GRASP’ing at the First 5 ~~Patterns~~ Principles

Shawn Bohner
Office: Moench Room F212
Phone: (812) 877-8685
Email: bohner@rose-hulman.edu

GRASP

- General Responsibility Assignment Software Patterns (or Principles)

- Focus for Chapter 17 and today
  1. Low Coupling
  2. High Cohesion
  3. Information Expert
  4. Creator
  5. Controller

Coupling

An Evaluative Principle

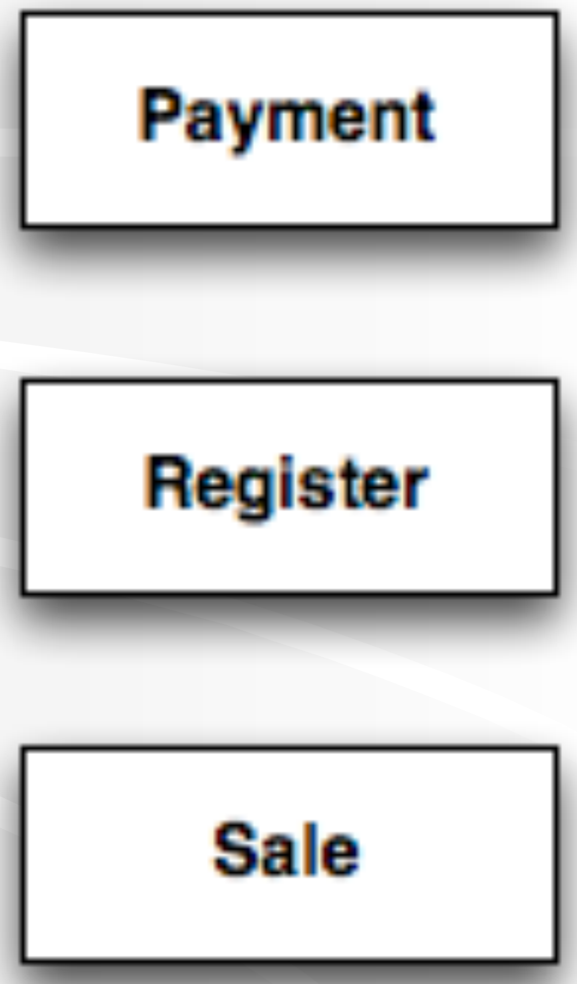
- A measure of how strongly one element:
  - is connected to,
  - has knowledge of, or
  - relies on other elements
- Want low (or weak) coupling
- What are some problems with high coupling?

Q1

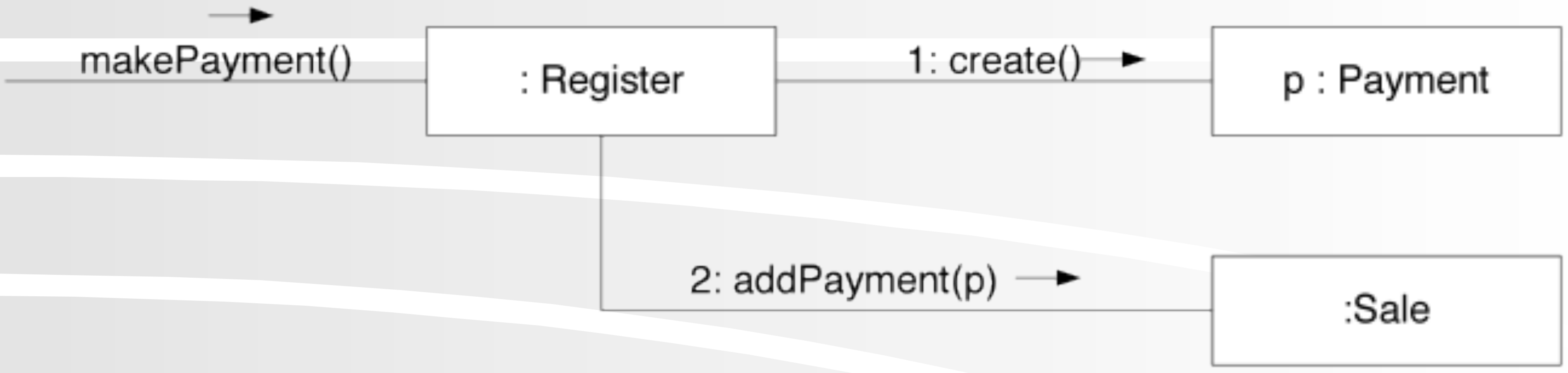
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Example

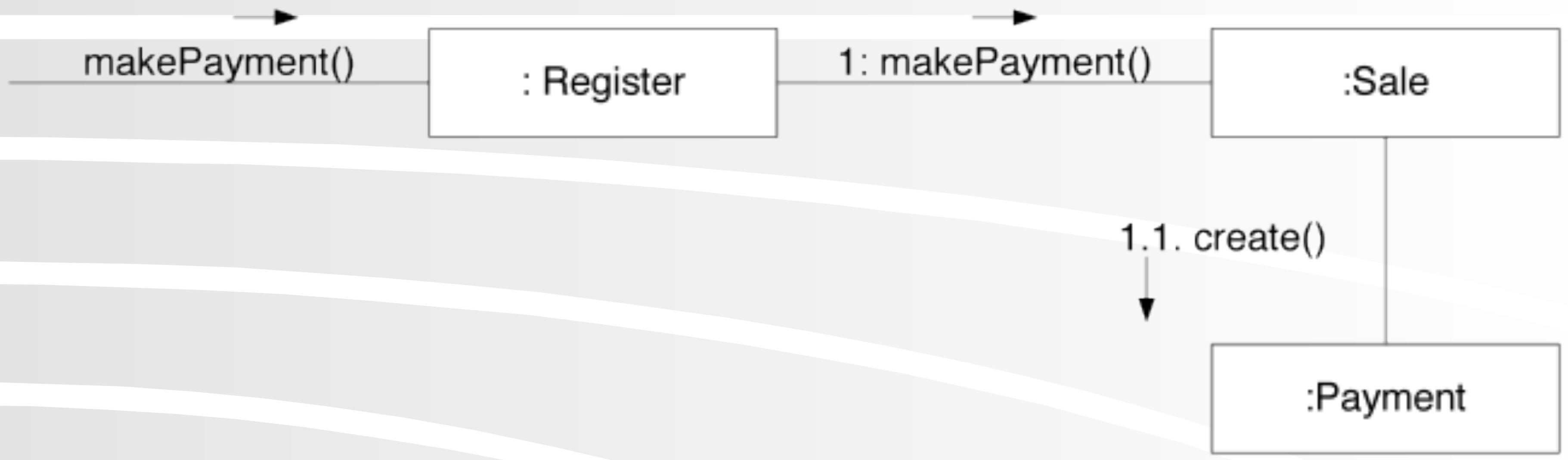
- Suppose we need to create a *Payment* instance and associate it with a *Sale*
- Who should be responsible?



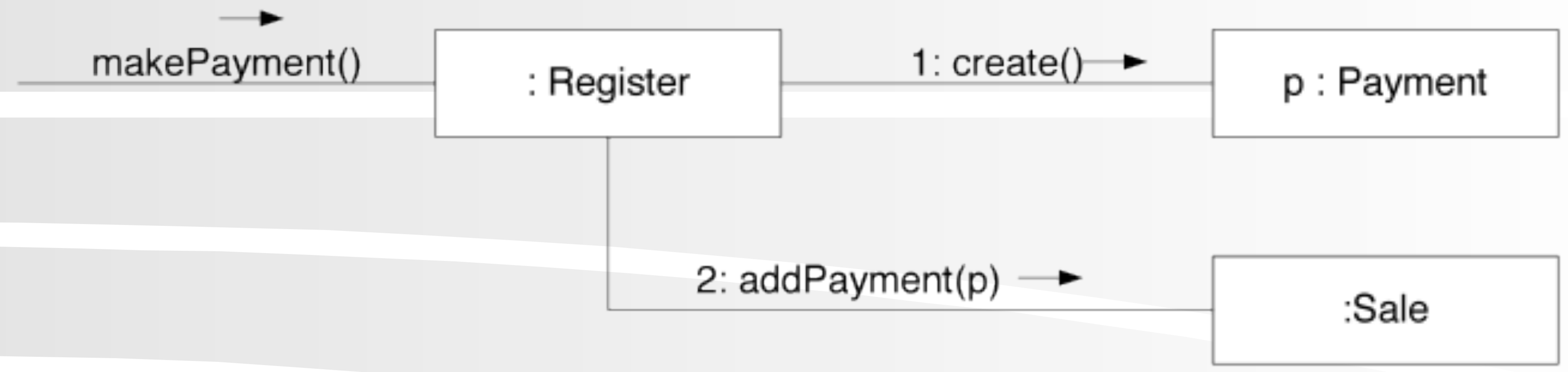
Option 1

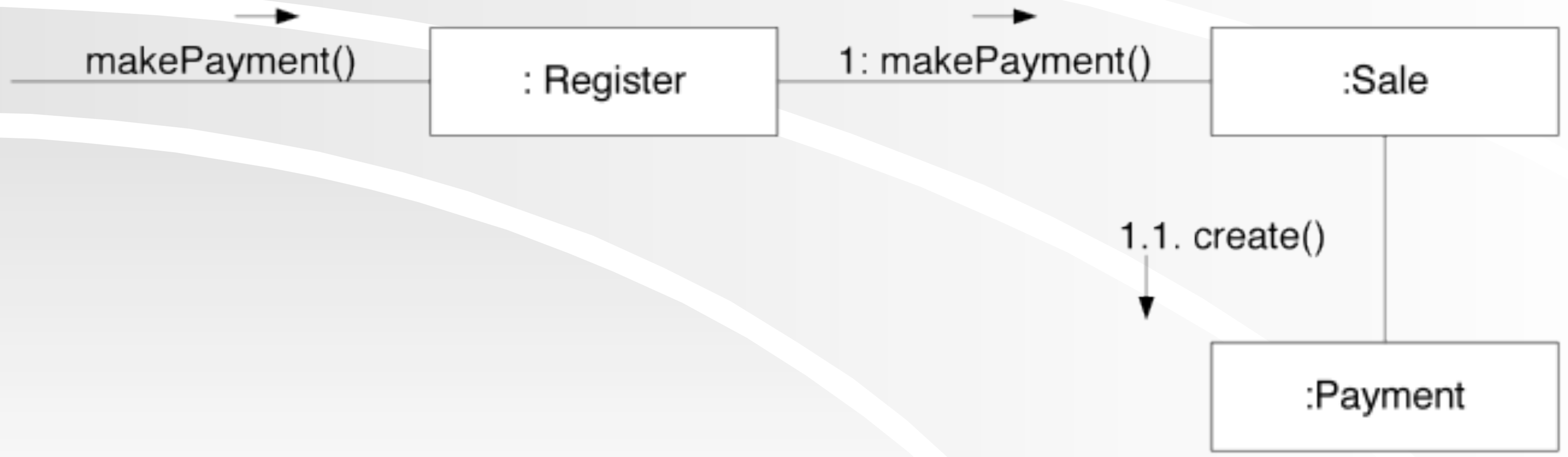

makePayment()
: Register
1: create()
p : Payment
2: addPayment(p)
:Sale

Option 2


makePayment()
: Register
1: makePayment()
:Sale
1.1. create()
:Payment

Lower Coupling?





Common Couplings

- *A* has an attribute of type *B*
- *A* calls a static method of *B*
- *A* has a method with a parameter or variable of type *B*
- *A* implements an interface *B*
- *A* is a subclass of *B*

Very strong coupling

Pick Your Battles

- Coupling to stable, pervasive elements isn't a problem
  - e.g., *java.util.ArrayList*
- Coupling to unstable elements can be a problem
  - Unstable interface, implementation, or presence
- Clearly can't eliminate coupling completely!

Cohesion

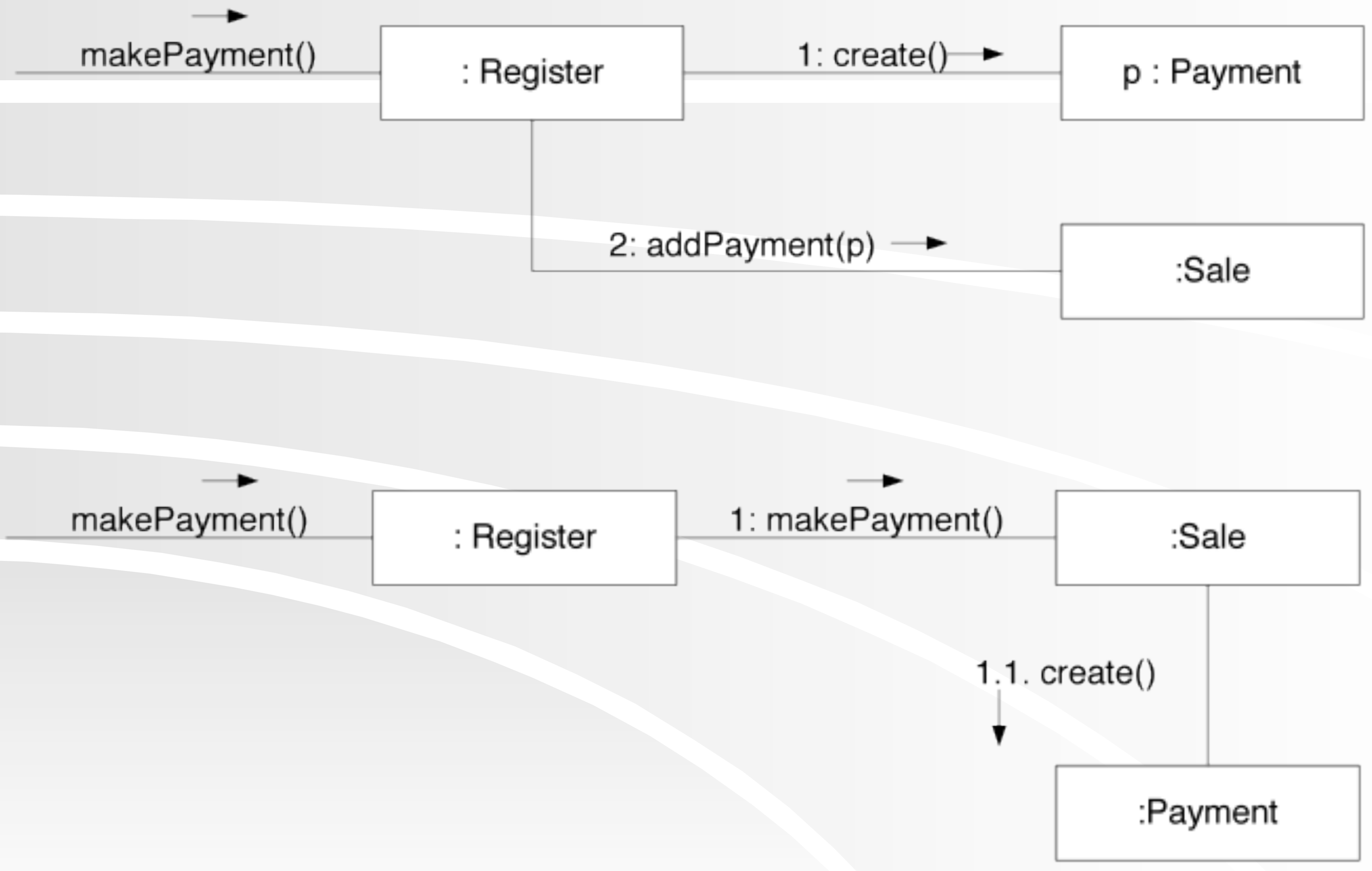
An Evaluative Principle

- A measure of how strongly related and focused the responsibilities of a class (or method or package…) are
- Want **high** cohesion
- What are some problems with low cohesion?

Q2

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

High Cohesion?



Guideline

A highly cohesive class…

- Has a small number of highly related methods
- Does not do “too much” work

Information Expert

- Problem: What is a general principle of assigning responsibilities?
- Solution: Assign a responsibility to the class that has the necessary information

the most general principle?

Q3,4

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Where do we look for classes?

- In the Design model if the relevant classes are there
- Otherwise:
  - Look to Domain model for motivation,
  - then add classes to the Design model

Information Expert Examples

- Who should be responsible for knowing the grand total of a *Sale*?
- Given that a *Piece* (in Monopoly) just landed on a *Square*, who should be responsible for calculating the rent due?

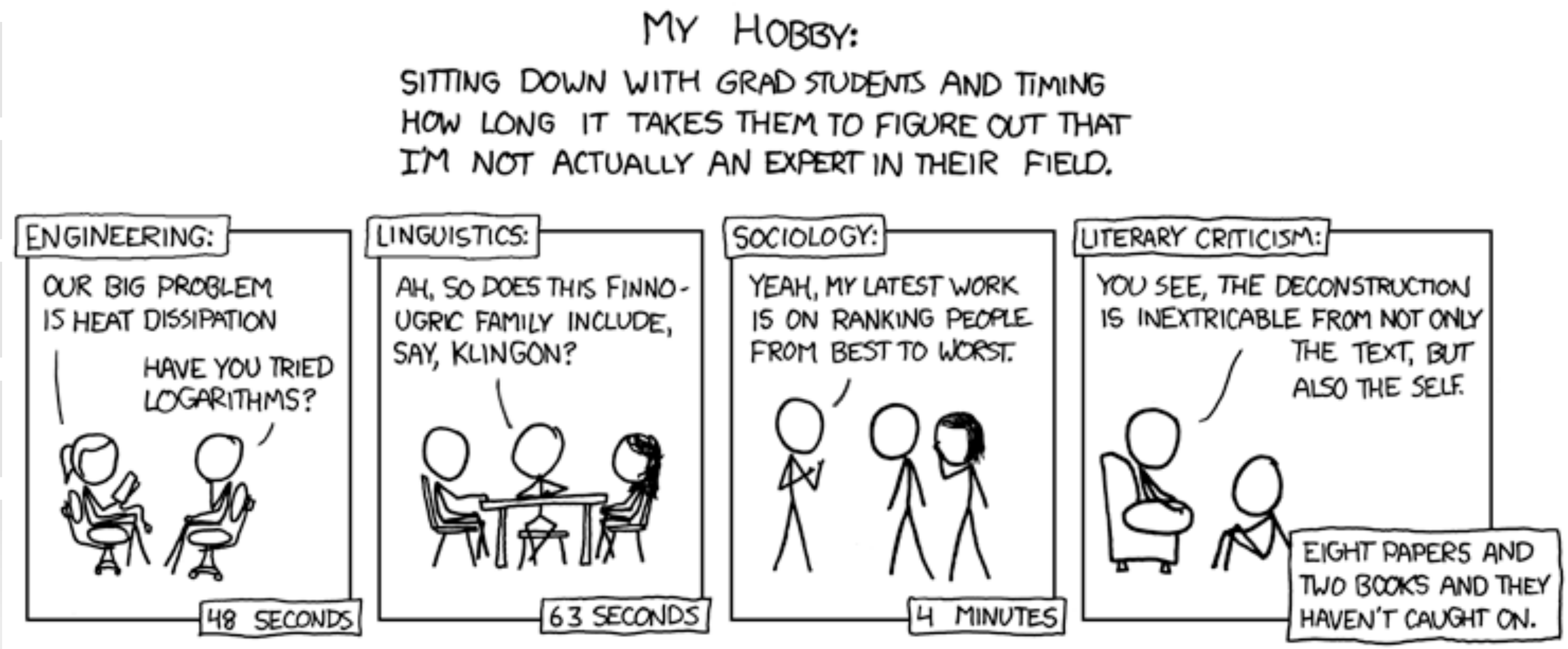
Q5

Information Expert Contraindications

- Sometimes Information Expert will suggest a solution that leads to coupling or cohesion problems
  - Consider: Who should be responsible for saving a *Sale* in a database?

Q6

Imposter


MY HOBBY:
SITTING DOWN WITH GRAD STUDENTS AND TIMING HOW LONG IT TAKES THEM TO FIGURE OUT THAT I'M NOT ACTUALLY AN EXPERT IN THEIR FIELD.
ENGINEERING:
OUR BIG PROBLEM IS HEAT DISSIPATION
HAVE YOU TRIED LOGARITHMS?
48 SECONDS
LINGUISTICS:
AH, SO DOES THIS FINNO-UGRIC FAMILY INCLUDE, SAY, KLINGON?
63 SECONDS
SOCIOLOGY:
YEAH, MY LATEST WORK IS ON RANKING PEOPLE FROM BEST TO WORST.
4 MINUTES
LITERARY CRITICISM:
YOU SEE, THE DECONSTRUCTION IS INEXTRICABLE FROM NOT ONLY THE TEXT, BUT ALSO THE SELF.
EIGHT PAPERS AND TWO BOOKS AND THEY HAVEN'T CAUGHT ON.

If you think this is too hard on literary criticism, read the Wikipedia article on deconstruction.

Creator

- Problem: Who should be responsible for creating a new instance of some class?

- Solution: Make *B* responsible for creating *A* if…
  - *B* contains or is a composition of *A*
  - *B* records *A*
  - *B* closely uses *A*
  - *B* has the data to initialize *A*

Most important

The more matches the better.

Q7,8

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

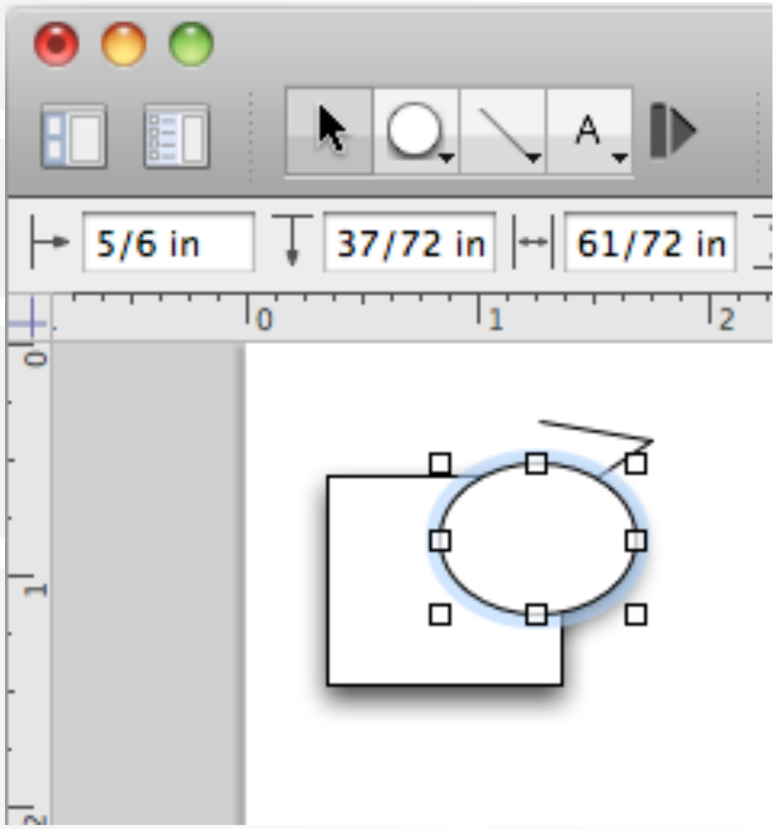
Creator Examples

- In Monopoly simulator, who should create…
  - *Squares?*
  - *Pieces?*
  - *Dice?*
- In NextGen POS, who should create…
  - *SalesLineItems?*
  - *ProductDescriptions?*

Q9

Creator Contraindications

- Complex creation scenarios
  - Recycling instances
  - Conditional creation



Team Creativity

Q11

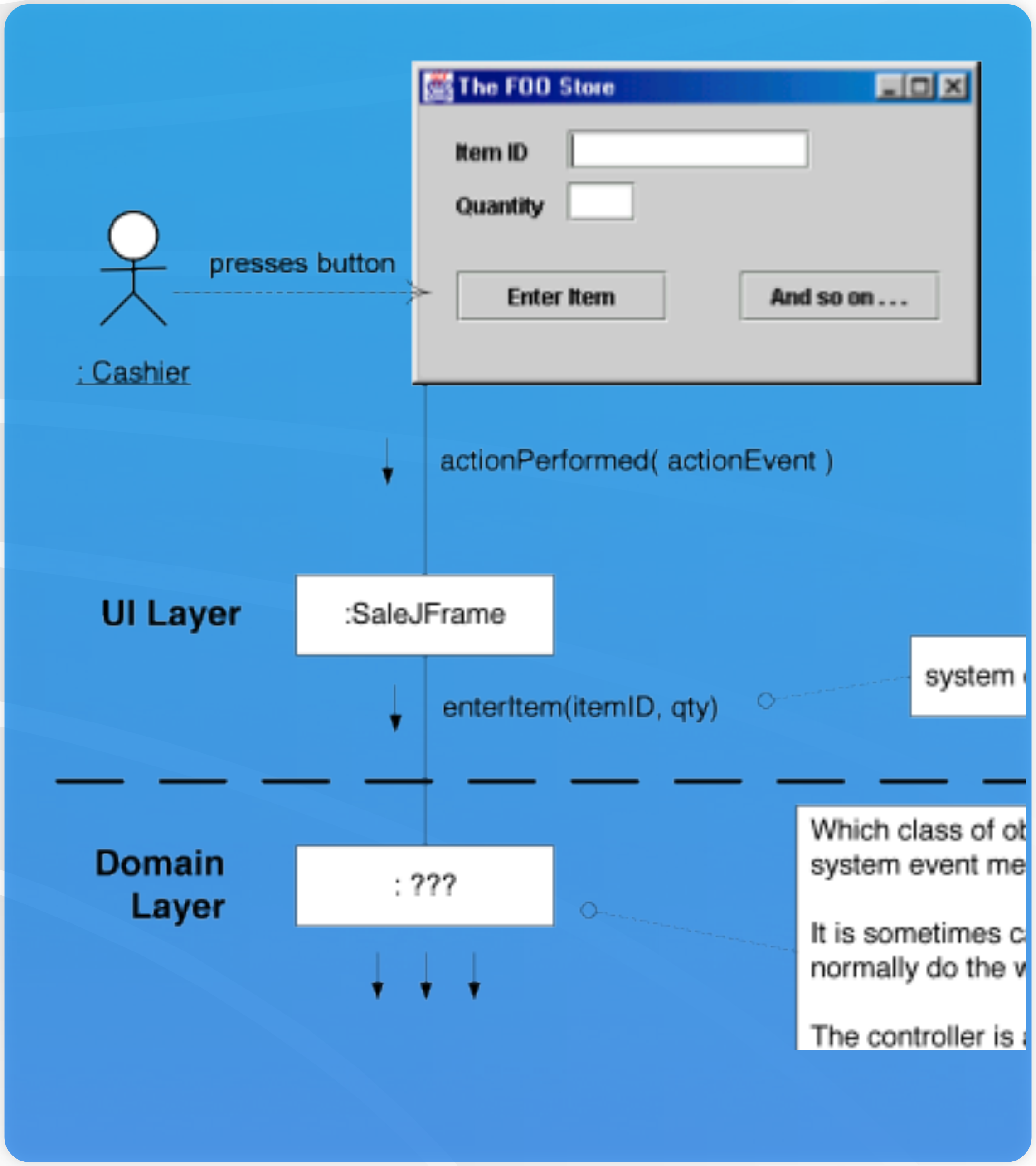
Controller

- Problem: What first object beyond the UI layer receives and coordinates a *system operation*
- Solution: Assign the responsibility to either…
  - A façade controller, representing the overall system and handling all system operations, or
  - A use case controller, that handles all system events for a single use case

Q12,13

Example

- What domain layer class should own handling of the *enterItem* system operation?



Guidelines

- Controller should delegate to other domain layer objects
- Use façade controller when…
  - There are a limited number of system operations, or
  - When operations are coming in over a single “pipe”
- Use use case controller when a façade would be bloated (low cohesion!)

Controller Benefits

- Increased potential for reuse
- Can reason/control the state of a use case
  - e.g., don’t close sale until payment is accepted

Controller Issues

Switch from façade to use case controllers

- Controller bloat—too many system operations
- Controller fails to delegate tasks
- Controller has many attributes

Delegate!

Team Control

Q14,15

Homework and Milestone Reminders

- Read Chapter 18
- Homework 4 – Dog-eDoctor System Preliminary Logical Architecture and Design
  - Today by 5:00pm (Tuesday, January 5th, 2010)
- Milestone 3 – Iteration 1: Junior Project
  - Finish Analysis Model (SSDs, OCs)
  - Logical Architecture - Package Diagrams, and
  - 1st (initial) Version of System
  - Due by 11:59pm on Friday, January 8th, 2009