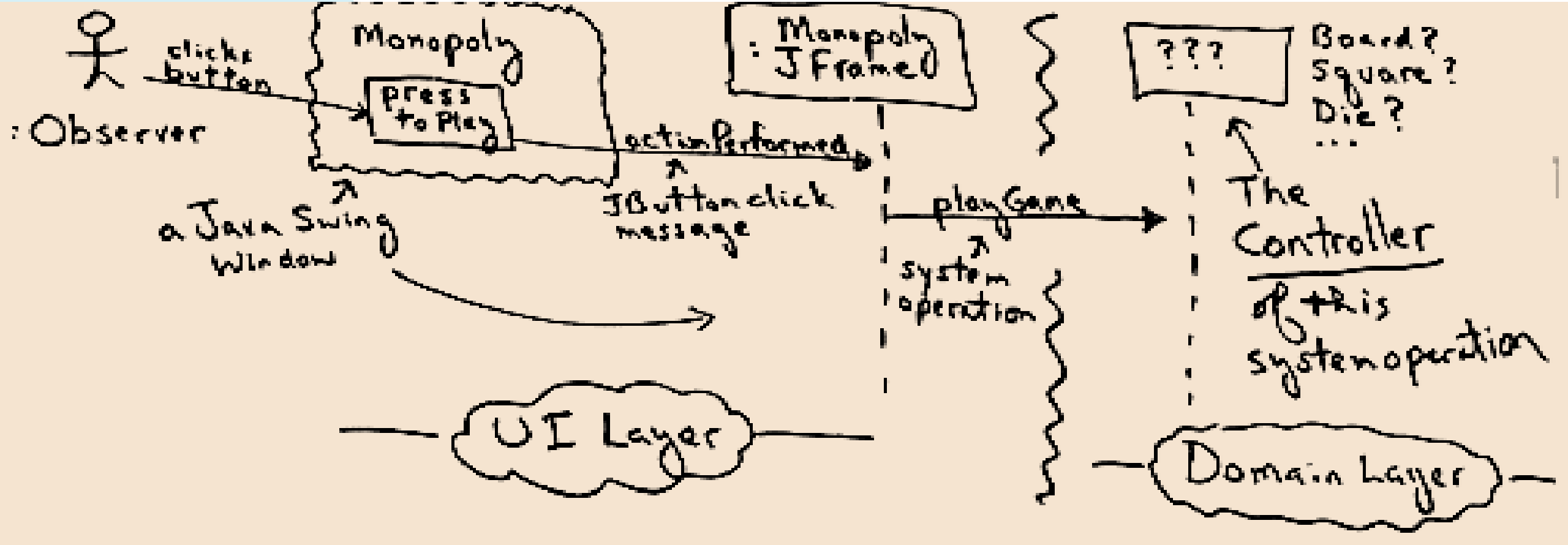
4.1 Mini Exercise 4

- **SSD for the Monopoly game**
  - **Note the playGame operation**



4.1 Mini Exercise 4

- **Assign responsibility to receive the system operations by an object which**
  - **Representing System/subsystem/device or**
  - **Handling just this use case**

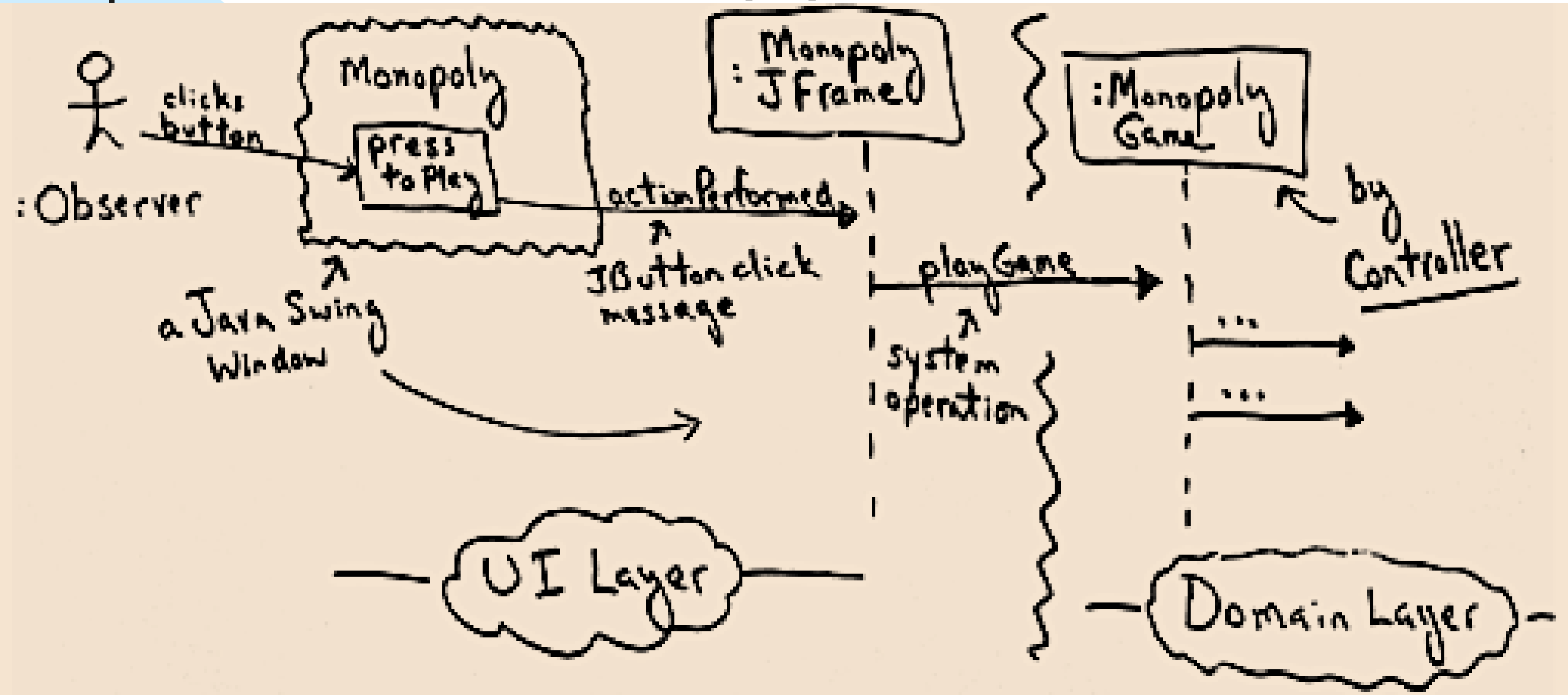


4.2 GRASP rule4: Controller

- **Name: Controller 控制器**
- **Problem:**
  - **What first object beyond the UI layer receives and co-ordinates (controls) a system operation**
- **Solution:**
  - **Assign the responsibility to a class representing one of the following choices:**
  - **1. Facade(外观) Controller:**
    - **represents the overall system, a root object, a device that the object is running within, or a major sub-system**
  - **2. Use Case or Session Controller（用例控制器、会话控制器）:**
    - **represents a use case scenario within which the system event occurs**

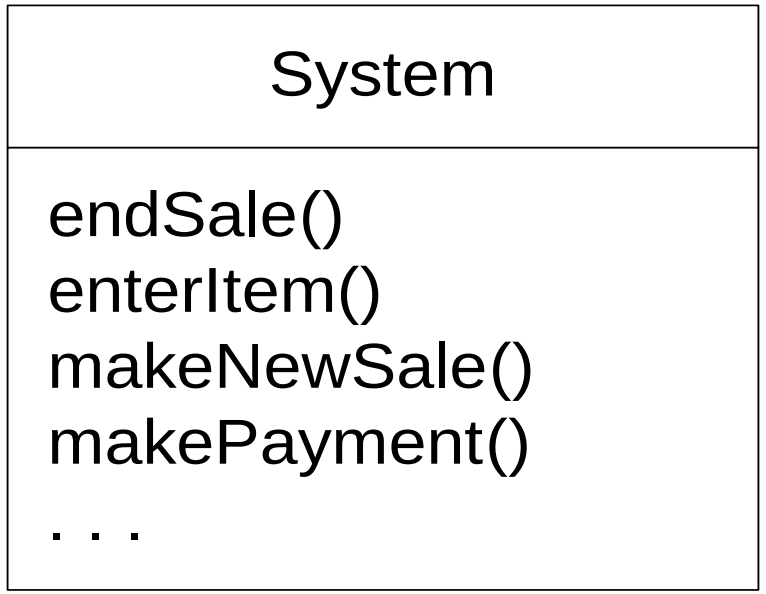
4.3 Mini Exercise 4 — Solution

- 能够代表整个系统的对象：MonopolyGame



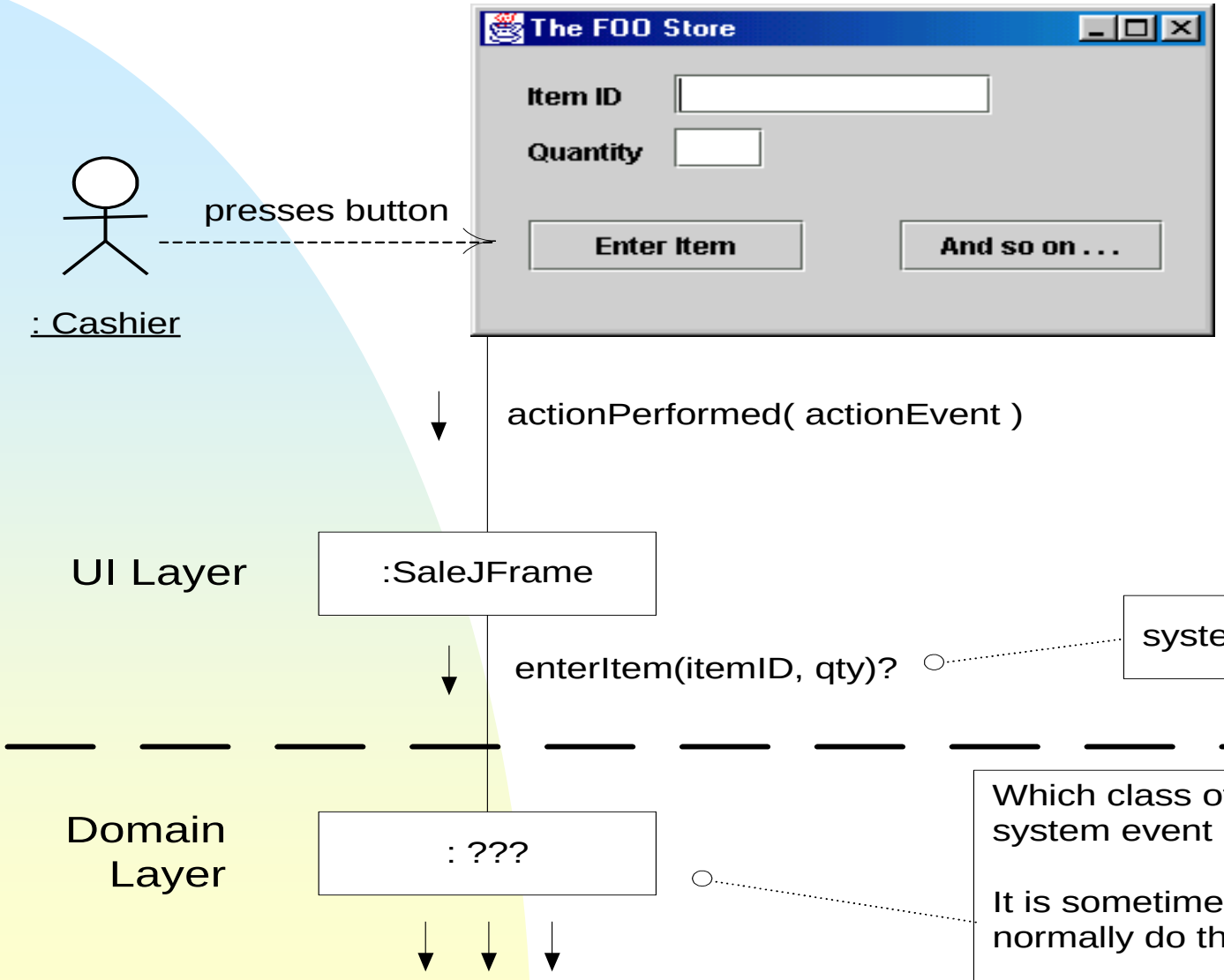
4.4 Controller — NextGen POS Example

- **Some system operations of the NextGen POS application**



- **Conceptual representation — what is the class that handles these operations?**
  - **during design, a controller class is assigned the responsibility for system operations**

4.4 Controller — NextGen POS Example

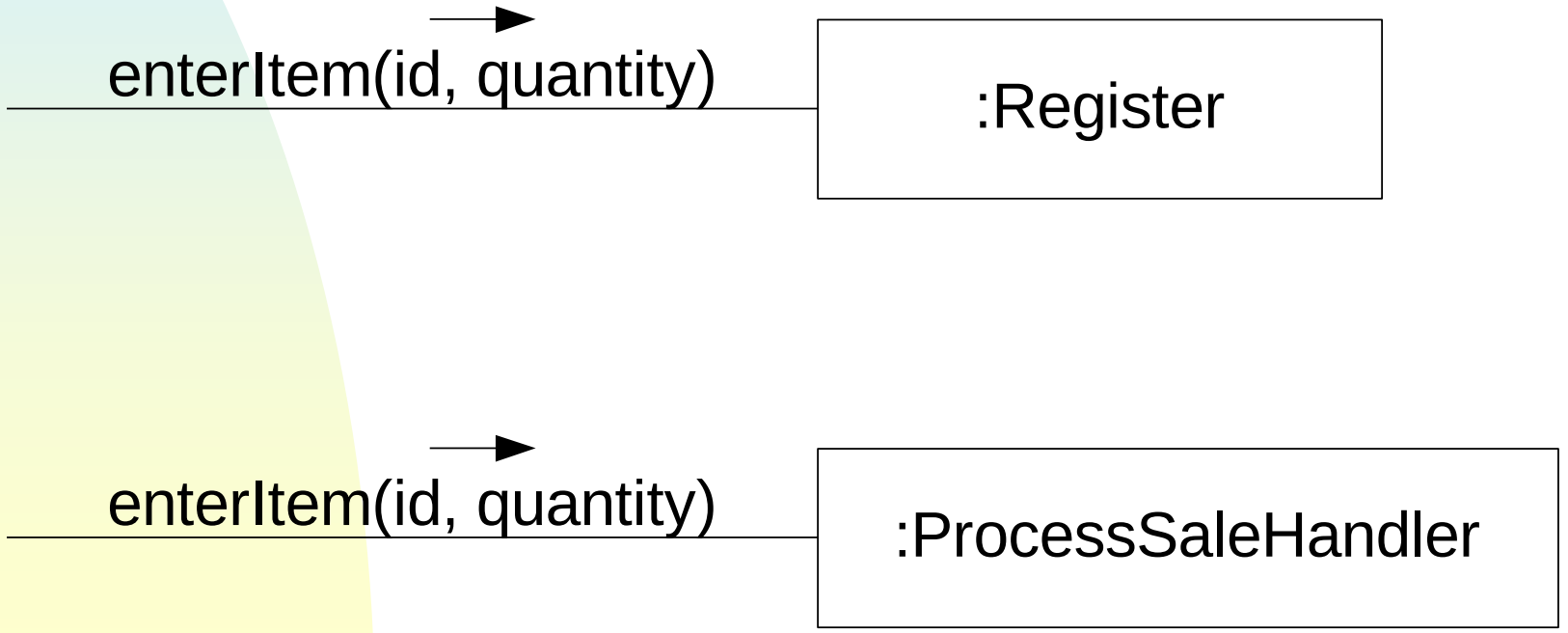


- **What object should be the Controller for enterItem?**

4.4 Controller — NextGen POS Example

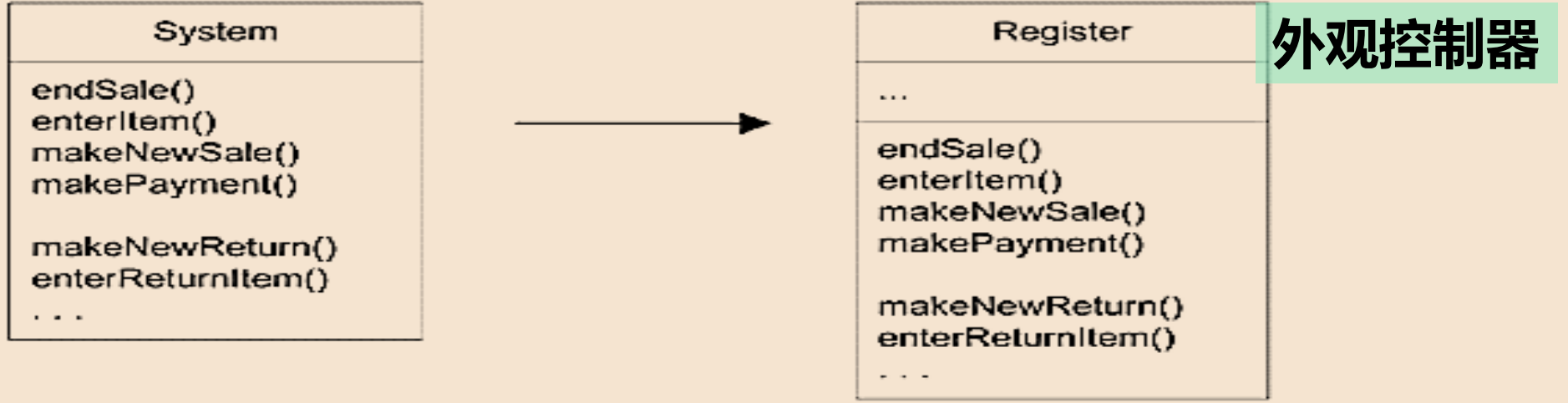
- (Solution) 2种方案
  - 外观控制器 Facade: Register, POSSystem
  - 会话控制器Session: ProcessSaleHandler, ProcessSaleSession

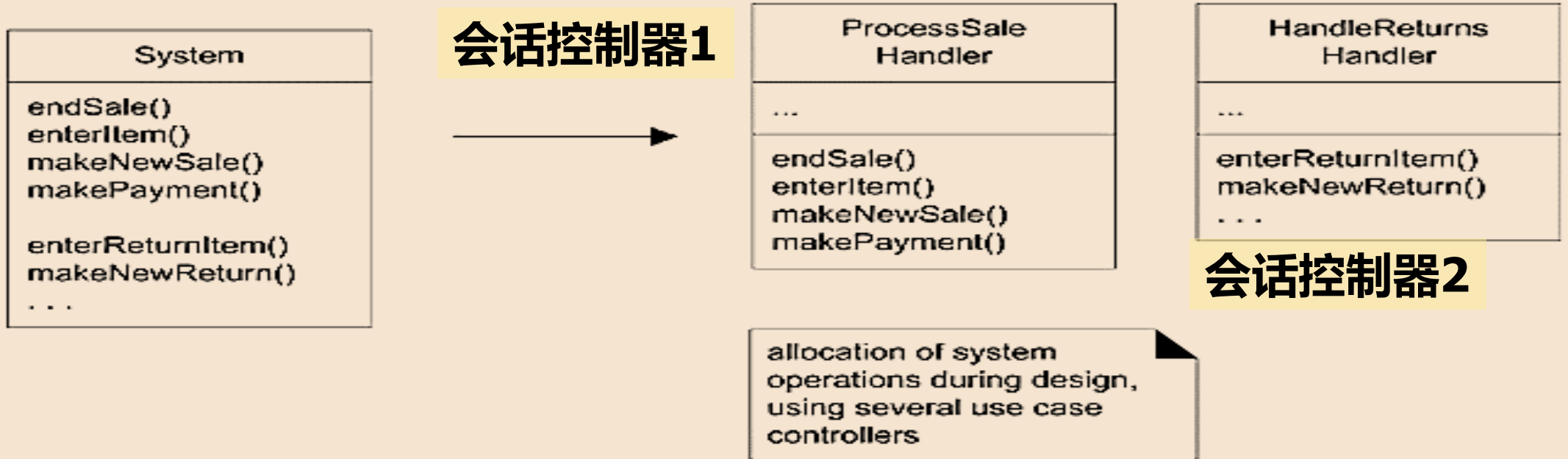
Controller choices



4.4 Controller

- **Allocation of system operations**





4.5 Controller — Observations

- 委托模式 Delegation pattern
  - 外部输入事件可以来自参与者（人）或者（其他系统）
  - Facade — 相当于领域层对外部世界的“脸”
    - Ex: Register
  - Handler —处理系统某个明确的功能集，比如相关的一组系统事件
    - Ex: ProcessSale

4.5 Controller — Facade

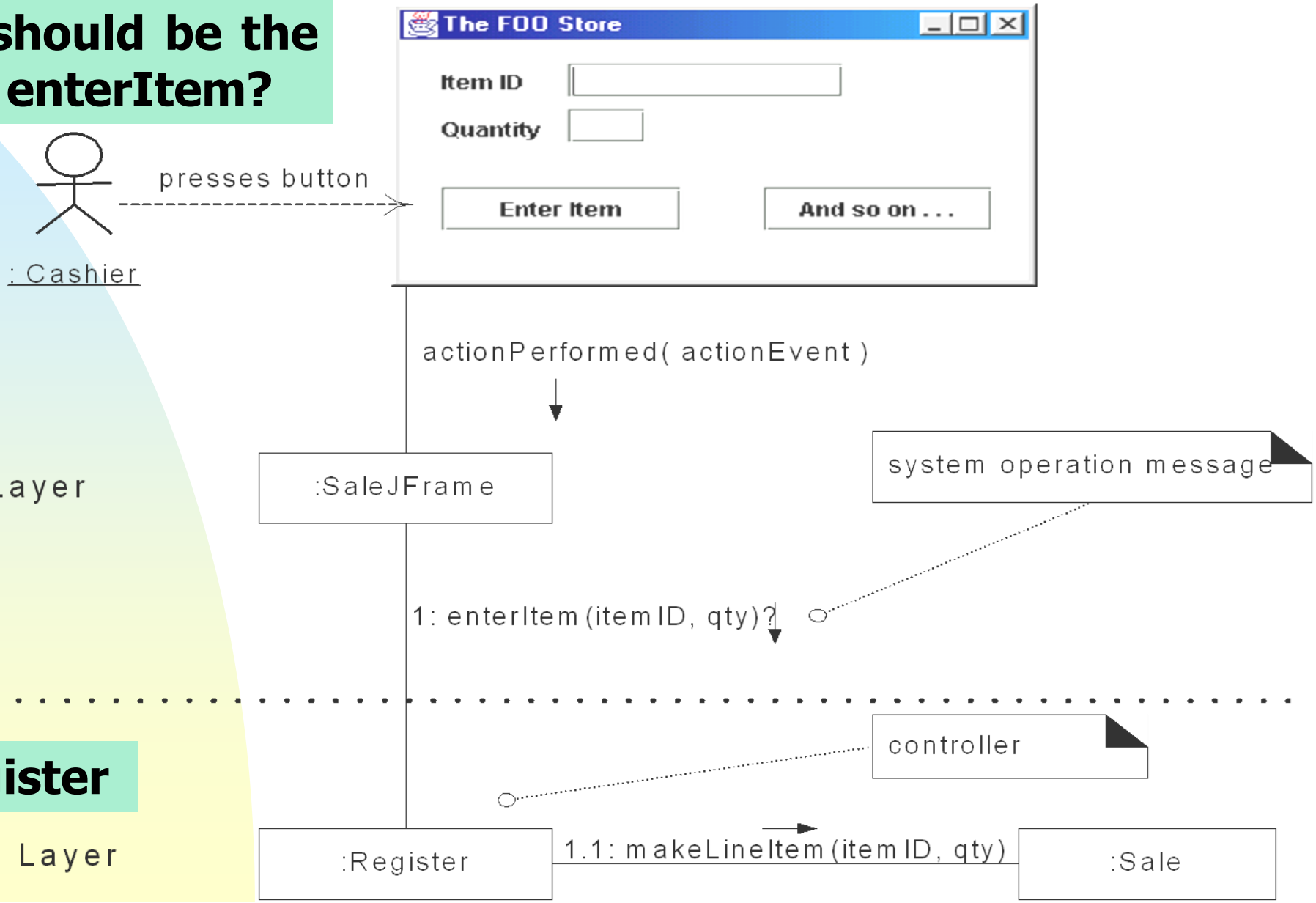
- 外观控制器 Facade
  - 为子系统中的一组接口提供一个一致的界面 Provide a consistent interface for a set of interfaces in subsystem “cover” over the other layers of the application
  - Abstraction of an overall physical unit
    - — Register, PizzaShop
  - The entire software system
    - — POSSystem
  - Abstraction of some other overall system or sub-system concept
    - — MonopolyGame
- 适用于
  - 相对较小的系统 relatively small systems
  - 有限数量的系统操作 and/or system with limited number of system operations
  - 在消息处理系统中，不能转发消息到可选的控制器时 in message handling system when can’t direct messages to alternative controllers
    - — Internet application servers

4.5 Controller -- Session

- 会话控制器
  - 一种纯虚构出来的概念 Pure Fabrication，即领域模型中没有的概念
  - 如， ProcessSaleHandler is not a domain concept in Domain Model
- 会话控制器的应用场合
  - 当采用外观控制器会导致高耦合、低内聚时 When assigning to facade may lead to high coupling or low cohesion (“Bloat”)
  - 很多系统事件跨越多个不同的处理过程 Many system operations across different processes
  - 概念上容易理解和构建 Conceptually easier to understand and build
    - 一个会话控制器负责一类系统事件
- 会话控制器的命名习惯 Session Controllers Naming conventions
    - <UseCaseName> Handler or
    - <UseCaseName> CoOrdinator or
    - <UseCaseName>Session
  - Use same controller class for all system operations in the use case scenario
  - Session is a type of conversation between the actor and the SUD 新系统

4.6 Register Controller: NextGen POS

What object should be the Controller for enterItem?



Answer: Register

4.7 Discuss: Controller vs. UI

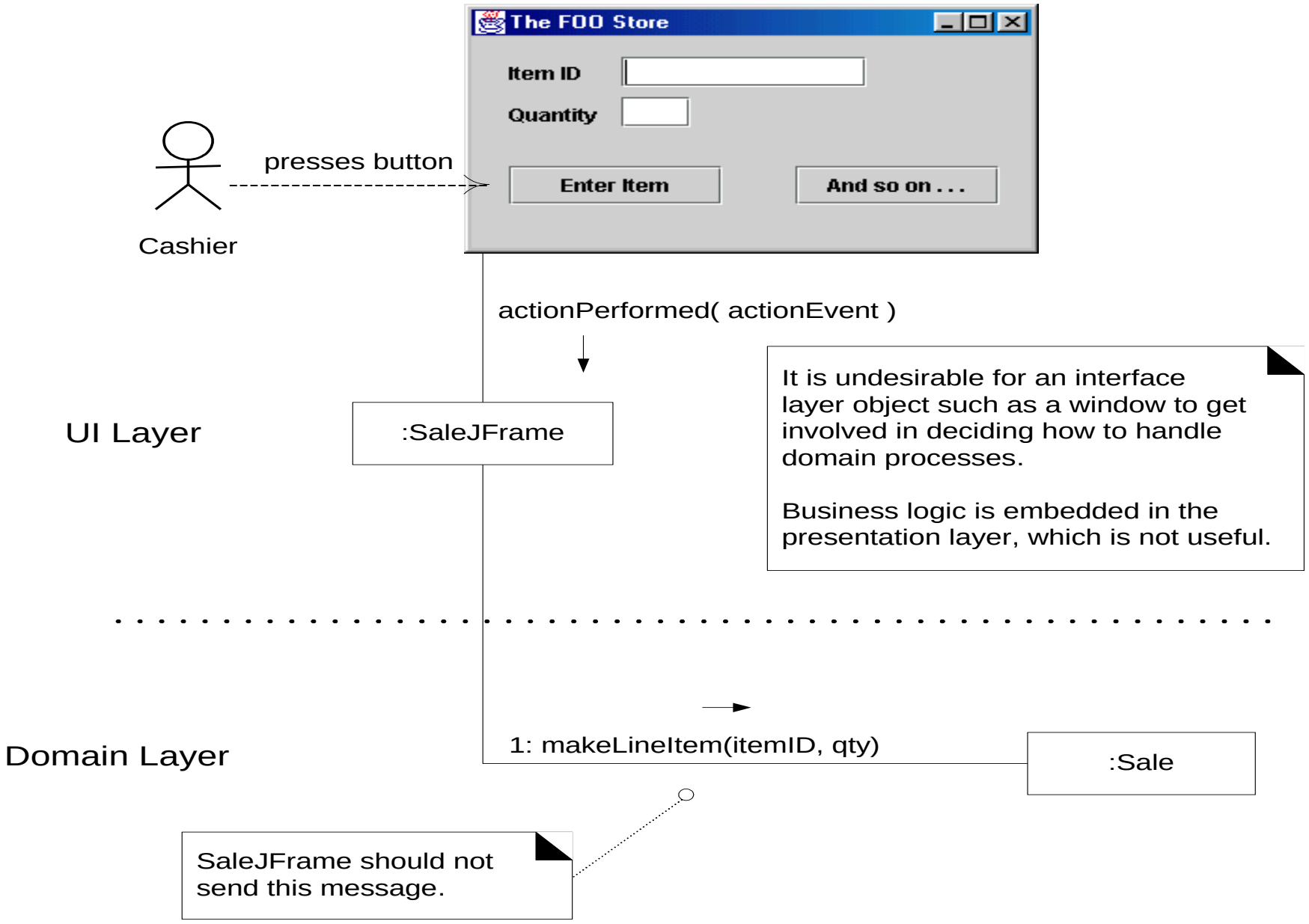
1）UI层不应负责处理系统操作

2）系统操作一定在领域层进行处理

本书把应用逻辑层又称为架构的领域层

3）控制器负责委托（转发）消息

4）右图，较差的设计方案：业务逻辑嵌入到展示层



4.7 Discuss: Controller — Benefits

- 控制器模式的优点
  - 容易适应UI层的变化 Allows for easy change of UI and/or alternative UI
  - 领域层代码易于重用（因为UI层一般与应用关系密切）Allows for reuse of domain layer code (UI is usually application specific)
  - 有助于保证应用所需要的操作顺序 Helps ensure sequence of operation which may differ from UI input
  - 可以对系统的状态进行推理（UI层不保存系统状态）Can reason about system state — UI does not preserve state

4.8 Bloated(臃肿的) Controllers

- 臃肿控制器的问题
  - 当一个外观控制器处理了大部分系统事件时 When have a facade controller handling all of many system events
  - 当一个控制器做了太多的事情，而不是委托给其他的对象去处理 When the controller performs many of the system operations instead of delegating
  - 当控制器掌握了太多的系统信息 When the controller has many attributes (much information) about the system
    - which should be distributed to or duplicates from elsewhere
  - 导致：低内聚 Low cohesion — 做事不专注，做了太多的事
- 臃肿控制器的解决方法
  - 增加更多的控制器 Add more controllers
  - 采用会话控制器替换外观控制器 “session controller” instead of facade
  - 控制器委托任务给别的对象，而不是自己做 Design the controller so that it delegates operations to other objects
  - 高内聚的理念 High Cohesion is itself a GRASP principle

- 本讲结束