CSSE 374: Persistent Frameworks with GoF Design Patterns & Deployment Diagrams

Shawn Bohner

Office: Moench Room F212

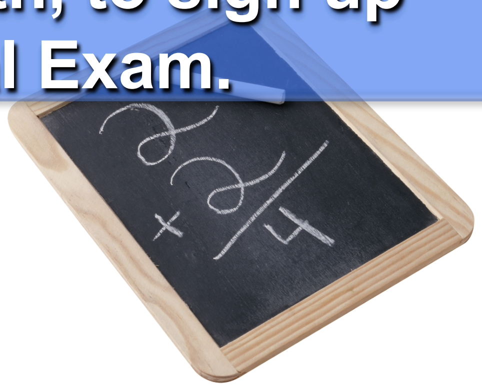
Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Optional Final Exam

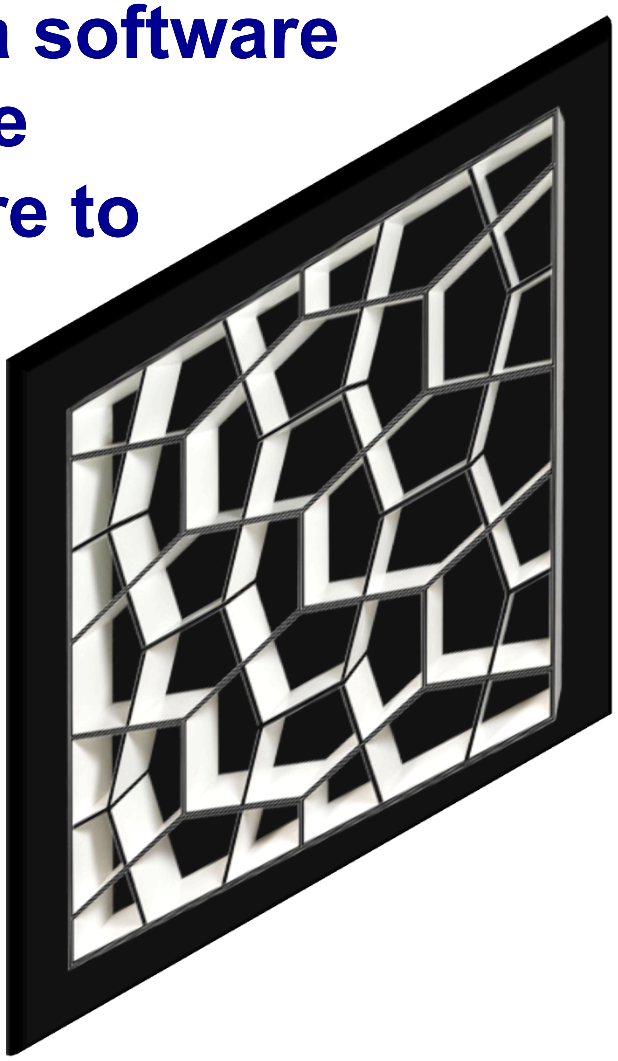
Email me by tomorrow, Feb. 15th, to sign up for Final Exam.



- 8:00am on Wednesday, Feb. 23rd
  - Room G317
- If you don't take the exam, we'll use your first exam grade as your final exam grade
- Sign-up for exam by Tuesday of 10th week
  - If you sign-up, you must take the exam
  - Taking the exam can improve or lower your grade

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
  - Finish up Template Pattern
  - State Pattern
  - Command Pattern
- Deployment Diagrams
- Design Studio with Team 2.5

Persistence Framework – a service to provide object to record mapping

In a Persistence Framework a record is to an object, as a __________ is to a graphical object in a GUI Framework.

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

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Recall: A Persistence Framework

Domain Layer

Persistence Framework

PersistenceFaçade

get(OID, class):Object
put(OID, object)

Store object in RDB

put(OID, Butler U.)

Relational Database

Name	City
RHIT	Terre Haute
Purdue	W. Lafayette
Indiana U.	Bloomington
Butler U.	Indianapolis

University Table

Recall: Maps between Persistent Object & Database

1

:University

name = Butler

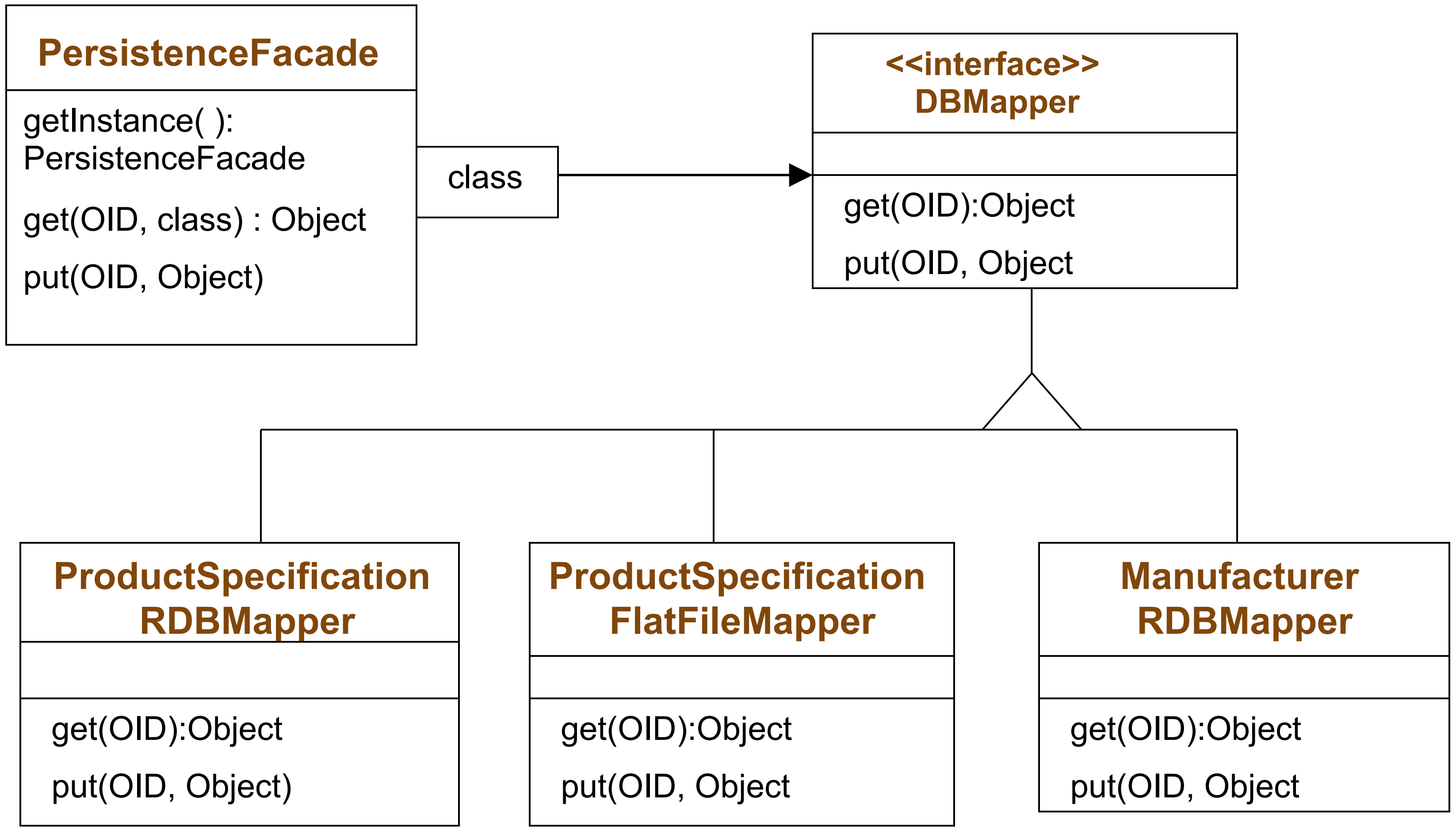
city = Indianapolis

oid = xyz123

University Table

OID	name	city
XI001	RHIT	Terre Haute
wxx246	Purdue	W. Lafayette
xxz357	Indiana U.	Bloomington
xyz123	Butler U.	Indianapolis

Recall: Façade Design Pattern w/Brokers

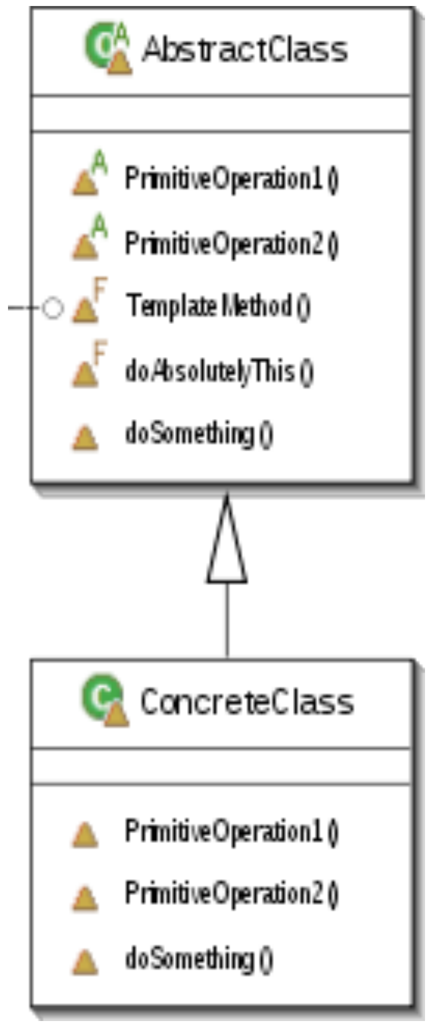


Each mapper gets and puts objects in its own unique way, depending on the kind of data store and format.

Recall: Template Method Pattern

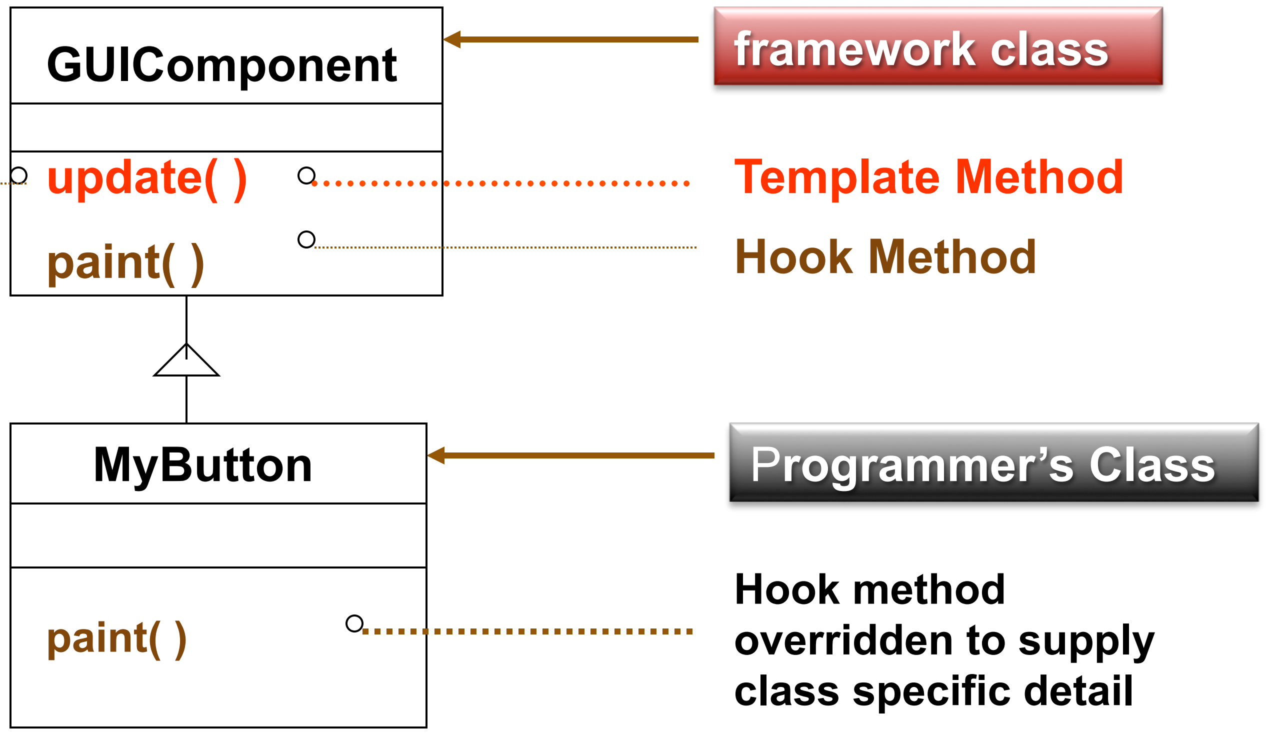
- Problem: How can we record the basic outline of an algorithm in a framework (or other) class, while allowing extensions to vary the specific behavior?

- Solution: Create a *template method* for the algorithm that calls (often abstract) helper methods for the steps. Subclasses can override/implement these helper methods to vary the behavior.

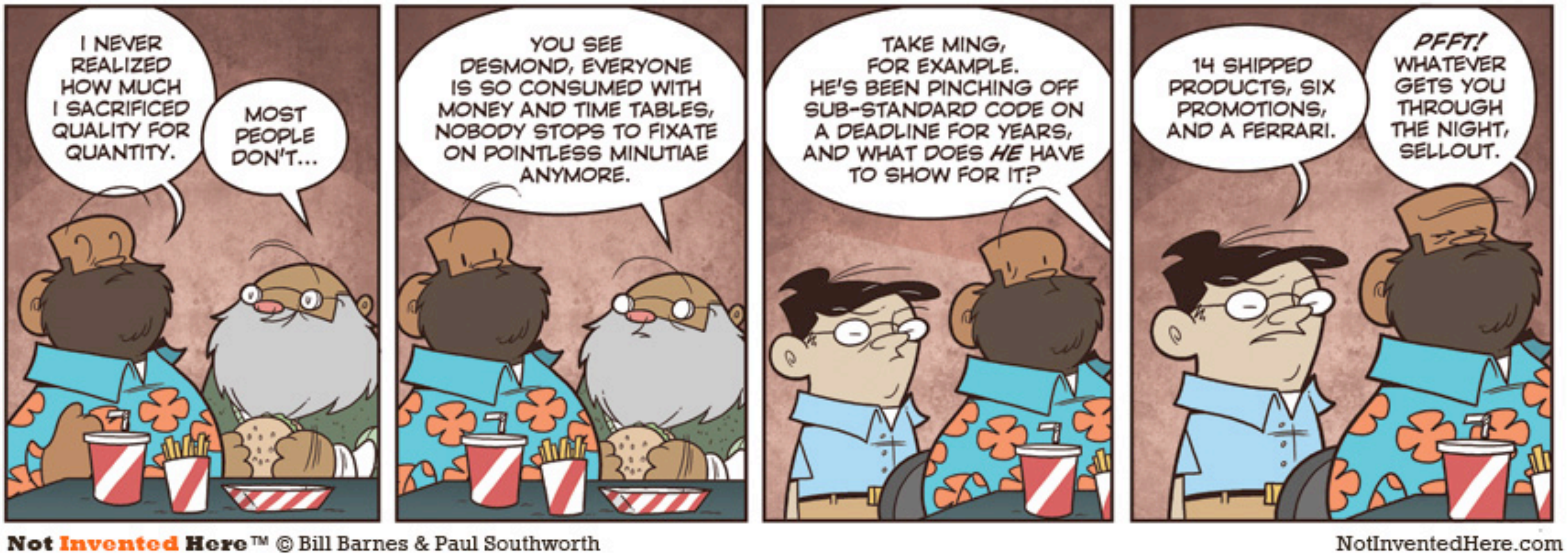


Recall Example: Template Method used for Swing GUI Framework

```
//unvarying part of algorithm
public void update {
    clearBackground( );
    //call the hook method
    paint( );
}
```

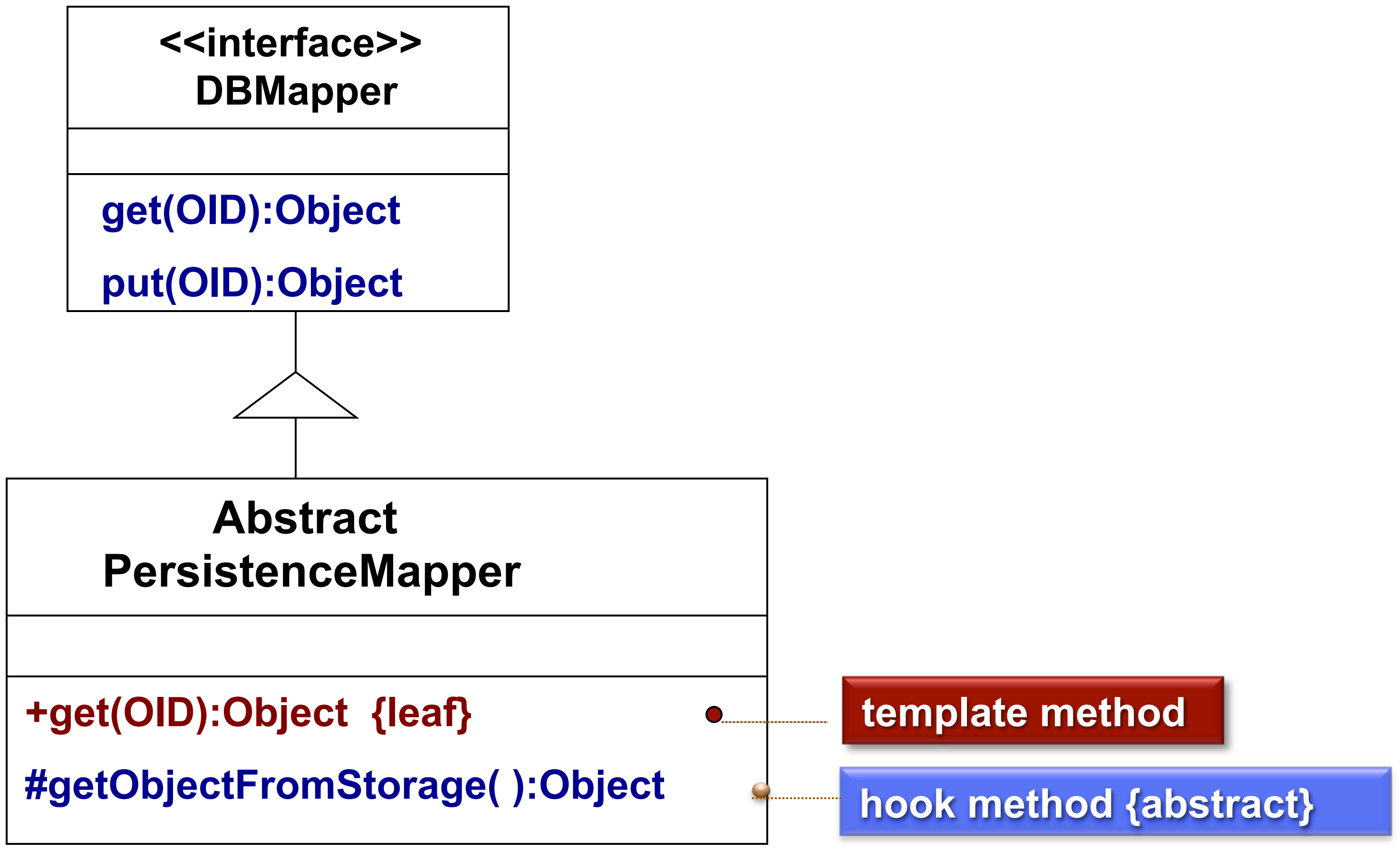


Sacrificing Quality for Quantity…


I NEVER REALIZED HOW MUCH I SACRIFICED QUALITY FOR QUANTITY.
MOST PEOPLE DON'T...
YOU SEE DESMOND, EVERYONE IS SO CONSUMED WITH MONEY AND TIME TABLES, NOBODY STOPS TO FIXATE ON POINTLESS MINUTIAE ANYMORE.
TAKE MING, FOR EXAMPLE. HE'S BEEN PINCHING OFF SUB-STANDARD CODE ON A DEADLINE FOR YEARS, AND WHAT DOES HE HAVE TO SHOW FOR IT?
14 SHIPPED PRODUCTS, SIX PROMOTIONS, AND A FERRARI.
PFFT! WHATEVER GETS YOU THROUGH THE NIGHT, SELLOUT.
Not Invented Here™ © Bill Barnes & Paul Southworth
NotInventedHere.com

- ***It's a bit like all you can eat fast food!***

Template Method in NexGen POS 1/2

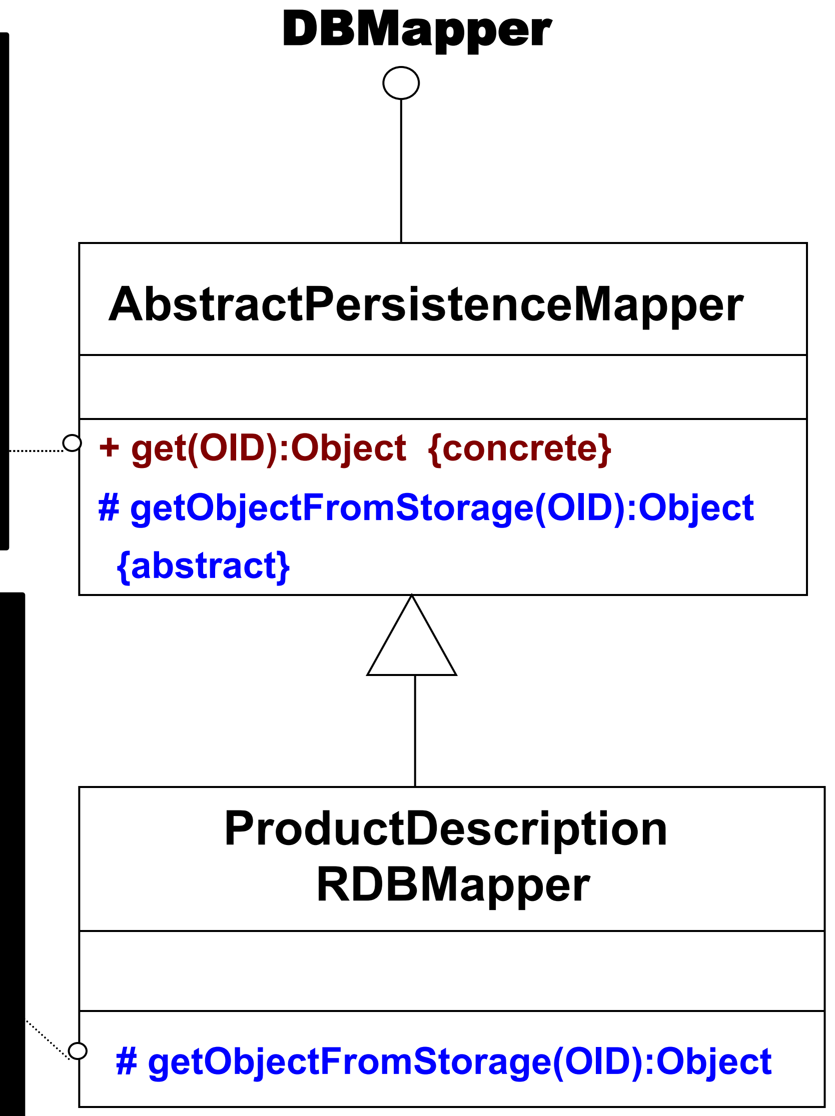


Q1

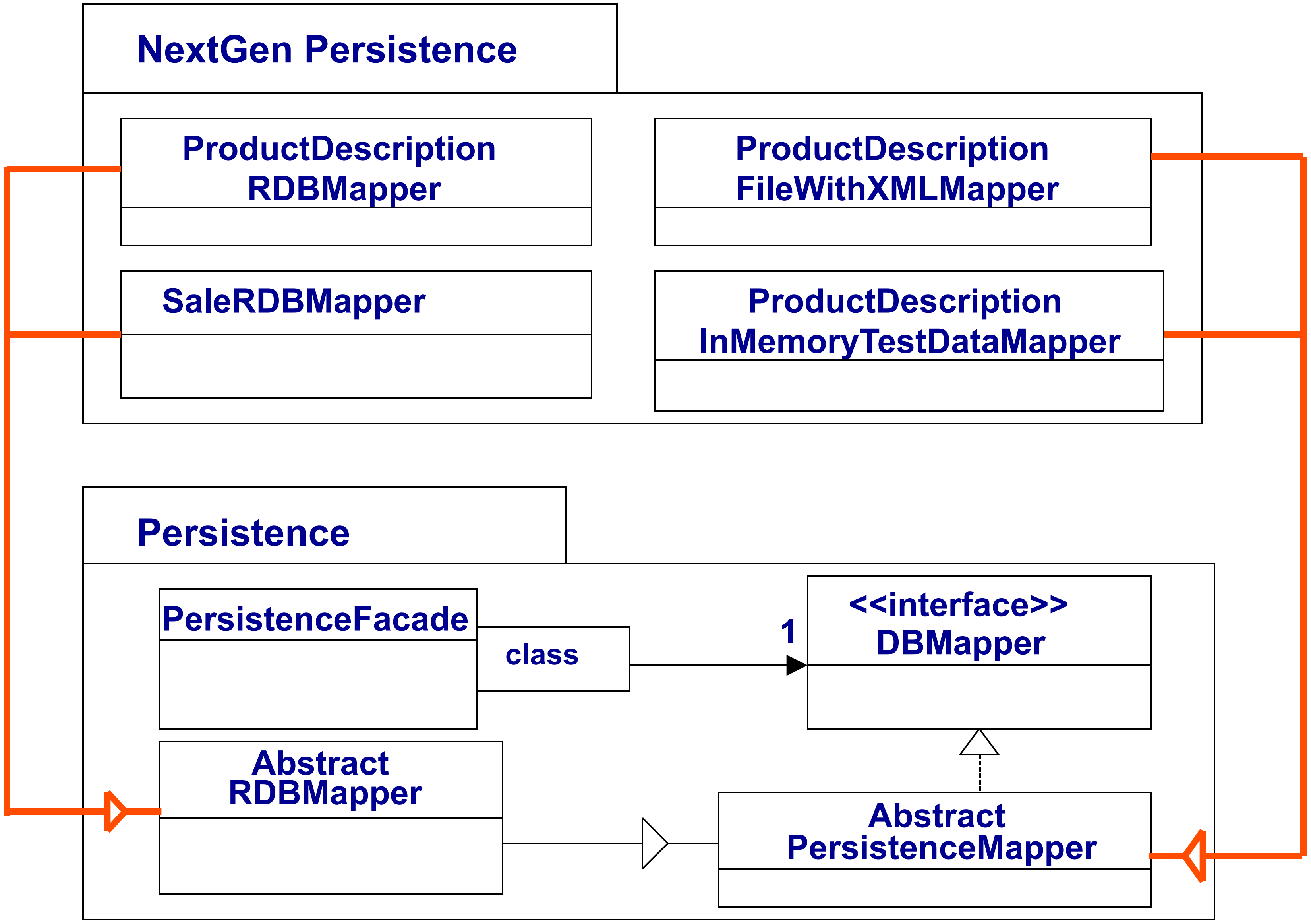
Template Method in NexGen POS 2/2

```
//template method
public final Object get(OID oid) {
     obj = cachedObjects.get(oid);
     if (obj == null) {
          //hook method
          obj = getObjectFromStorage(oid);
          cachedObject.put(oid, obj);  }
      return obj;  }
```

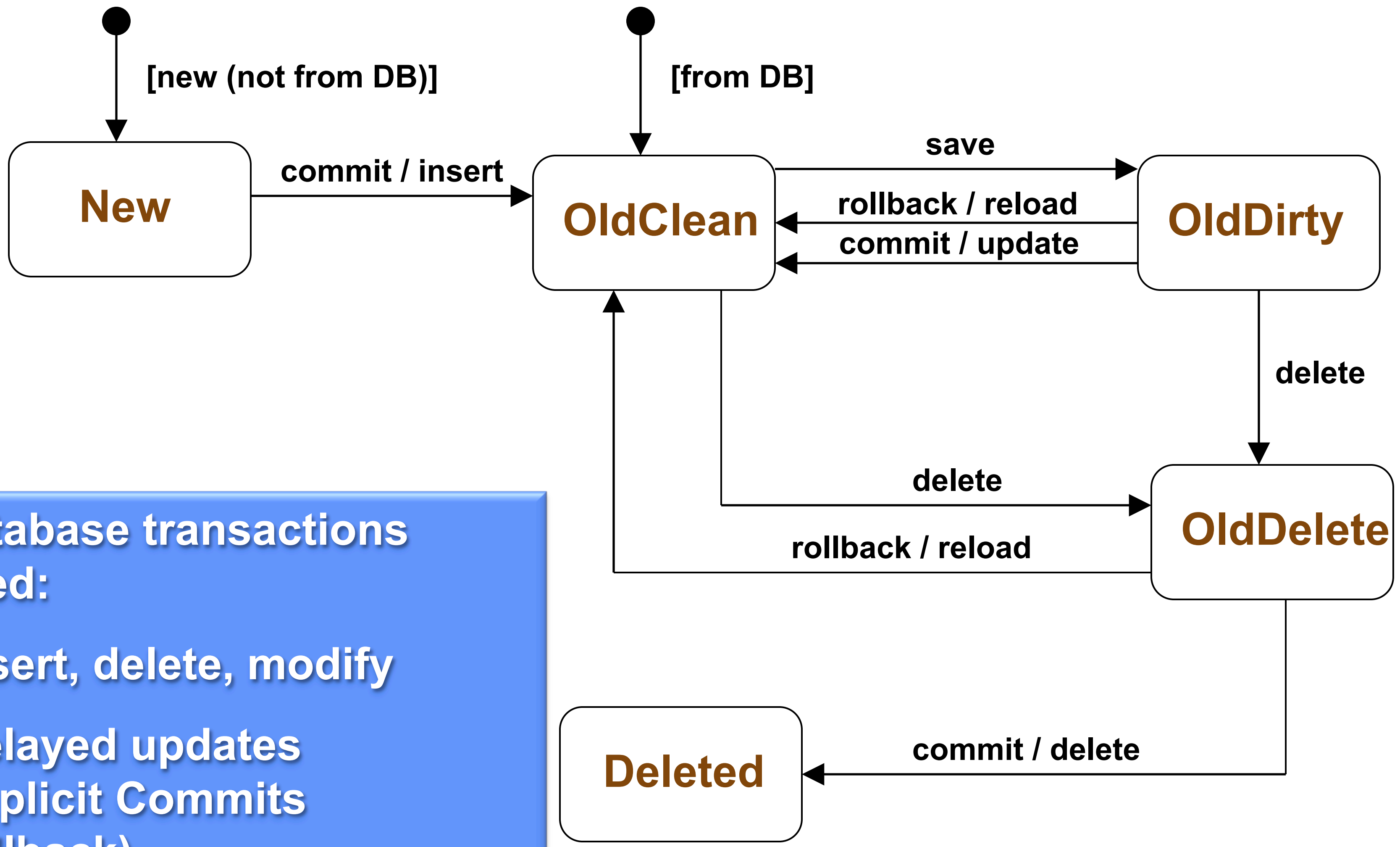
```
//hook method override
protected Object getObjectFromStorage(OID oid) {
     String key = oid.toString( );
     dbRec = SQL execution result of
          “Select* from PROD_DESC where key =“ +key
ProductDescription = new ProductDescription();
pd.setPrice(dbRec.getColumn(“PRICE”);
…etc
```


DBMapper
AbstractPersistenceMapper
+ get(OID):Object {concrete}
# getObjectFromStorage(OID):Object {abstract}
ProductDescription RDBMapper
# getObjectFromStorage(OID):Object

Persistence Framework


NextGen Persistence
ProductDescription RDBMapper
ProductDescription FileWithXMLMapper
SaleRDBMapper
ProductDescription InMemoryTestDataMapper
Persistence
PersistenceFacade
class
1
<<interface>> DBMapper
Abstract RDBMapper
Abstract PersistenceMapper

Transactional States & the State Pattern

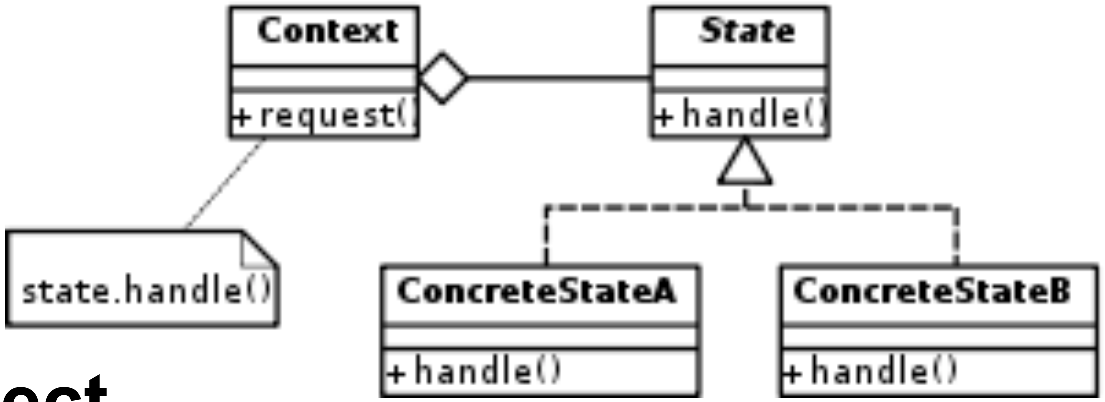


Database transactions need:

-insert, delete, modify

-Delayed updates /Explicit Commits (rollback)

State Pattern

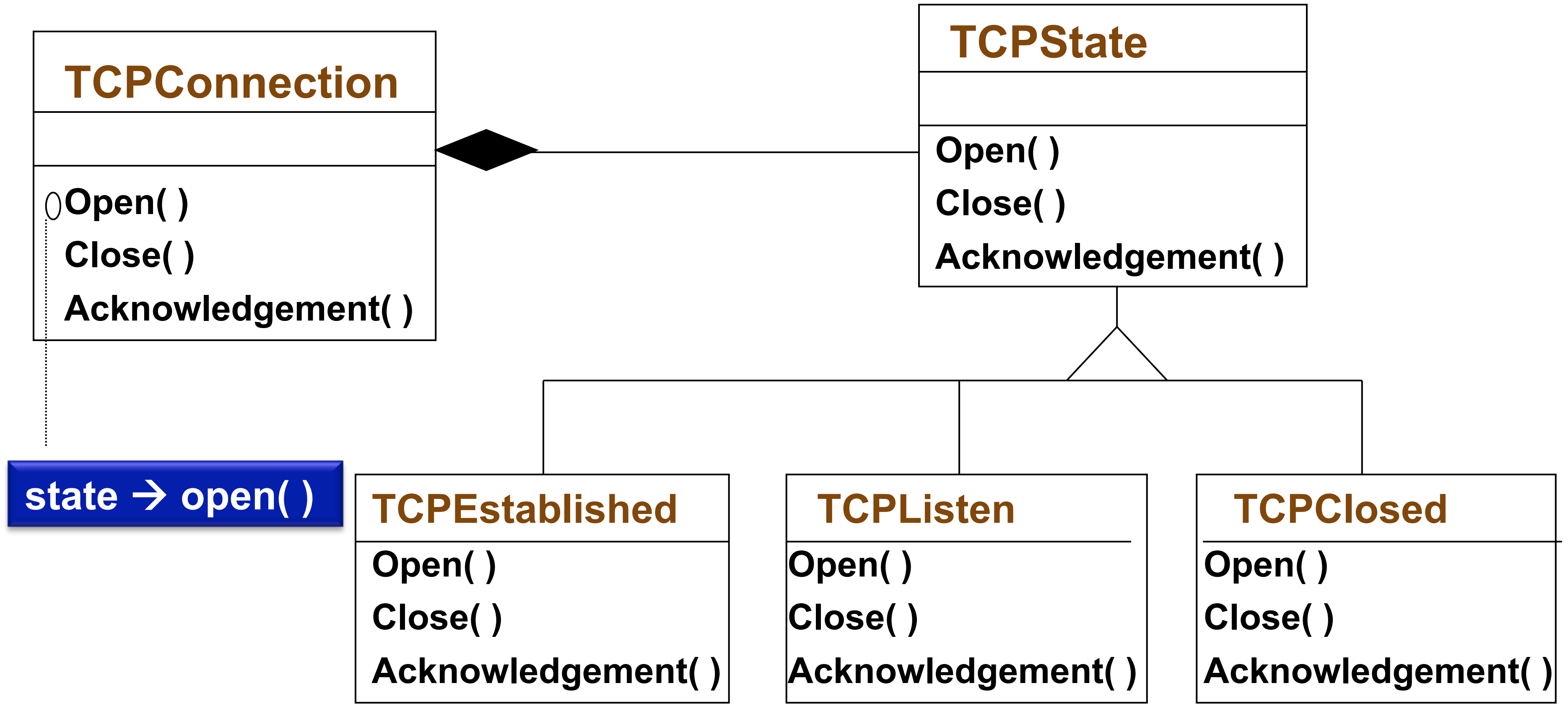


Problem: When the behavior of an object, *obj*, changes depending on its state, how can we avoid complicated conditional statements?

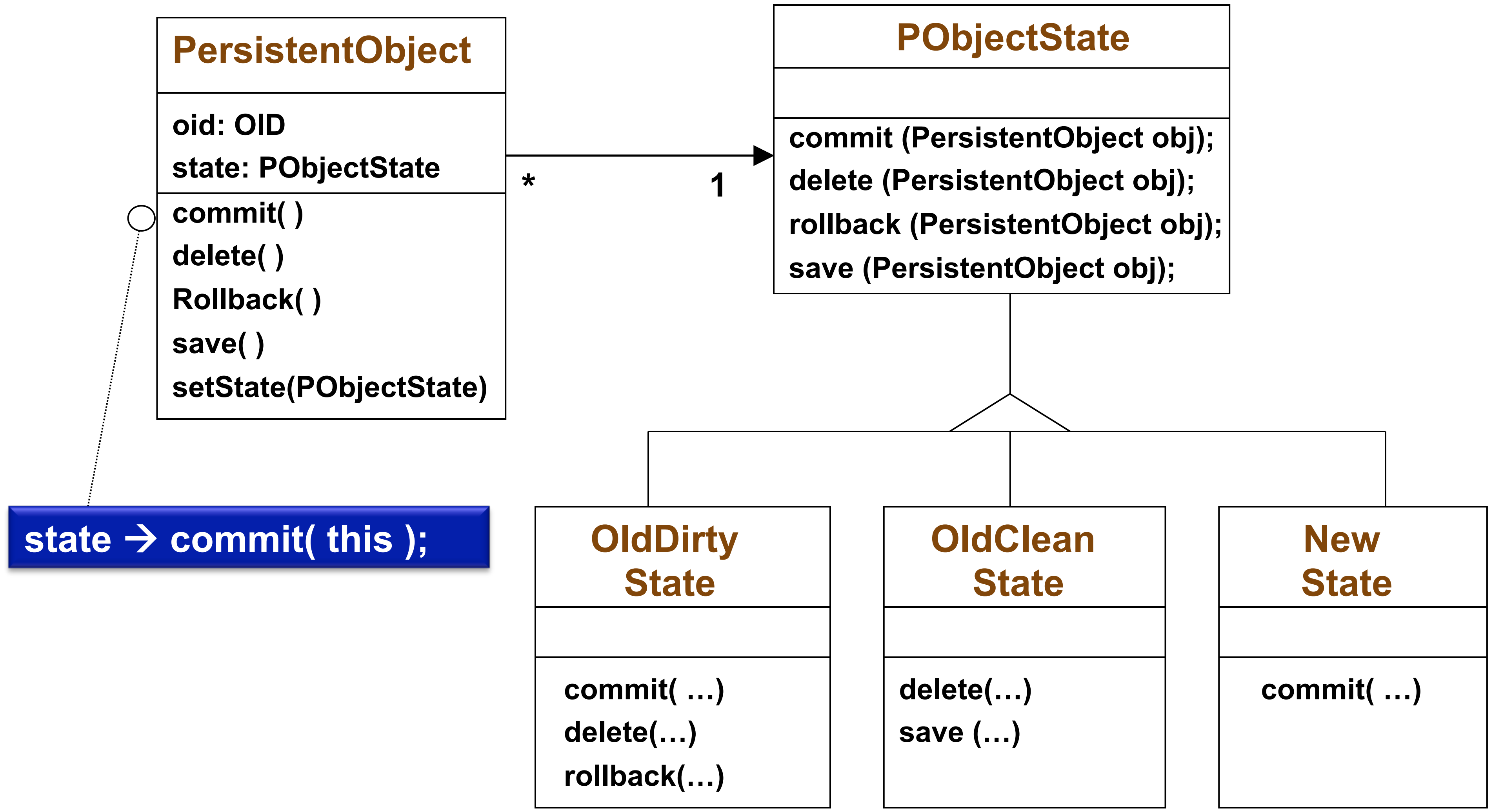
Solution: Create ***state classes*** implementing a common interface. Delegate state-dependent methods from *obj* to the current state object.

Q2,3

Example: State Pattern in TCP

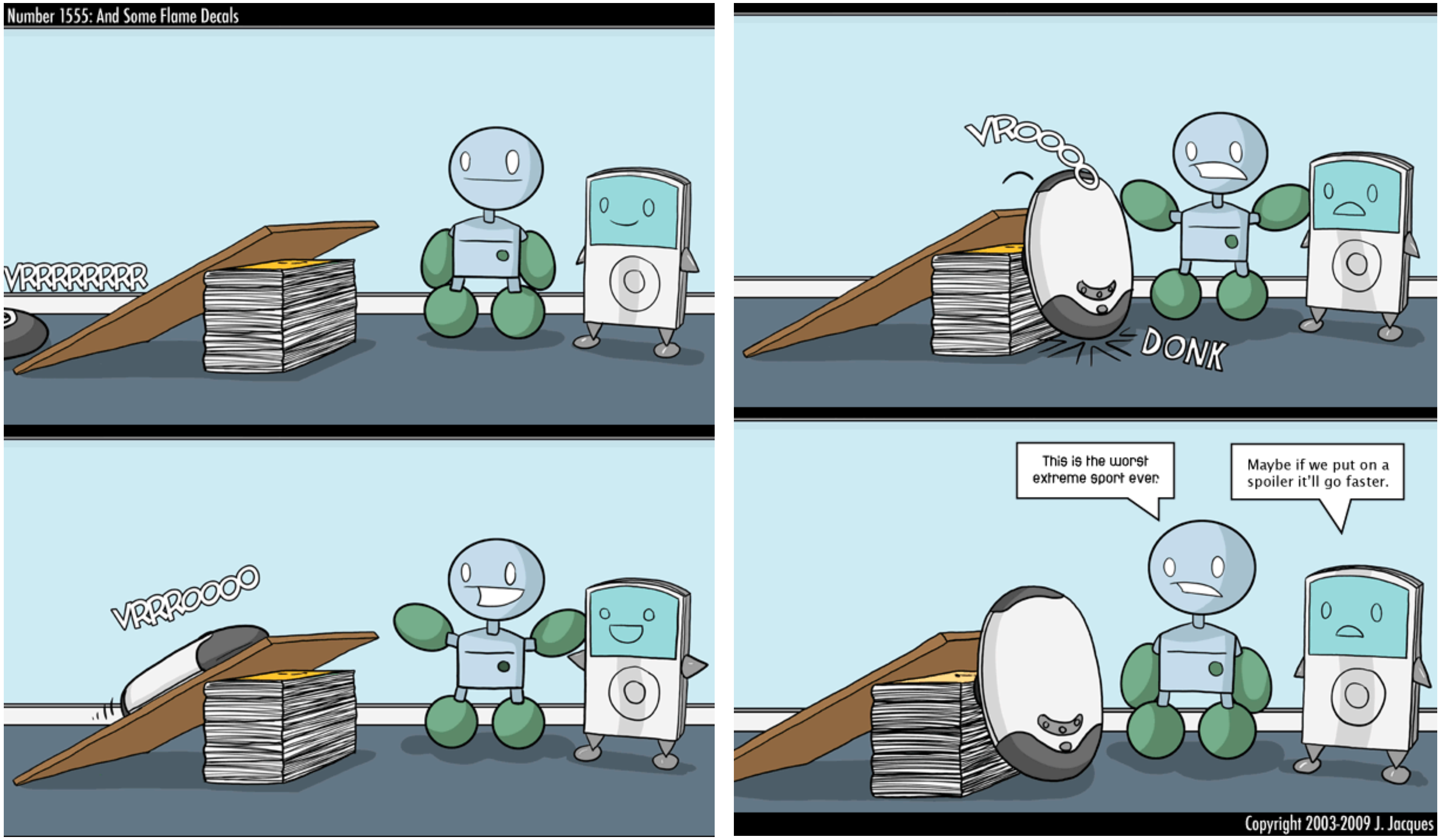


State Pattern in Persistence Framework



Q4

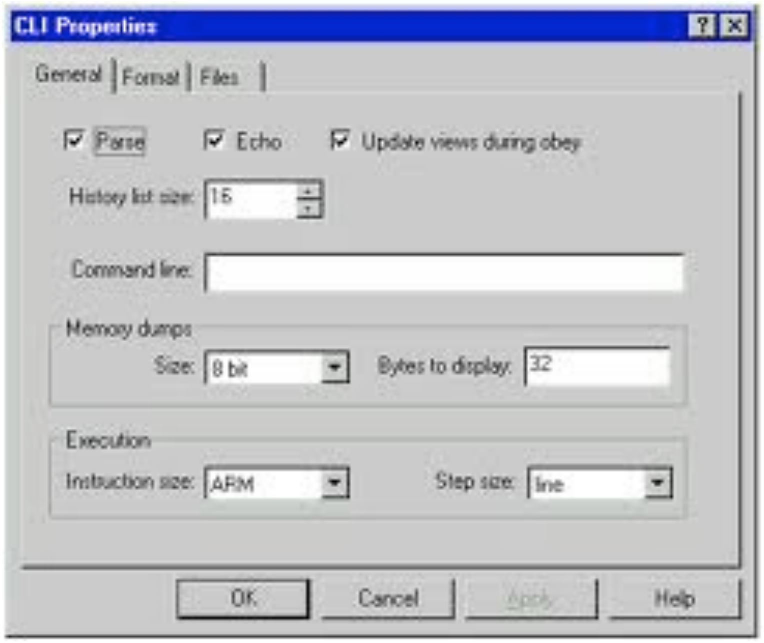
Cartoon of the Day


Number 1555: And Some Flame Decals
VRRRRRRRR
VRRROOOOO
VROOOO
DONK
This is the worst extreme sport ever.
Maybe if we put on a spoiler it'll go faster.
Copyright 2003-2009 J. Jacques

Command Pattern

Problem: When we need to record operations so we can undo them, or execute them later, what should we do?

Solution: Define a Command interface that represents all possible operations. Create subclasses of it for each kind of operation and instances for each actual operation.



☑ Parse ☑ Echo ☑ Update views during obey

History list size: 16

Command line:

Memory dumps

Size: 8 bit Bytes to display: 32

Execution

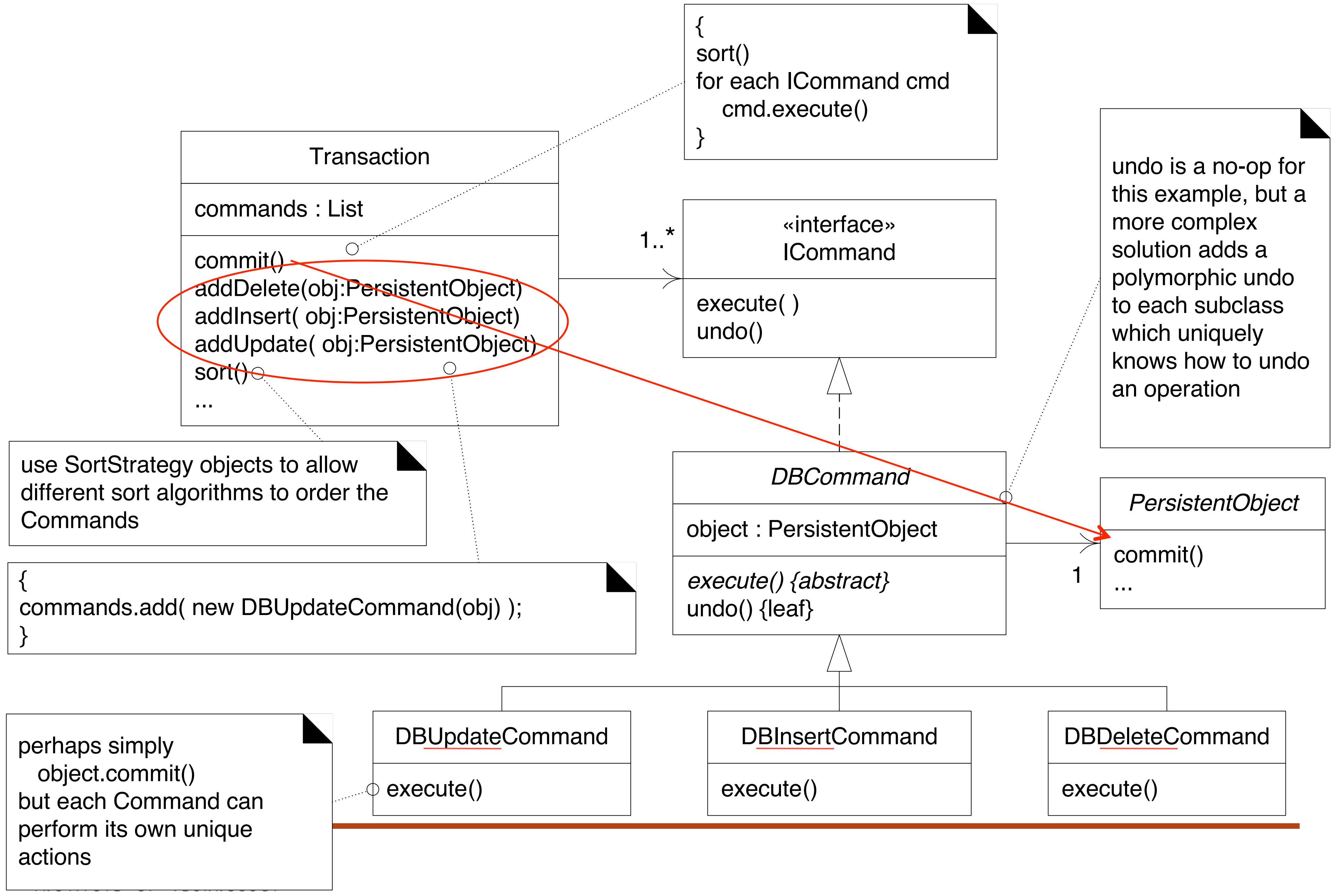
Instruction size: ARM Step size: line

Q5,6

Uses for the Command Pattern

- Undo/redo
- Prioritizing and Queuing operations
- Composing multi-part operations
- Progress bars
- Macro recording

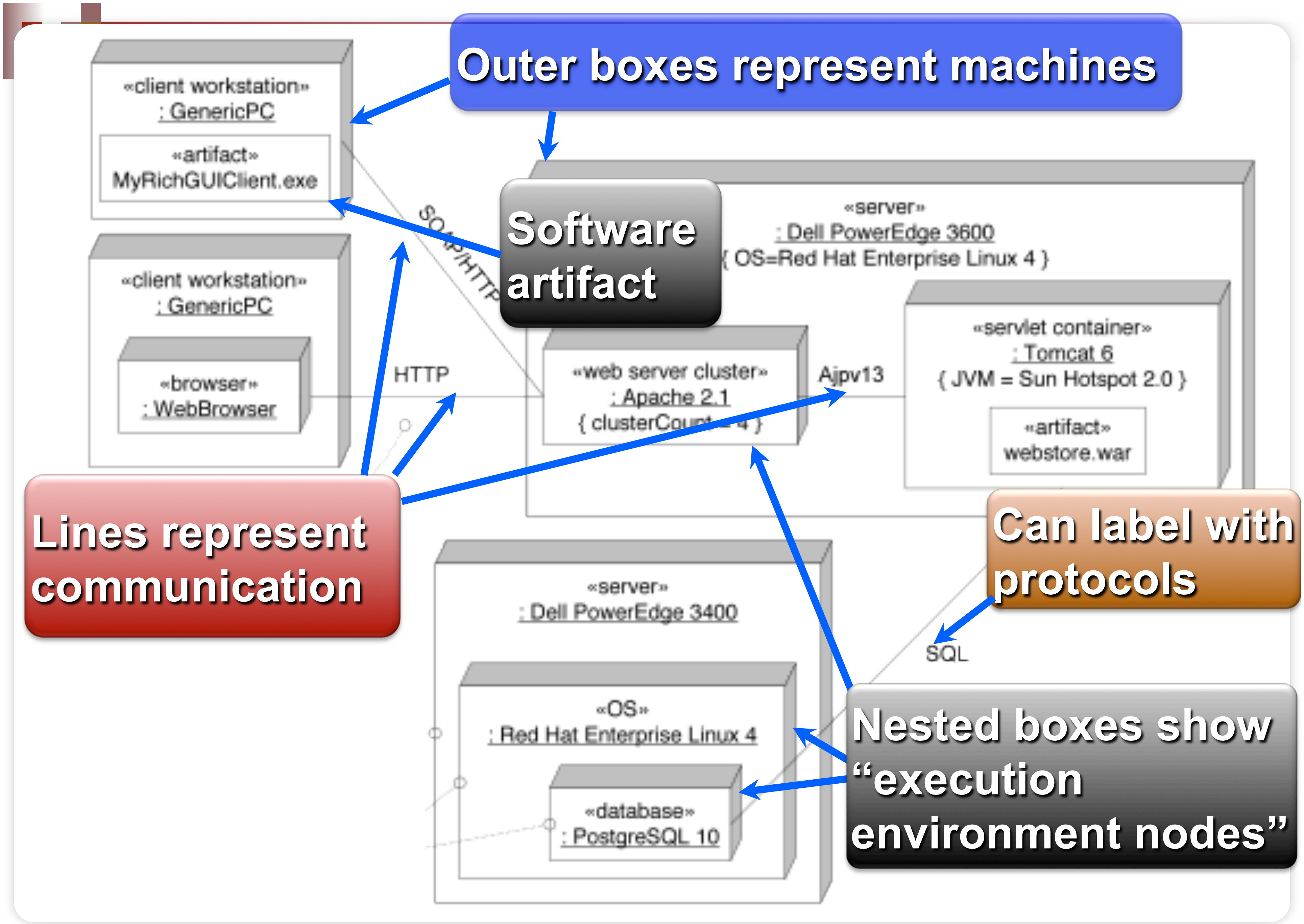
Command Pattern in NextGen POS


{
sort()
for each ICommand cmd
cmd.execute()
}
Transaction
commands : List
commit()
addDelete(obj:PersistentObject)
addInsert(obj:PersistentObject)
addUpdate(obj:PersistentObject)
sort()
...
1..*
«interface»
ICommand
execute( )
undo()
undo is a no-op for this example, but a more complex solution adds a polymorphic undo to each subclass which uniquely knows how to undo an operation
use SortStrategy objects to allow different sort algorithms to order the Commands
{
commands.add(new DBUpdateCommand(obj));
}
DBCommand
object : PersistentObject
execute() {abstract}
undo() {leaf}
PersistentObject
commit()
...
1
perhaps simply object.commit() but each Command can perform its own unique actions
DBUpdateCommand
execute()
DBInsertCommand
execute()
DBDeleteCommand
execute()

Deployment Diagrams

- Recall two key Architectural views:
  - Logical Architecture
  - Deployment Architecture

- Deployment Diagrams provide the means to express how the physical components of the system are organized


Outer boxes represent machines
Software artifact
Lines represent communication
Can label with protocols
Nested boxes show "execution environment nodes"
«client workstation» : GenericPC
«artifact» MyRichGUIClient.exe
«client workstation» : GenericPC
«browser» : WebBrowser
SOAP/HTTP
HTTP
Ajpv13
SQL
«server» : Dell PowerEdge 3600 { OS=Red Hat Enterprise Linux 4 }
«web server cluster» : Apache 2.1 { clusterCount = 4 }
«servlet container» : Tomcat 6 { JVM = Sun Hotspot 2.0 }
«artifact» webstore.war
«server» : Dell PowerEdge 3400
«OS» : Red Hat Enterprise Linux 4
«database» : PostgreSQL 10

Design Studio Calendar

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

Homework and Milestone Reminders

- 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 2.1 Design Studio Tomorrow
- Reminder: Bring Laptops Tomorrow!
- Thursday a Project Focus Day in Class