CSSE 374: More Object Design with Gang of Four Design Patterns

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

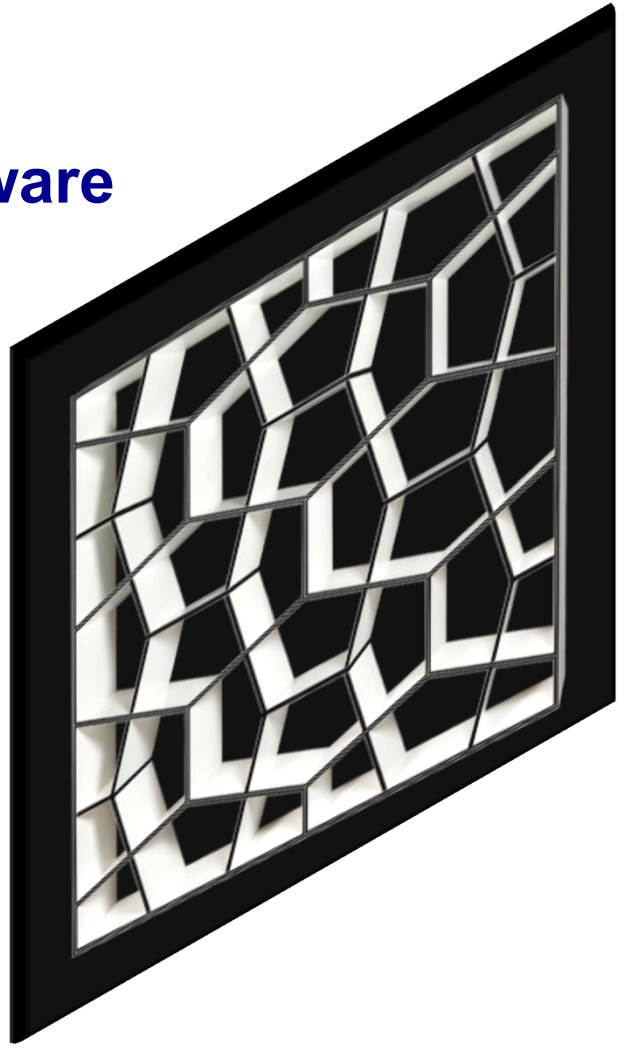
Email: bohner@rose-hulman.edu

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.

- Using GoF Patterns in Iteration 3
  - Local caching
  - Failover to local services
  - Support for third-party POS devices
  - Handling payments
- Exercise (if time)
- Design Studio with Team 2.3



Gang of Four Design Patterns

Behavioral

- Interpreter
- Template Method
- Chain of Responsibility
- ✓ Command
- Iterator
- Mediator
- Memento
- ✓ Observer
- State
- ✓ Strategy
- Visitor

Creational

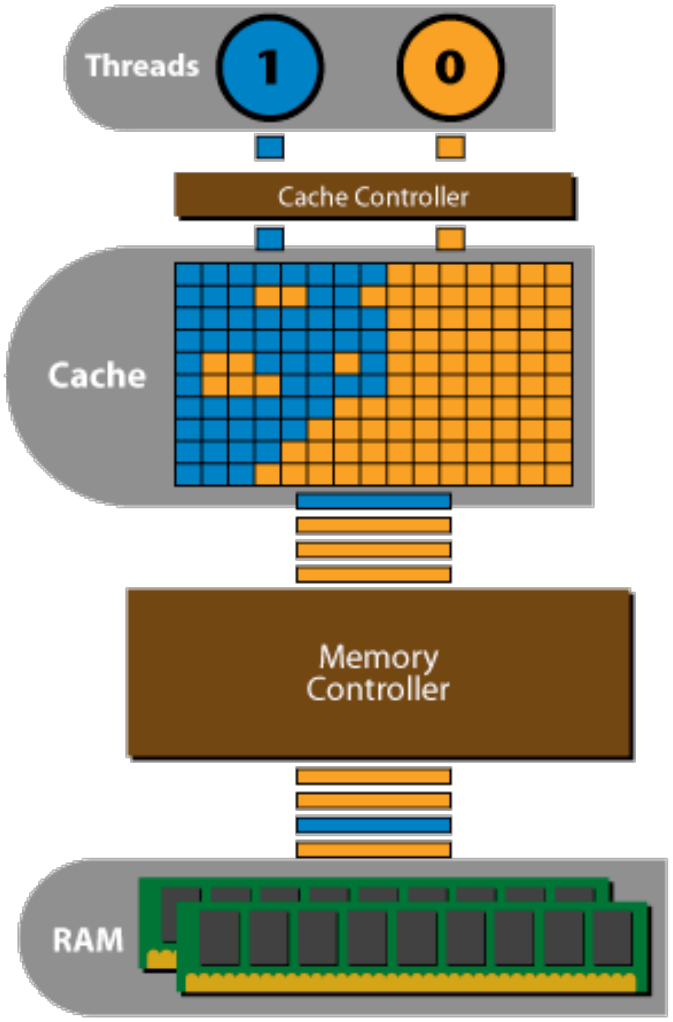
- ✓ Factory
- ✓ Abstract Factory
- Builder
- Prototype
- ✓ Singleton

Structural

- ✓ Adapter
- Bridge
- ✓ Composite
- ✓ Decorator
- ✓ Façade
- Flyweight
- ✓ Proxy

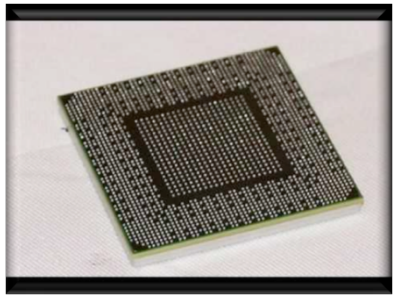
Failover & Performance with Local Caching

- What is a *cache*? How does a cache usually work?
- Why use a local cache for NextGen POS?
  - Performance
  - Improve recoverability



Search Strategy for Product Information

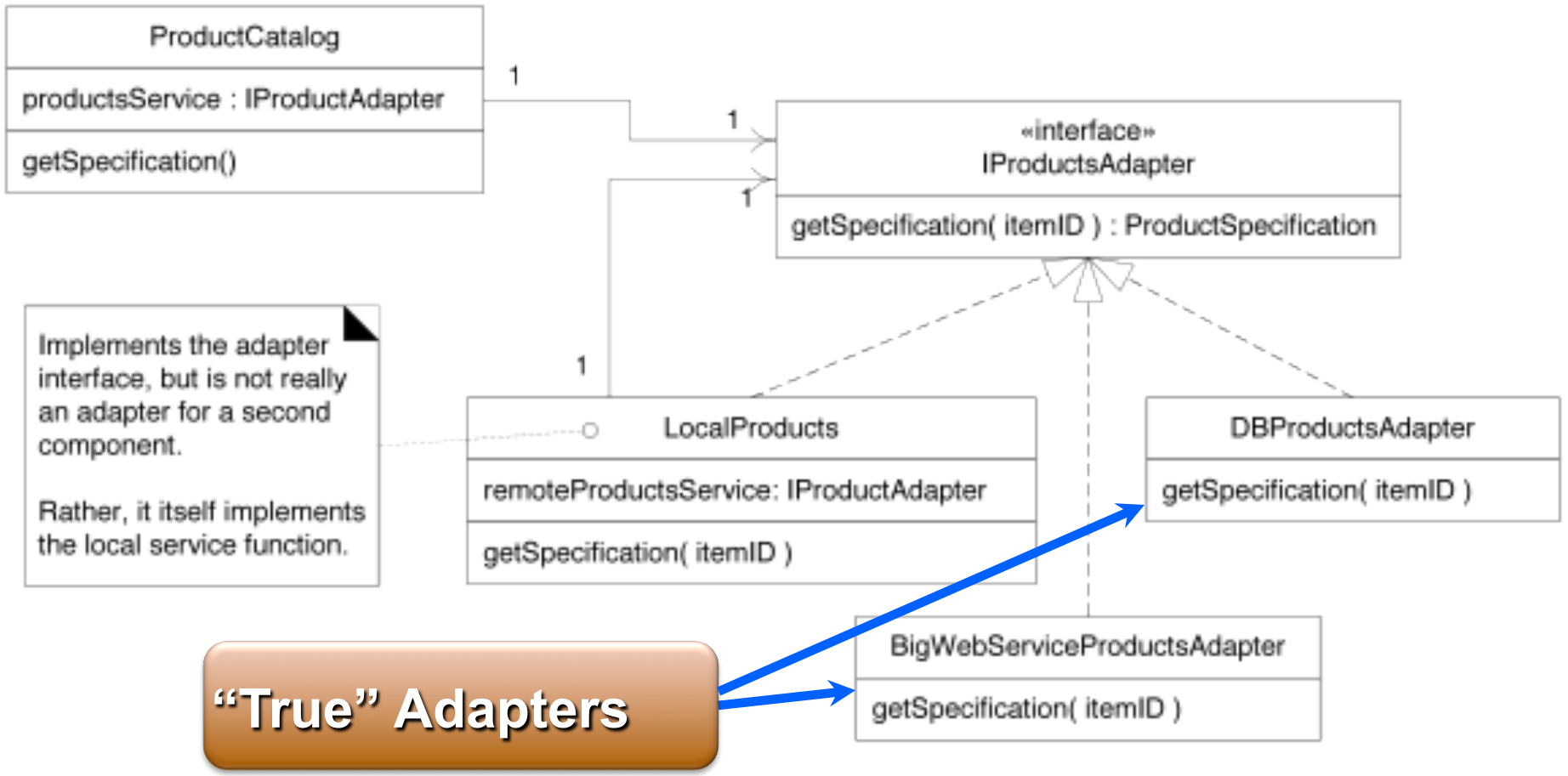
1. Look in memory (in map stored by *ProductCatalog*)



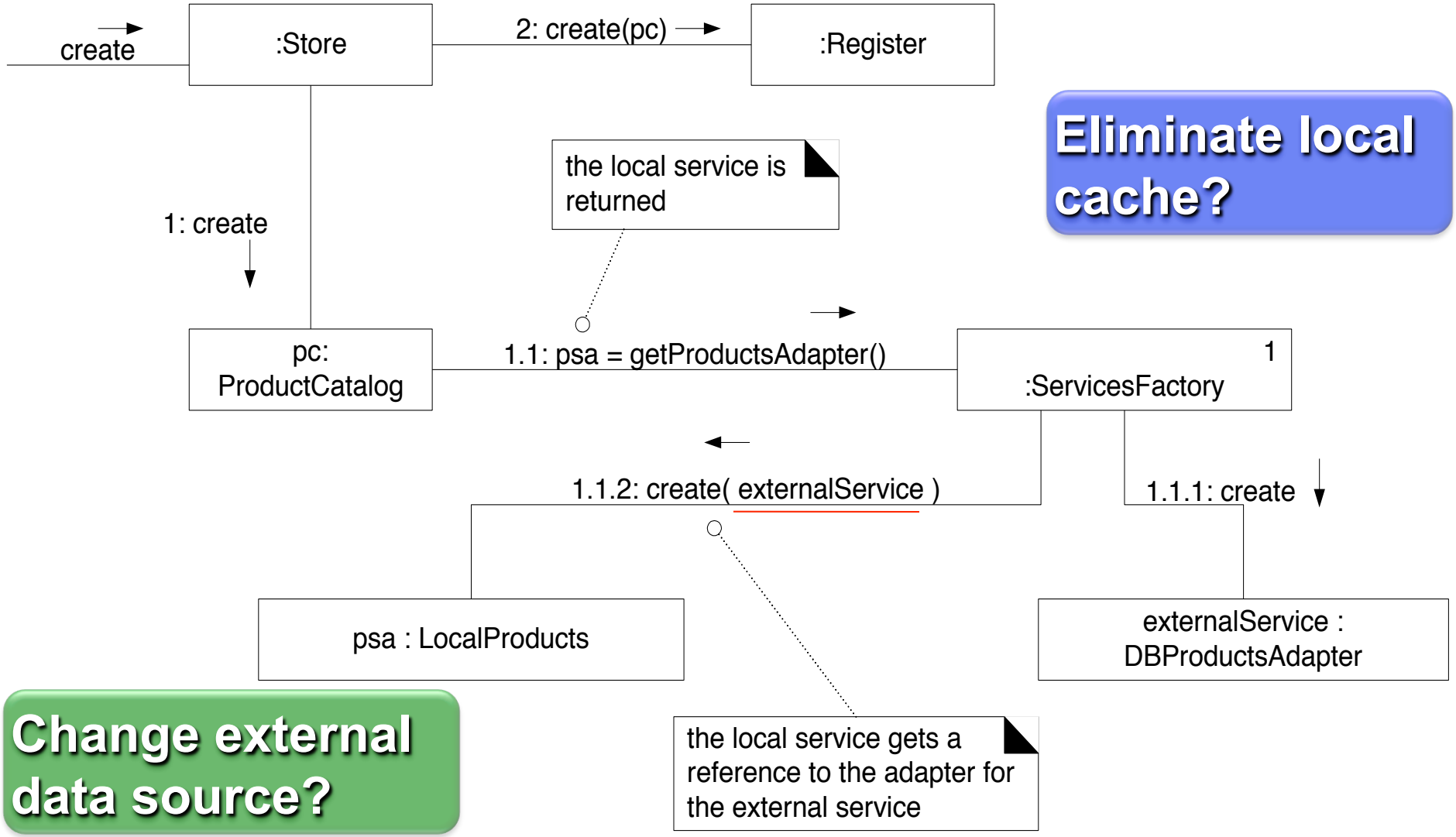
2. Look on local hard drive cache

3. Retrieve from remote persistence service

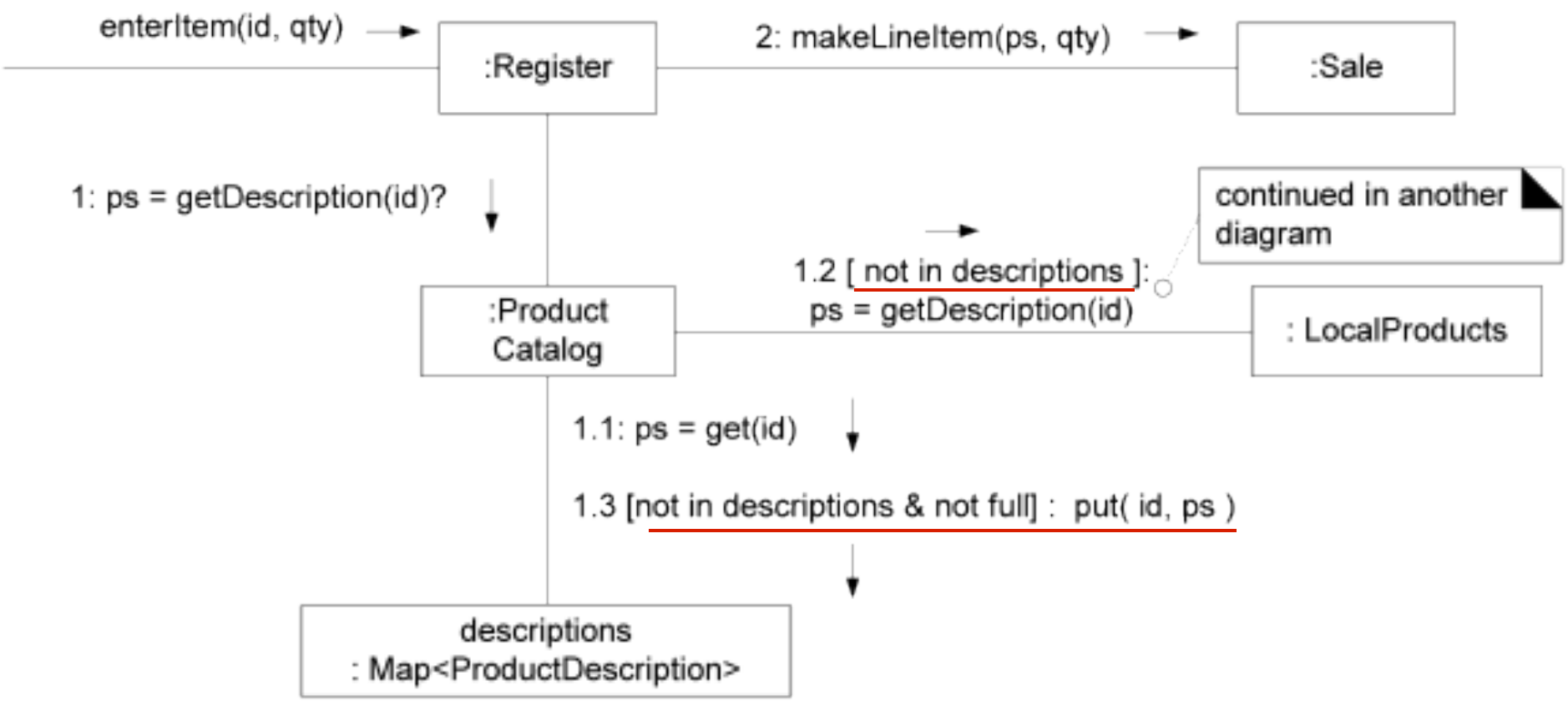
Applying the Adapter Pattern



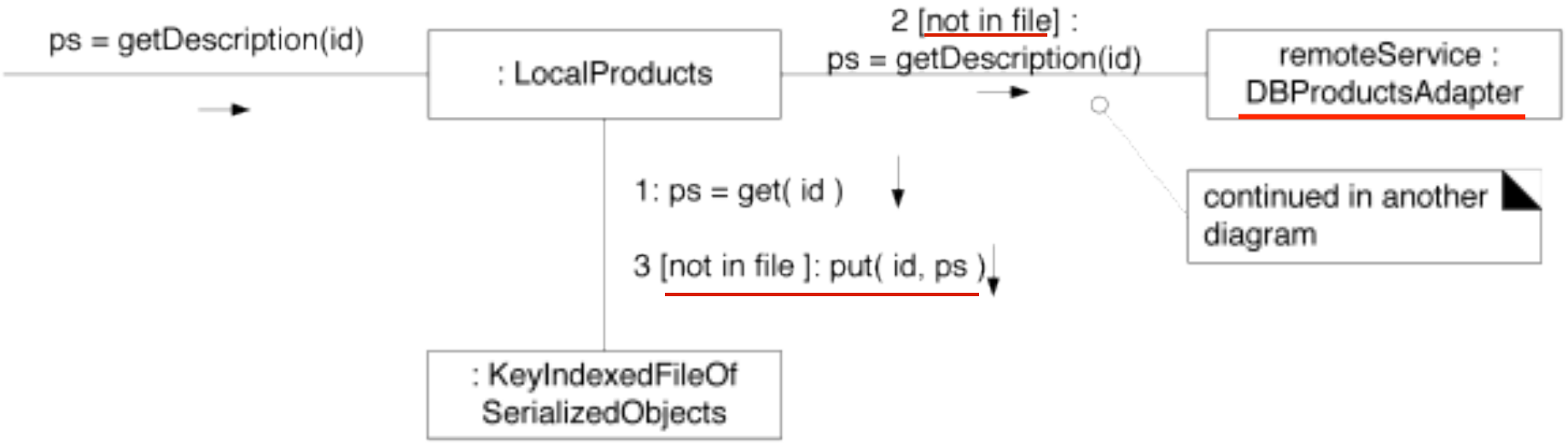
Using a Factory to Set Up for Local Caching

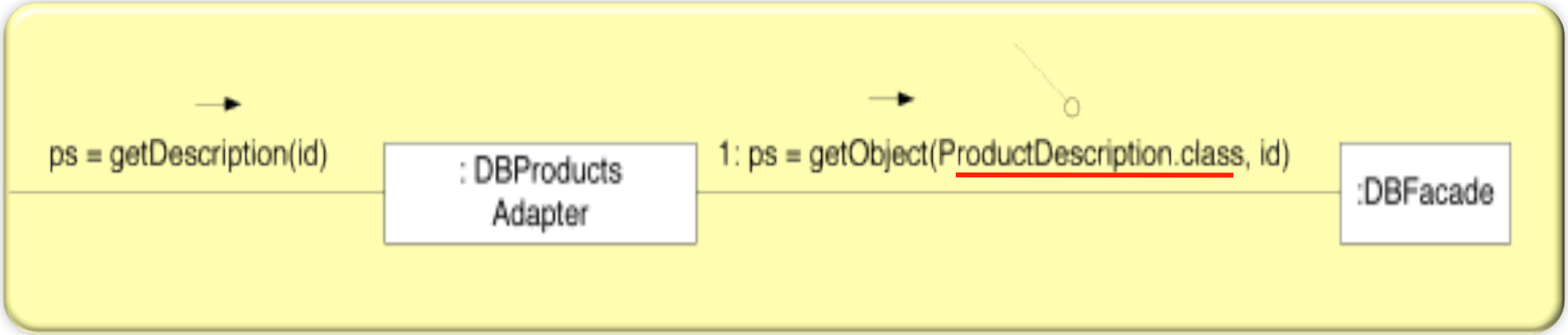


Product Lookup with In-memory Miss

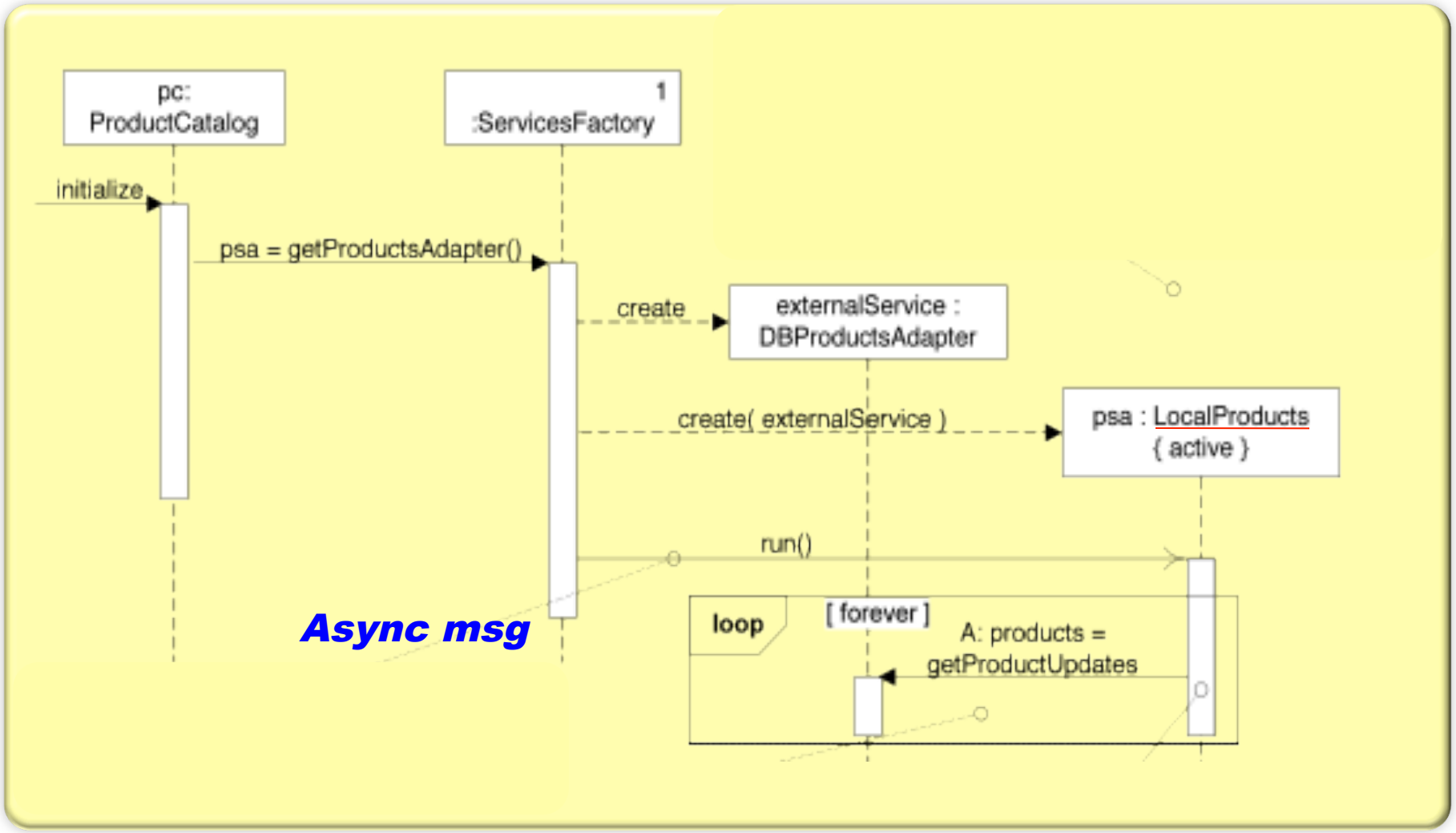


Product Lookup with Local Cache Miss

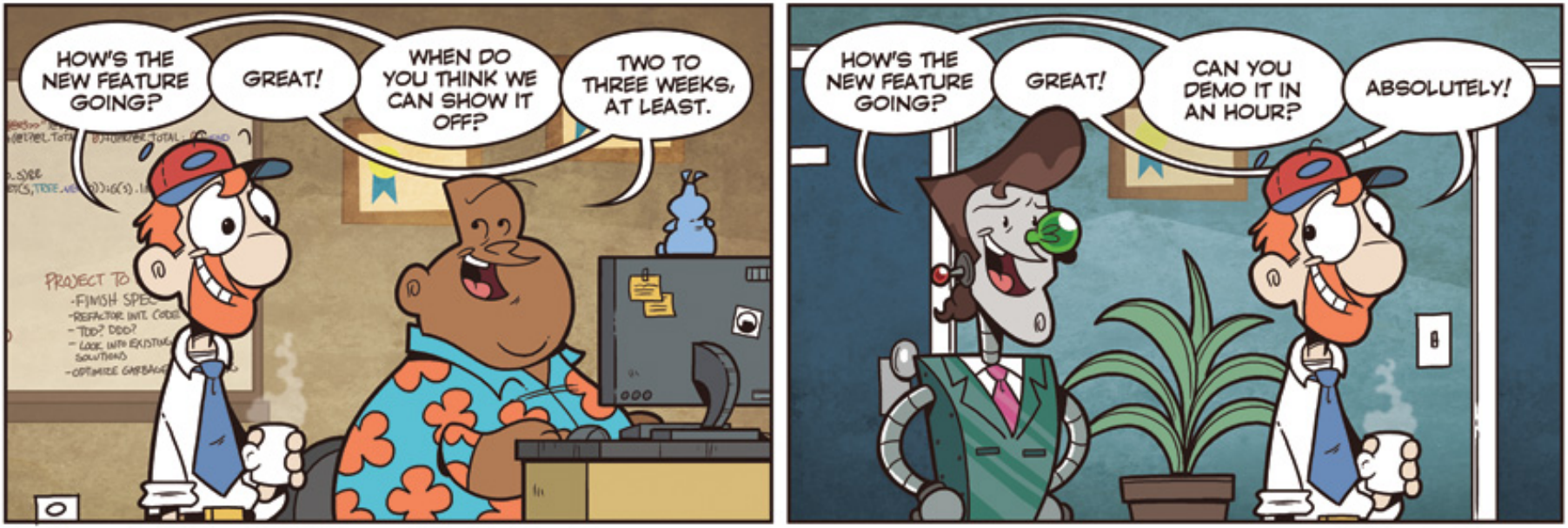




Using Threads to Freshen the Cache



How's the final iteration going?



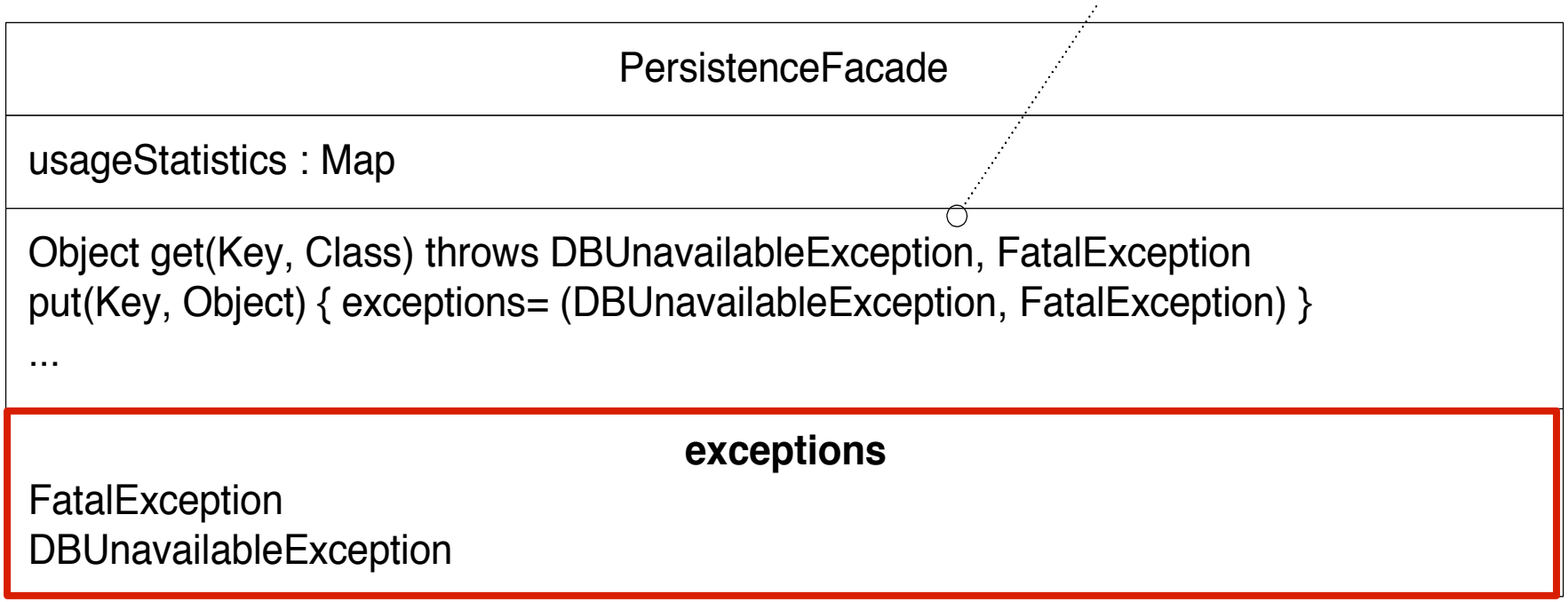
Used by permission. http://notinventedhe.re/on/2009-12-28

Handling Failure in NextGen POS:

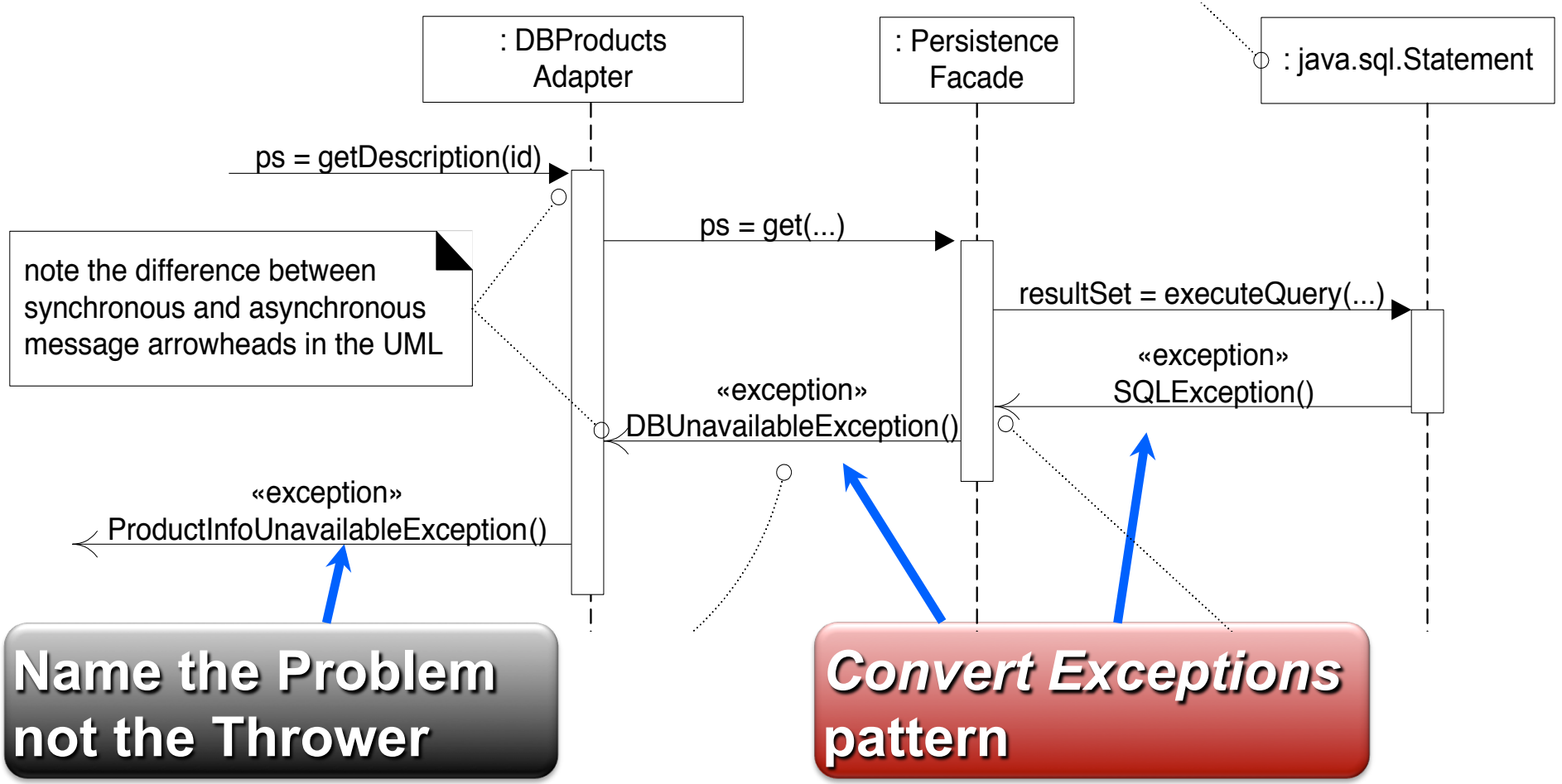
What should happen if there is a local cache miss and the external product information service fails?

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

DCD: Exceptions Caught and Thrown



Showing Exception in Sequence Diagrams



How should NextGen POS handle this exception?

Common exception handling patterns

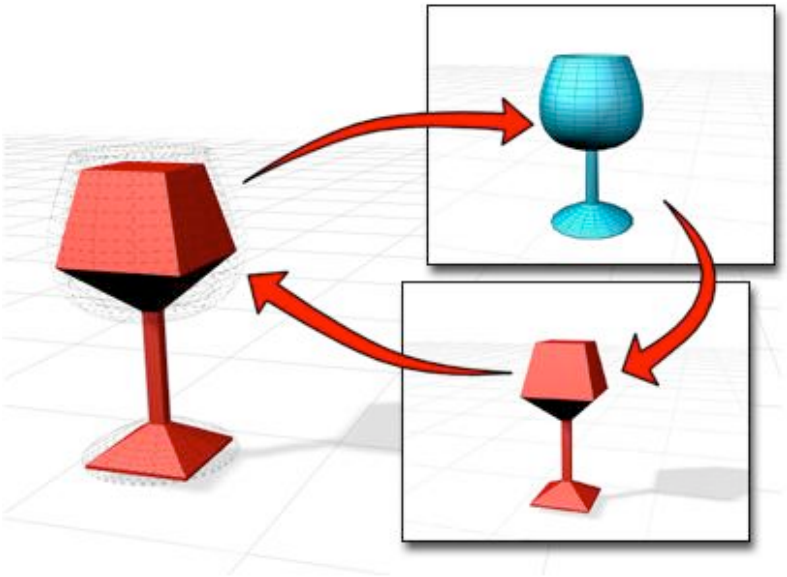
- Use a central error logging object to record all exceptions for diagnosis by developers

- Use a standard, application-independent, non-UI object to notify users
  - Can delegate to multiple different UI notifications
  - Protected Variation for changes in reporting

Failover to Local Services with a Proxy

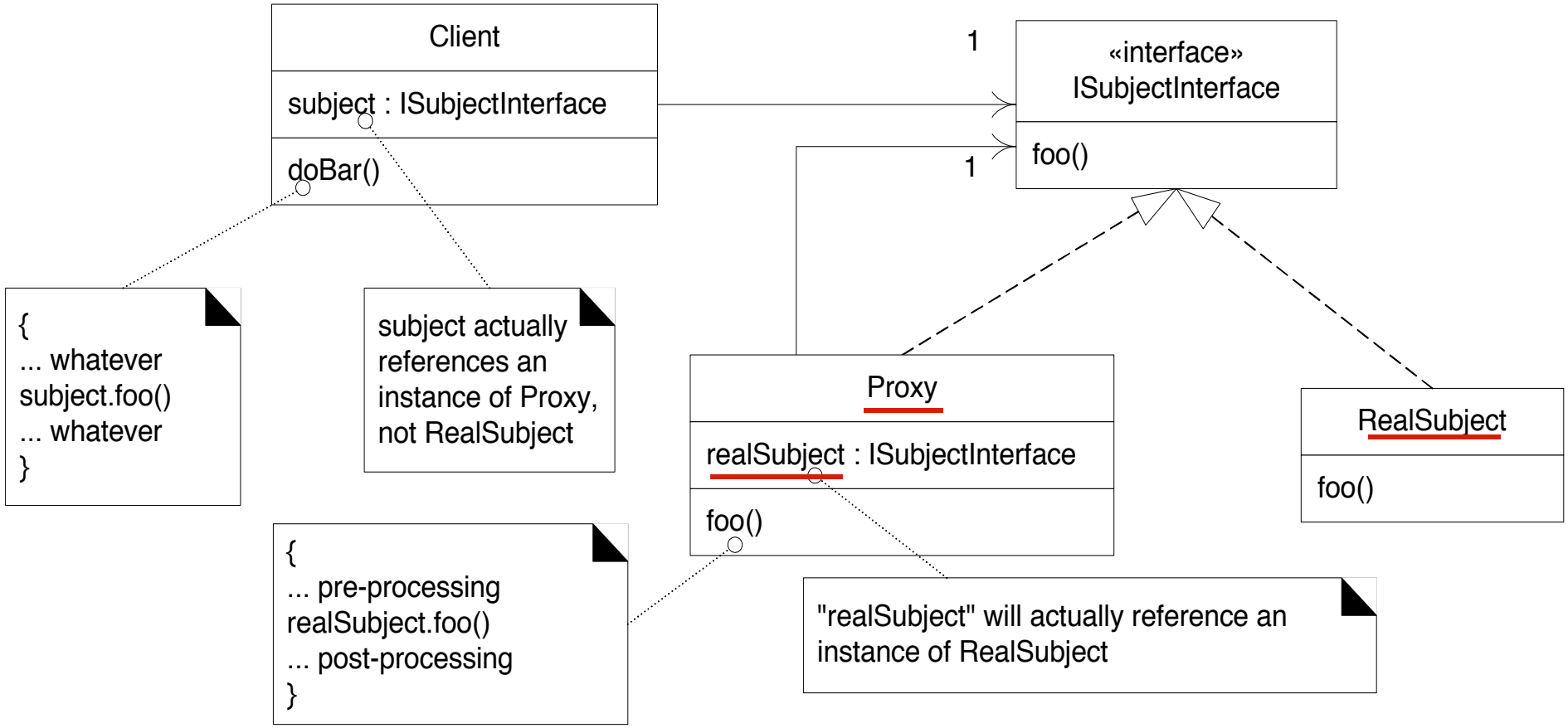
Proxy GoF Pattern

Problem: How do we control access to some *subject* object if we want to avoid giving direct access?



Solution: Add a level of indirection with a *proxy* object that implements the same methods as the *subject* and conditionally delegates to it.

Structure of the Proxy Pattern

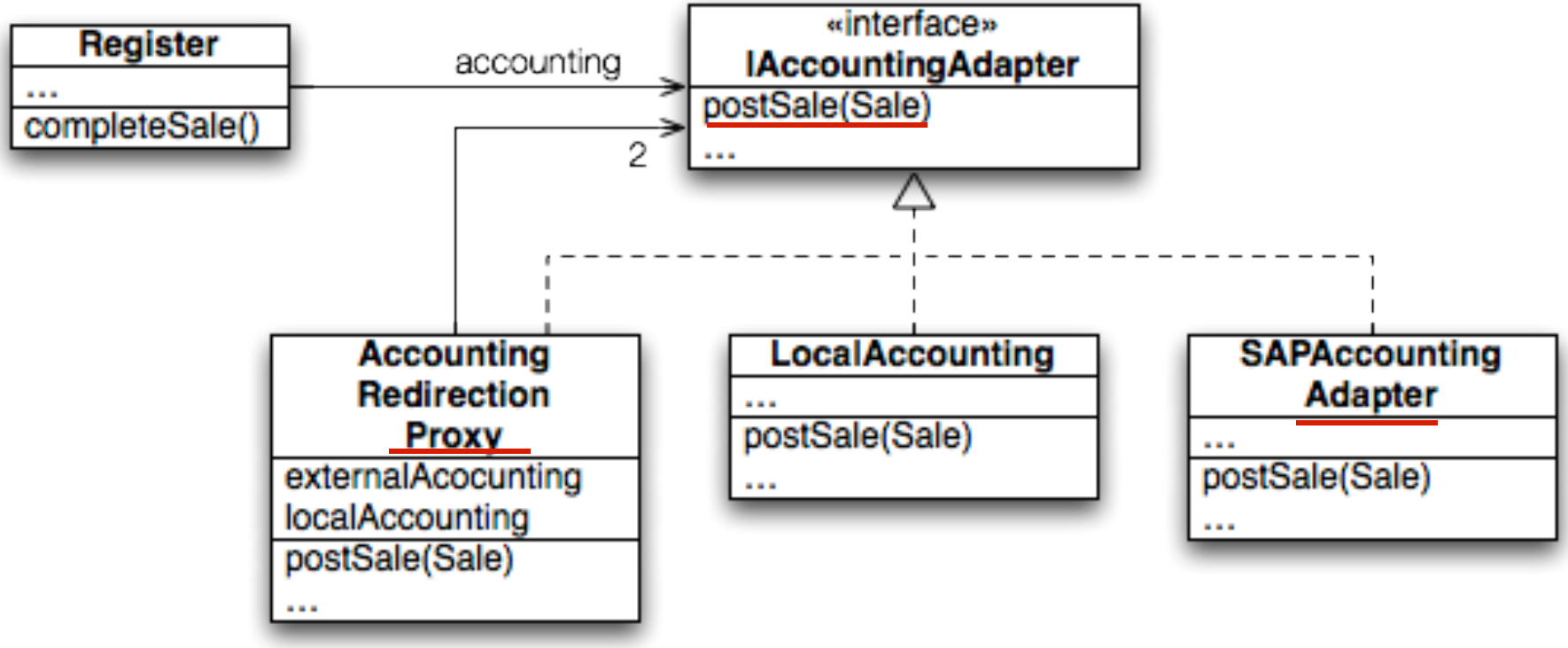


Proxy in NextGen POS

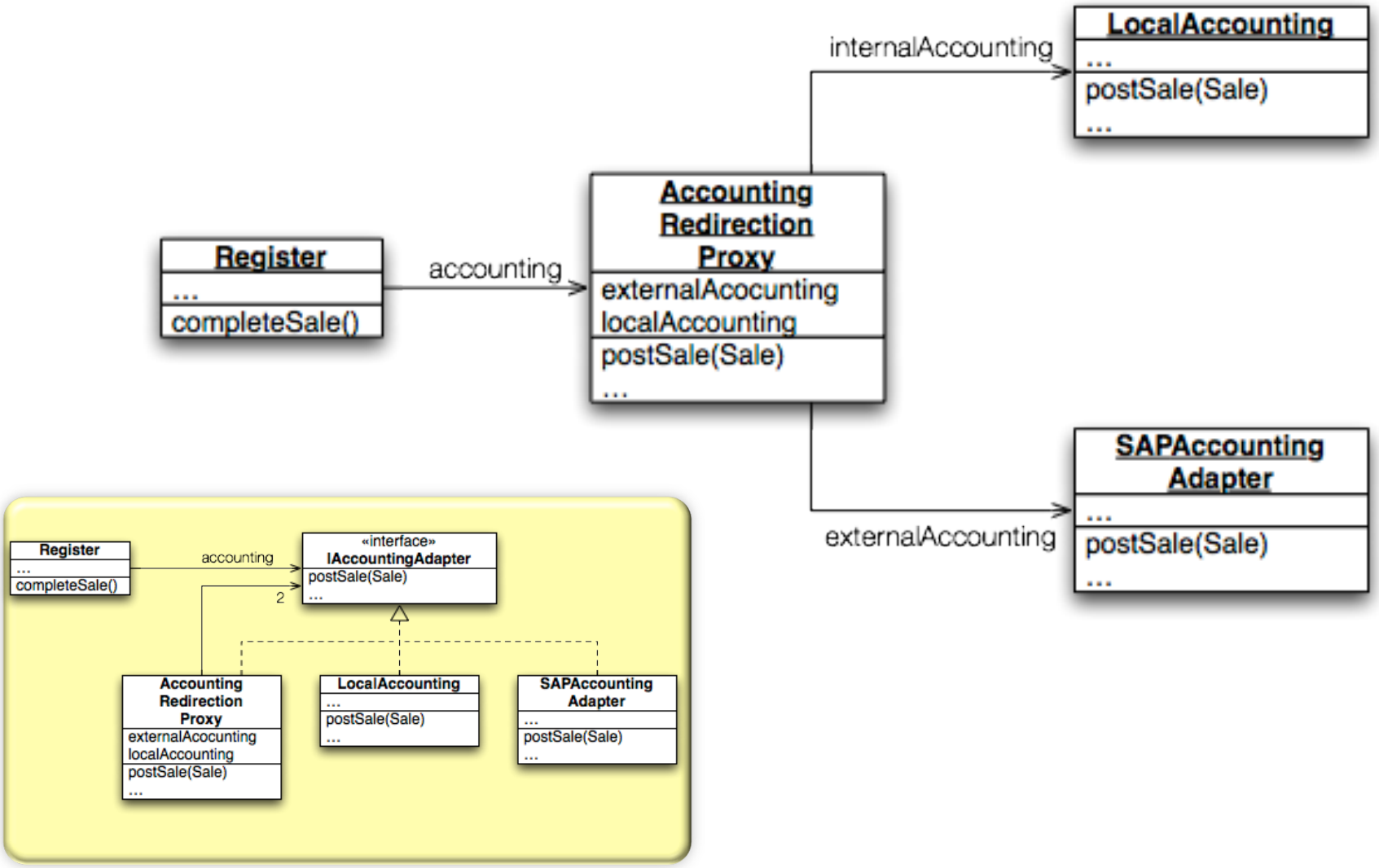
Posting sales to the accounting service

- Send *postSale(Sale)* to a *redirection proxy*
- *Proxy* attempts to post to external service
  - If it fails, then *proxy* stores result locally

Proxy in NextGen POS — DCD



Proxy in NextGen POS: Object Diagram



Exercise: Proxy for Failover

- Break up into your teams
- Consider how NextGen POS can use a Proxy to failover to local storage if the remote accounting service is down.
- Sketch a communication diagram depicting the above situation.

Design Studio Calendar

	Monday	Tuesday	Thursday
8th week		Team 2.4	Team 2.1
9th week	Team 2.2	Today Team 2.3	Team 2.5
10th week	Team 2.4	Team 2.1	Course Wrap-up

Homework and Milestone Reminders

- Read Chapter 37
- Milestone 5 – Final Junior Project System and Design
  - Preliminary Design Walkthrough on Friday, February 11th, 2011 during weekly project meeting
  - Final due by 11:59pm on Friday, February 18th, 2011
- Team 2.5 Design Studio