Getting a GRASP on Designing with Responsibilities

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

Q1

Plus / Delta Feedback

- Pace
  - 1 – much too fast
  - 12 – somewhat too fast
  - 4 – Somewhat too slow
  - 0 – much too slow
- Working well
  - Diagramming Design examples in class+++, Handouts, explicit answers to quiz on slides(?), Learning topics for milestones, Material is better organized…
- Improvements
  - More depth on purpose of diagrams, More fundamentals, Active learning a examples done on board, slow down/speed up(?), More concrete examples, Use machine to draw examples in class, Don't rush when we get behind.

Plus / Delta Feedback

- Quizzes
  - 6 – Very helpful
  - 9 – somewhat helpful
  - 2 – somewhat unhelpful
  - 0 – Very unhelpful

- Working well
  - Working examples via quiz in class, Point to key areas in lecture, Reading questions, Indicators where Q'a are in lecture, Focuses lecture, …
- Improvements
  - Easier questions, Trimming quizzes, More short-answer, Answers in the text of slides (?), Don't be too arbitrary, Shorten them, Sometimes consume too much time -- loose focus.

Plus / Delta Feedback

- Reading
  - 1 – all of it
  - 8 – most of it
  - 7 – little of it
  - 1 – none of it

- Homework Difficulty
  - 0 – much too difficult
  - 15 – a bit too difficult
  - 1 – a bit too easy
  - 0 – much too easy

Plus / Delta Feedback

- Homework helpfulness
  - 1 – very helpful
  - 8 – somewhat helpful
  - 7 – somewhat unhelpful
  - 1 – very unhelpful
- Working well
  - Re-enforces techniques/tools, Opportunity to experiment on what is presented in class, Practice before milestones, PDFs, Well-written task descriptions…
- Improvements
  - Clarify assignments better, Redo's on HW, Provide even more examples to clarify assignments, Give firmer rubric, Quicker returns to support milestones, Give less homework (?), Too much reading, Simplify project.

Plus / Delta Feedback

- Workload
  - 0 – much higher than average
  - 15 – somewhat higher than average
  - 1 – somewhat lower than average
  - 0 – much lower than average

- General Comments
  - Yikes - Milestones are taking off, speeding through too many slides – focus on a few a key things (and reading will cover rest)
  - Stop reading XKCD in class – humor lost in my translation

Summary of Plus / Delta Actions

- Active learning a examples done on board - Yes
  - More concrete examples
  - More depth on purpose and fundamentals
- Slow down/speed up – will try not to get behind
- Will make the quizzes less distracting via short answer and giving more time in lecture
- Will clarify assignments better
  - More examples to clarify assignments and a firmer rubric,
  - Quicker returns to support milestones now with longer lead
- Milestones are taking off – getting to the meat!
  - Give focused homework, reading, and project assignments
  - Will modulate, but do not want diminish value
- Use machine to draw examples in class (no luck doing this expediently yet – will keep trying)

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Mastering Object-Oriented Design

- A large set of soft principles
- It isn't magic. We learn it with:
  - Patterns (named, explained, and applied)
  - Examples
  - Practice

"The critical design tool for software development is a **mind well-educated in design principles**."

Q2

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Responsibility-Driven Design

- Responsibility Driven Design (RDD)
  - Pioneered by Wirfs-Brock in early 1990s
- Think of objects in terms of what they do or know (the human worker metaphor!)
- An object's obligation or contract that it offers to other objects

Responsibilities for an Object

- Doing
  - a *Sale* is responsible for creating instances of *SalesLineItem*
- Knowing
  - a *Sale* is responsible for knowing its *total* cost

Q3,4

Knowing and Doing (continued)

- "Doing" Responsibilities
  - Create another object
  - Perform a calculation
  - Initiate an action in an object
  - Control/coordinate activities of objects
- "Knowing" Responsibilities
  - Knowing it's own encapsulated data
  - Knowing about other objects
  - Knowing things it can derive or calculate

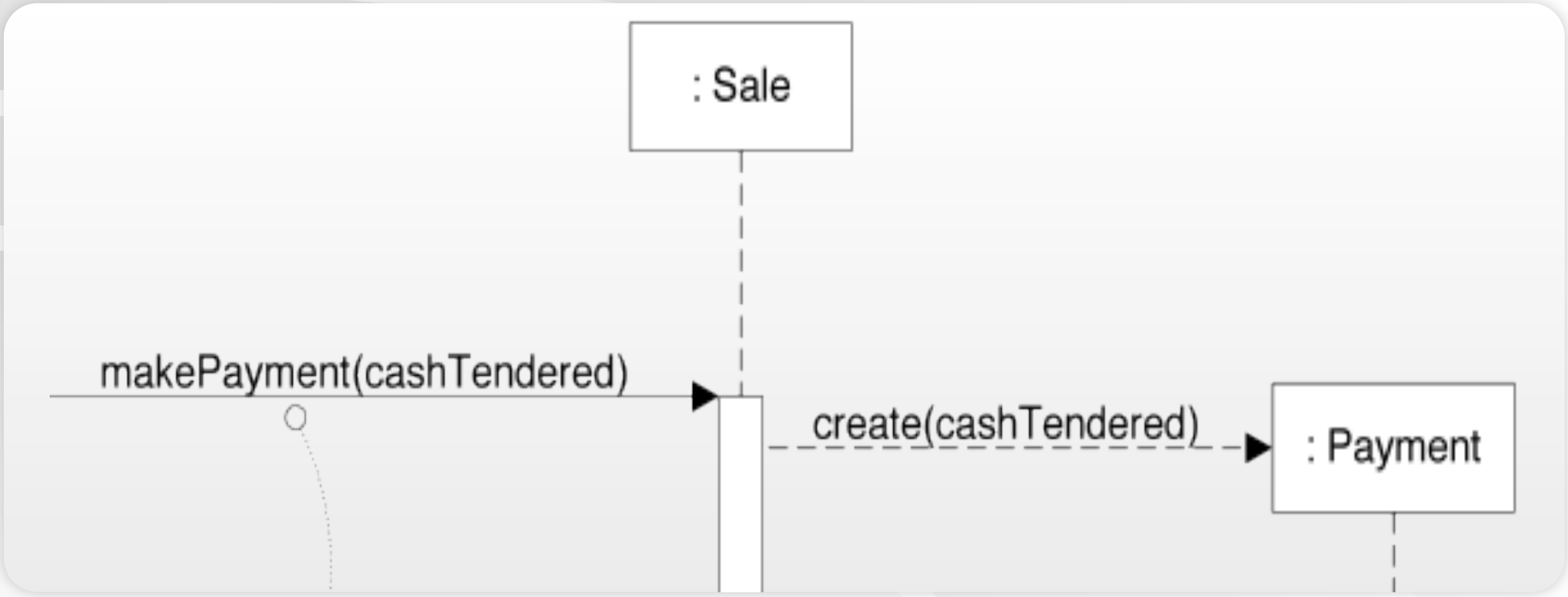
Responsibilities Come in All Sizes

- Big: provide access to a relational database
- Small: create a Sale

A responsibility is **not** the same thing as a method

When Do We Assign Responsibilities?

- While coding
- While modeling
  - UML is a low-cost modeling tool
  - Can assign responsibilities with minimal investment



Introducing GRASP

- GRASP: General Responsibility Assignment Software Patterns (or Principles)
  - A set of patterns for assigning responsibilities to software objects
- What is a Pattern?
  - A pattern is a named and well-known problem-solution pair that can be applied in a new context

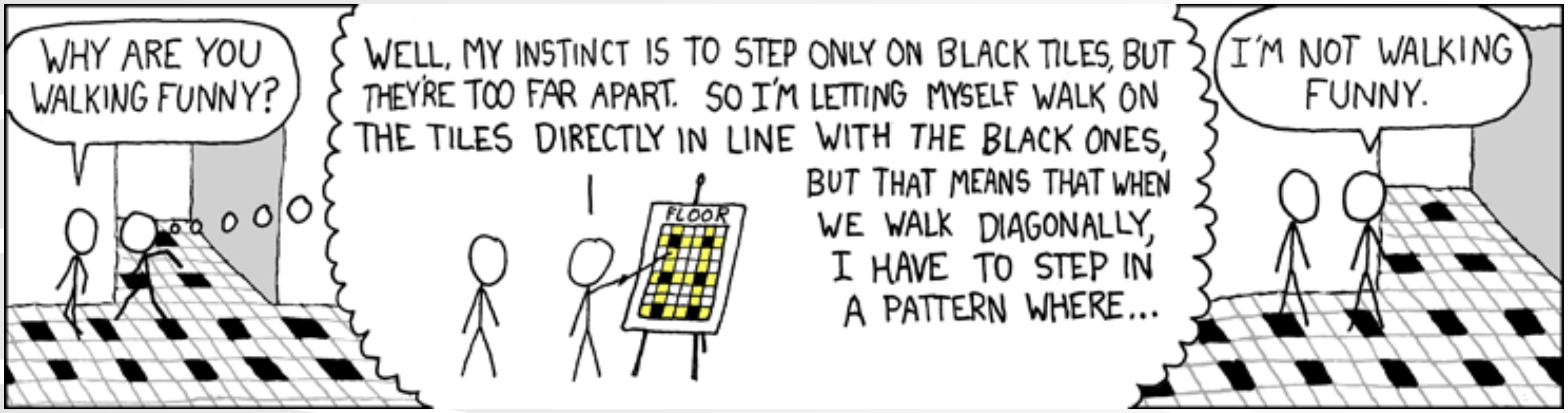
Nine GRASPs

- Five In this chapter
  1. Creator
  2. Information Expert
  3. Low Coupling
  4. Controller
  5. High Cohesion

- Four Later In Chapter 25

Polymorphism	Pure Fabrication
Indirection	Protected Variations

Floor Tiles


WHY ARE YOU WALKING FUNNY?
WELL, MY INSTINCT IS TO STEP ONLY ON BLACK TILES, BUT THEY'RE TOO FAR APART. SO I'M LETTING MYSELF WALK ON THE TILES DIRECTLY IN LINE WITH THE BLACK ONES, BUT THAT MEANS THAT WHEN WE WALK DIAGONALLY, I HAVE TO STEP IN A PATTERN WHERE...
FLOOR
I'M NOT WALKING FUNNY.

The worst part is when sidewalk cracks are out-of-sync with your natural stride.

Example Pattern

Names Matter!

Pattern Name	Information Expert
Problem	What is a basic principle by which to assign responsibilities to objects?
Solution	Assign a responsibility to the class that has the information needed to fulfill it.

"New pattern" is an oxymoron!

Q5,6,7

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

History

- ***A Pattern Language: Towns, Buildings, Construction***
  by Alexander, Ishikawa, and Silverstein
- **Kent Beck, Ward Cunningham**
- ***Design Patterns: Elements of Reusable Object-Oriented Software***
  Gamma, Helm, Johnson, Vlissides

The Gang of Four

Homework and Milestone Reminders

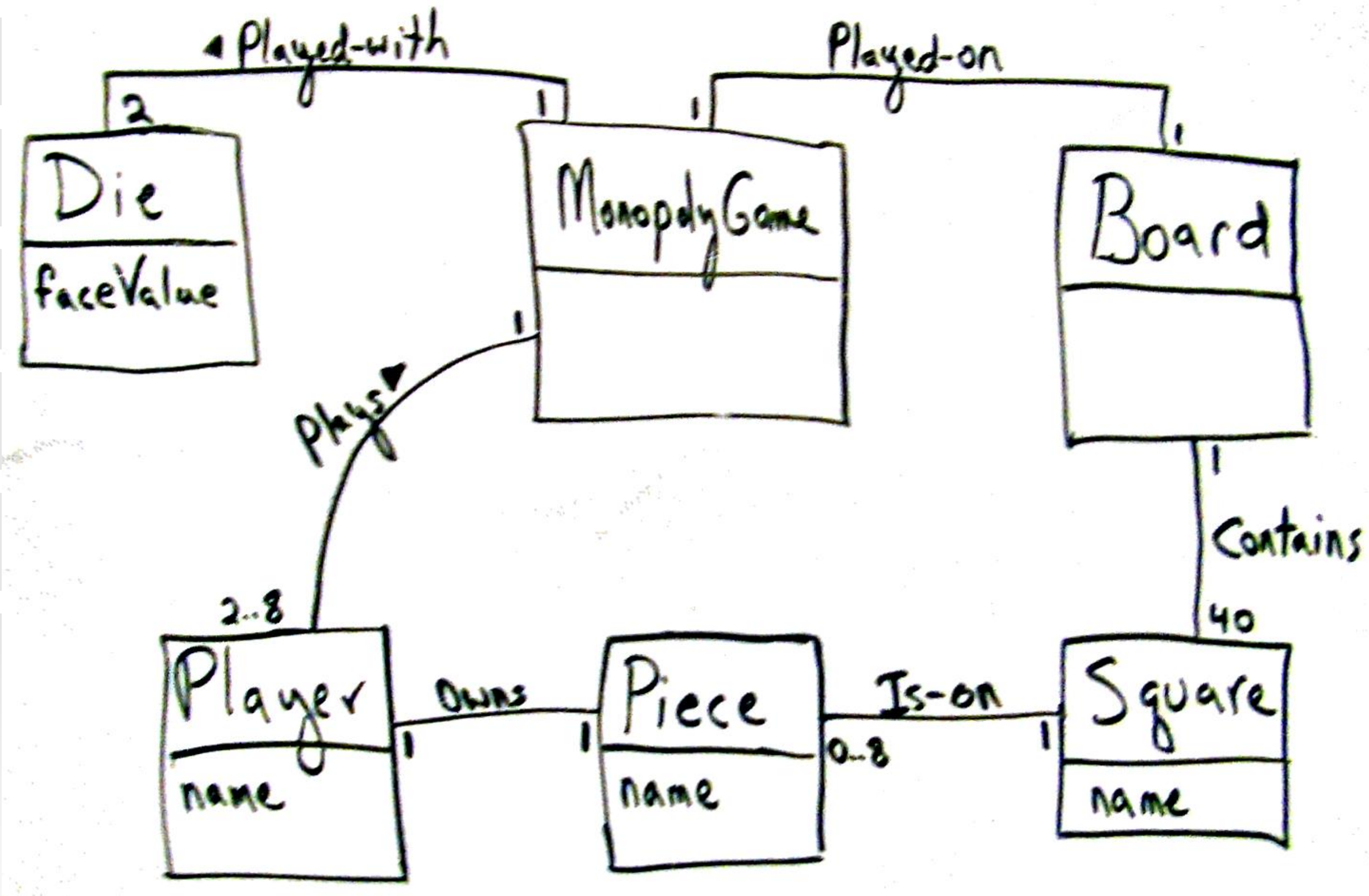
- Read Chapter 17 on GRASP (Rest of Chapter)
- Homework 4 – Dog-eDoctor System Preliminary Logical Architecture and Design
  - Due by 5:00pm on Tuesday, January 5th, 2010
  - Extra credit if you get it in by 5:00pm this Friday!
- Milestone 3 – Iteration 1: Junior Project
  - Finish Analysis Model (SSDs, OCs)
  - Logical Architecture - Package Diagrams, and
  - 1st (initial) Version of System
  - Due by 11:59pm on Friday, January 8th, 2009

Creator Pattern

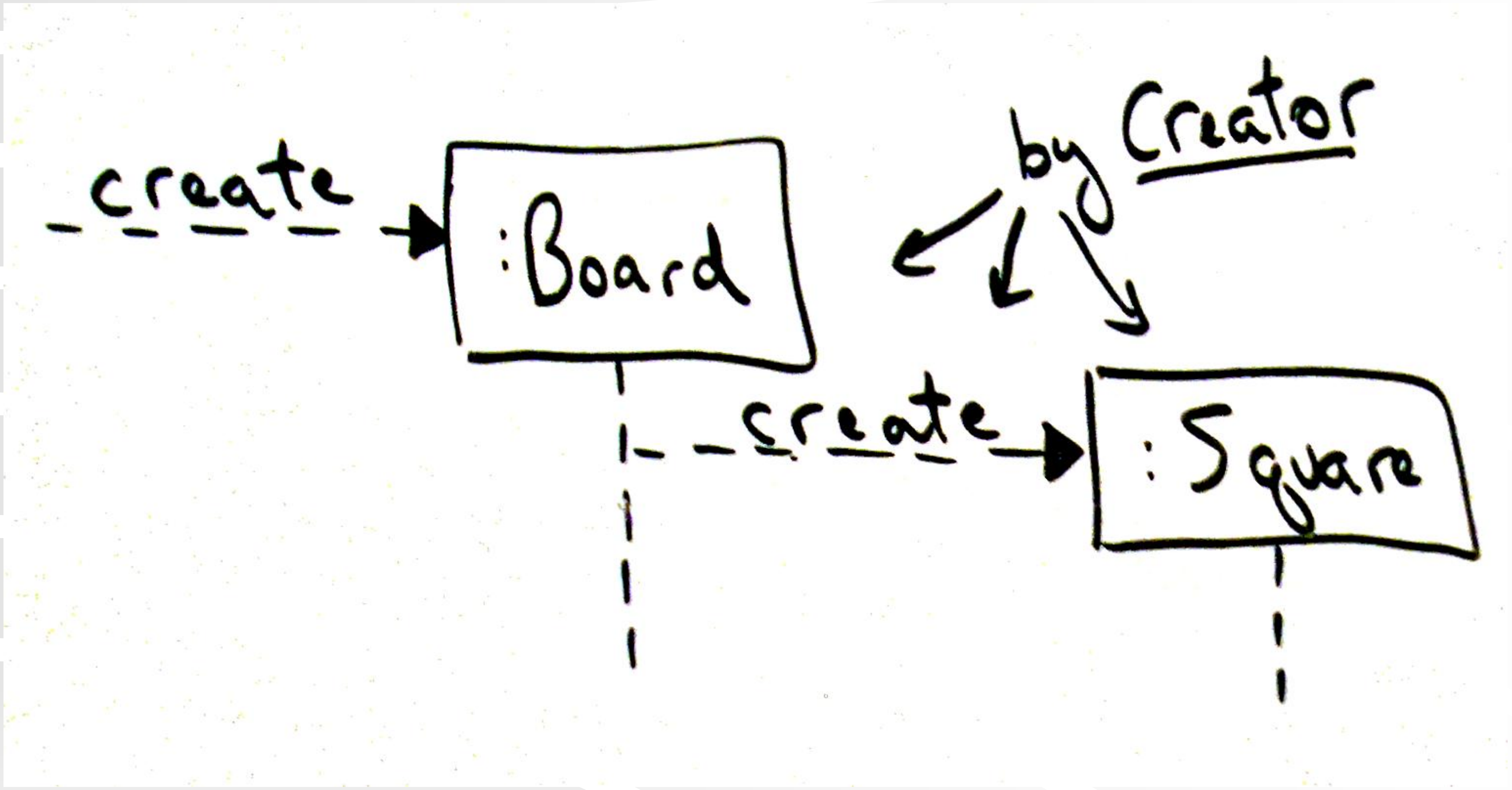
- Who should create object A?
  - Solution (advice):
    - Let B do it if:
      - B contains or aggregates A
      - B records A
      - B closely uses A
      - B has the initializing data for A

- Monopoly Board Example
  - When you start a game, who creates the squares for the board?
  - Let Board create them since it *contains* the squares

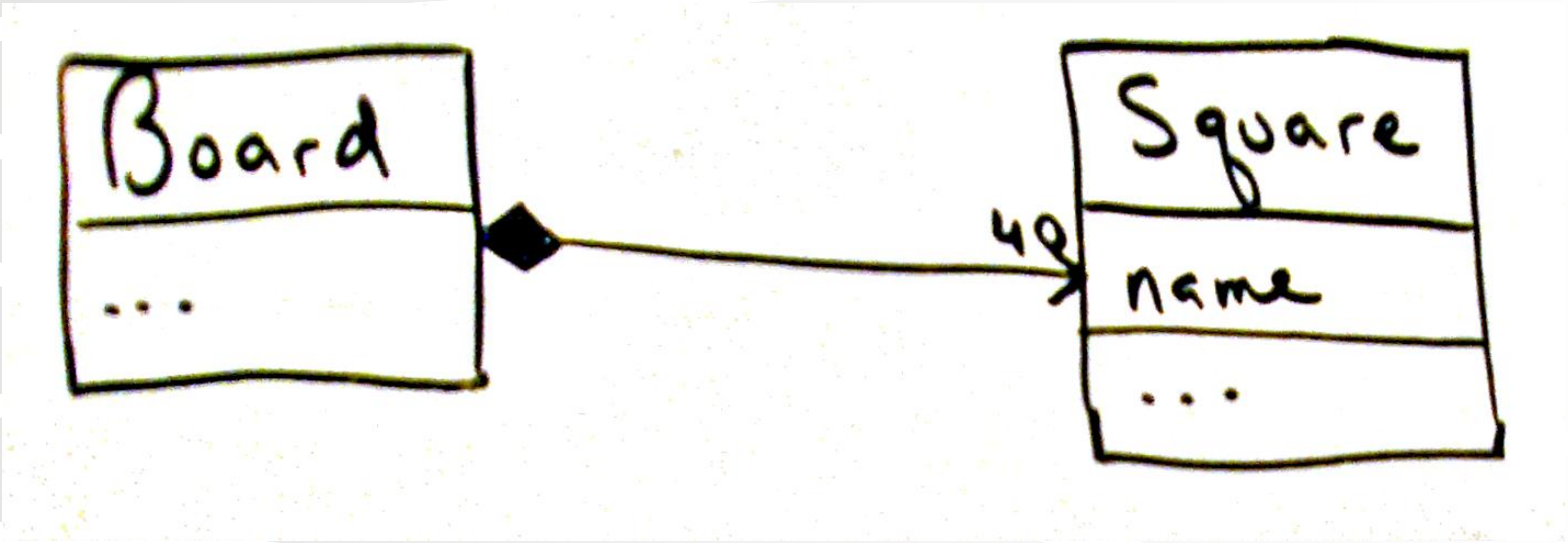
Monopoly Example



Create in Action



Composition



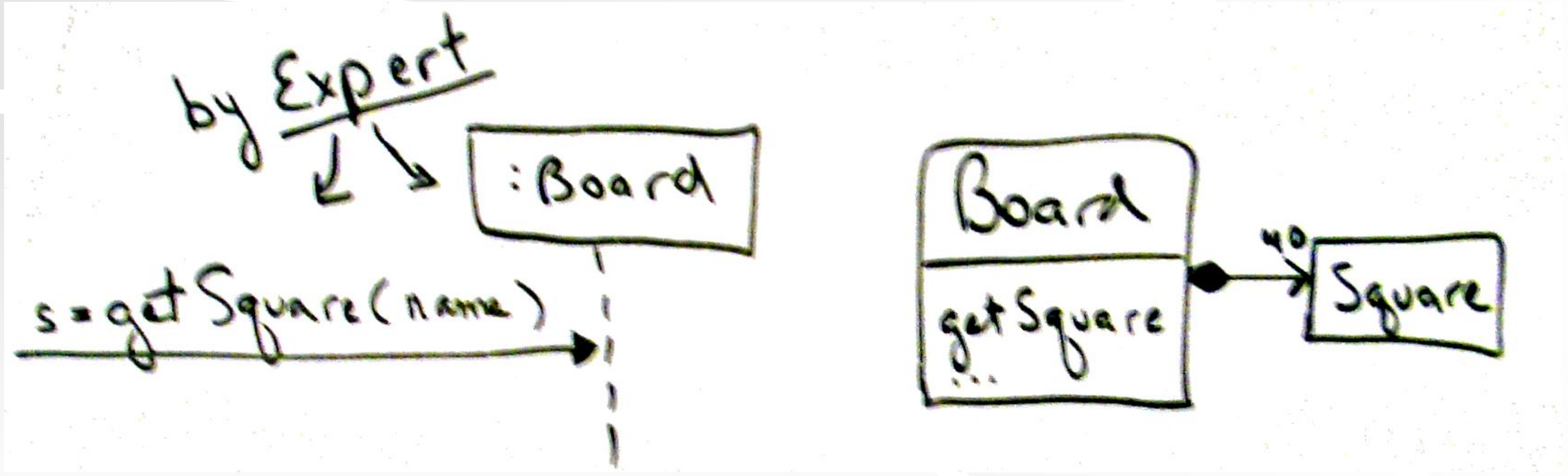
- Board has a composition relationship with a set of squares

Experts and Unique Identifiers

- What is a basic principle of RDD?

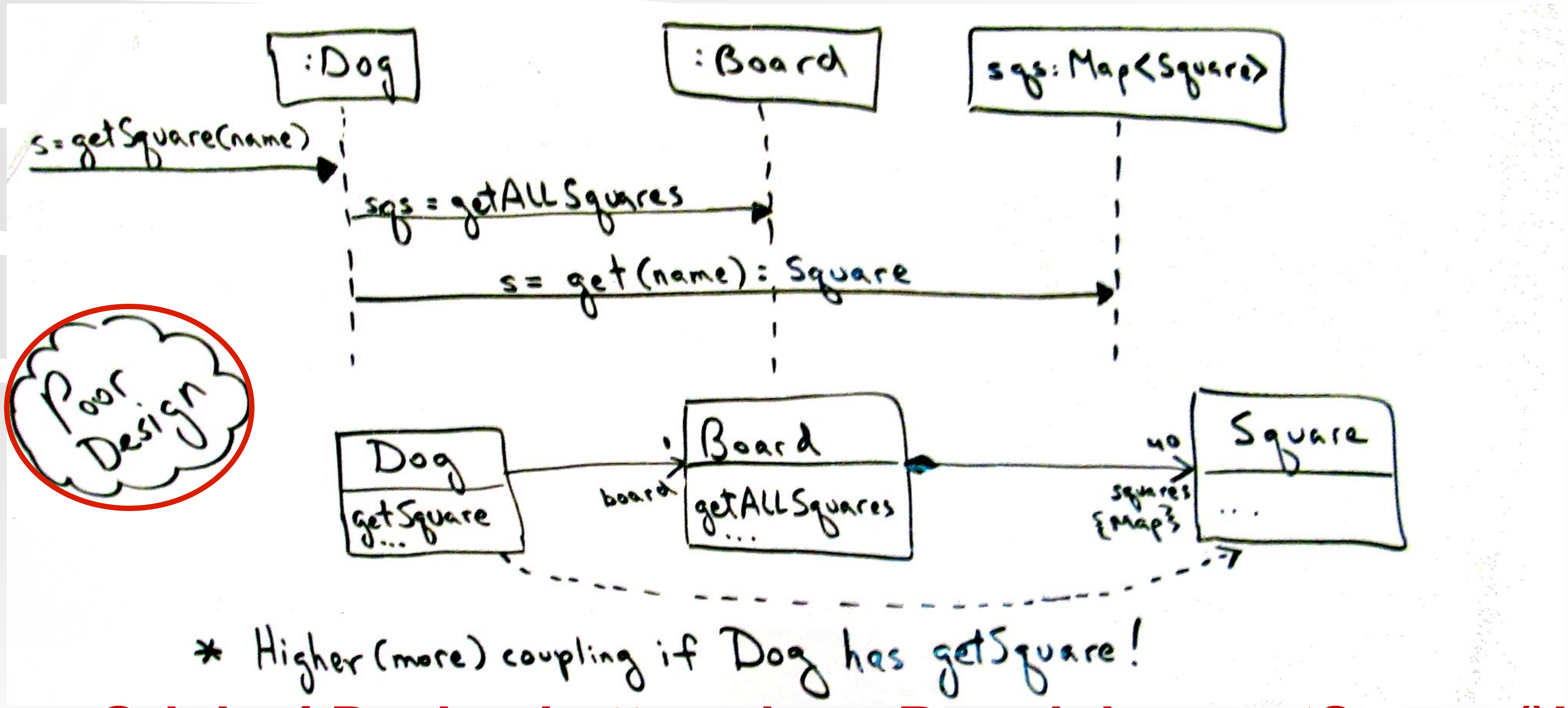
  …Assign responsibility to the object that has the required information

  - "Tell the expert to do it!"
- Who should get a square given a unique ID?
  - Let the Board do it because it knows about the squares



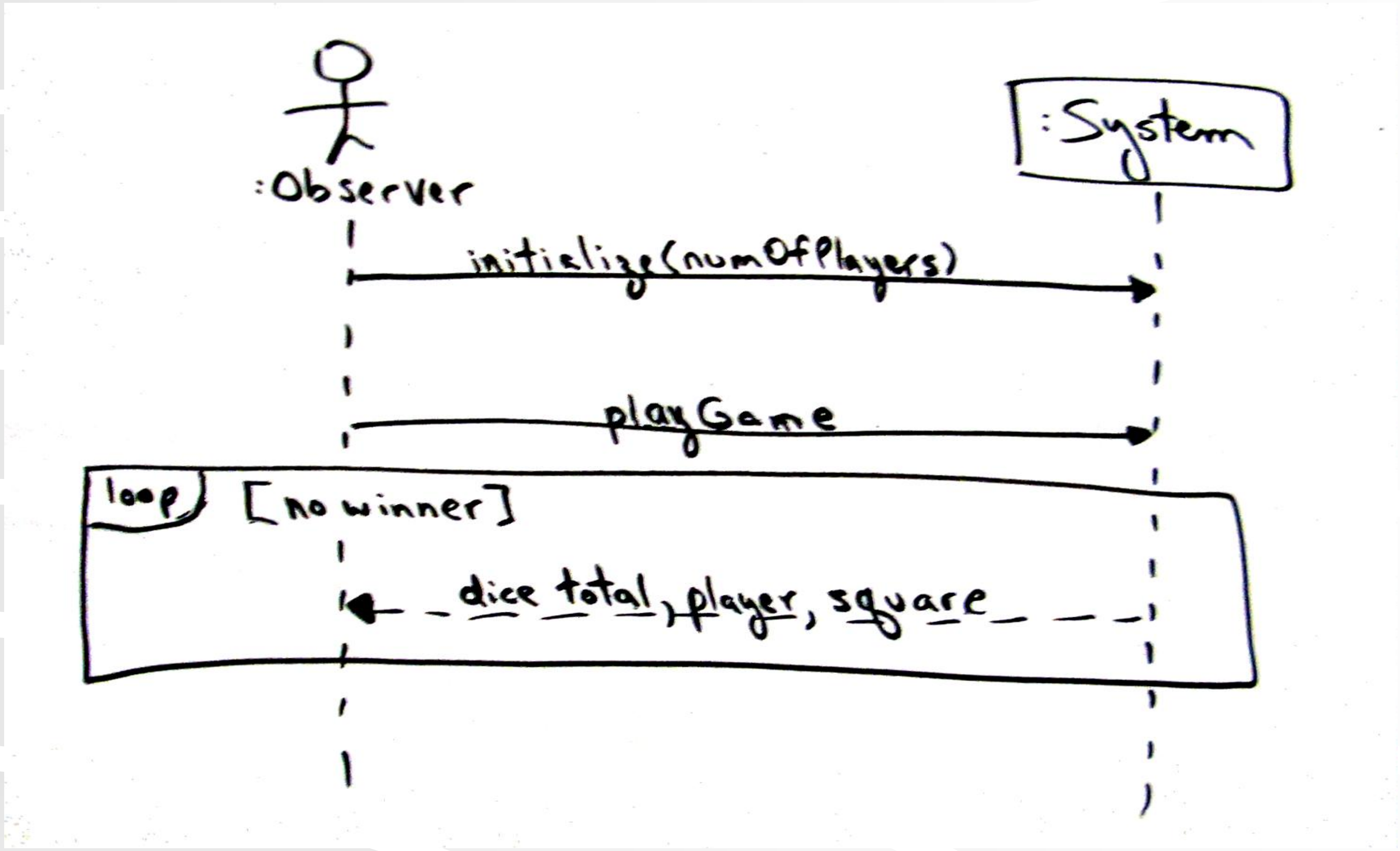
Low Coupling

- Low Coupling Reduces the Impact of Change
  - Evaluate design alternatives to get minimal coupling
  - Assign responsibility to minimize object coupling
  - Can use "Simple chain of command"



Original Design better where Board does getSquare()!

Coupling User Interface & Domain Layer

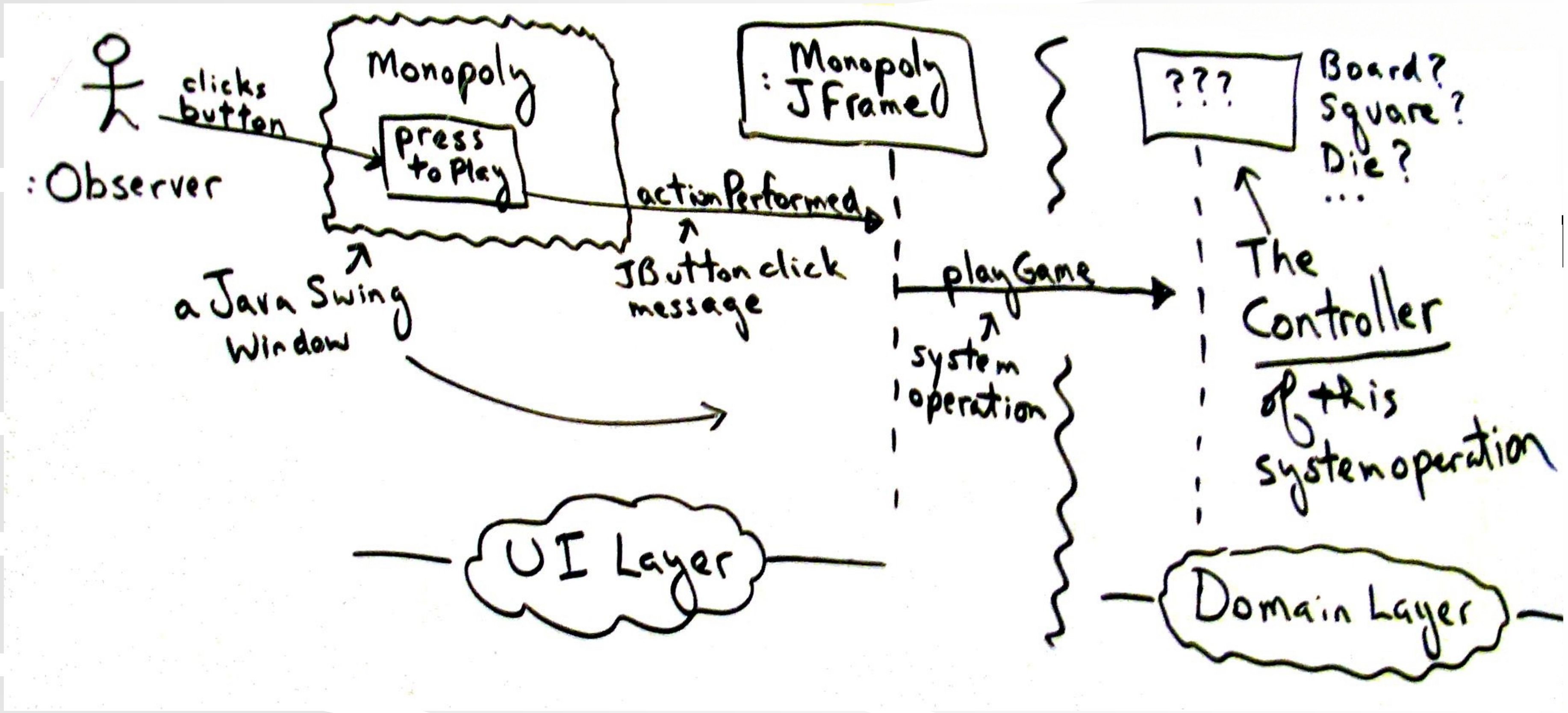


SSD for playing a Monopoly Game

- User directly interacts with a GUI, not the domain layer
- Which object should relay system operations from UI to domain layer? Let's take a look…

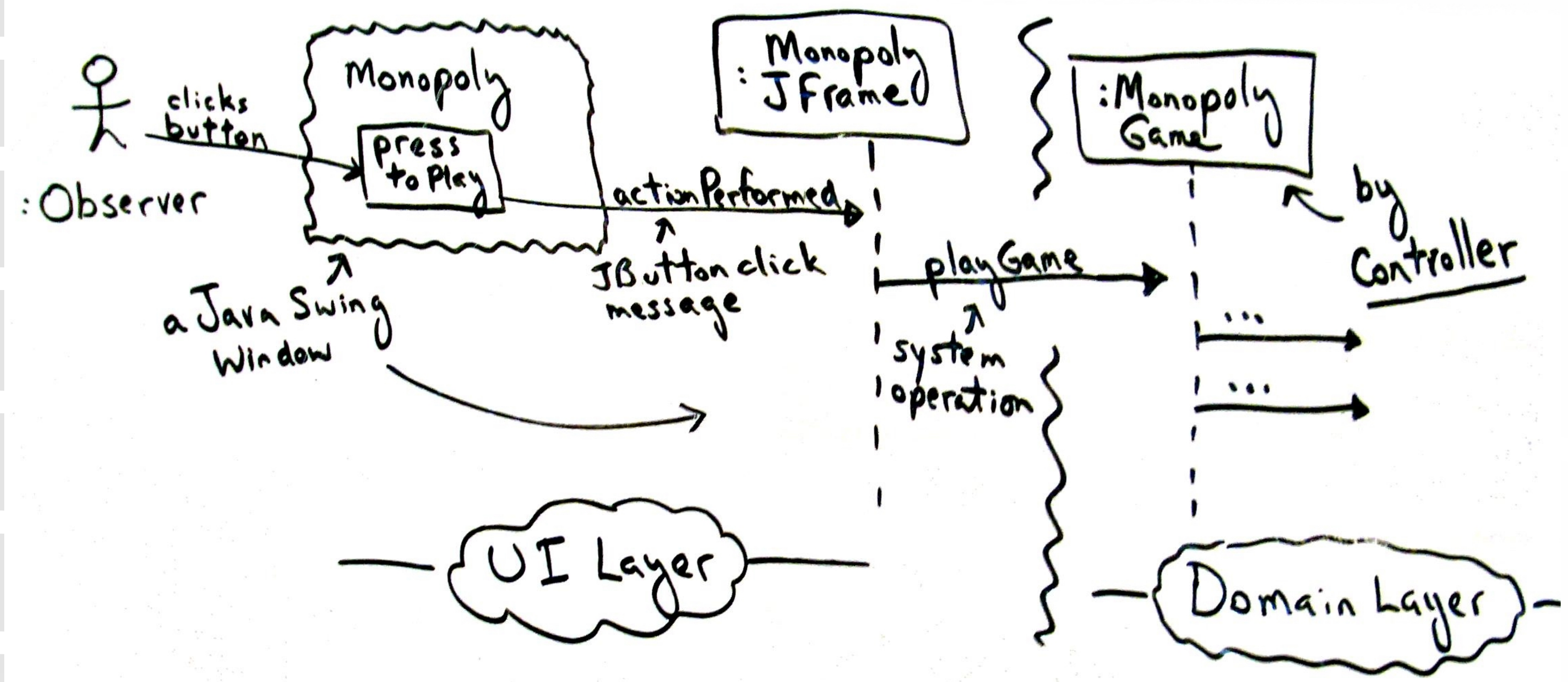
ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Layered view of Monopoly Game



- ❖Must translate UI event into system operation
- ❖Who mediates between UI and Domain layers? Hmmm?

More on Monopoly



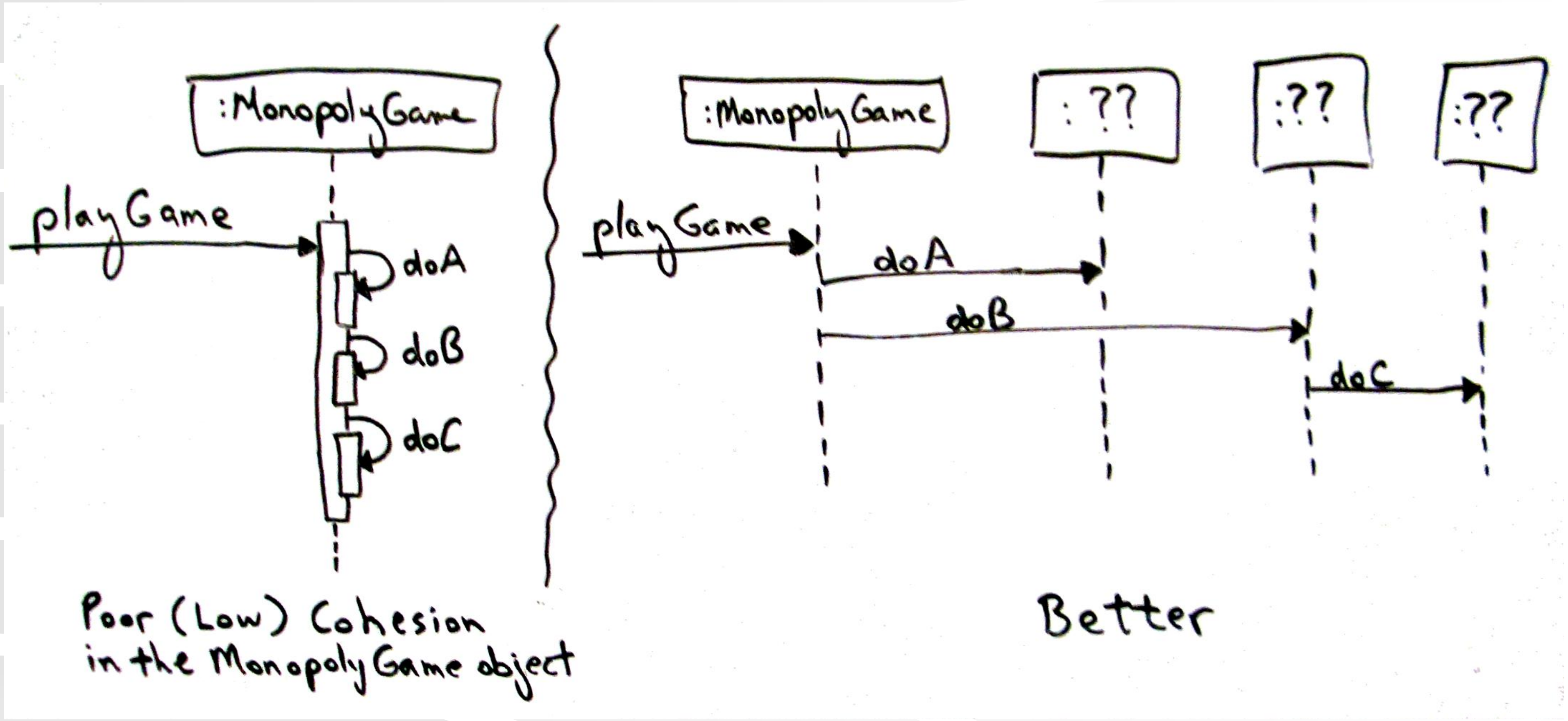
- Let MonopolyGame be controller …
- It represents the system and there aren't many system operations!

High Cohesion

- Keep objects focused, understandable and maintainable with Cohesion Principle
- How to support low coupling?
  - Assign so object's responsibilities are closely related
  - Evaluate alternatives to optimize cohesion
  - "Don't spread the responsible objects too thin"
  - "Teamwork"
- Inherent trade-offs of Cohesion and Coupling
  - To minimize coupling, a few objects have all responsibility
  - To maximize cohesion, a lot of objects have limited responsibility
  - Trade-off from alternative designs for best results

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Design Alternatives for High Cohesion



:MonopolyGame
playGame
doA
doB
doC
Poor (Low) Cohesion in the MonopolyGame object
:MonopolyGame
: ??
:??
:??
playGame
doA
doB
doC
Better