CSSE 374: 3½ Gang of Four Design Patterns

Shawn Bohner

Office: Moench Room F212

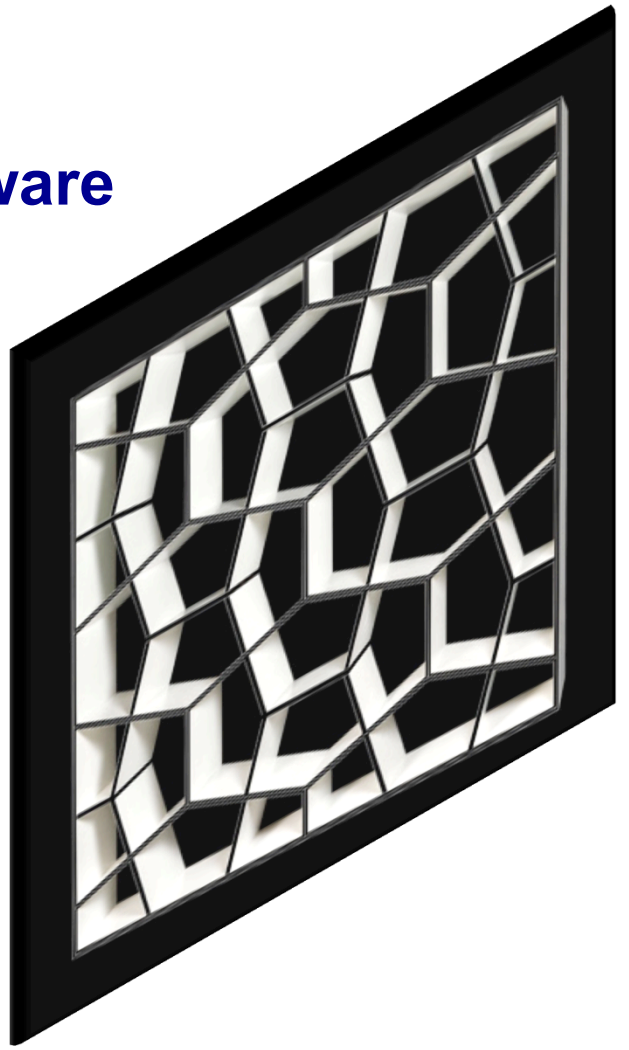
Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

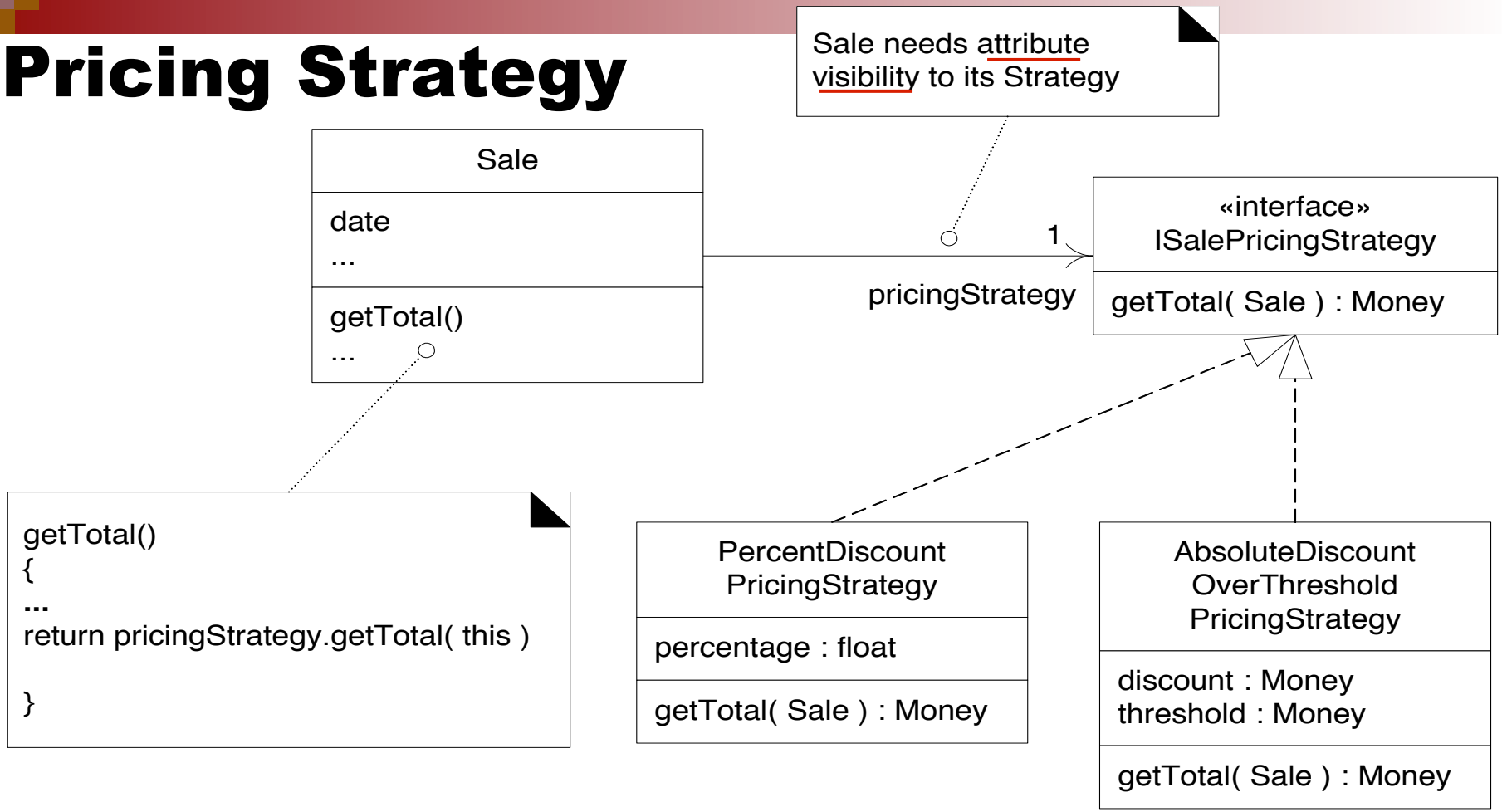
Learning Outcomes: Patterns, Tradeoffs

Identify criteria for the design of a software system and select patterns, create frameworks, and partition software to satisfy the inherent trade-offs.



- Describe and use GoF Patterns
  - Composite
  - Façade
  - Observer
  - Intro to Abstract Factory
- Design Studio with Team 2.3

Pricing Strategy

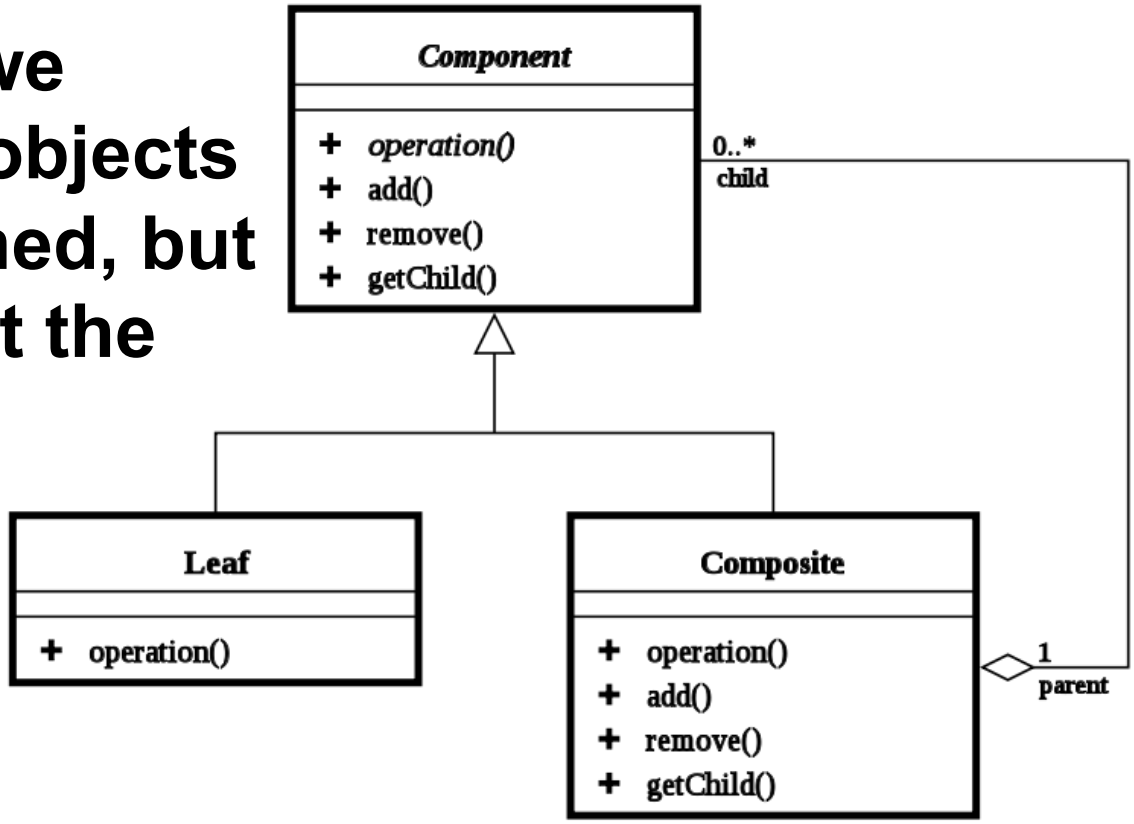


But how do we handle multiple, conflicting pricing policies?

- **Preferred customer discount, 15% off sales of $400**
- **Buy 1 case of Darjeeling tea, get 15% off entire order**
- **Manic Monday, $50 off purchases over $500**

Composite: Structural Pattern

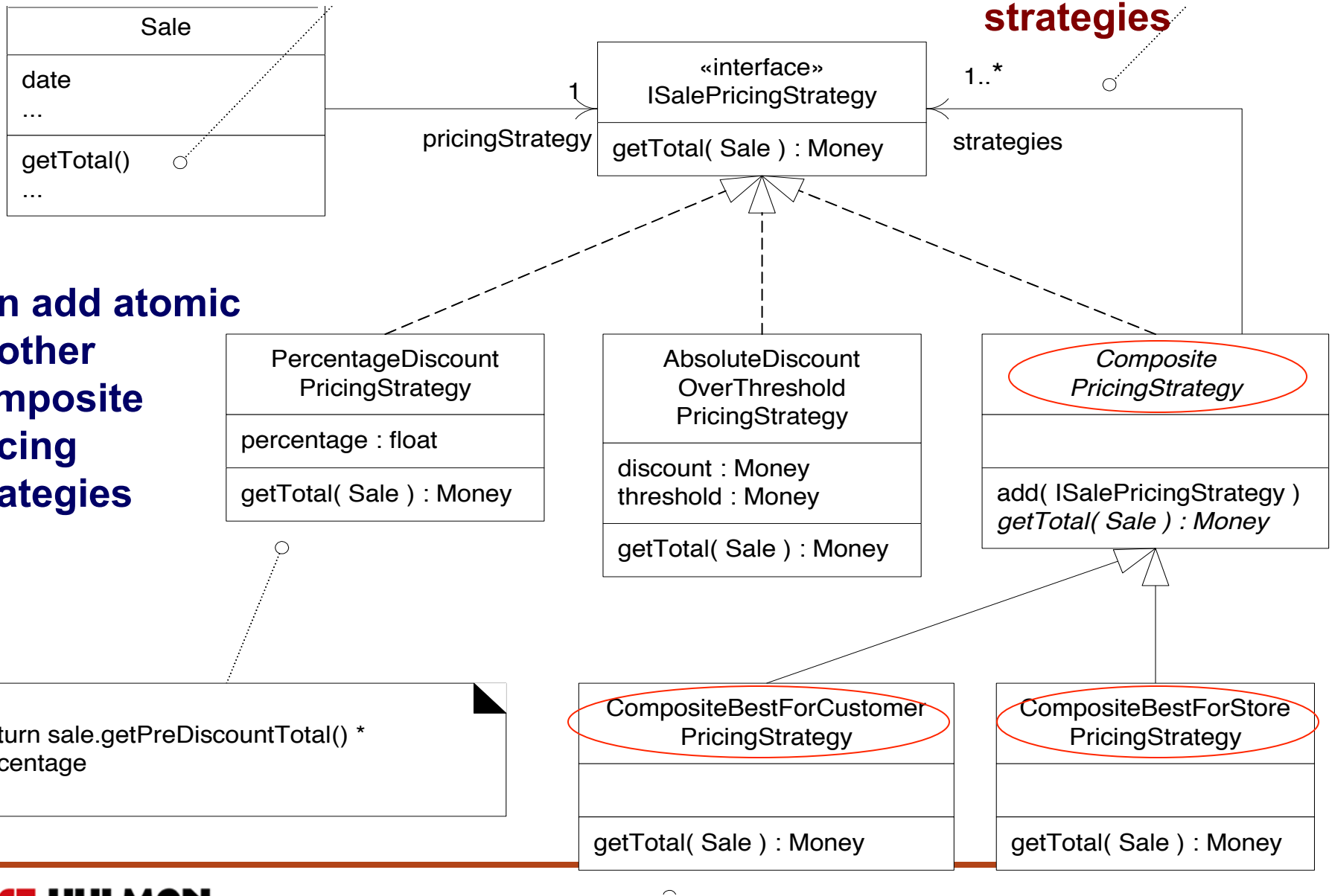
Problem: How do we handle a group of objects that can be combined, but should still support the same polymorphic methods as any individual object in the group?



Solution: Define a *composite* object that implements the same interface as the individual objects.

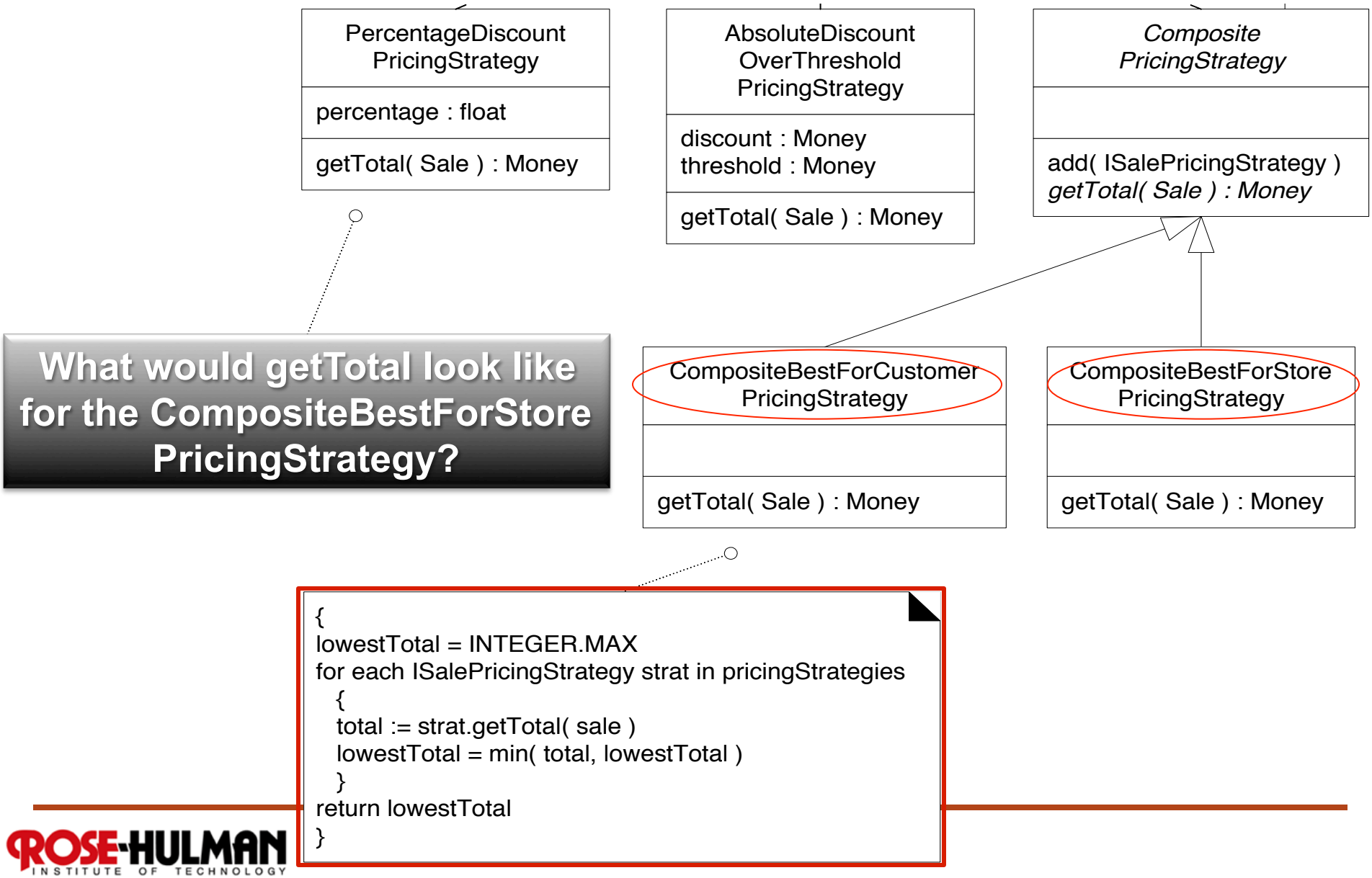
Composite Pricing Strategy

Composites have list of contained strategies

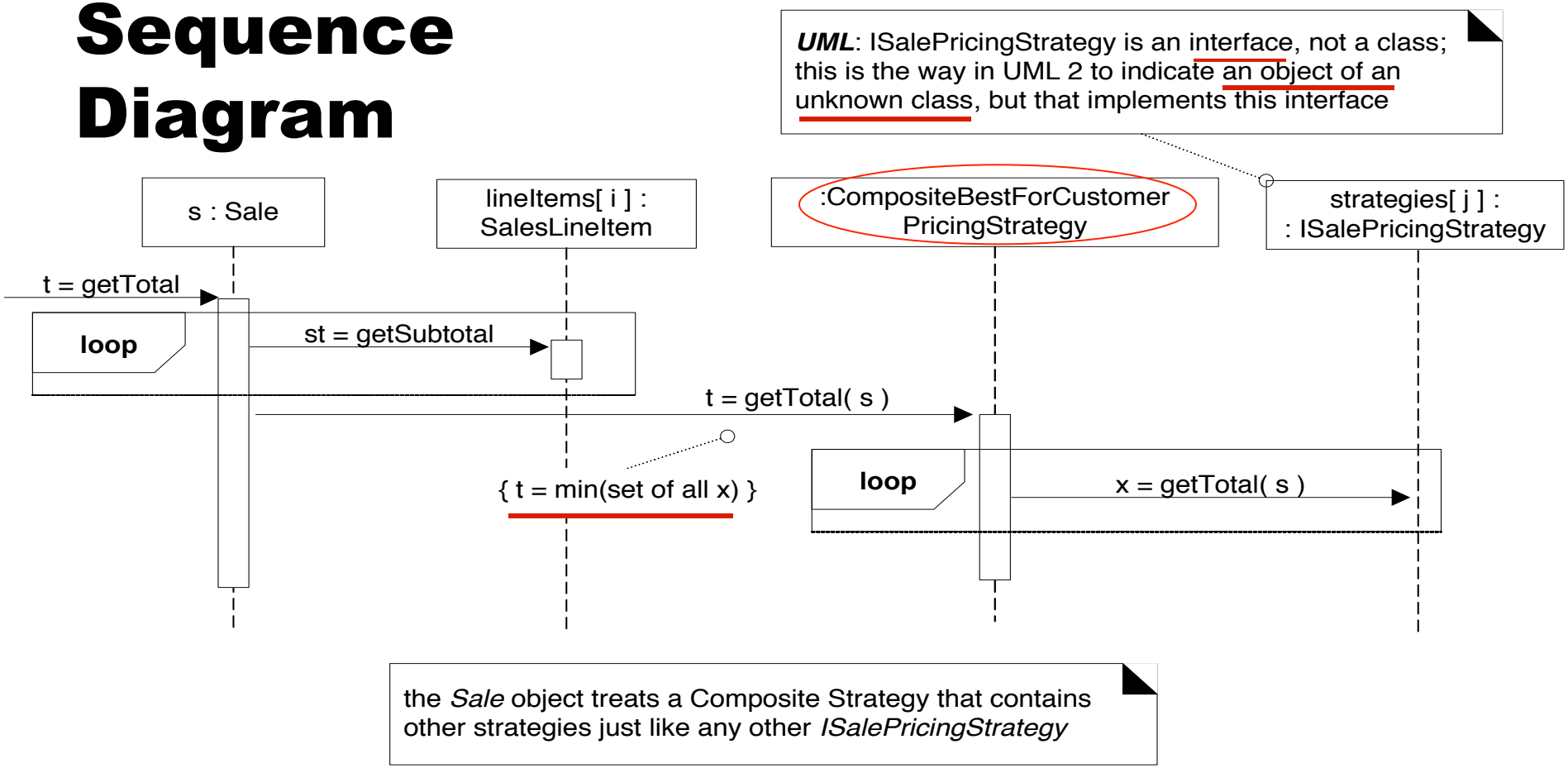


Can add atomic or other composite pricing strategies

Composite Pricing Strategy (continued)



Composite Sequence Diagram



Composite object iterates over its collection of atomic strategy objects

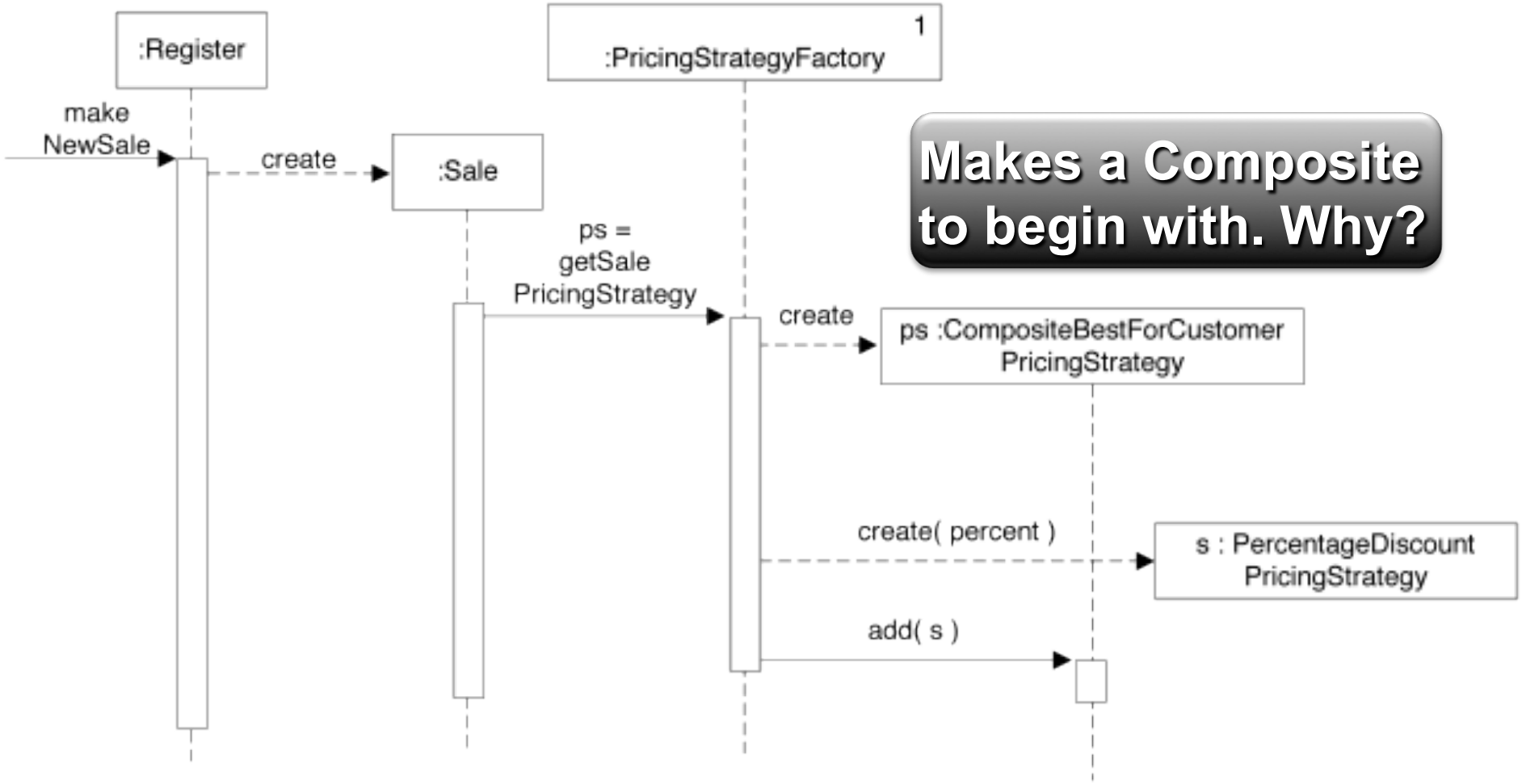
How do we build a Composite Strategy?

Three places in example where new pricing strategies can be added:

1. When *new sale is created*, add store discount policy
2. When *customer is identified*, add customer-specific policy
3. When a *product is added to the sale*, add product-specific policy

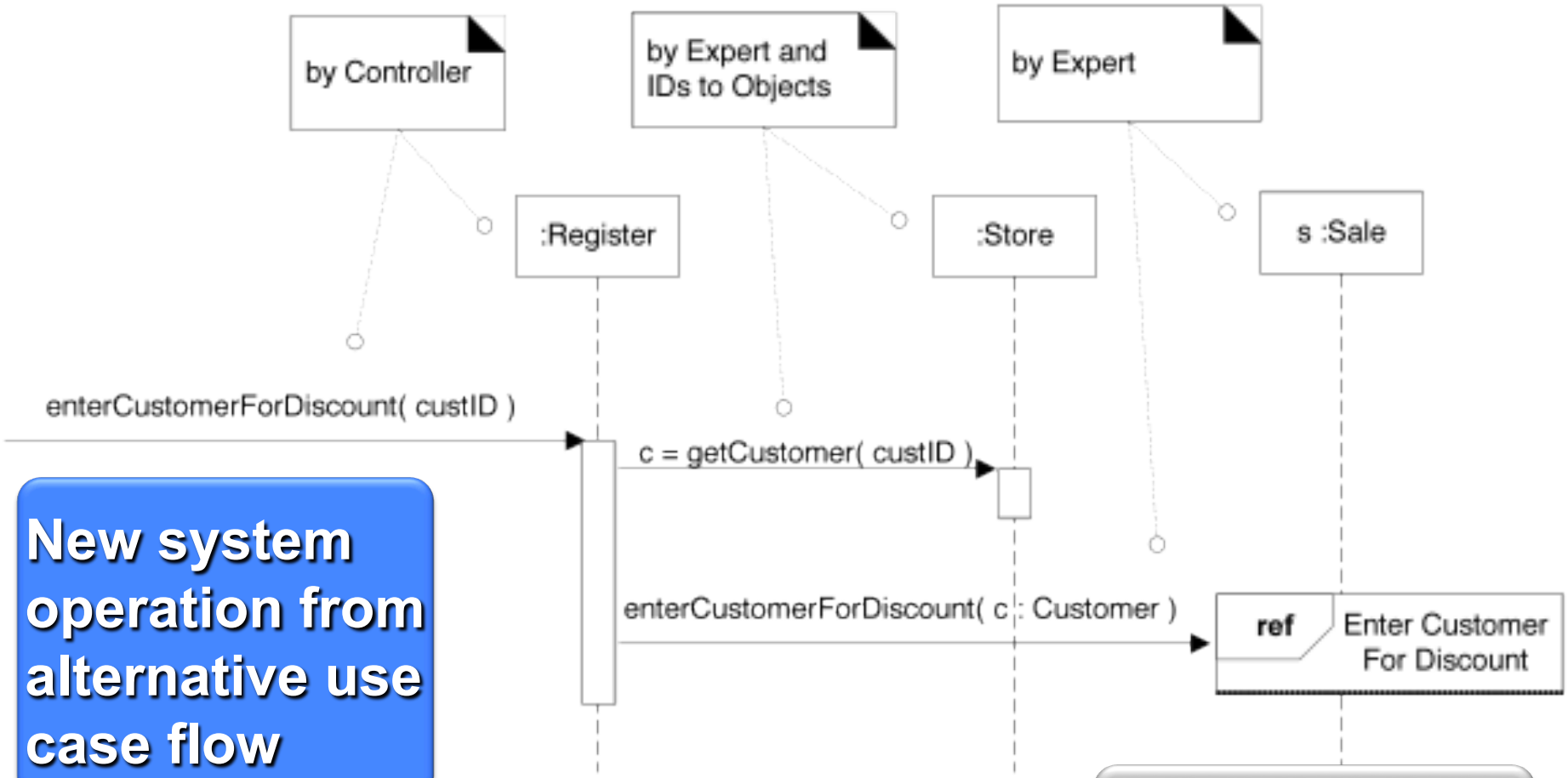
1. Adding Store Discount Policy

Singleton, Factory



Makes a Composite to begin with. Why?

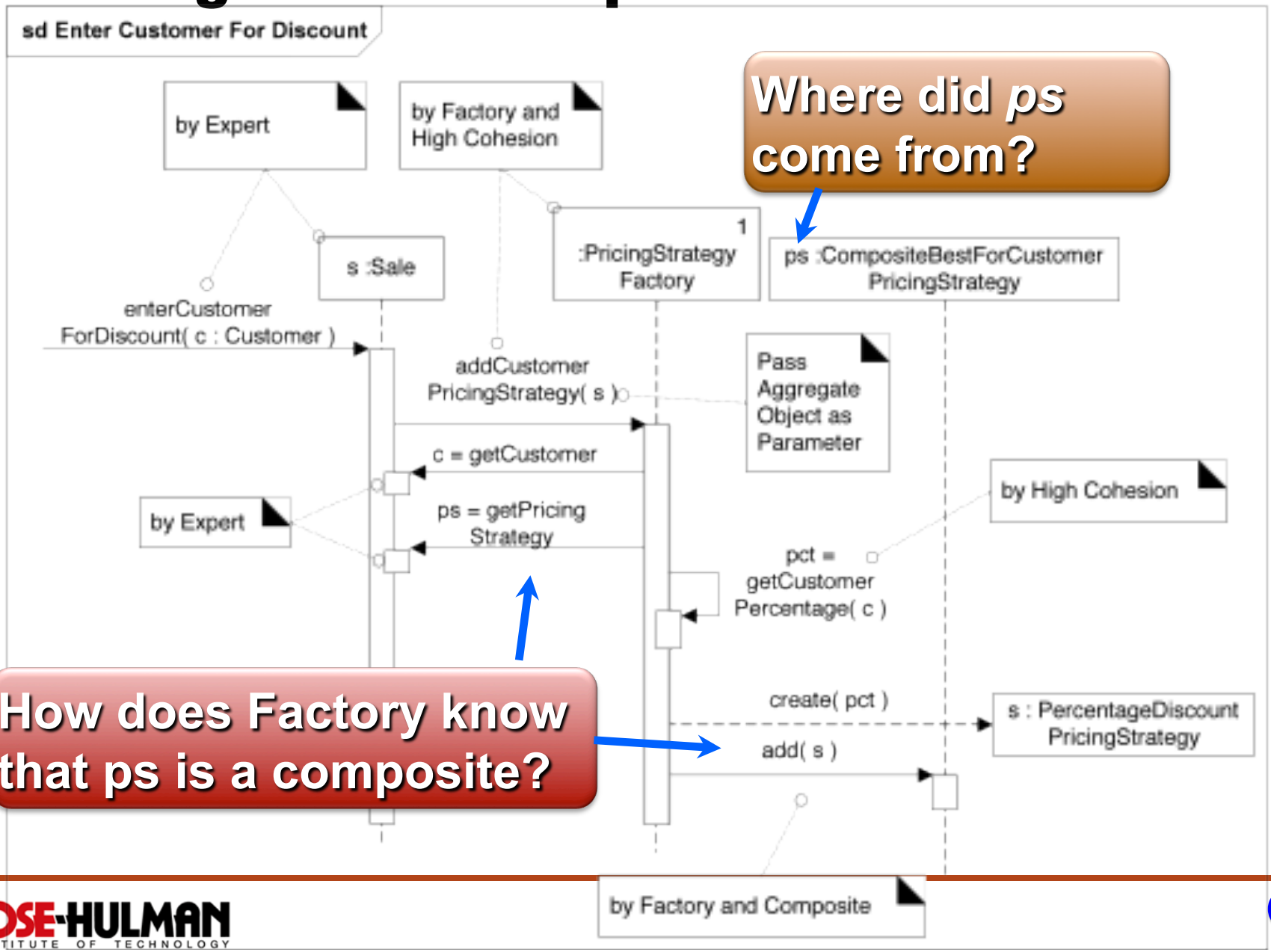
2. Adding Customer Specific Discount Policy



New system operation from alternative use case flow

Recall: What's a ref frame?

2. Adding Customer Specific Discount Policy



Recall BrickBusters Video Store. Identify a situation where Composite might be applicable.

- Think for 15 seconds…
- Turn to a neighbor and discuss it for a minute

Façade

More general than just Façade Controllers

- NextGen POS needs *pluggable business rules*
- Assume rules will be able to disallow certain actions, such as…
  - Purchases with gift certificates must include just one item
  - Change returned on gift certificate purchase must be as another gift certificate
  - Allow charitable donation purchases, but max. of $250 and only with manager logged-in

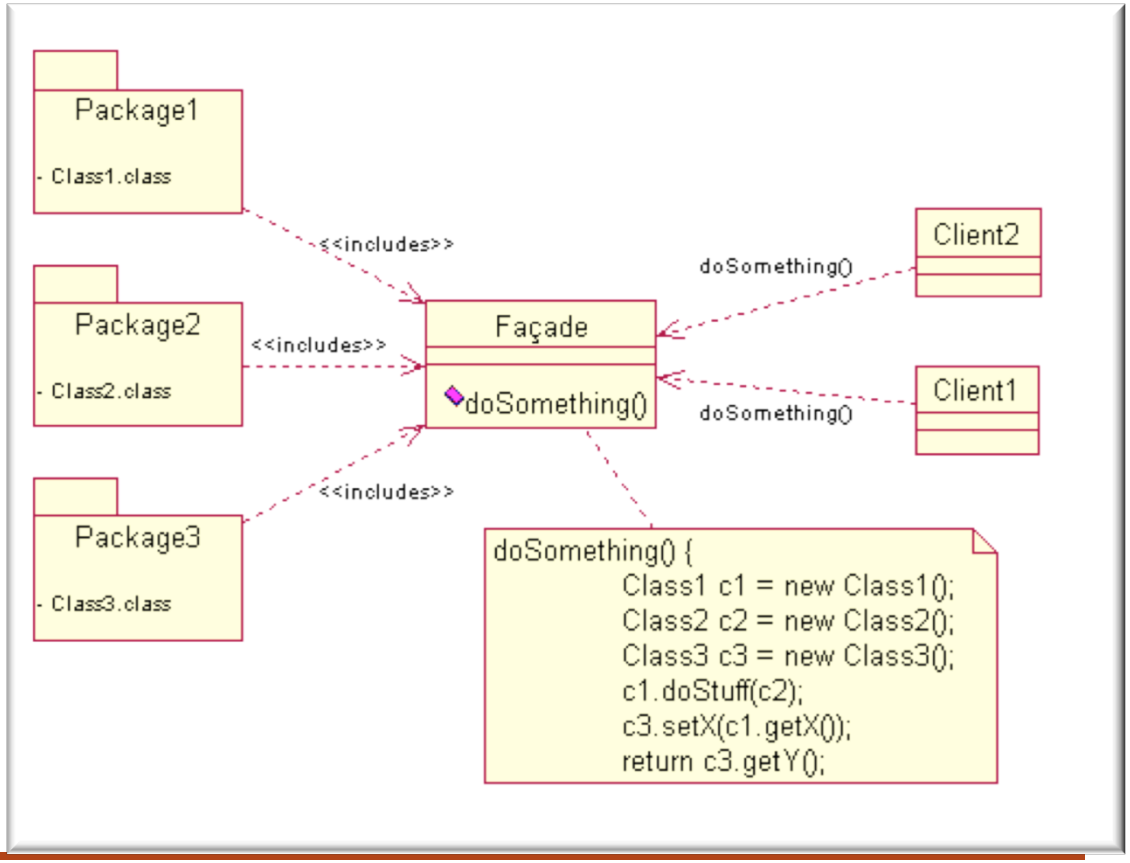
Some Conceivable Implementations

- Strategy pattern
- Open-source rule interpreter
- Commercial business rule engine

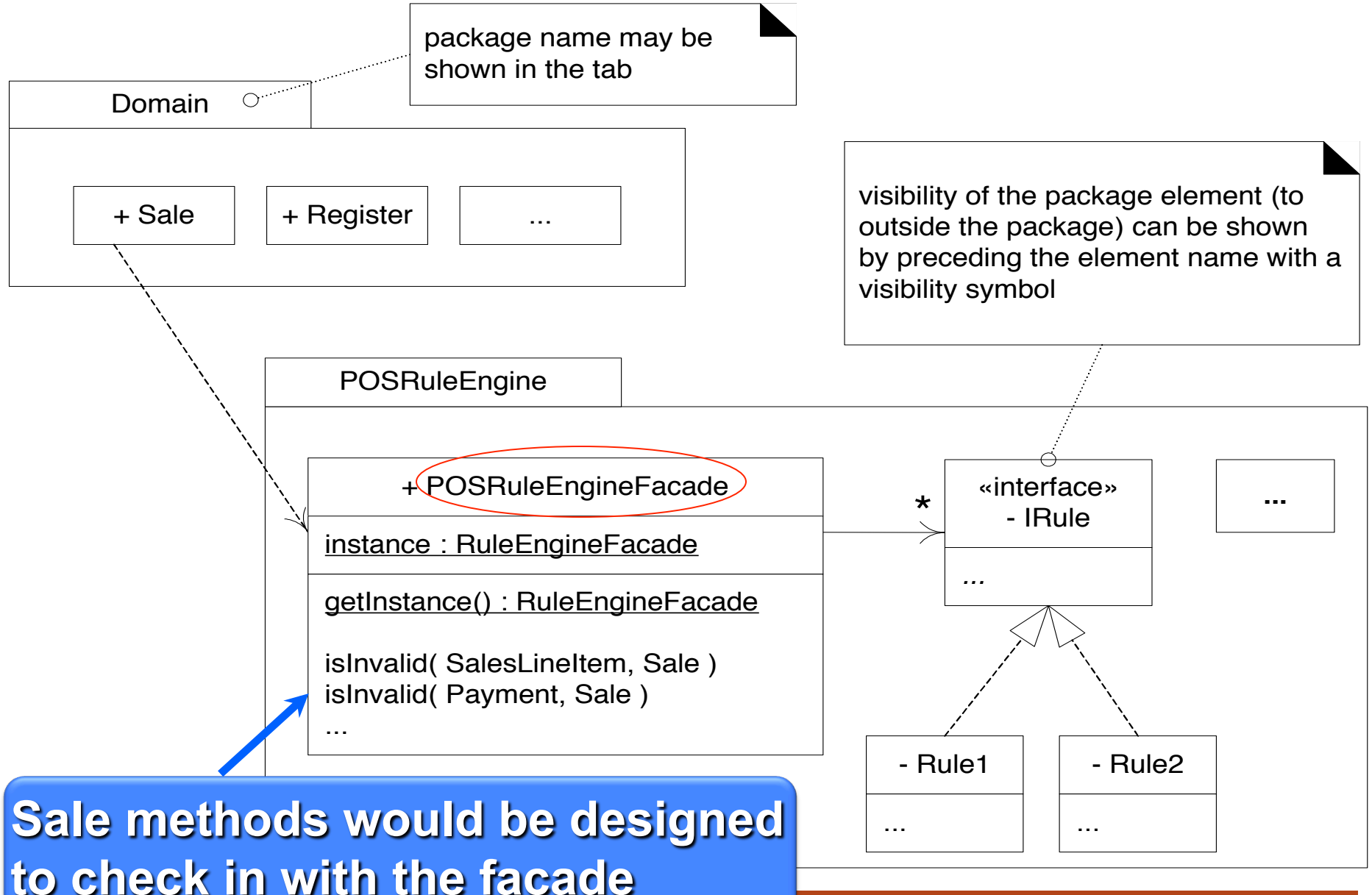
Façade

Problem: How do we avoid coupling to a part of the system whose design is subject to substantial change?

Solution: Define a *single point of contact* to the variable part of the system—a *façade object* that wraps the subsystem.

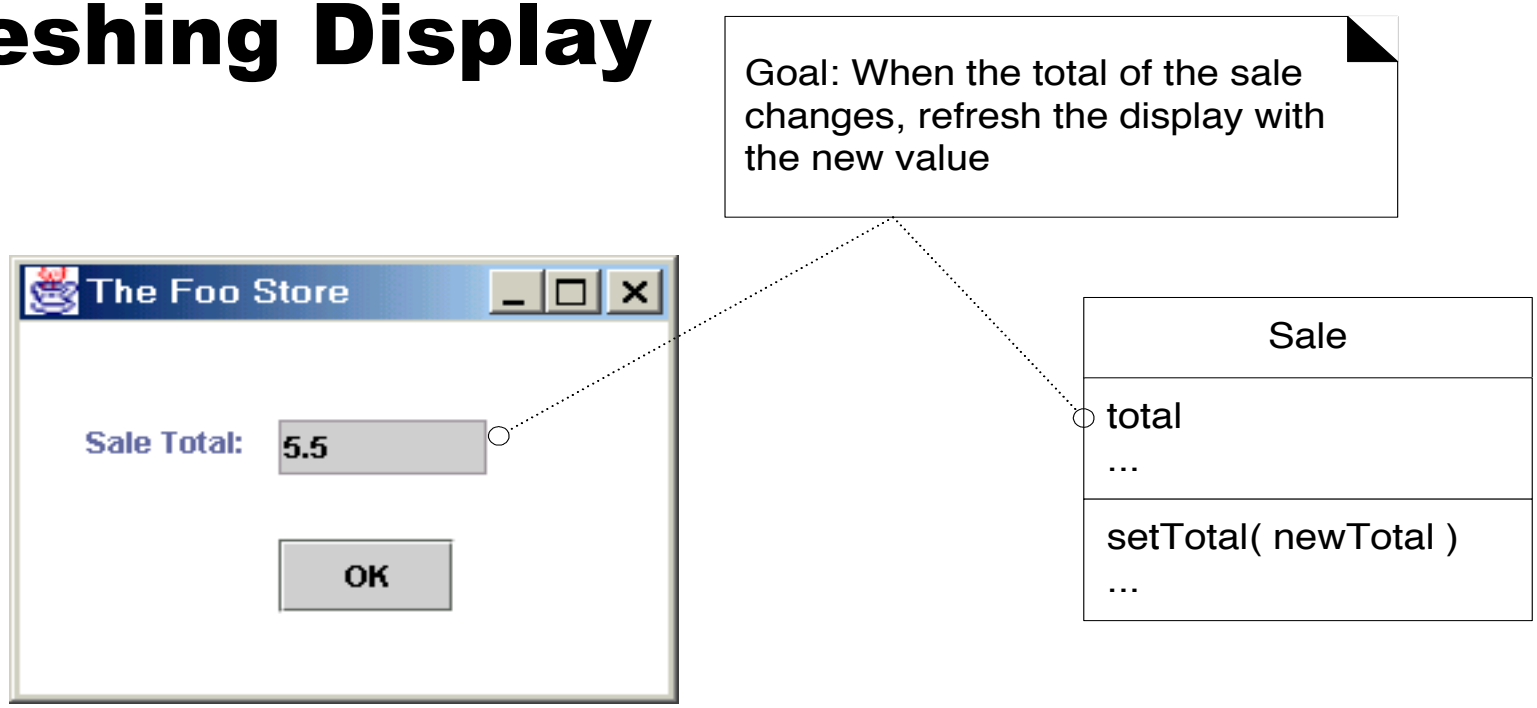


Façade Example



Sale methods would be designed to check in with the façade

Refreshing Display



How do we refresh the GUI display when the domain layer changes *without coupling the domain layer back to the UI layer*?

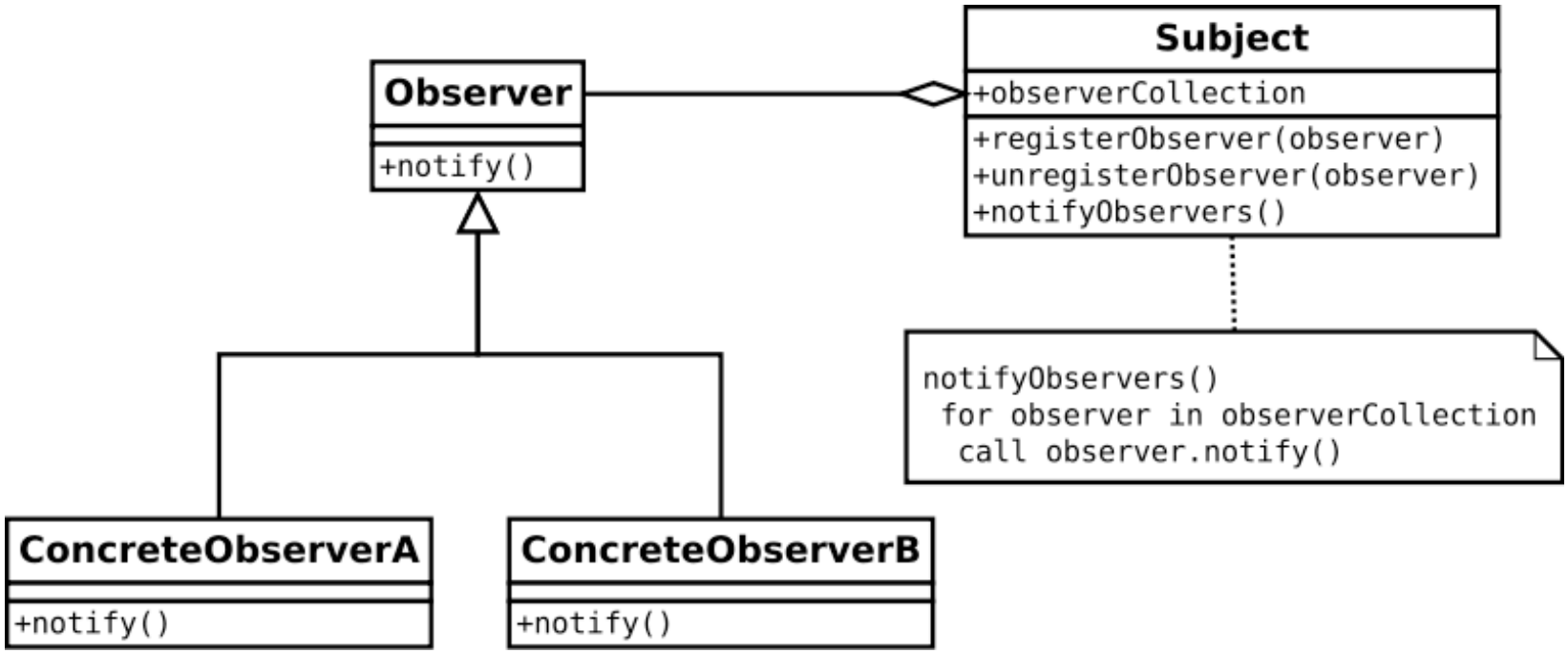
Model-View Separation

Observer (aka Publish-Subscribe/Delegation)

Problem: *Subscriber* objects want to be informed about events or state changes for some *publisher* object. How do we do this while maintaining low coupling from the publisher to the subscribers?

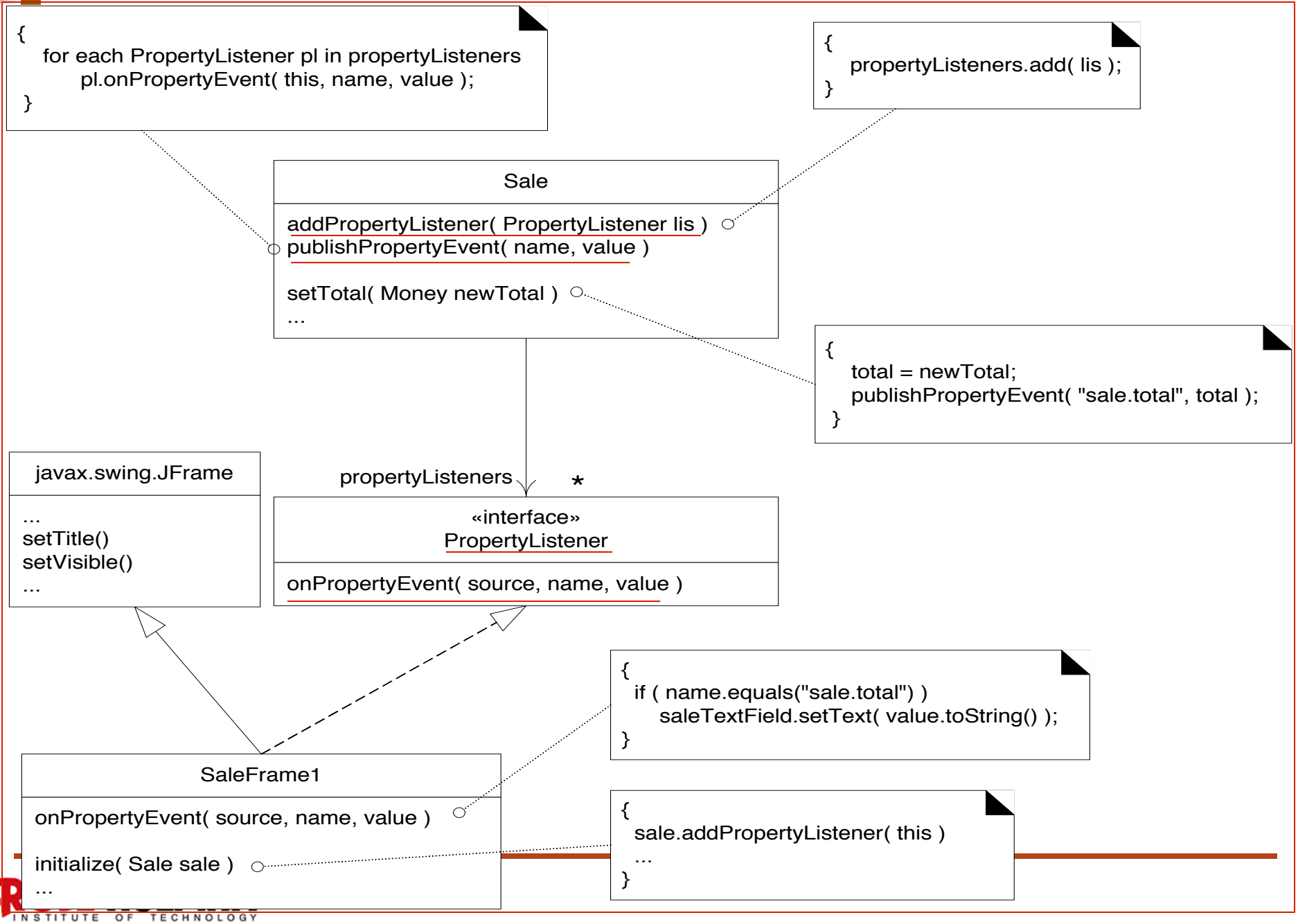
Solution: Define an subscriber interface that the subscriber objects can implement. Subscribers register with the publisher object. The publisher sends notifications to all its subscribers.

Observer: Behavioral Pattern

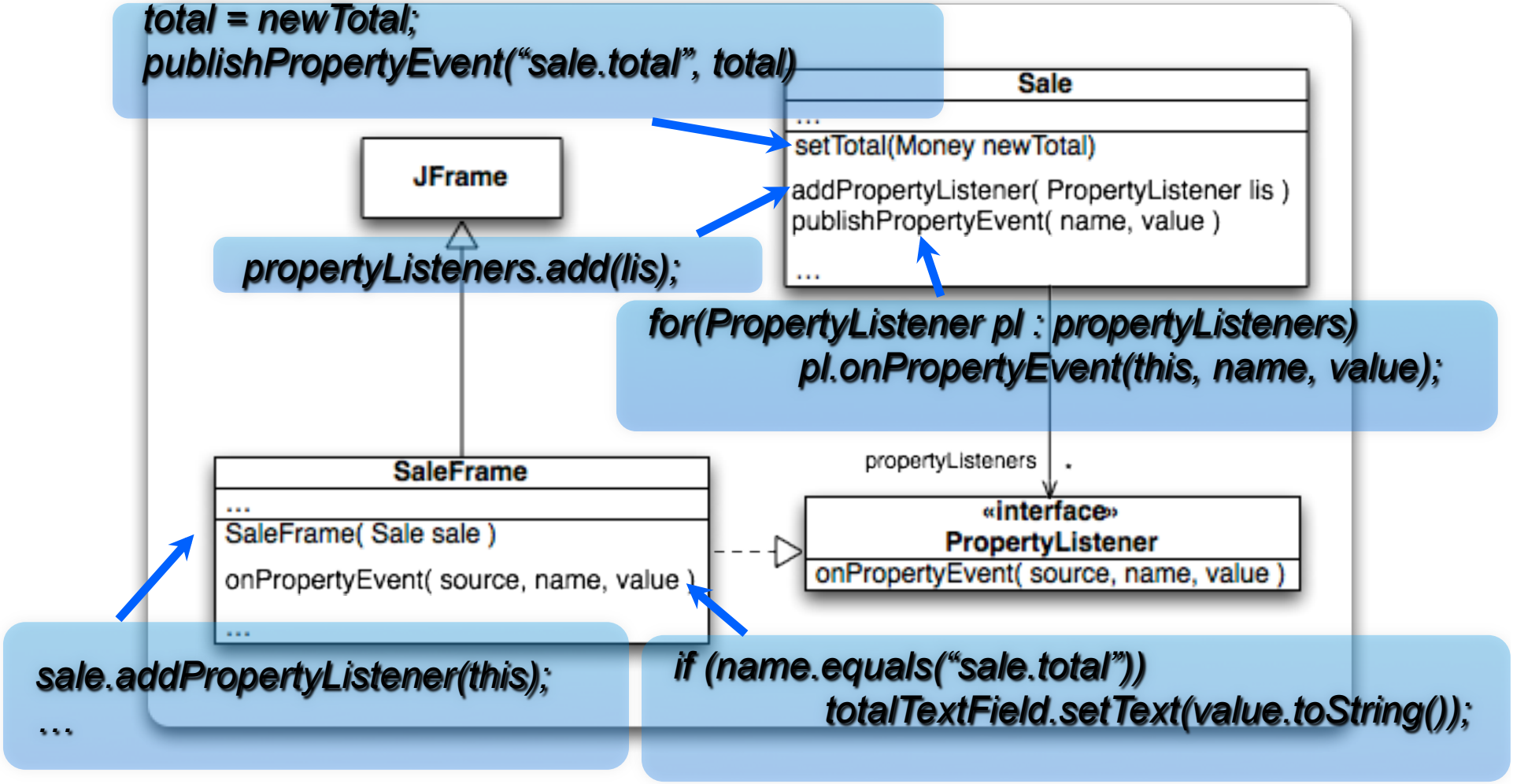


Observer pattern is a 1:N pattern used to notify and update all dependents automatically when one object changes.

Sale has a List of Listeners

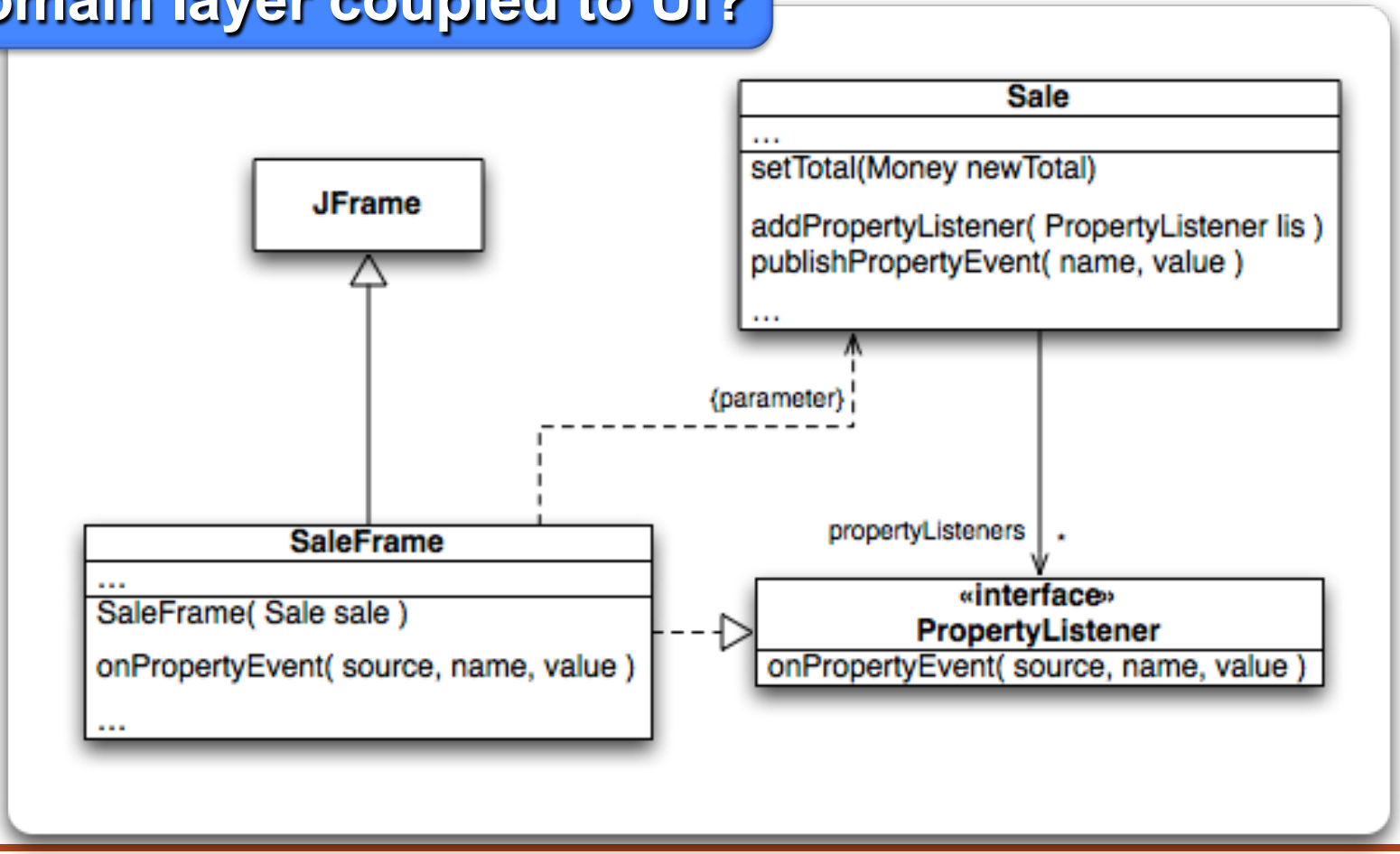


Example: Update SaleFrame when Sale's Total Changes



Example: Update SaleFrame when Sale's Total Changes (continued)

Is UI coupled to domain layer?
Is domain layer coupled to UI?



Observer: Not just for GUIs watching domain layer...

- GUI widget event handling
- Example:

```
JButton startButton = new JButton("Start");
startButton.addActionListener(new Starter
());
```

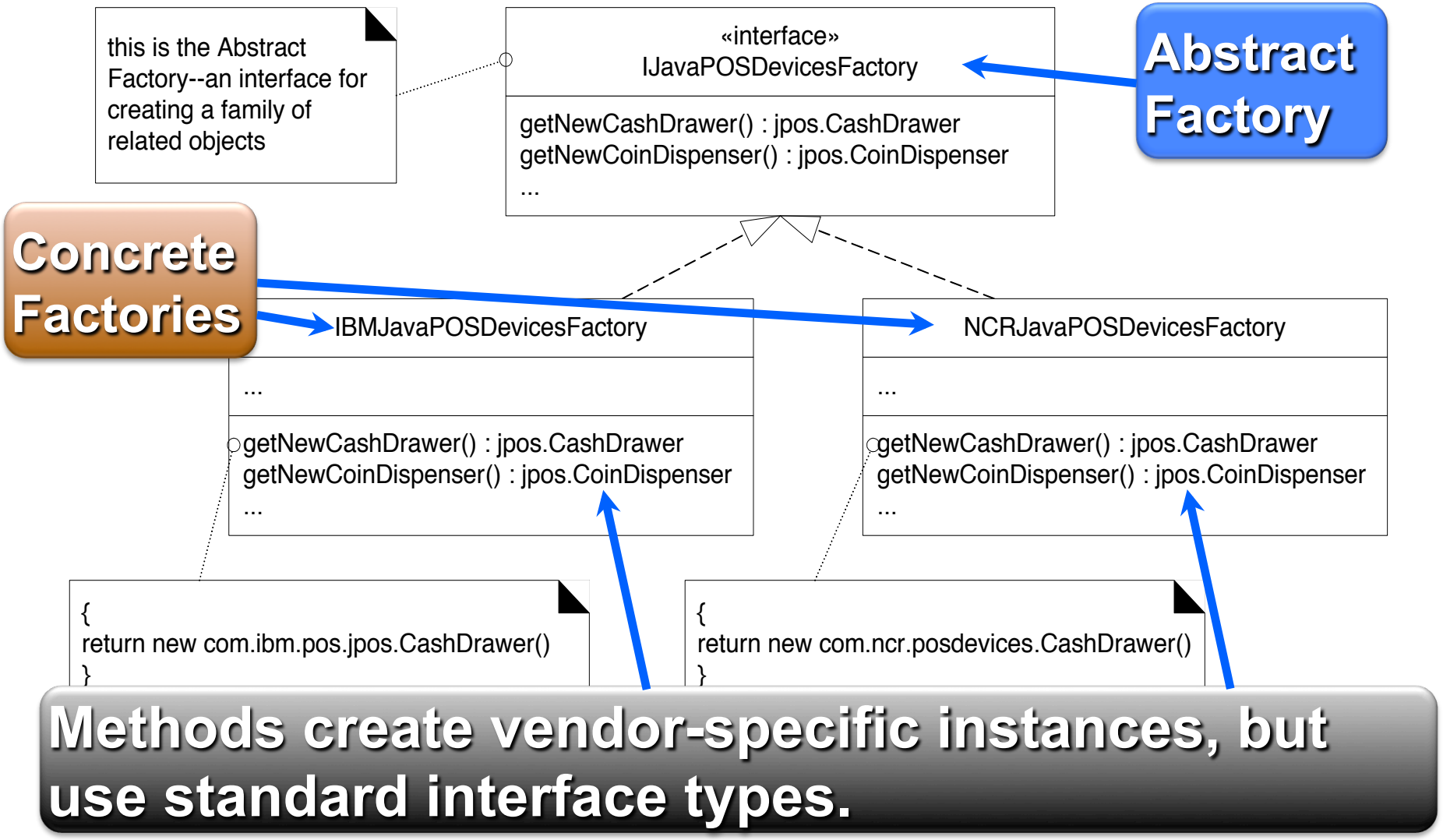
- Publisher: *startButton*
- Subscriber: *Starter* instance

Abstract Factory: Creational Pattern

Problem: How can we create families of related classes while preserving the variation point of switching between families?

Solution: Define an *abstract factory* interface. Define a *concrete factory* for each family.

Abstract Factory Example



Methods create vendor-specific instances, but use standard interface types.

Design Studios

Objective is to share your design with others to communicate the approach or to leverage more eyes on a problem.

- **Minute or so to set up…**
- **5-6 minute discussion**
- **1-2 minute answering questions**

1. **Team 2.3 – Evaluation GUI Tool**

Homework and Milestone Reminders

- Read Chapter 27 and 28
- Homework 5 – BBVS Design using more GRASP Principles
  - Due by 11:59pm Tuesday, January 25th, 2011
- Milestone 4 – Junior Project Design with More GRASP'ing
  - Due by 11:59pm on Friday, January 28th, 2011