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

3、GRASP原则三: 低耦合 Low Coupling

- How to support low dependency, low change impact and increased reuse?
- 如何保证设计方案支持低的依赖性、低的变化影响度、增加可重用性?

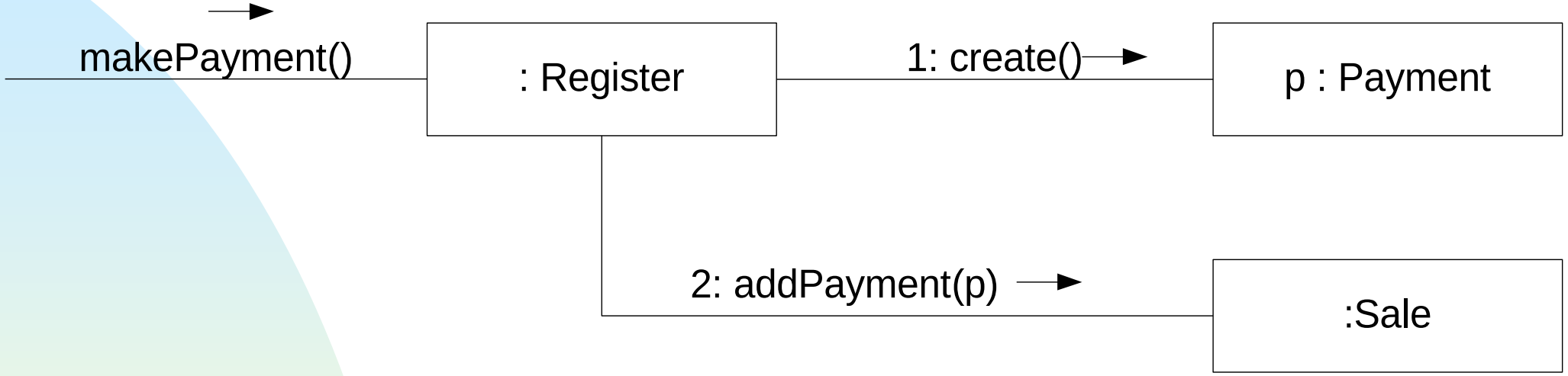
3.1 Mini-Exercise 3

- **Who creates the Payment?**

- **makePayment method invoked in Register**
- **A Payment has to be associated with the Sale**
- **Who creates the Payment instance?**
  - **By creator → Register, Sale are candidates**

3.1 Mini-Exercise 3

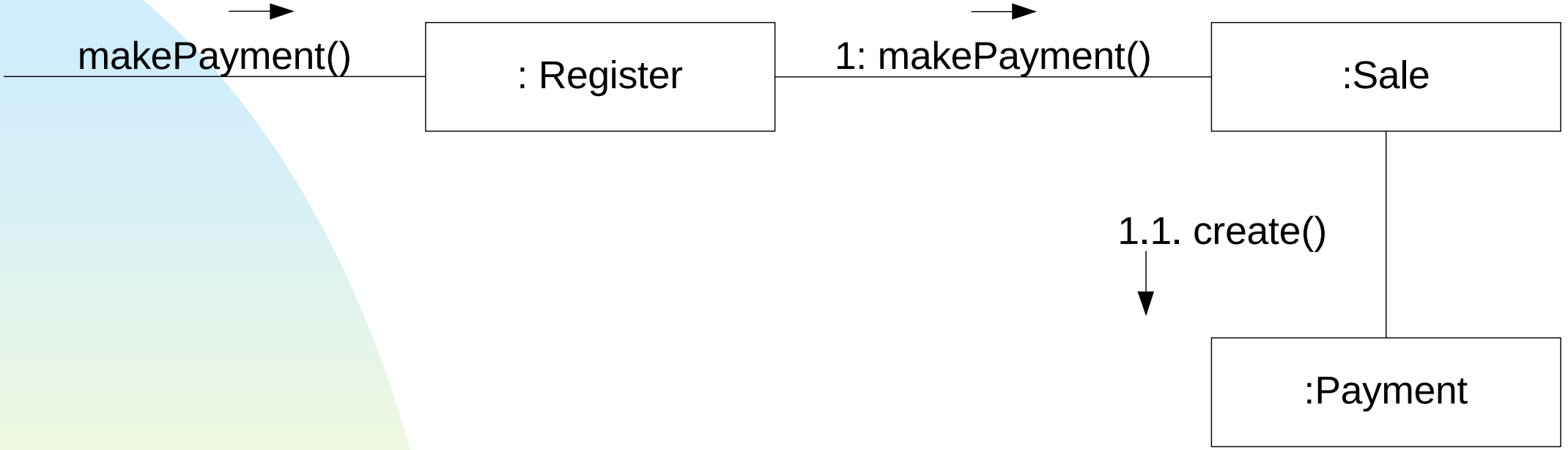
- **Solution 1**



- **Solution 1 点评**
  - **Register records Payment**
    - **Creator pattern**
  - **addPayment method passes p instance of Payment as parameter**
  - **Register coupled to Payment class**

3.1 Mini-Exercise 3

- **Solution 2**



- **Solution 2 点评**
  - **Register delegate Payment responsibility to Sale**
  - **Sale create instance of Payment**
    - **By Creator**
  - **Register has no relation with Payment**

3.2 耦合的定义Coupling

- 耦合：一个元素与其它元素的联接、感知以及依赖程度的度量 Measure of how strongly one element is: connected to、has knowledge of、relies on another element
- 比较
  - Cohesion，内聚: 模块内的操作之间联系紧密的程度
  - Coupling，耦合: 两个子模块之间联系的强度
- 高耦合带来的问题 Problems with High Coupling
  - “牵一发动全身” Forced local changes because of changes in related class
    - A依赖B，A与B之间有耦合，一旦B变化了，A就会受影响
  - 元素孤立是无法理解 Harder to understand in isolation
  - 元素很难重用 Harder to reuse — drags in more classes

3.3 GRASP rule3： Low Coupling

- **Name: Low Coupling**
- **Problem:**
  - **How to support low dependency, low change impact and increased reuse?**
- **Solution:**
  - **Assign responsibility so coupling remains low.**
    - **Use this principle to evaluate alternatives**
    - **All other things being equal prefer the low coupling solution**
- **Note: Information Expert encourages Low Coupling**
- **Why most secret service (spy) is one-way contact? 为什么特务工作都是单线联系?**

3.4 Discuss： Low Coupling

- 低耦合是所有设计决策时自然而然要考虑的原则 Low Coupling is a principle to keep in mind during all design decisions; it is an underlying goal to continually consider
- 低耦合是一种评估原则，是设计师用来对设计方案进行评价的一种指标 It is an evaluative principle that a designer applies while evaluating all design decisions
- X与Y存在耦合的情况，例如
  - X has an attribute that refers to Y
  - X calls on services of Y
  - X has a method that references Y (parameter, local variable, or return value)
  - X is a direct or indirect subclass of Y
  - Y is an interface, and X implements that interface

3.4 Discuss： Low Coupling

- 低耦合支持类的设计相对独立，减少了变化带来的相互影响 Low Coupling supports the design of classes that are more independent, which reduces the impact of change
- 低耦合与其他的原则，如信息专家、高内聚必须综合考虑
- 不能单独考虑低耦合
  - “是药，三分毒”
- 继承关系中，子类与父类的耦合非常紧密 A subclass is strongly coupled to its superclass.
  - 所以， 能用组合的地方不要用继承
- There is no absolute measure of when coupling is too high
  - 低耦合不具备可操作性！类似“只可意会，不可言传”

3.4 Discuss：Low Coupling

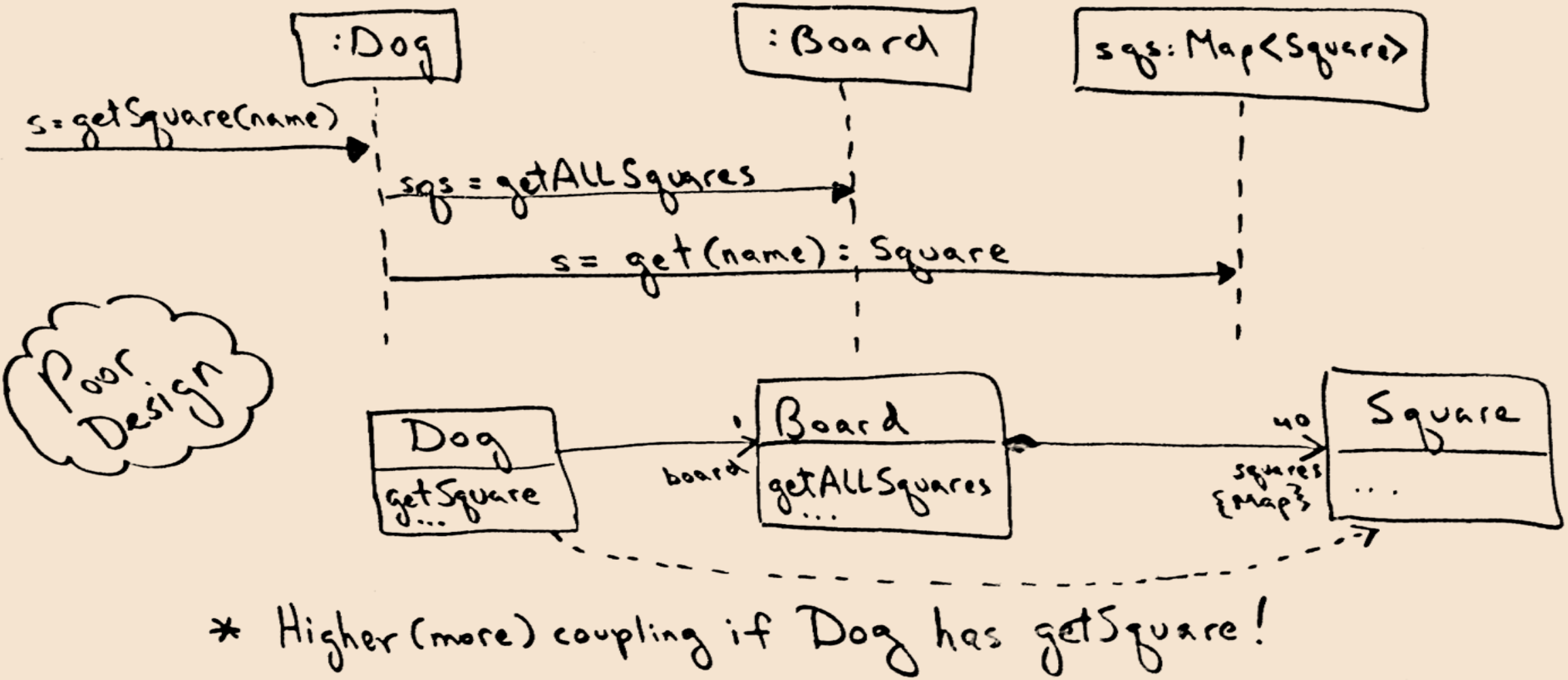
- 极端情况，类之间没有耦合
  - 不希望这种情况出现
    - because a central metaphor of object technology is a system of connected objects that communicate via messages
    - Too little coupling means we aren't a "collaborating community of objects"
  - 这样会形成很差的设计
    - 产生少量的不内聚、臃肿、行为复杂的对象，承担了全部的工作，这些类可能独立工作，成为一个个简单的数据仓库
- 类之间存在适度的耦合是正常的、必须的
  - 这样才能产生面向对象系统，其任务是由相互连接的对象通过协作来完成 to create an object-oriented system in which tasks are fulfilled by a collaboration between connected objects

3.4 Discuss：Low Coupling— When Not To

- “背靠大树好乘凉”
  - 因为大树不会倒、不常变、稳定
- High coupling to stable elements and to pervasive elements is seldom a problem
  - e.g. language libraries
  - For example, a Java J2EE application can safely couple itself to the Java libraries (java.util, and so on)
  - because they are stable and widespread

3.5 Example: Low Coupling

- **Why not assign getSquare to Dog? (i.e., some arbitrary other class)**
- **Minimal coupling will reduce change (dependency)**



- 本讲结束