More Object Design with GoF Patterns

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

Today’s Agenda

- More Object Design with GoF Patterns
  - More tomorrow
- Design Studio:
  Team 12 – XNA Games

Applying Patterns to NextGen POS Iteration 3

- Local caching
- Failover to local services
- Support for third-party POS devices
- Handling payments

Failover & Performance with Local Caching

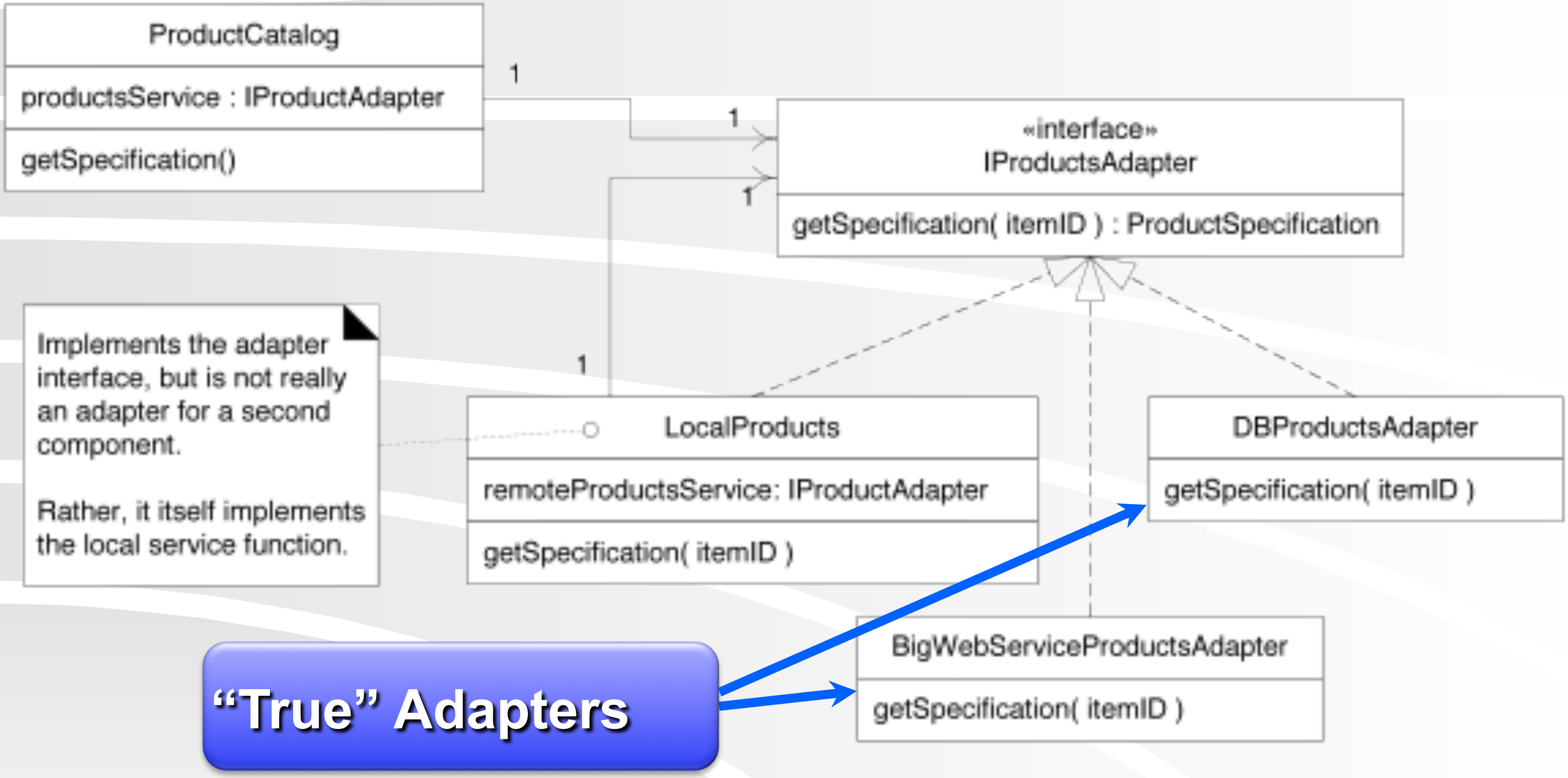
- What is a *cache*?
  How does a cache usually work?

- 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



ProductCatalog

productsService : IProductAdapter

getSpecification()

«interface»
IProductsAdapter

getSpecification(itemID) : ProductSpecification

LocalProducts

remoteProductsService: IProductAdapter

getSpecification(itemID)

DBProductsAdapter

getSpecification(itemID)

BigWebServiceProductsAdapter

getSpecification(itemID)

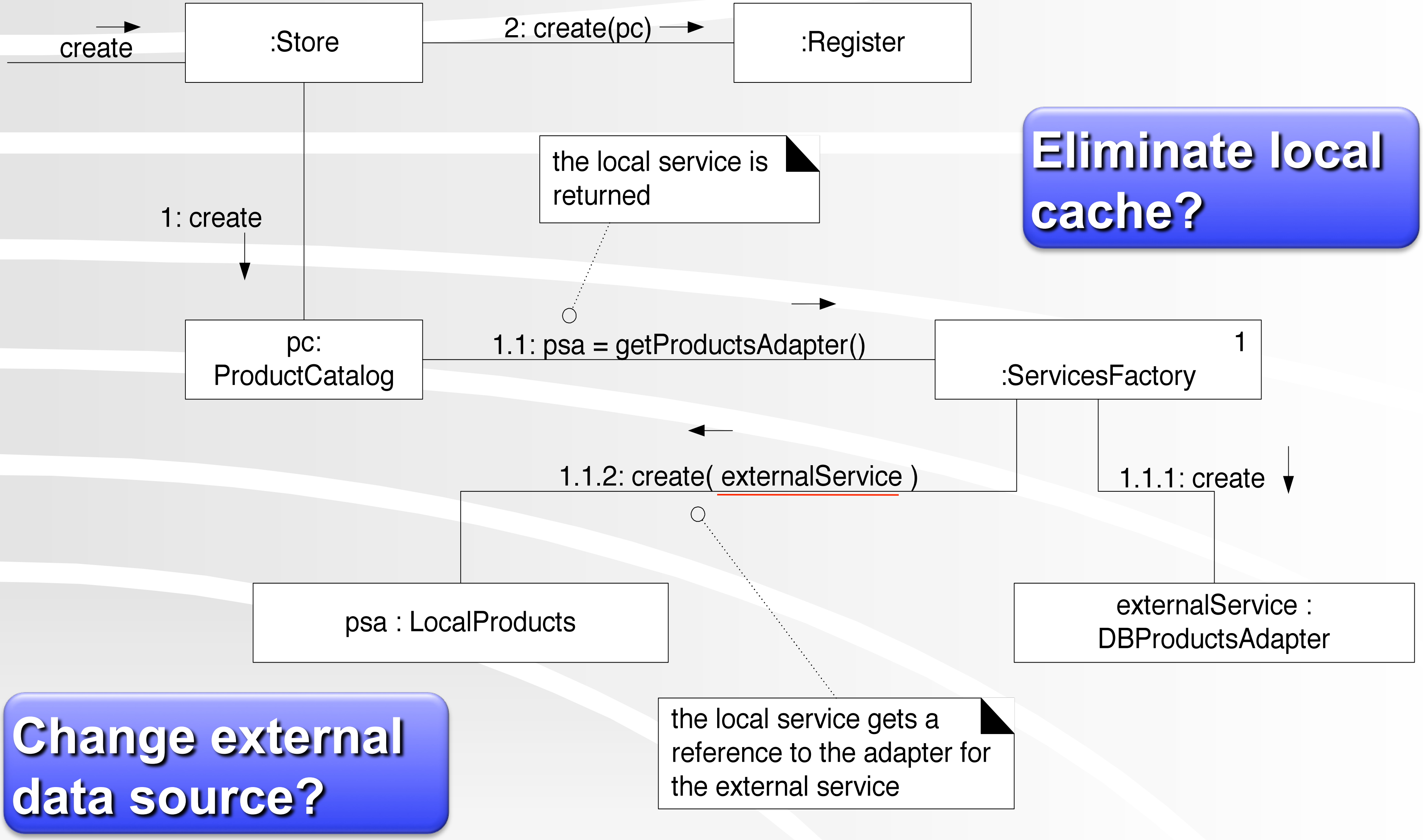
Implements the adapter interface, but is not really an adapter for a second component.

Rather, it itself implements the local service function.

“True” Adapters

Q1

Using a Factory to Set Up for Local Caching

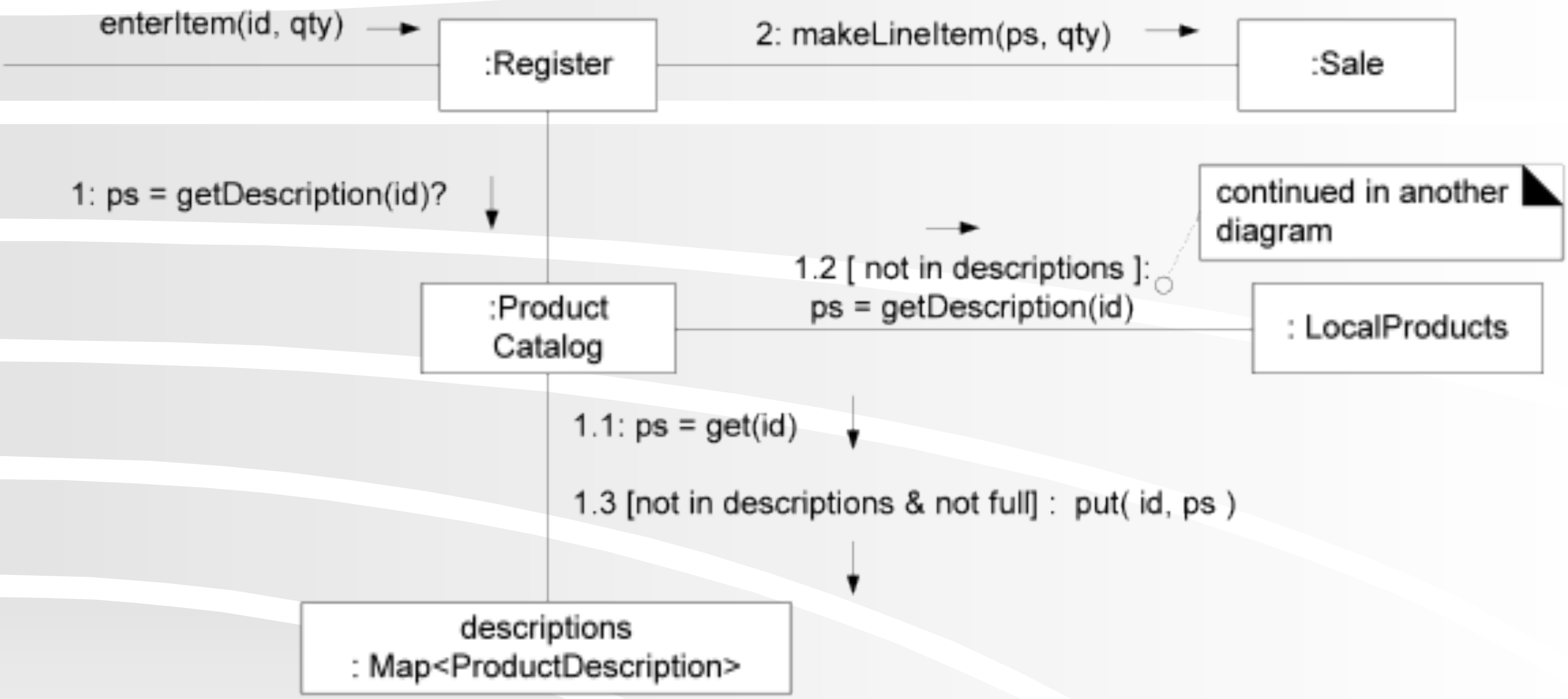

create
:Store
2: create(pc)
:Register
1: create
the local service is returned
pc:
ProductCatalog
1.1: psa = getProductsAdapter()
1
:ServicesFactory
1.1.2: create(externalService)
1.1.1: create
psa : LocalProducts
externalService :
DBProductsAdapter
the local service gets a reference to the adapter for the external service

Eliminate local cache?

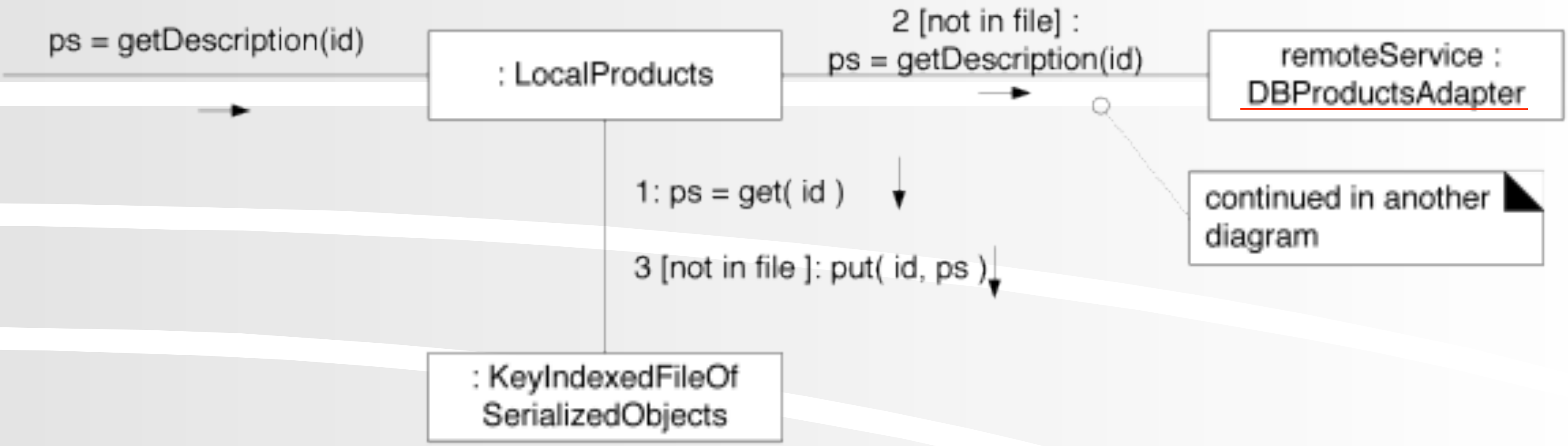
Change external data source?

Q2

Product Lookup with In-memory Miss


enterItem(id, qty)
:Register
2: makeLineItem(ps, qty)
:Sale
1: ps = getDescription(id)?
continued in another diagram
1.2 [not in descriptions]:
ps = getDescription(id)
:Product Catalog
: LocalProducts
1.1: ps = get(id)
1.3 [not in descriptions & not full] : put(id, ps)
descriptions
: Map<ProductDescription>

Product Lookup with Local Cache Miss



ps = getDescription(id)

: LocalProducts

2 [not in file] :
ps = getDescription(id)

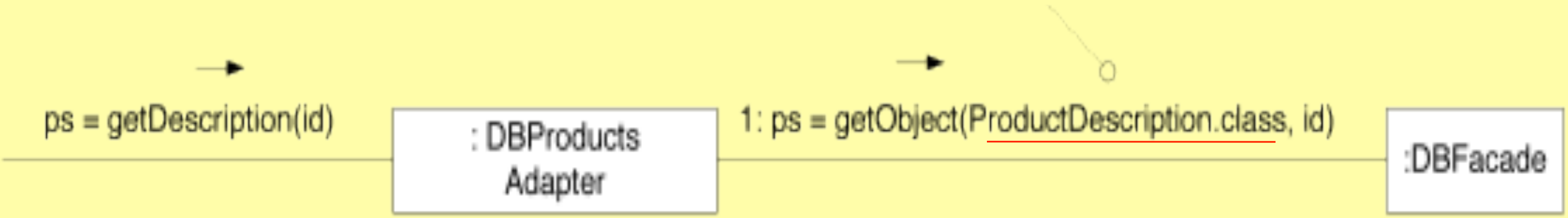
remoteService :
DBProductsAdapter

1: ps = get(id)

3 [not in file]: put(id, ps)

continued in another diagram

: KeyIndexedFileOf
SerializedObjects



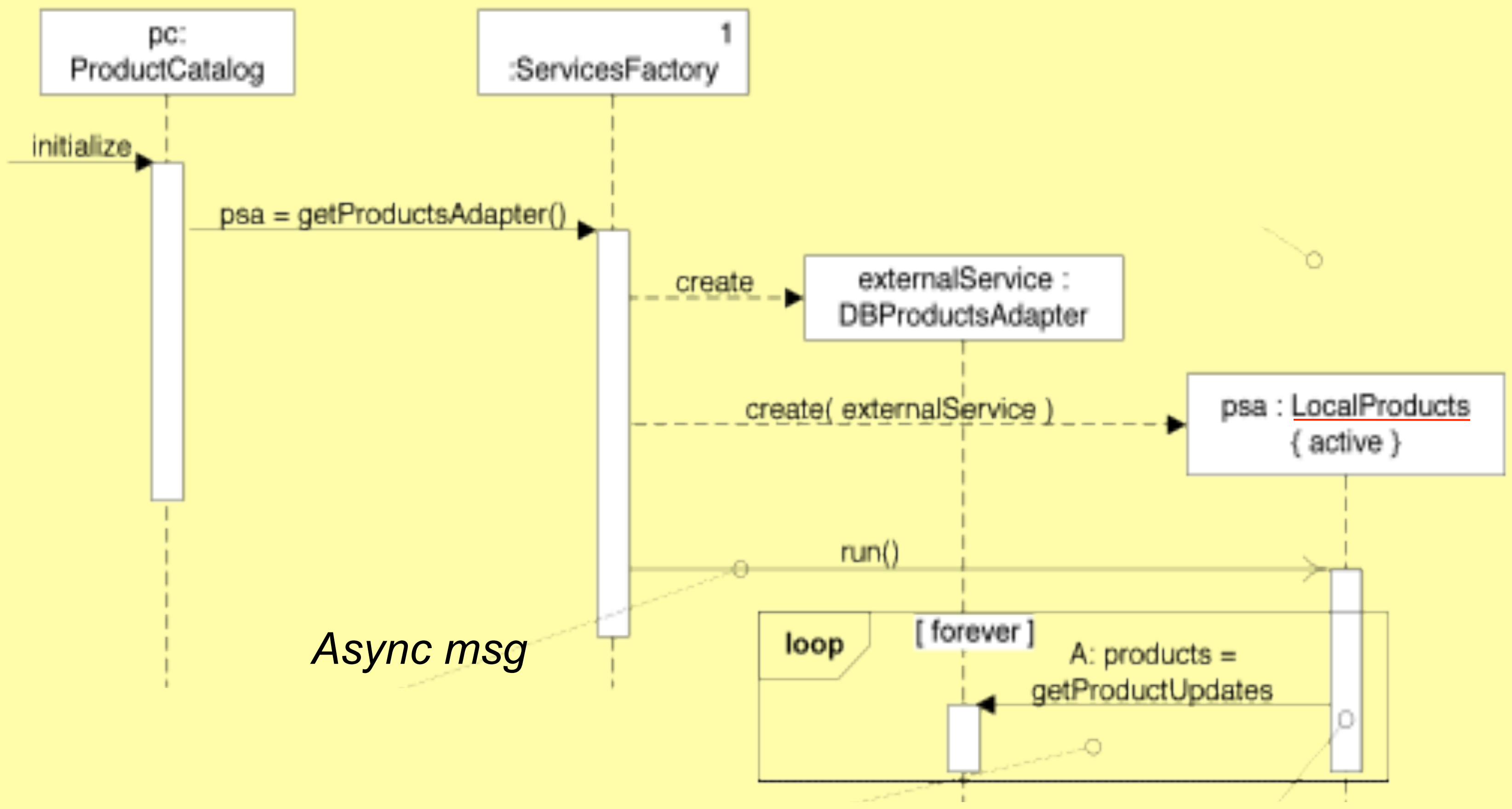
ps = getDescription(id)

: DBProducts
Adapter

1: ps = getObject(ProductDescription.class, id)

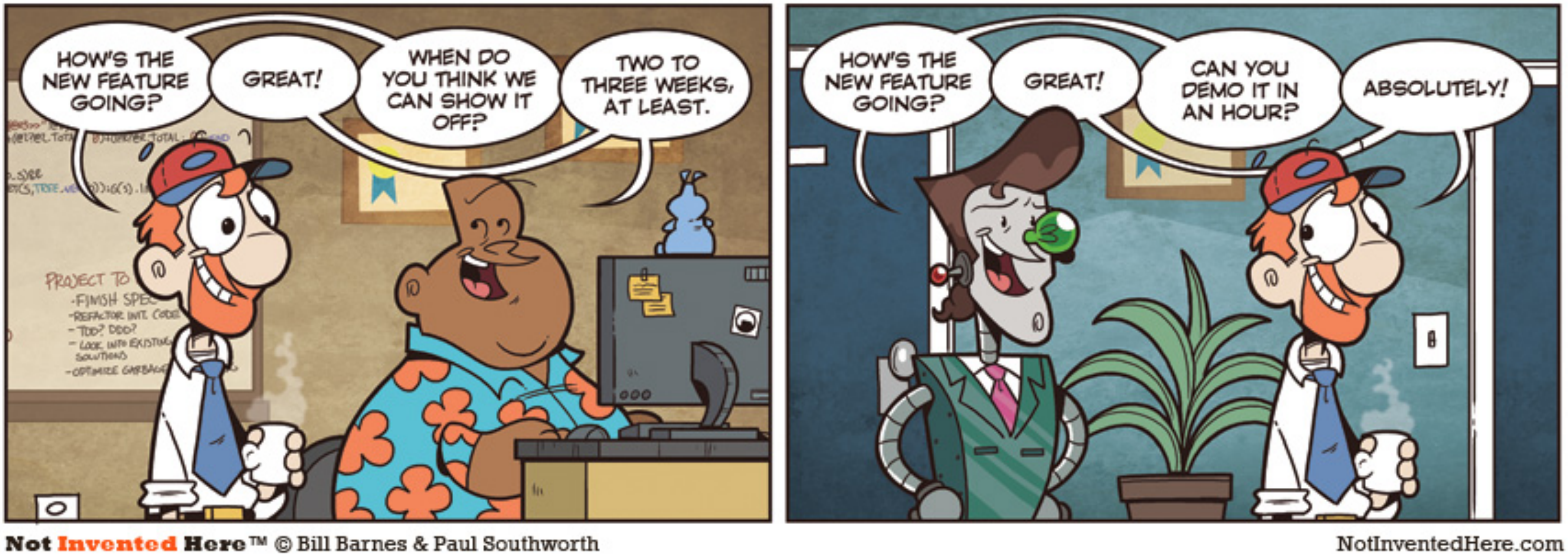
:DBFacade

Using Threads to Freshen the Cache



Async msg

How’s the final iteration going?

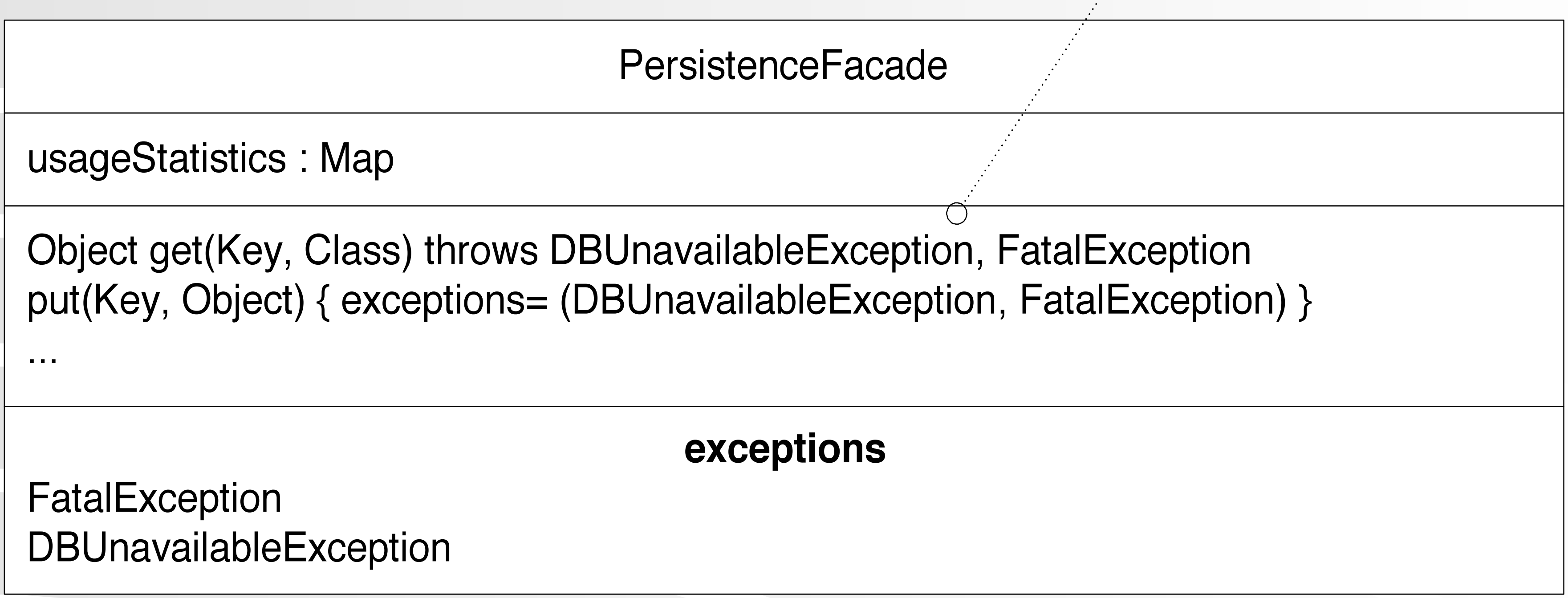

HOW'S THE NEW FEATURE GOING?
GREAT!
WHEN DO YOU THINK WE CAN SHOW IT OFF?
TWO TO THREE WEEKS, AT LEAST.
PROJECT TO
-FINISH SPEC
-REFACTOR INIT CODE
-TDD? DDD?
-LOOK INTO EXISTING SOLUTIONS
-OPTIMIZE GARBAGE
HOW'S THE NEW FEATURE GOING?
GREAT!
CAN YOU DEMO IT IN AN HOUR?
ABSOLUTELY!
Not Invented Here™ © Bill Barnes & Paul Southworth
NotInventedHere.com

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?

DCD: Exceptions Caught and Thrown



PersistenceFacade

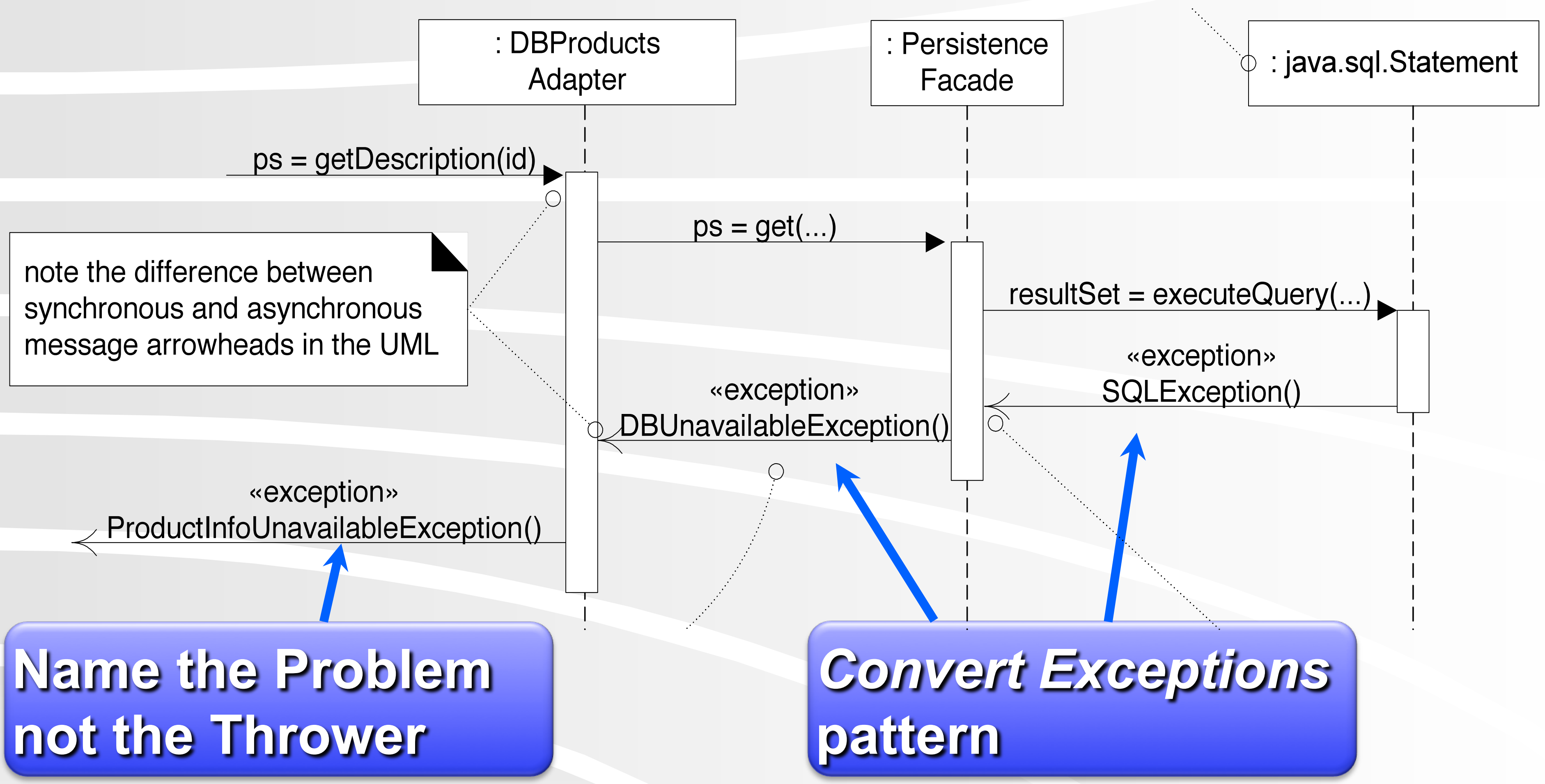
usageStatistics : Map

Object get(Key, Class) throws DBUnavailableException, FatalException
put(Key, Object) { exceptions= (DBUnavailableException, FatalException) }
...

exceptions

FatalException
DBUnavailableException

Showing Exception in Sequence Diagrams



: DBProducts Adapter

: Persistence Facade

: java.sql.Statement

ps = getDescription(id)

ps = get(...)

resultSet = executeQuery(...)

«exception» SQLException()

«exception» DBUnavailableException()

«exception» ProductInfoUnavailableException()

note the difference between synchronous and asynchronous message arrowheads in the UML

Name the Problem not the Thrower

Convert Exceptions **pattern**

Q3

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

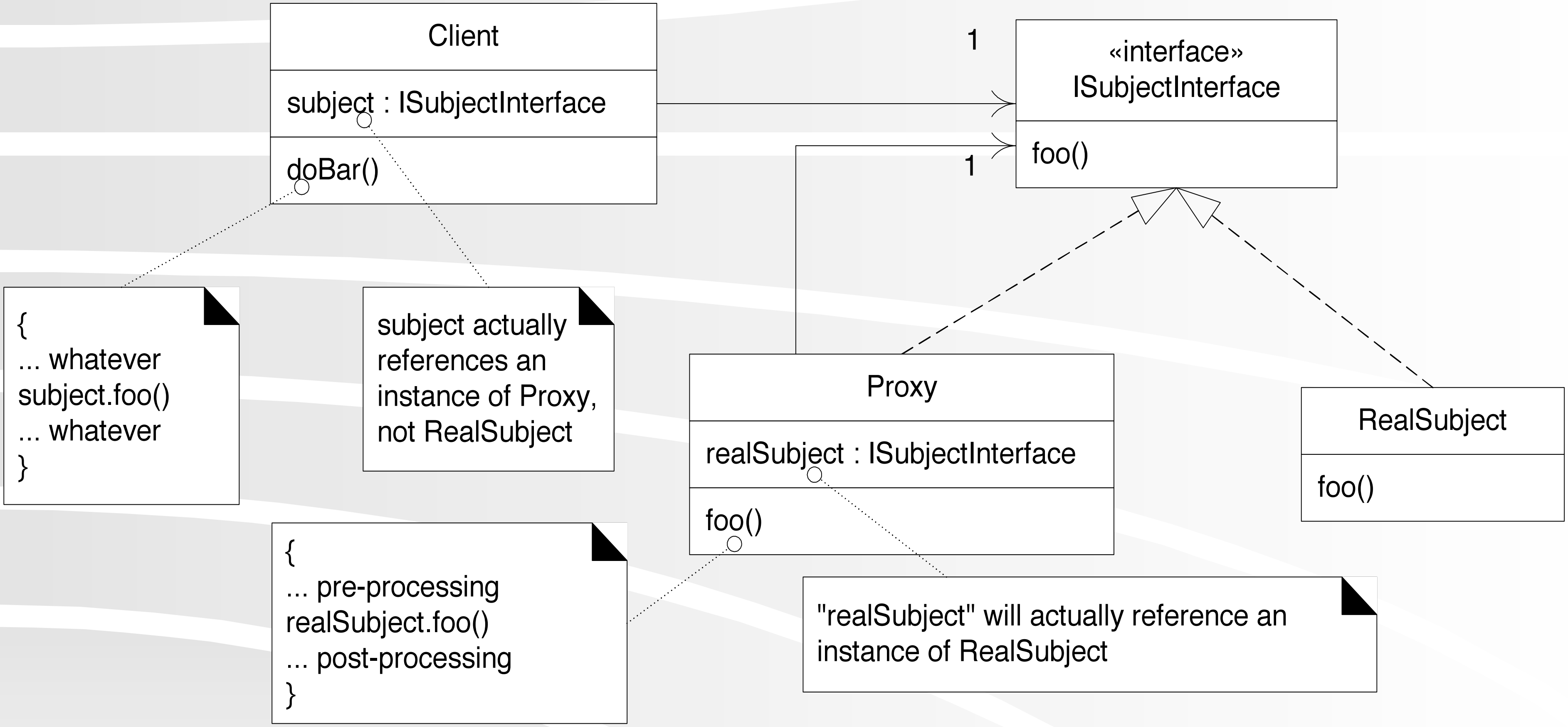
Proxy GoF Pattern

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.

Q4,5

Structure of the Proxy Pattern



{
... whatever
subject.foo()
... whatever
}

subject actually references an instance of Proxy, not RealSubject

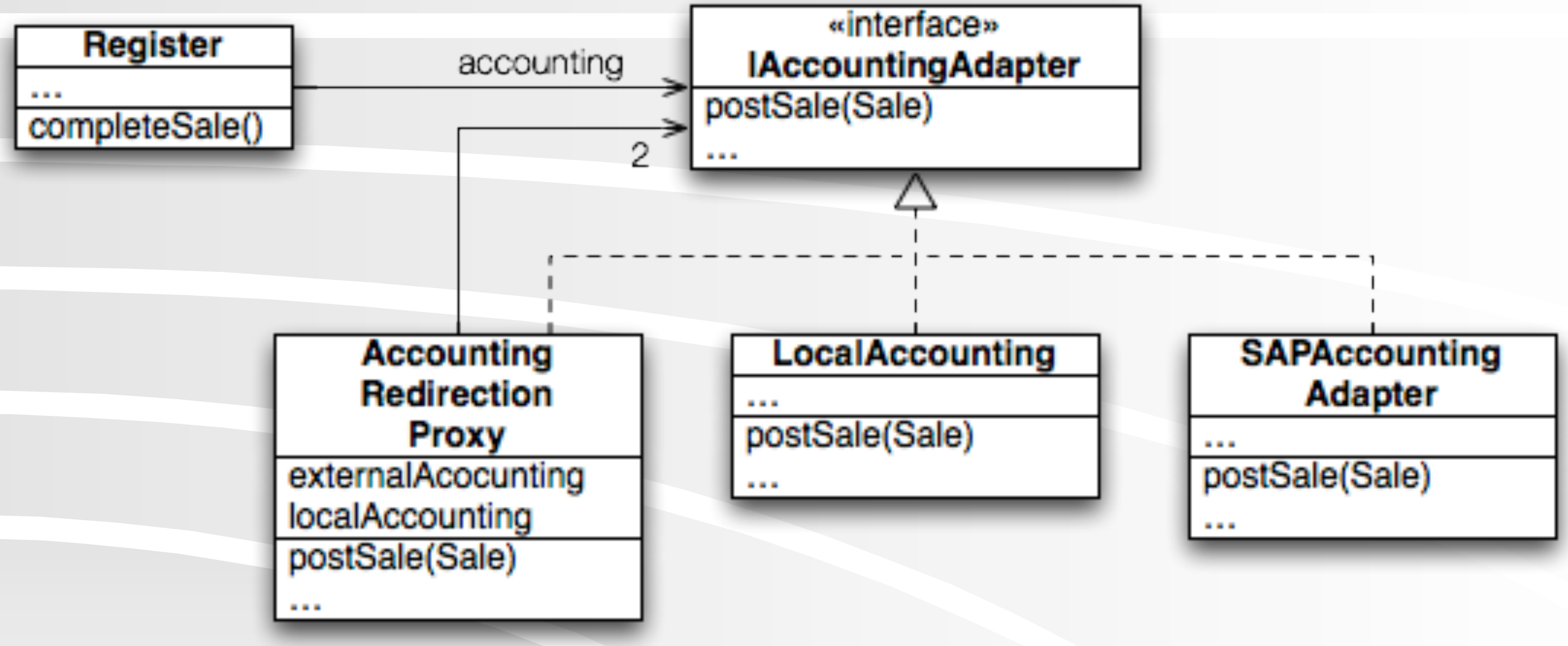
{
... pre-processing
realSubject.foo()
... post-processing
}

"realSubject" will actually reference an instance of RealSubject

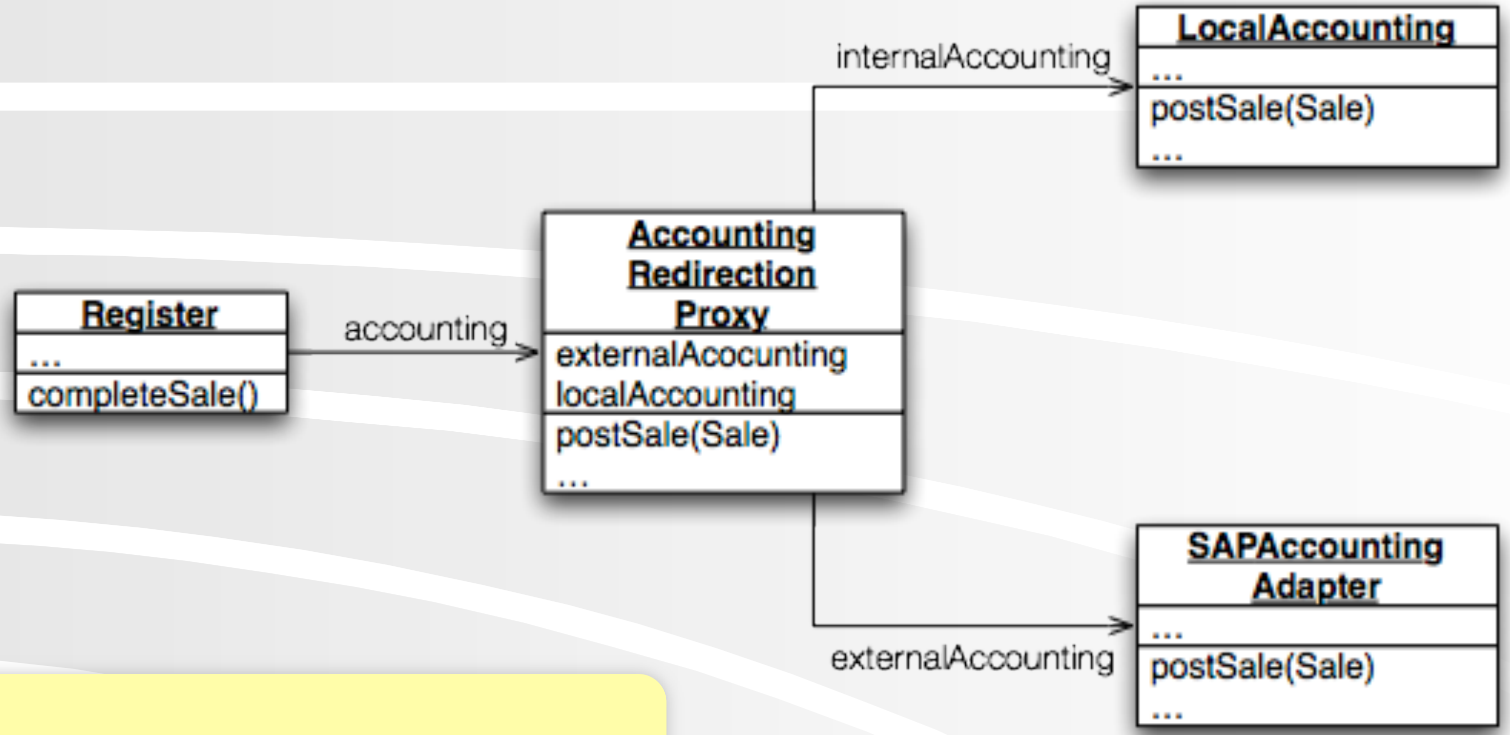
Proxy in NextGen POS

- 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



Register
...
completeSale()

accounting

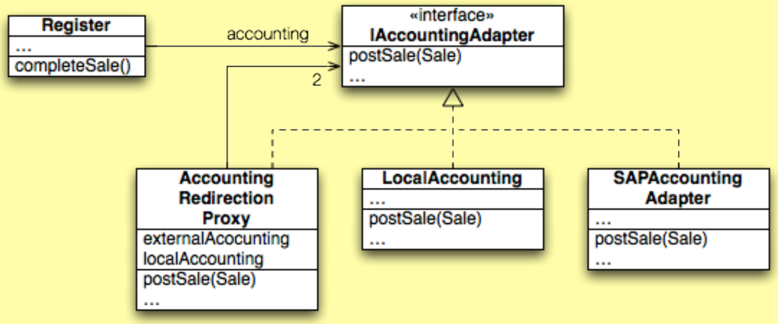
Accounting Redirection Proxy
externalAcocunting
localAccounting
postSale(Sale)
...

internalAccounting

LocalAccounting
...
postSale(Sale)
...

externalAccounting

SAPAccounting Adapter
...
postSale(Sale)
...



Register
...
completeSale()

accounting

2

«interface»
IAccountingAdapter
postSale(Sale)
...

Accounting Redirection Proxy
externalAcocunting
localAccounting
postSale(Sale)
...

LocalAccounting
...
postSale(Sale)
...

SAPAccounting Adapter
...
postSale(Sale)
...

Q6

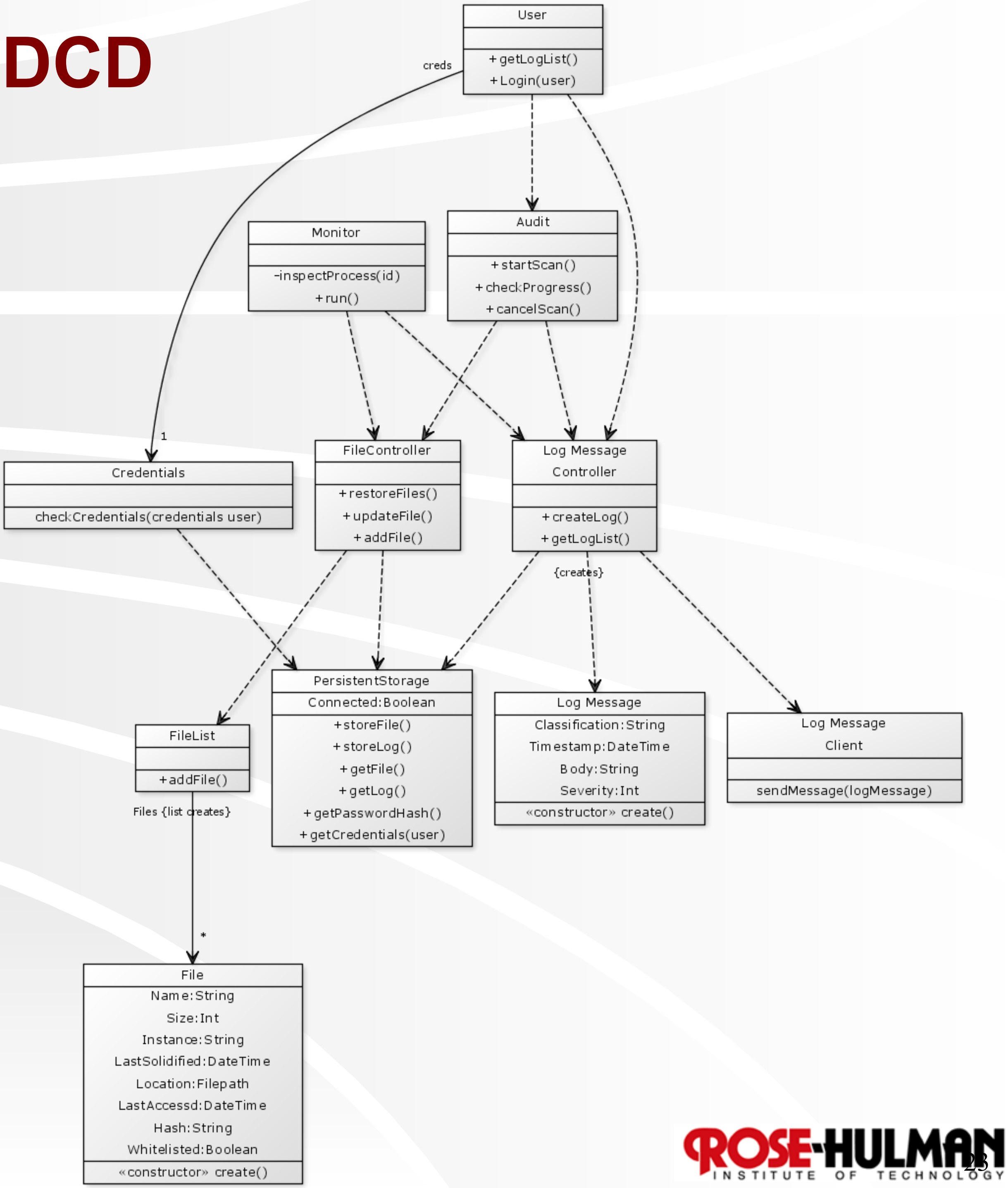
Design Studio:
Team 12: Security API

~5 minutes:
Team describes problem and current solution (if any)

~3 minutes:
Class thinks about questions, alternative approaches Q7

~12 minutes:
On-board design with team modeling and instructor advising/facilitating

Security API DCD



User
+getLogList()
+Login(user)

Monitor
-inspectProcess(id)
+run()

Audit
+startScan()
+checkProgress()
+cancelScan()

Credentials
checkCredentials(credentials user)

FileController
+restoreFiles()
+updateFile()
+addFile()

Log Message Controller
+createLog()
+getLogList()

FileList
+addFile()

PersistentStorage
Connected:Boolean
+storeFile()
+storeLog()
+getFile()
+getLog()
+getPasswordHash()
+getCredentials(user)

Log Message
Classification:String
Timestamp:DateTime
Body:String
Severity:Int
«constructor» create()

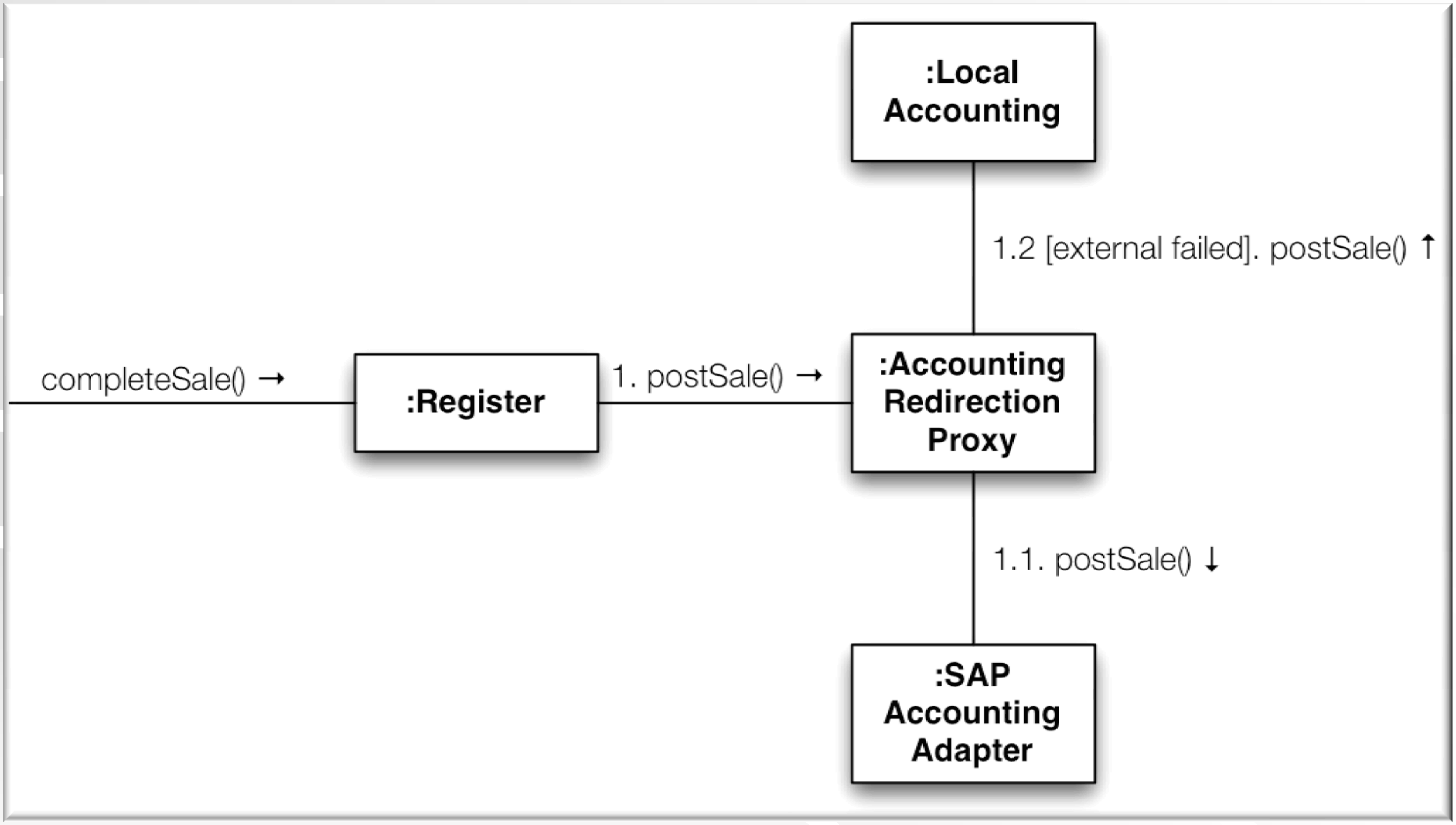
Log Message Client
sendMessage(logMessage)

File
Name:String
Size:Int
Instance:String
LastSolidified:DateTime
Location:Filepath
LastAccessd:DateTime
Hash:String
Whitelisted:Boolean
«constructor» create()

Homework and Milestone Reminders

- Read Chapter 37
- Milestone 5 – Iteration 3 Junior Project System with finalized Design Document
  - Preliminary Design Walkthrough on Friday, February 12th, 2010 during project meeting.
  - Final Project Due by 11:59pm Friday, February 19th, 2010.

CD for Proxy to failover to local storage if the remote accounting service is down


:Local Accounting
1.2 [external failed]. postSale() ↑
completeSale() →
:Register
1. postSale() →
:Accounting Redirection Proxy
1.1. postSale() ↓
:SAP Accounting Adapter