Puzzle of the Day

```
package puzzle;

public class LinePrinter {
  public static void main(String[] args) {
    // Note: \u000A is Unicode representation of linefeed (LF)
    char c = 0x000A;
    System.out.println(c);
  }
}
```

What does this program do?

Puzzle of the Day – Java Interpretation

```
package puzzle;

public class LinePrinter {
  public static void main(String[] args) {
    // Note:
is Unicode representation of linefeed (LF)
    char c = 0x000A;
    System.out.println(c);
  }
}
```

Puzzle of the Day – Fix

```
package puzzle;

public class LinePrinter {
  public static void main(String[] args) {
    System.out.println();
    System.out.println();
  }
}
```

CSSE 374
Software Design

Mark Hays

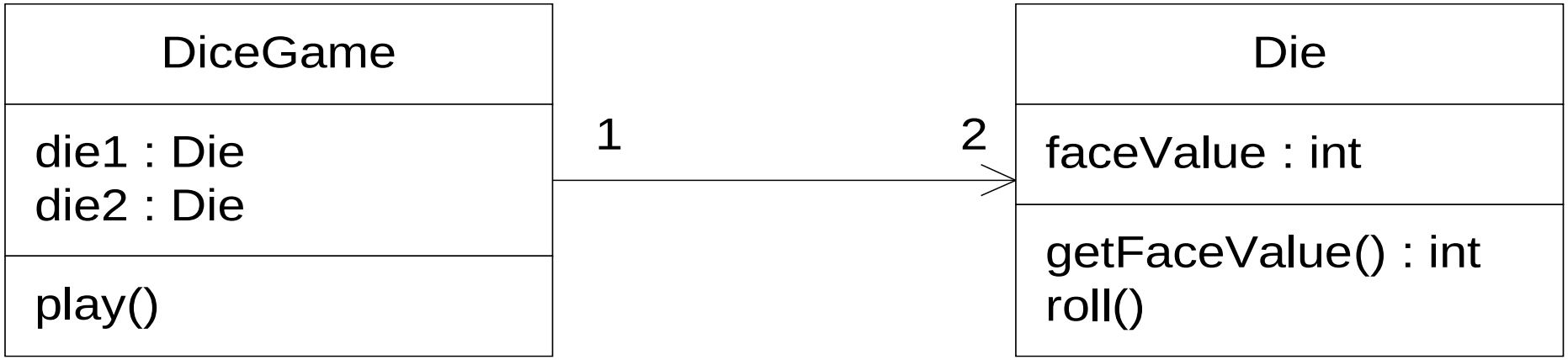
ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

What is Software?

- Think for 30 seconds…
- Then let's discuss it

What is Design?

- Think for 30 seconds…
- Then let's discuss it



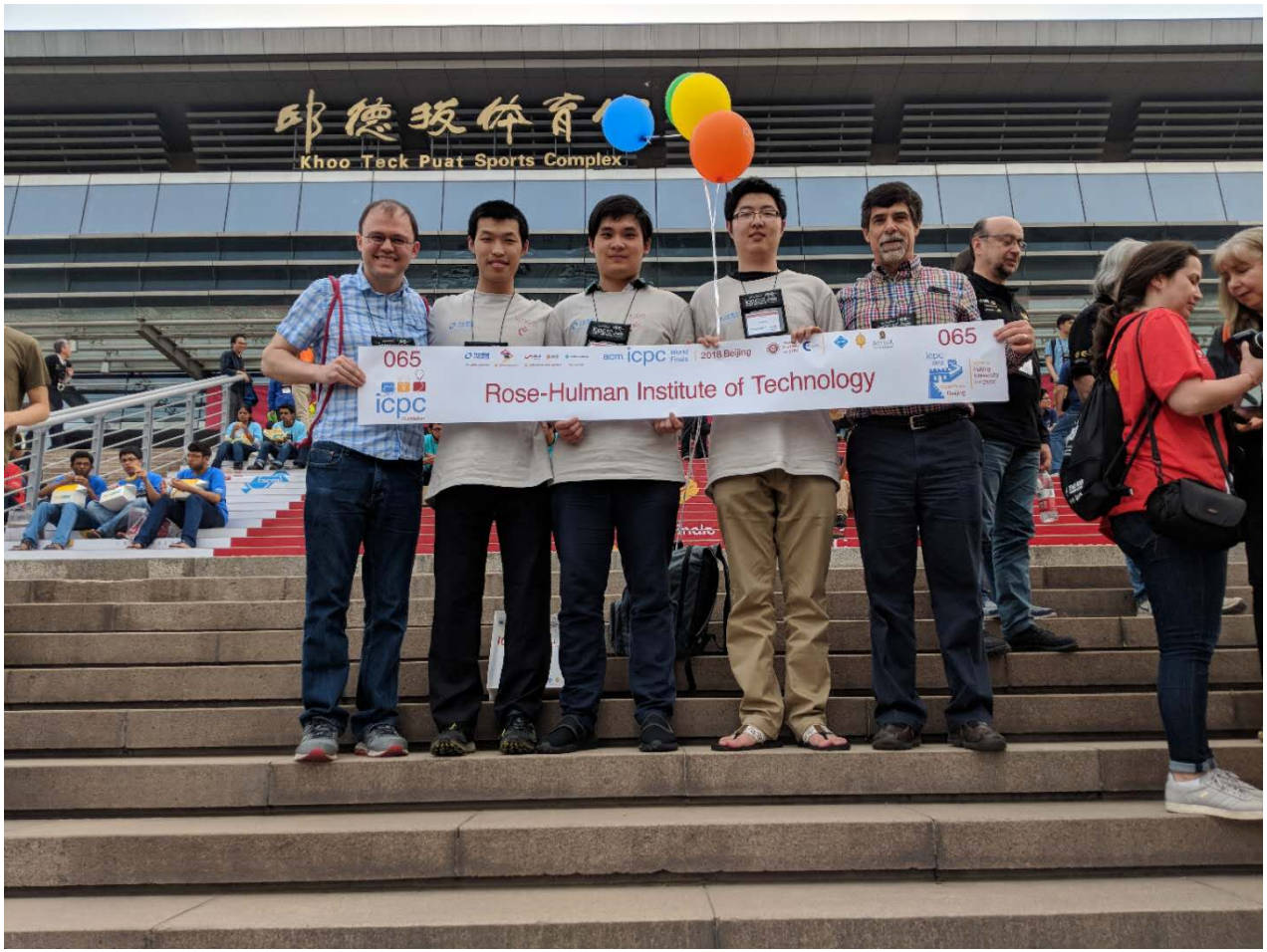
Is Design what Innovators do?

“In most people's vocabularies, design means veneer. It's interior decorating. It's the fabric of the curtains, of the sofa. But to me, nothing could be further from the meaning of design.

Design is the fundamental soul of a human-made creation that ends up expressing itself in successive outer layers of the product or service.”

Steve Jobs

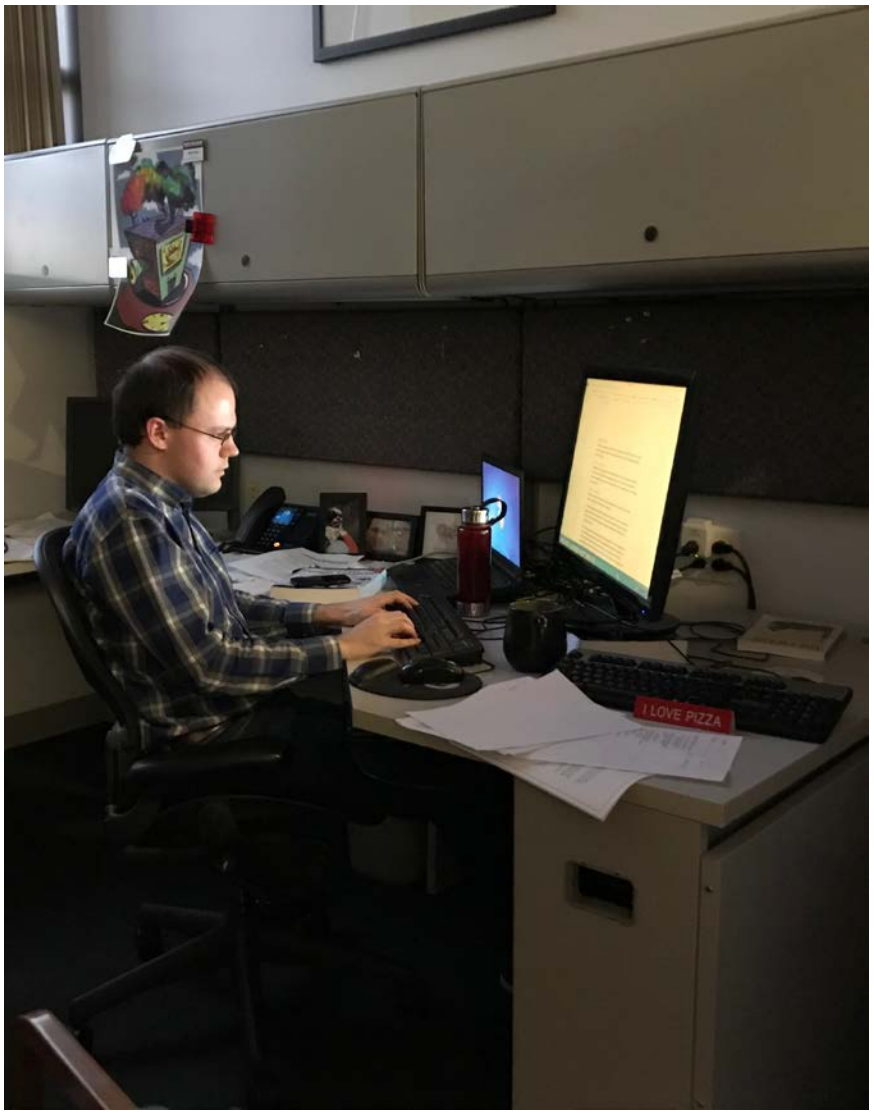
About me – coach of the Competitive Programming Team



About me – will carry to Bronze

About me – life of the Thanksgiving table

Open door policy



A typical week

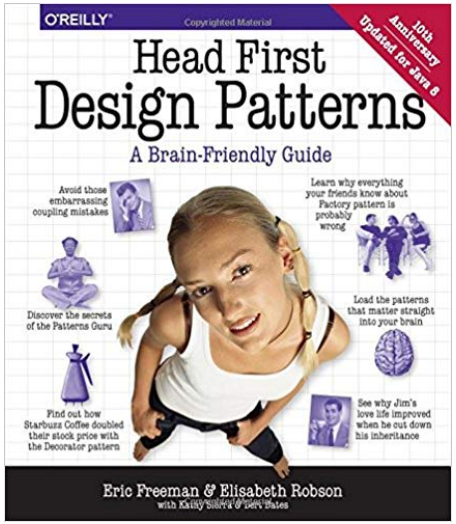
- Monday:
  - Reading quiz due Sunday night
  - Questions on the reading
  - Homework assigned
- Tuesday:
  - Design lecture
  - Design studio exercise
- Thursday:
  - Discuss term-long project
- Friday:
  - Homework due
  - Have a lab TA check off the implementation by 5:30PM

This week

- Monday:
  - Design lecture
- Tuesday:
  - **Reading quiz due Tuesday 6AM**
  - Questions on the reading
  - Homework assigned
- Thursday:
  - Project kickoff
- Friday:
  - Homework due
  - Have a lab TA check off the implementation by 5:30PM

Course Textbook and Readings

- Weekly reading quizzes on Moodle
- First quiz posted, due tomorrow morning



Academic Honesty

Be transparent.

Give credit where credit is due.

Do not share code.

Example – citing external code

```
// Print “hello” from the Linux kernel to the currently active console.
// (Outdated) reference:
// http://www.linuxtopia.org/online_books/Linux_Kernel_Module_Programming_Guide/x1141.html

current_task -> signal -> tty -> driver -> ops -> write(current_task -> signal -> tty,
                "hello\r\n",
                strlen("hello\r\n"));
```

The code for Linux 2.6 is mine, but I derived it from a dated source about 2.4. I **still** cite my source.

Example – discussing ideas with others

```
// Emily Richardson gave me the idea to implement
// Strategy pattern with a Map.
Map<String, LineReader> lineReadersByCompany = …
```

Exams

- Two evening exams
- Final exam during finals week

Project

- Five milestones, usually due every two weeks
  - Whole-class critique of “initial design”
  - End-of-milestone demo with instructor

- Look for CATME team formation survey in your email this evening

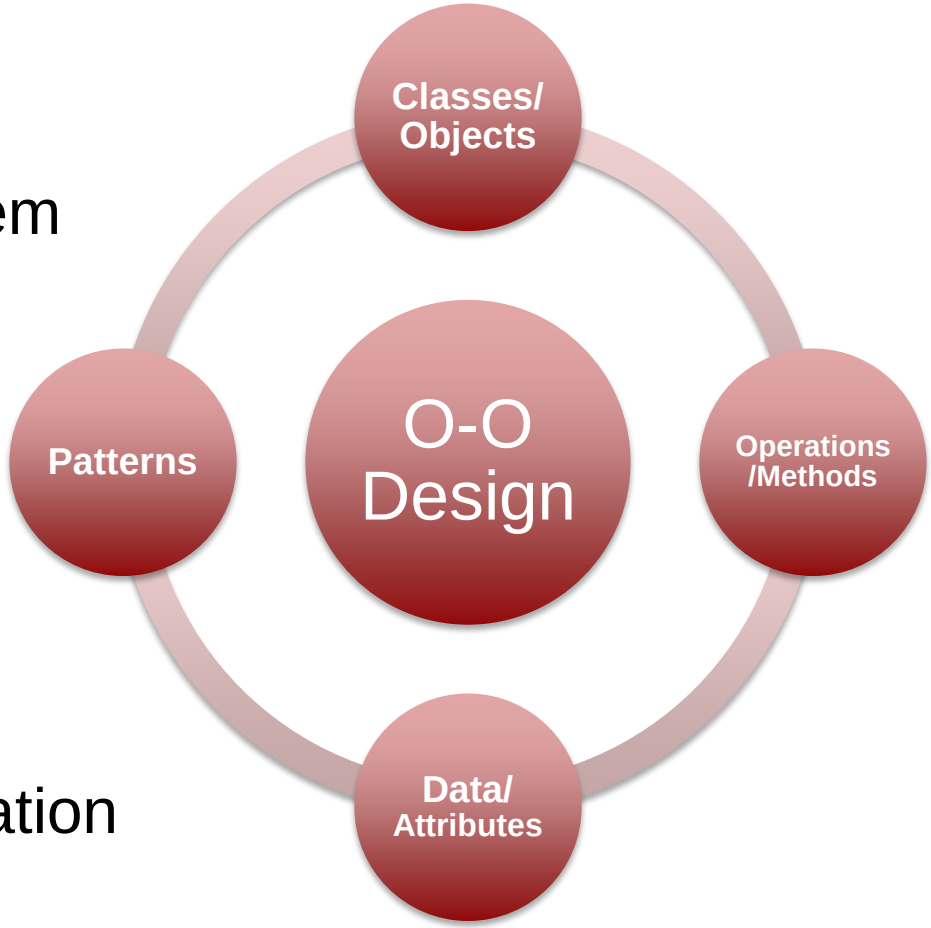
Where do designs come from?

This week's design lecture

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

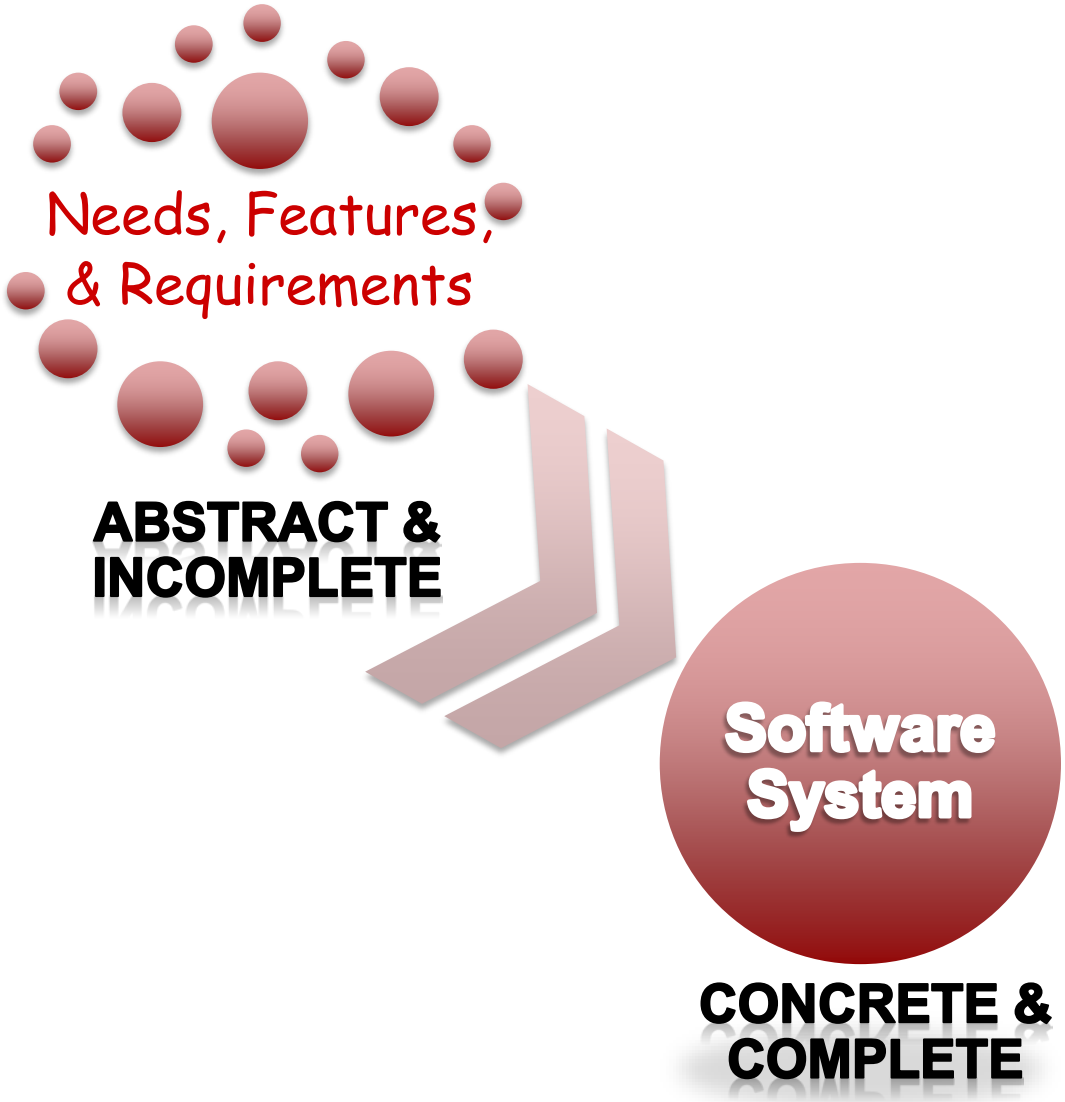
Analysis versus Design

- Analysis ➔ Validate
  - Investigation of the problem and requirements, rather than a solution
  - Do the right thing…
- Design ➔ Verify
  - A conceptual solution, rather than its implementation
    - Excludes low level details
  - Do the thing right…

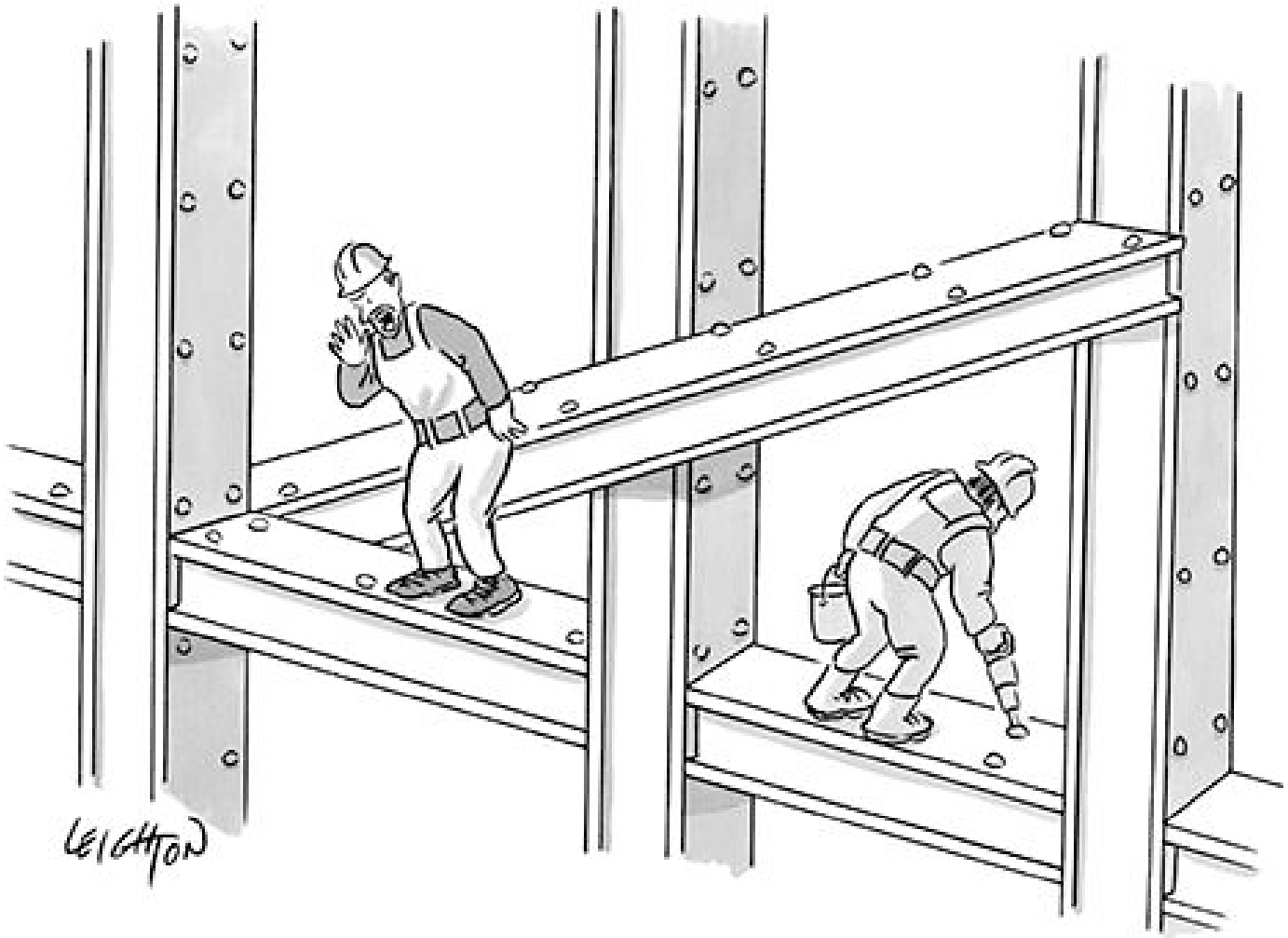


Why is Software Design Important?

- Size
- Complexity
- Constraints
- Performance
- Communication

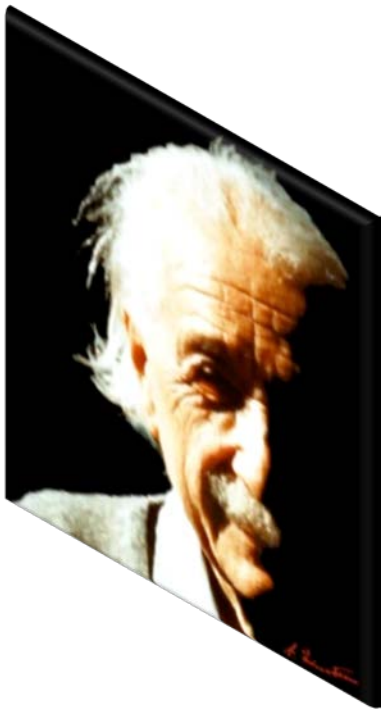


How CSSE220 students tend to design software



Thinking at the Right Level

- Abstraction - hiding irrelevant details to focus attention at right level
- Process of component identification is top-down, decomposing the system into successively smaller, less complex components
- Process of integration, which is bottom-up, building (composing) the target system by combining components in useful ways



Key Questions for Object-Oriented Design

1. What classes do we get from the application domain?
2. How should responsibilities be allocated to classes?
3. How should objects collaborate?

Guided by design patterns

Process of design

1. Define a Domain Model
2. Assign Object Responsibilities
3. Determine Interactions
4. Construct Class Diagrams

Let's do a Quick Example: Dice

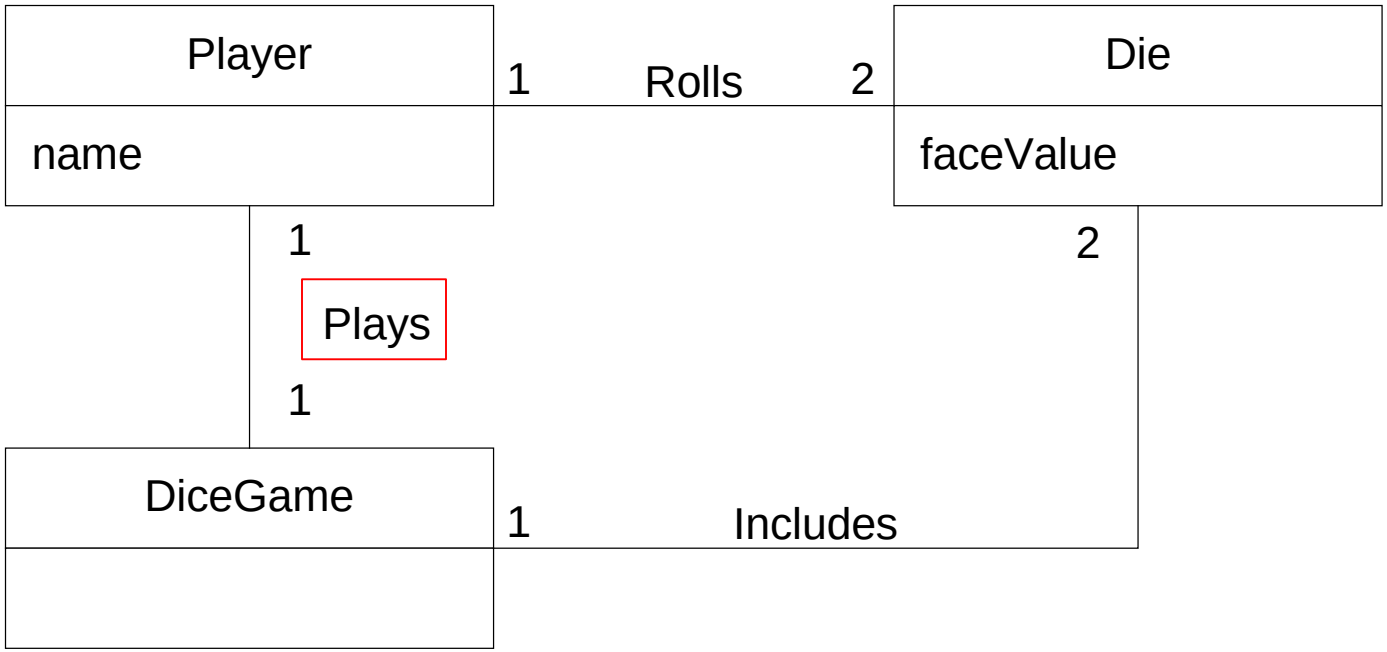
Use Case Description

- Play a dice game: Players requests to roll the dice. System presents results: If the dice face value totals seven, player wins; otherwise player loses

1. Define a Domain Model
2. Assign Object Responsibilities
3. Determine Interactions
4. Construct Class Diagrams

1. Define a Domain Model

2. Assign Responsibilities



Who is playing whom?

Message passing metaphor

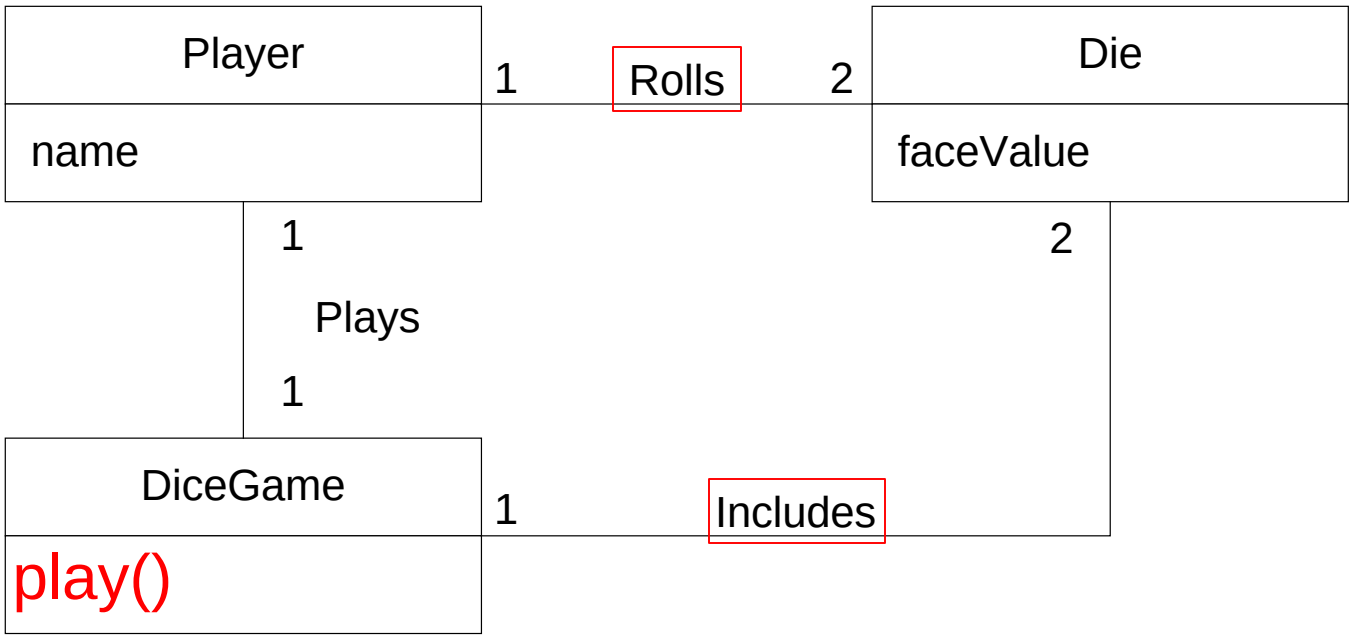
- “A Player plays a DiceGame”

- In OO speak:

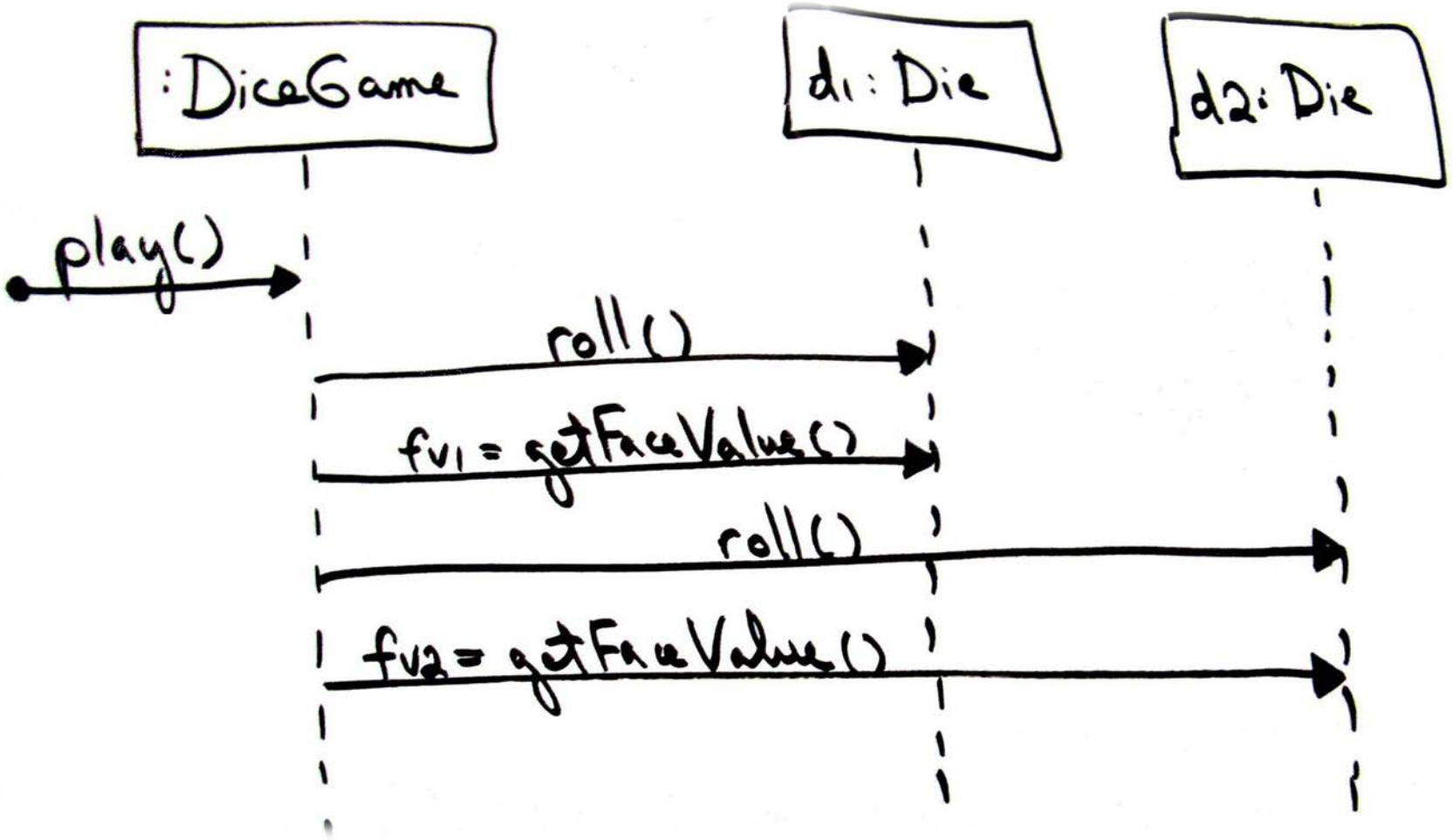
  A Player sends a play() message to a DiceGame.
  A DiceGame knows how to receive a play() message.

- In Java: DiceGame has the play() method

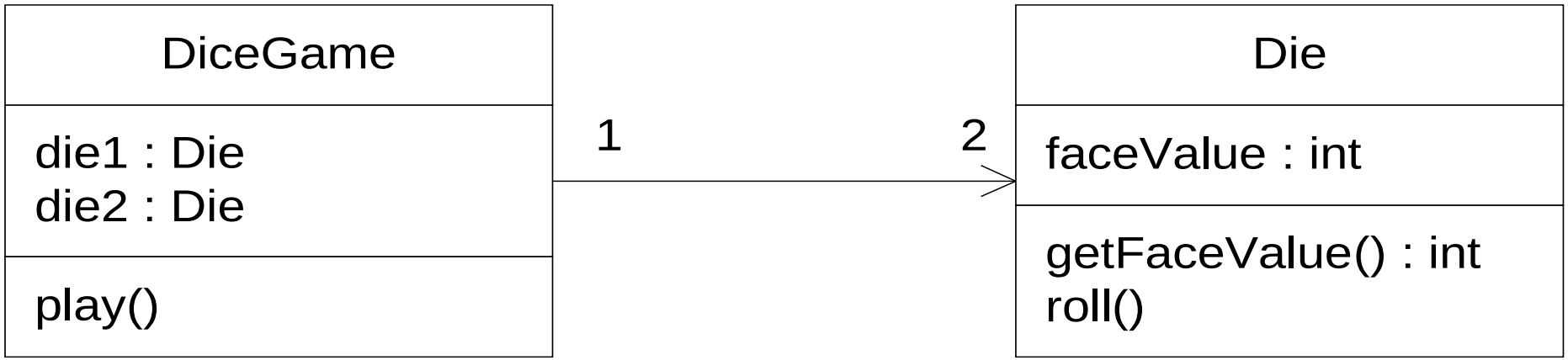
2. Assign Responsibilities



3. Sequence Diagram for Rolling Dice



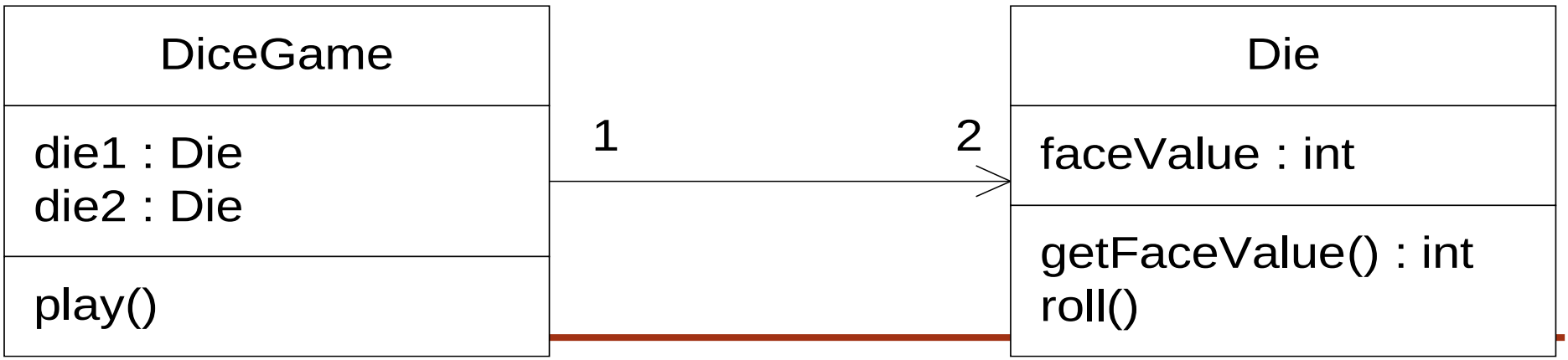
4. Construct Class Diagram



How does it differ from the domain model?

Recap – overall process of design

1. Define a Domain Model
2. Assign Object Responsibilities
3. Determine Interactions
4. Construct Class Diagrams

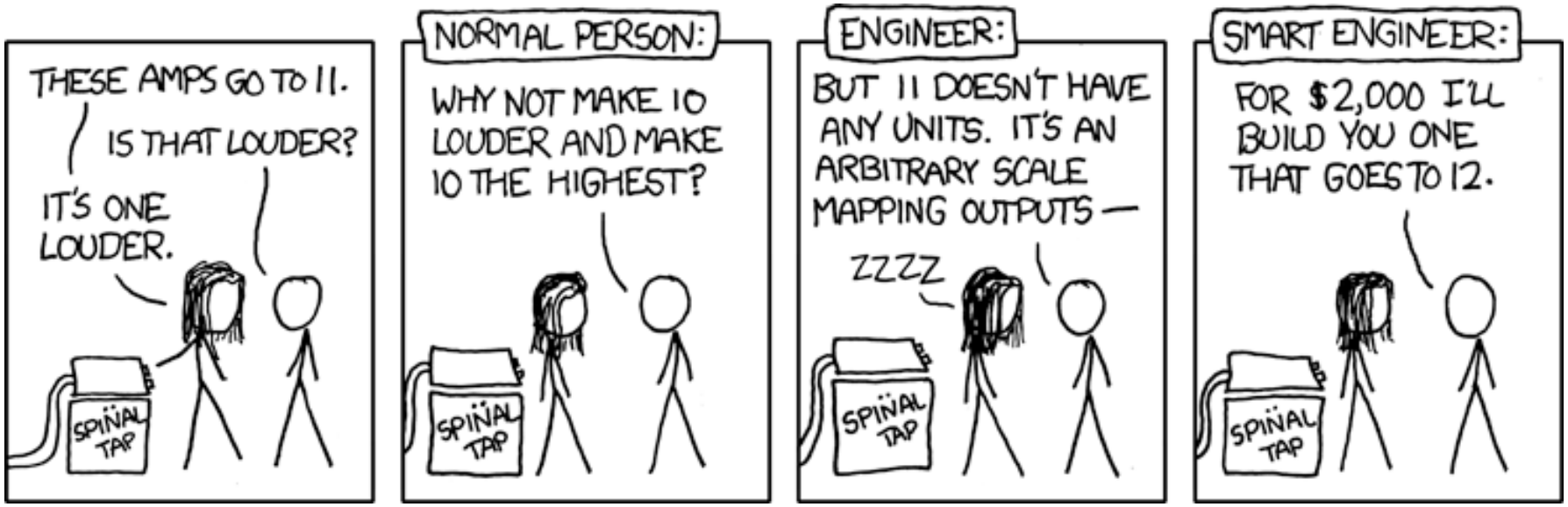


Break – 5 minutes

Four mysteries remain

1. We created a domain model – somehow
2. We came up with ideas for classes – somehow
3. We kept some association lines, but not others
4. How did we decide on the attributes?

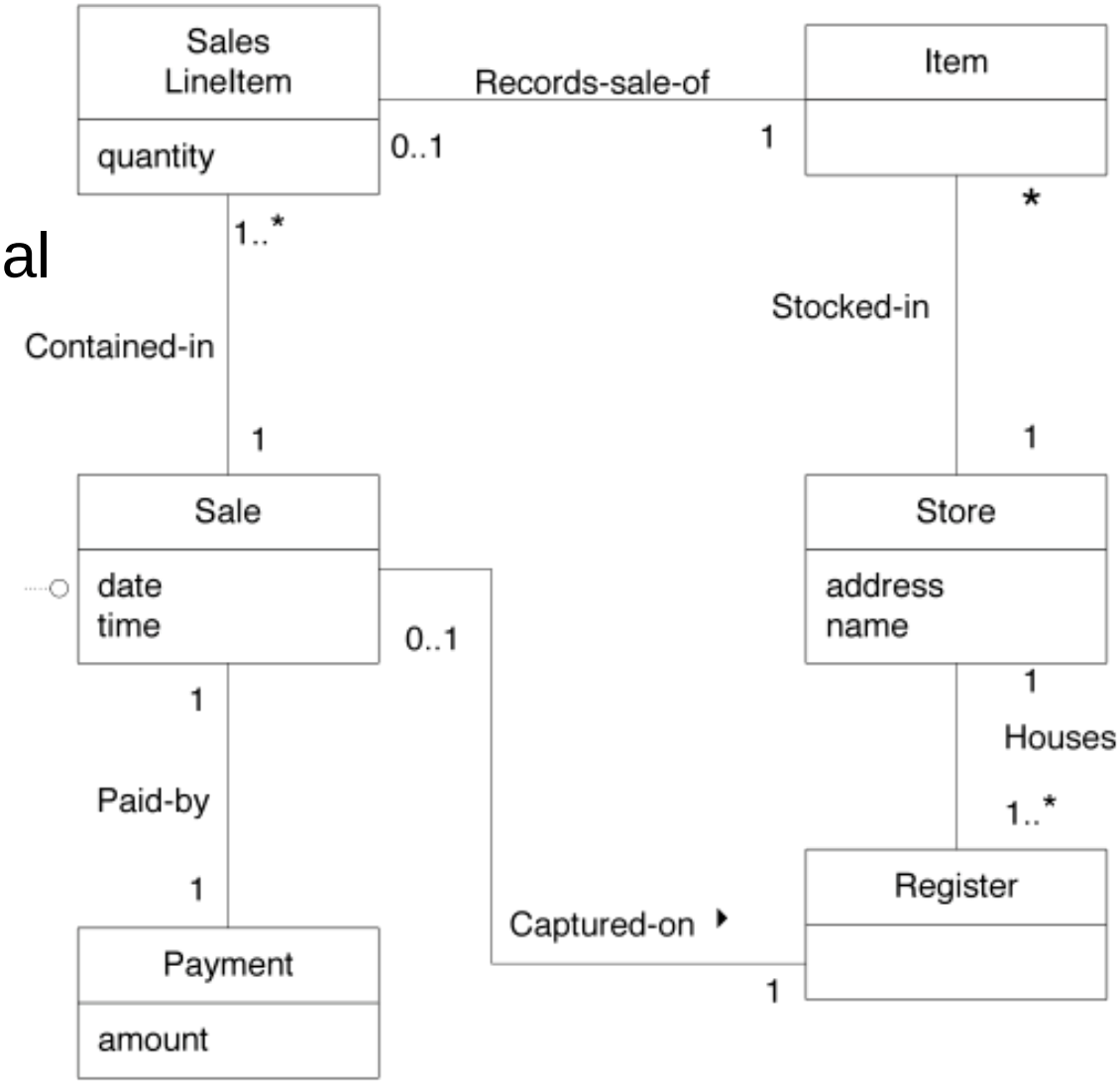
Mystery 1: Domain Models



Wow, that's less than $200 per ... uh ... That's a good deal!

What is a Domain Model? (1 of 2)

- Key Object-Oriented Analysis Model
- Abstraction of Conceptual Classes
- Illustrates noteworthy domain concepts
- Provides inspiration for naming design objects
- Notation trivial, but it takes practice to build a useful model



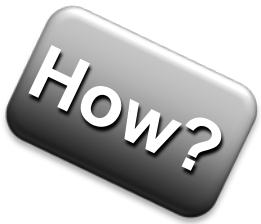
Source: Craig Larman

What is a Domain Model? (2 of 2)

- Visual representation of conceptual classes and their relationships
- Focuses on one domain
- Illustrated using UML class diagrams without operations

Why Create a Domain Model?

- Domain Model Easier for Users to Understand
- Names from domain model used (inspire) in naming in design models
- Goal: lower representational gap
  - Think like a mapmaker!
- Helps us:
  - Understand the software
  - Maintain the software



How to Create a Domain Model

1. Find the conceptual classes

2. Draw them as classes in a UML class diagram

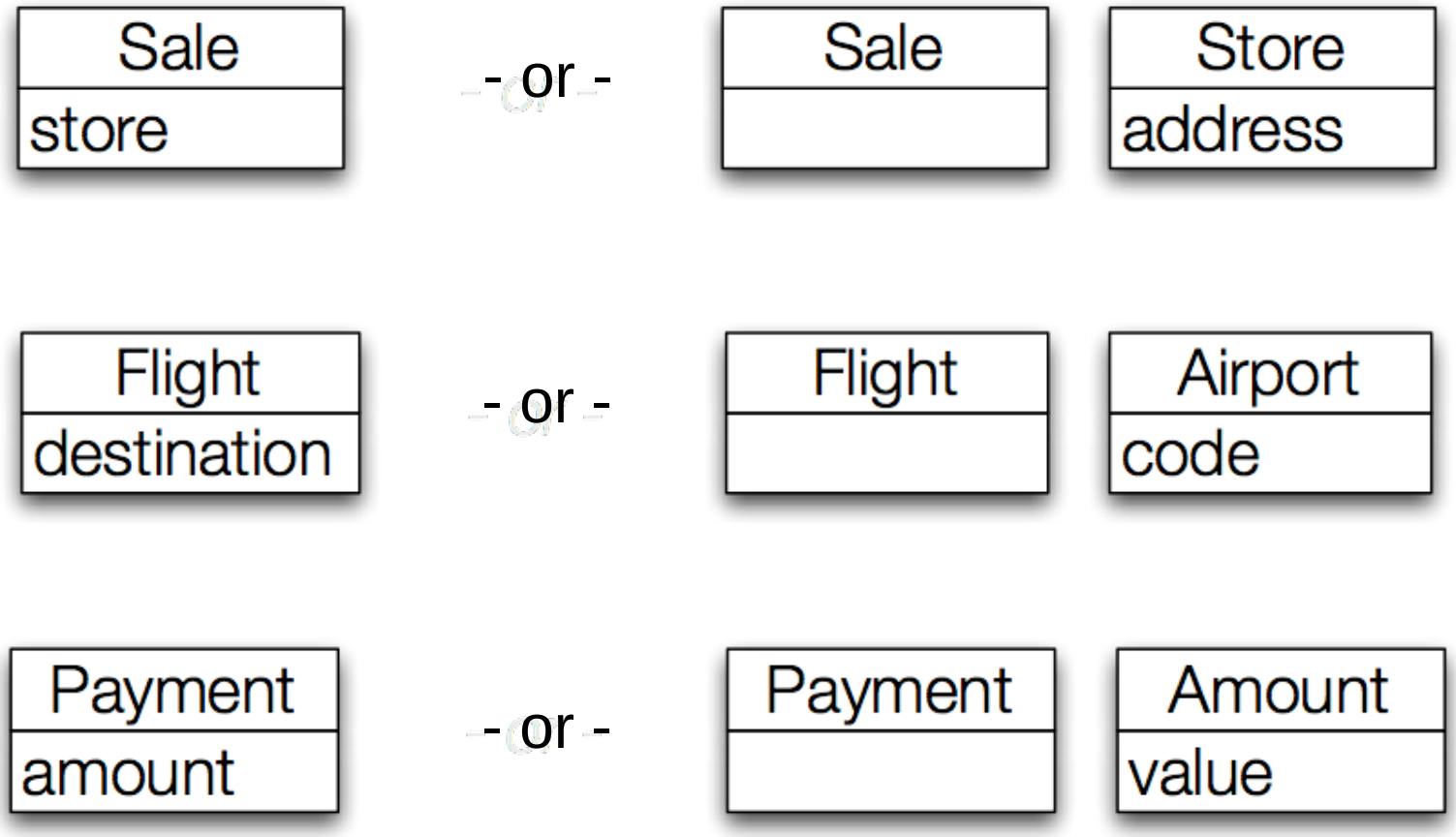
3. Add associations and attributes (but not operations)

Mystery 2: Conceptual Classes

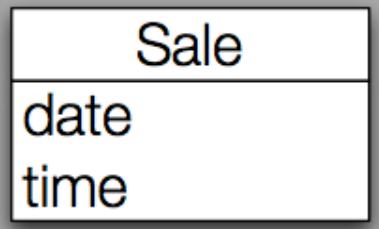
- A conceptual class is an idea, thing, or object
- Formally, a conceptual class can be represented as:
  - a symbol, it's intension, or it's extension

- Rules of thumb
  - If it takes up space, then it is probably a conceptual class
  - If you can't think of a thing as a number or text, then it is probably a conceptual class

Attribute or Class?

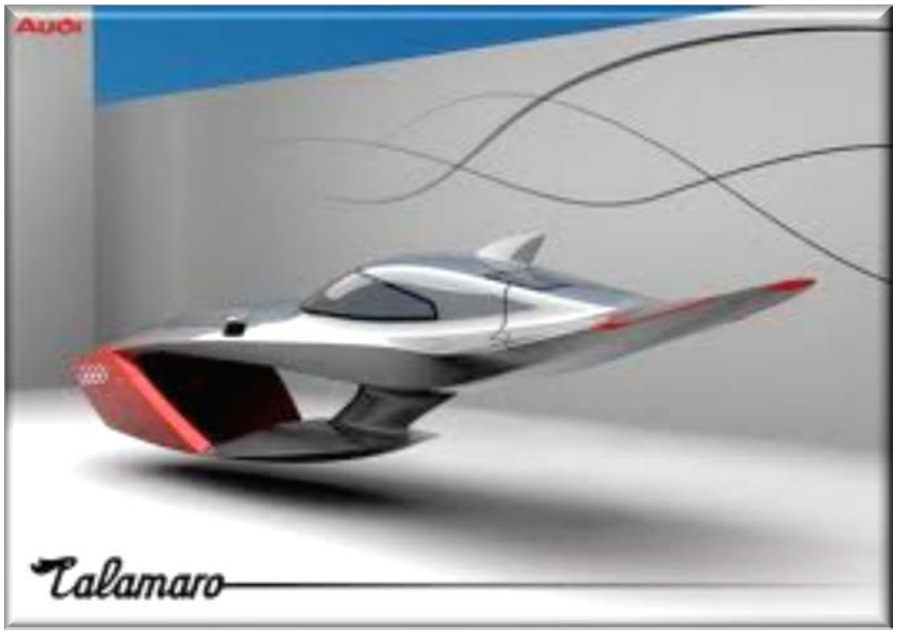


Conceptual Class, More Formally

	Example	Not Just an OO Idea
Symbol		$\mathbb{N}$
Intension	"A sale represents the event of a purchase transaction. It has a date and time."	$\{x \in \mathbb{Z} \mid x \geq 0\}$
Extension	*sale-1, 198702131914* *sale-2, 19691216084212* *sale-3, 19920815202830*	$\{0, 1, 2, 3, \ldots\}$

Strategies to Find Conceptual Classes

1. Reuse or modify existing models
2. Identify noun phrases; linguistic analysis
3. Use a category list

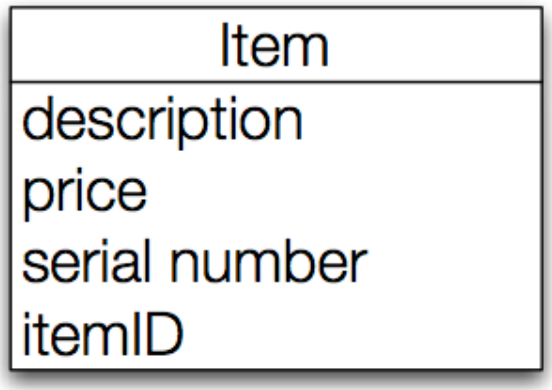


Category Lists for Conceptual Classes

Conceptual Class Category	Examples
Business transactions *Here's where the $ is!*	Sale, Payment
Physical objects *Important for control systems, simulations*	Item, Register
Containers of things	Store, Aisle, Bin
…	…

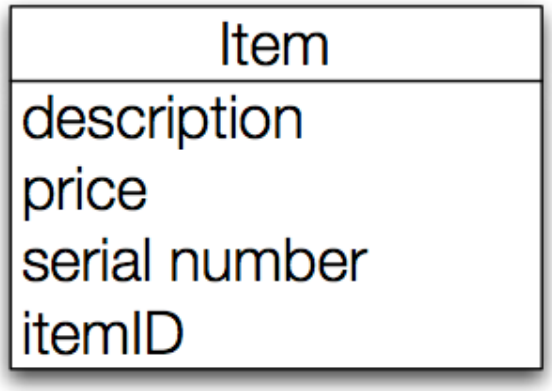
Aside: how not to be sensitive about design reviews...

Where things go wrong



- Assume an *Item* instance represents a physical item in a store
- Item data only recorded within *Item* instances
- When a real-world item is solid, we remove the software *Item* from a collection and it's garbage collected

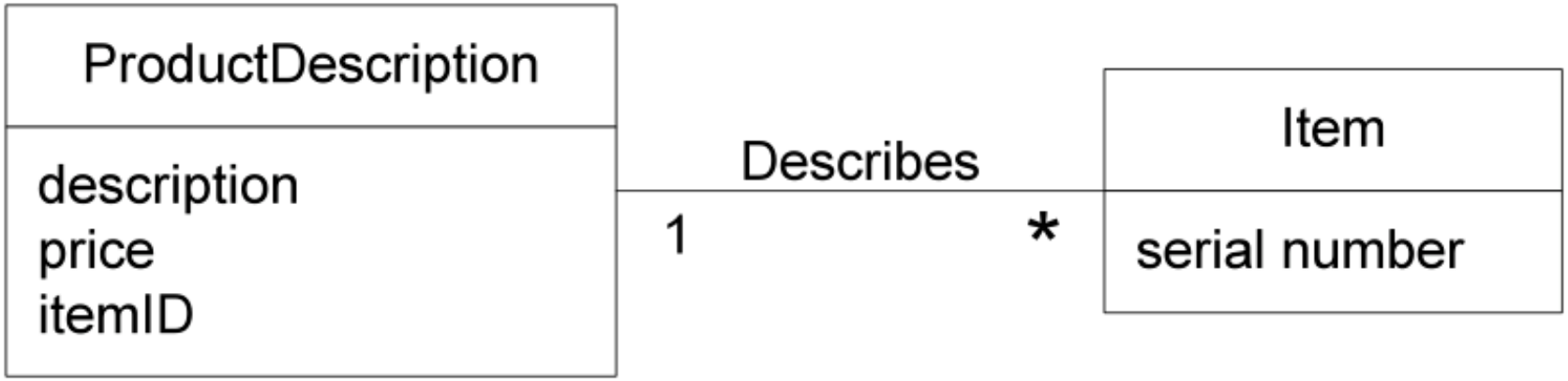
Problems



- Lose memory of the price, etc., if no *Item* instances remain in the system

- Duplicate data
  - Wasted space
  - Error-prone

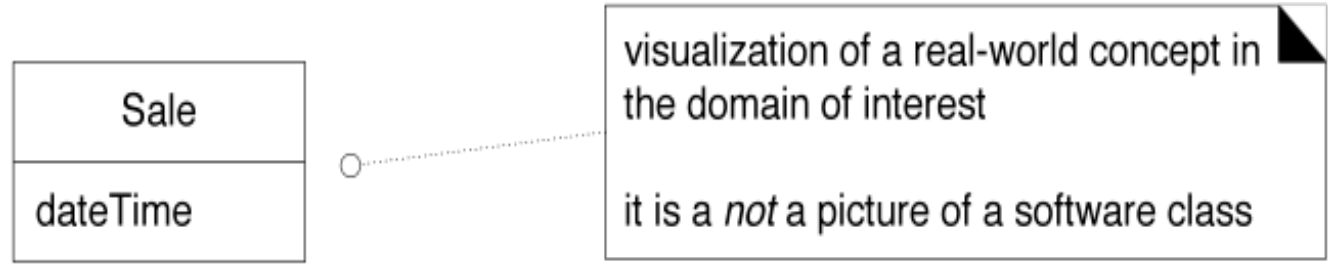
Solution: Use Description Class



- When information must be retained independent of existence of instances of the described item
- When deleting the described item could result in information loss
- When it reduces redundant information

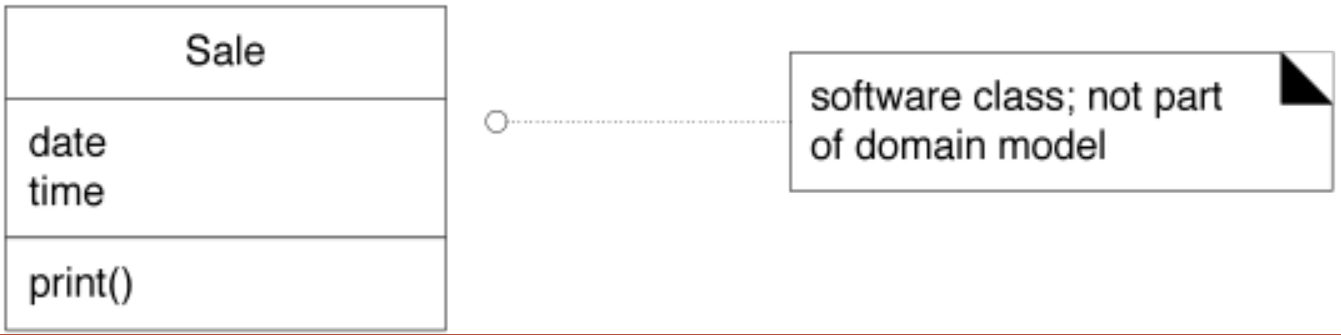
Avoid Premature Design

Good

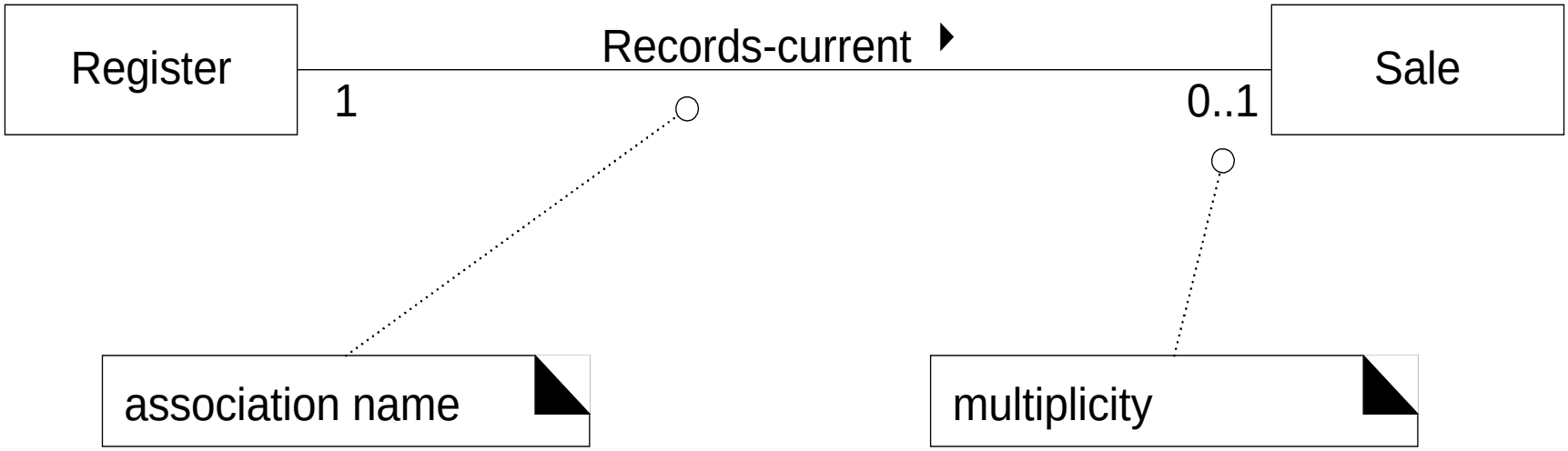


Avoid

Avoid



Mystery 3: Associations



- A meaningful relationship between objects
- Knowledge of a relationship must be preserved (some memory of a relationship)
- Associations are NOT
  - A model of data flows
  - Instance variables
  - DB foreign keys
  - SW object connections

Check your understanding...

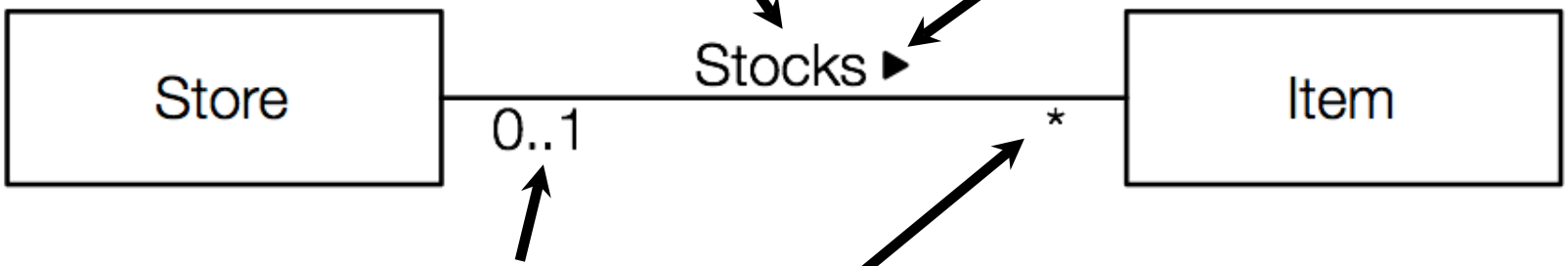
Do we likely need to remember that:

1. SalesLineItems are associated with a Sale?
2. A Piece is on a BoardSquare?
3. A Piece is owned by a Player?
4. A roll of the Dice calculated a BoardSquare to move to?
5. A Cashier looked up a ProductDescription?

Association Notation

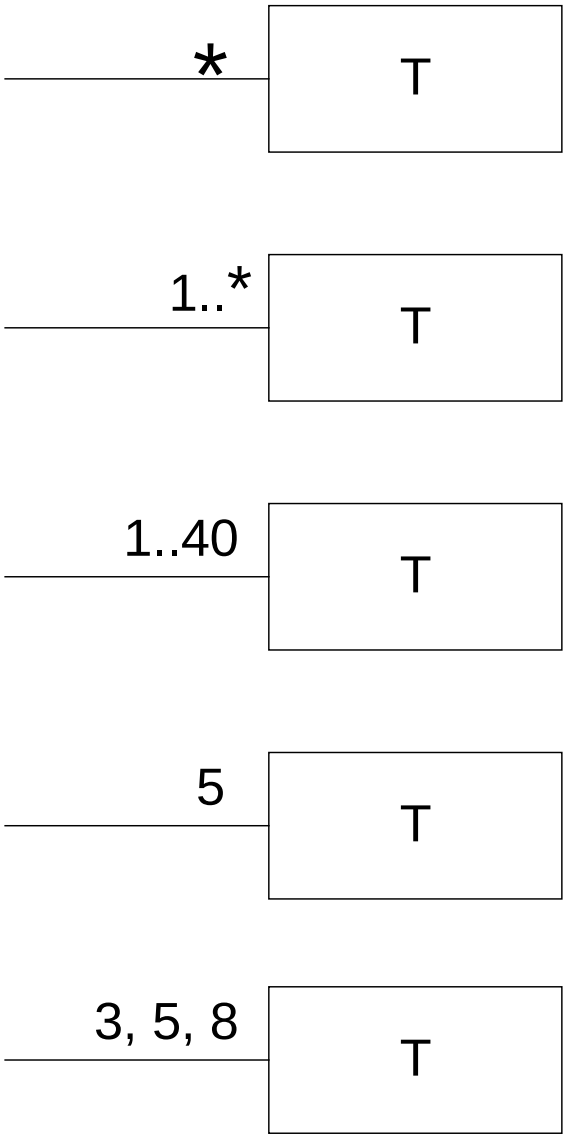
Association name:
- ✓ Use verb phrase
- ✓ Capitalize
- ✓ Typically camel-case or hyphenated
- ✓ Avoid “has”, “use

Reading direction: Can exclude if association reads left-to-right or top-to-bottom

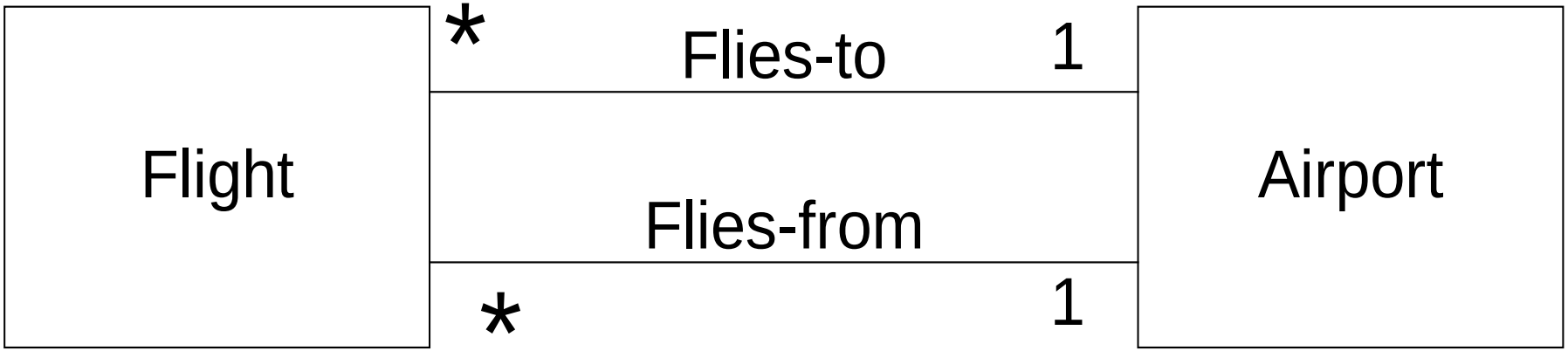


Multiplicity (Cardinality):
- ✓ ‘*’ means “many”
- ✓ x..y means from x to y inclusively

Cardinality (AKA Multiplicity)



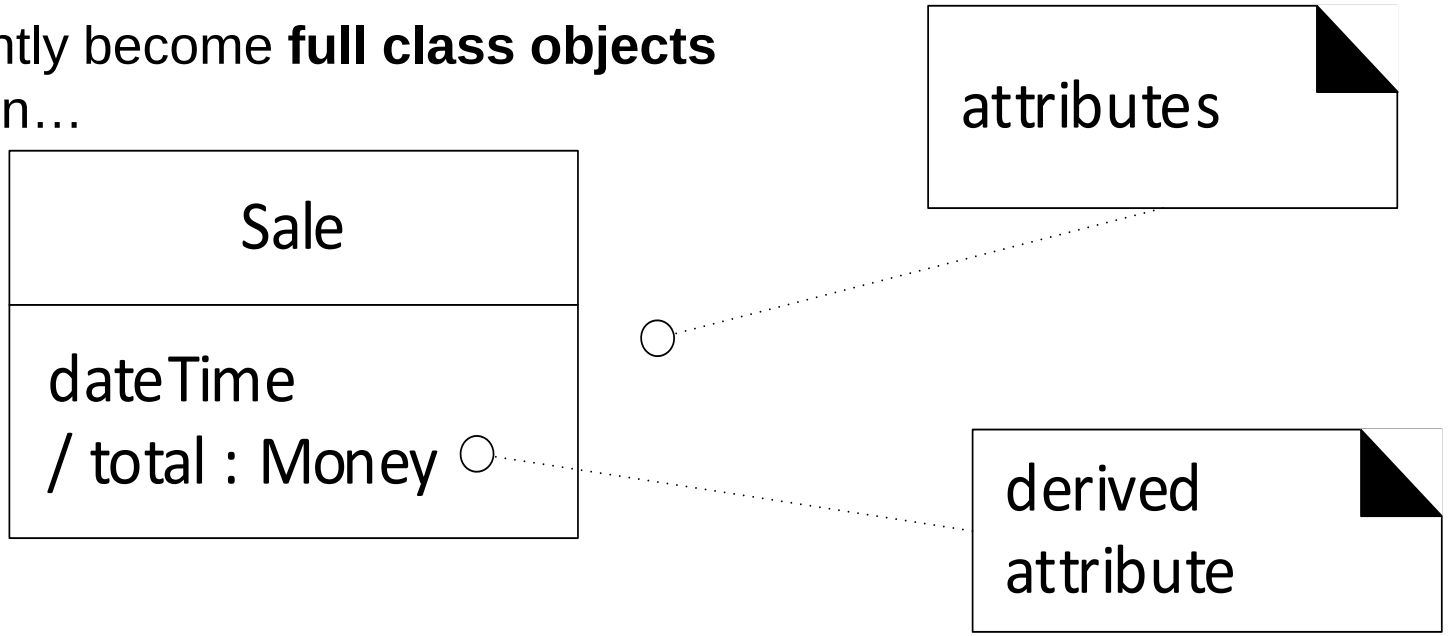
Let's Associate some more...



- **Classes can have multiple associations**
- **Classes can also self-associate!**
  - **Object creates itself**
  - **Object modifies itself**
  - **Object moves itself…**

Domain Model: Attributes

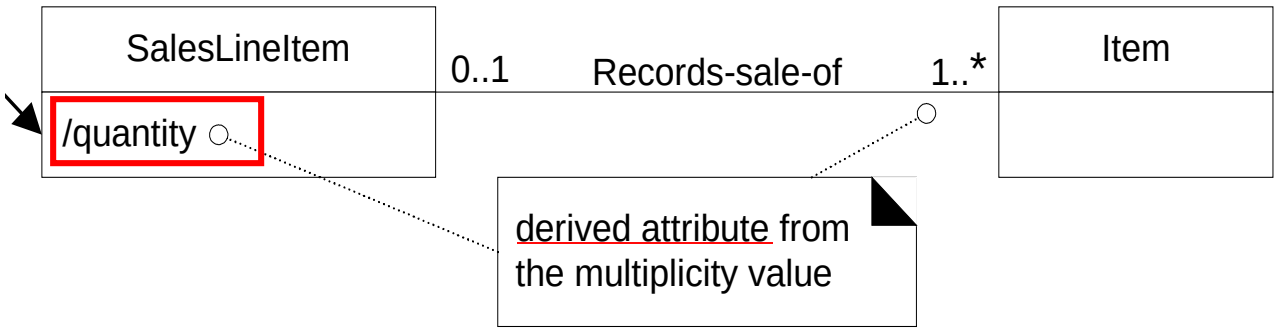
- An object's logical data value that must be remembered
  - Some attributes are derived from other attributes
- Usual primitive attributes (data types not shown in DM)
  - Numbers, characters, booleans
- Common compound attributes
  - Date, time, address, SSN, phone number, bar codes, etc.
  - Frequently become **full class objects** in design…



Recording Quantity of Items

Each line item records a separate item sale.
For example, 1 tofu package.

Each line item can record a group of the same kind of items.
For example, 6 tofu packages.



Recap – overall process of design

1. We created a domain model by:
2. Identifying the conceptual classes
   - Ideas and things that exist in the problem domain
   - That are not primitive data
3. Drawing associations for relations that need to be remembered
   - Recording their meaningful relationship and multiplicity
4. Making attributes for associations with primitive data
   - … but avoid duplication and inconsistency

Homework and Reading Reminders

- Reading Chapter 1 of Text, quiz on Moodle.
  - Due 8:00am TOMORROW, November 27th, 2018
- Review the UML reference material on Moodle
- Complete CSSE 374 CATME Survey
  - Due by 5:00pm Wednesday, November 28th, 2018