Visibility and Mapping Designs to Code

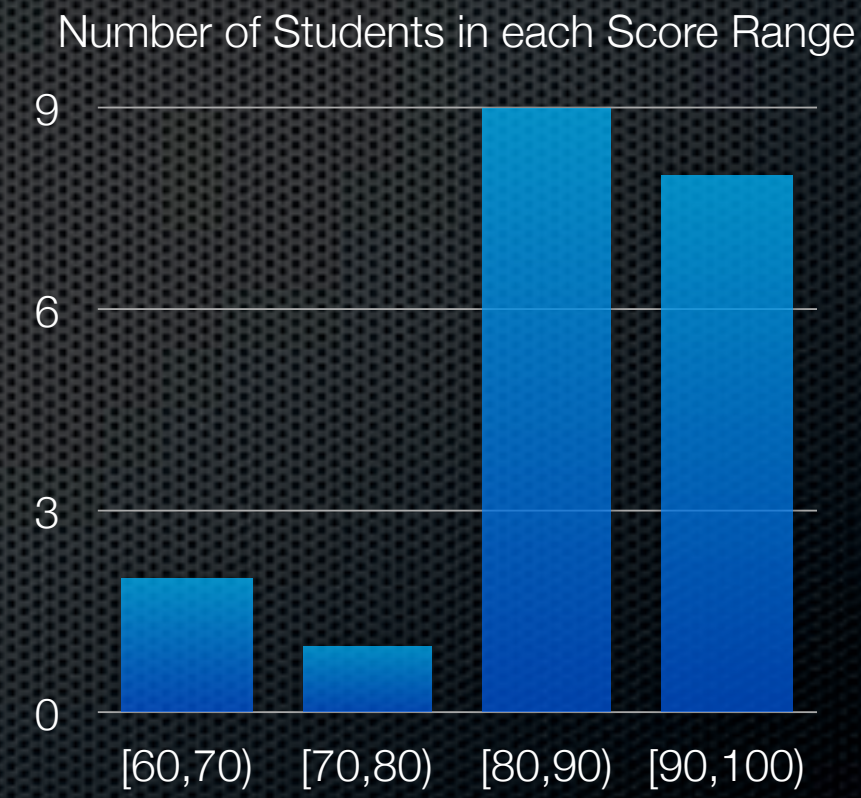
Curt Clifton

Rose-Hulman Institute of Technology

Q1

Exam 1 Results

- Max: 96
- Mean: 85.7
- Min: 67.5



Visibility

Visibility

- An object *B* is **visible** to an object *A* if *A* can send a message to *B*
- Related to, but not the same as:
  - Scope
  - Access restrictions (*public*, *private*, etc.)
- What are four common ways that *B* can be visible to *A*?

Attribute Visibility

- Object *A* has **attribute visibility** to object *B* if …
  - *A* has an attribute that stores *B*
- Quite permanent
- Most common

Parameter Visibility

- Object *A* has **parameter visibility** to object *B* if …
  - *B* is passed in as an argument to a method of *A*
- Not permanent, disappears when method ends
- Second most common
- Methods often convert parameter visibility to attribute visibility

Local Visibility

- Object *A* has **local visibility** to object *B* if …
  - *B* is referenced by a local variable in a method of *A*
- Not permanent, disappears when leaving variable's scope
- Third most common
- Methods often convert local visibility to attribute visibility

Q3

Global Visibility

- Object *A* has **global visibility** to object *B* if …
  - *B* is stored in a global variable accessible from *A*
- Very permanent
- Least common (but highest coupling risk)

Q4,5

Cartoon of the Day

Used with permission. http://notinventedhe.re/on/2009-9-23

From Design to Code

So Far…

Depending on the system, many of these steps might just be sketches!

- Created **Domain Model** from requirements and use cases
- Used **System Sequence Diagrams** to identify **system operations**
- Clarified system operations with **Operation Contracts**
- Assigned “doing” responsibilities with **Interaction Diagrams** (**Communication** and **Sequence Diagrams**)
- Assigned “knowing” responsibilities with **Design Class Diagrams**

Q6,7

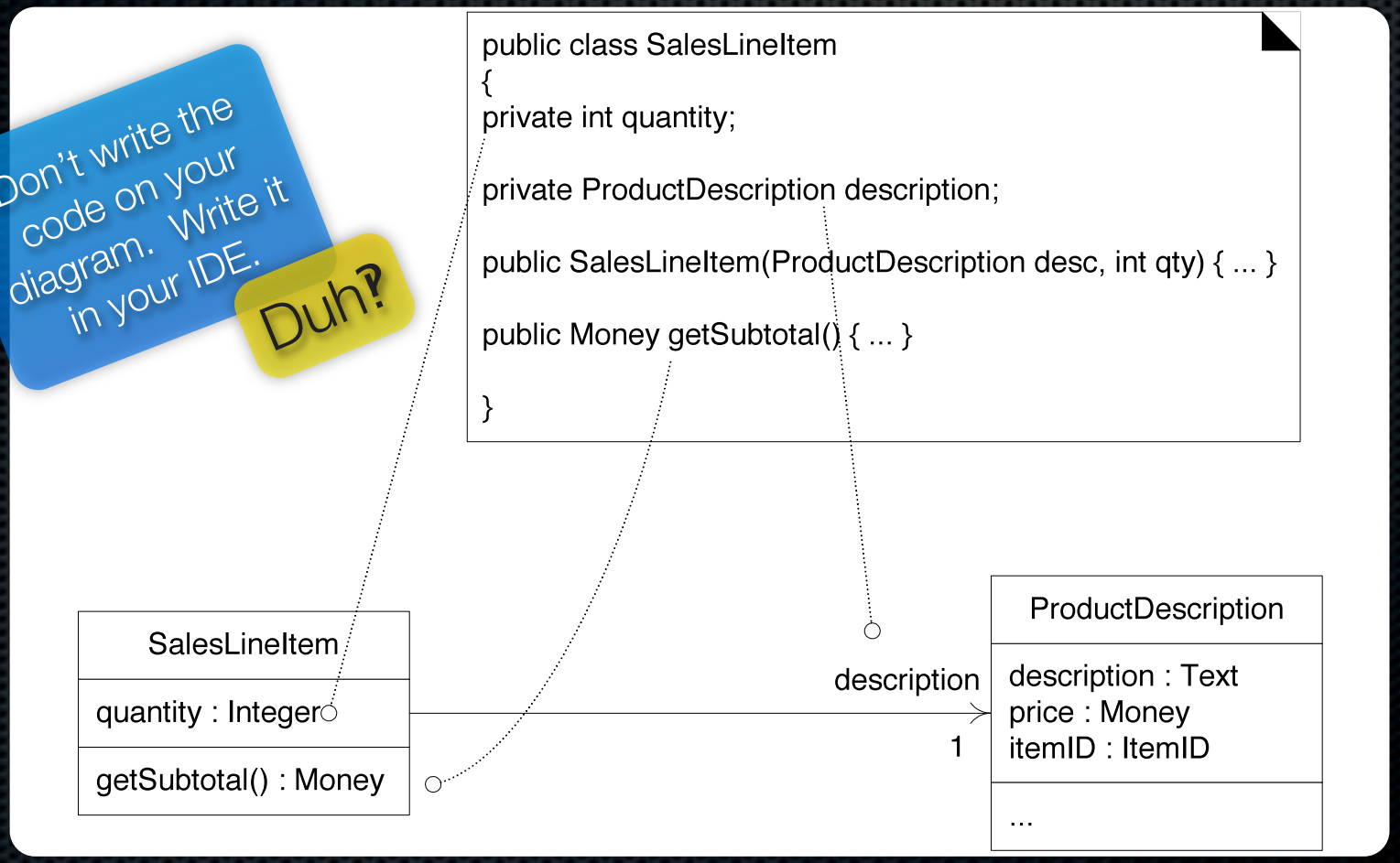
Next Up

- Use design documents to start writing code

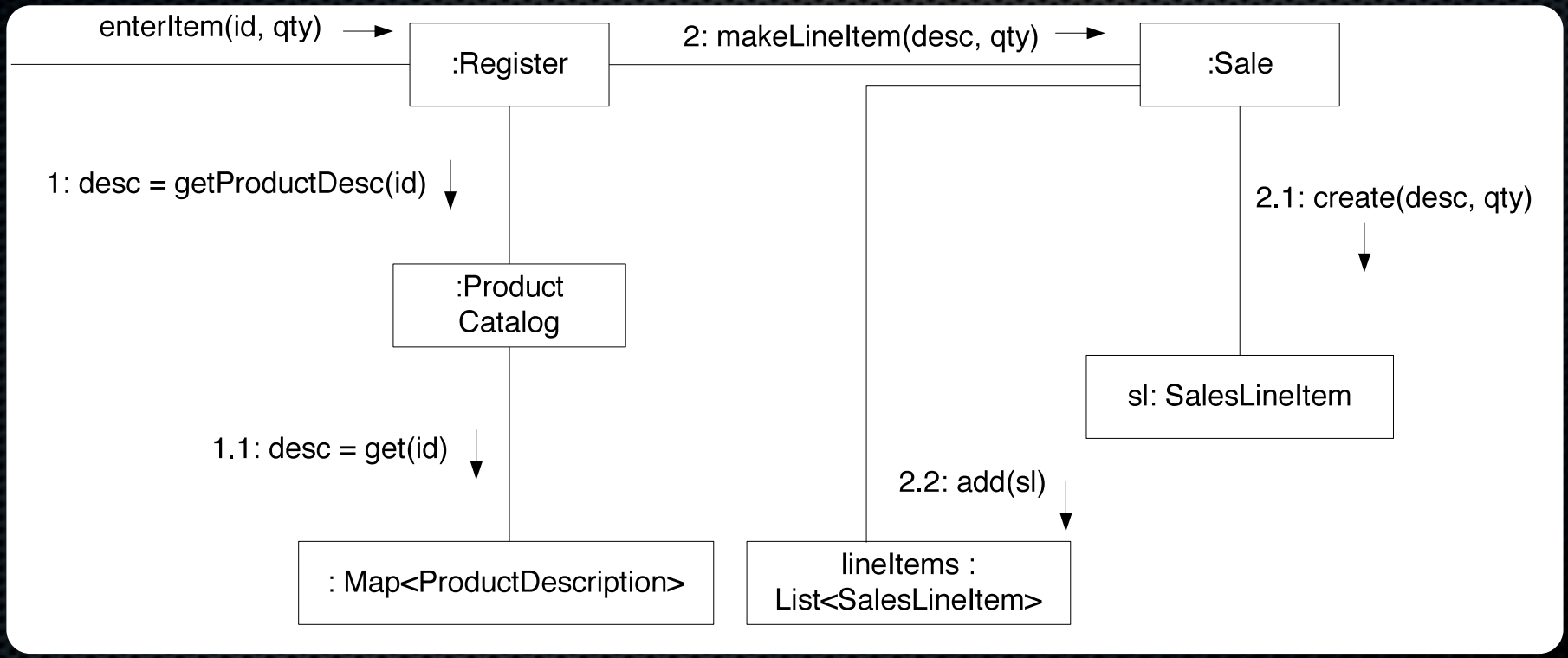
But the Design Isn't Done!

- More design will happen during coding
- The design documents (or sketches) provide a starting point
- More design (and probably more analysis) will be done in future iterations

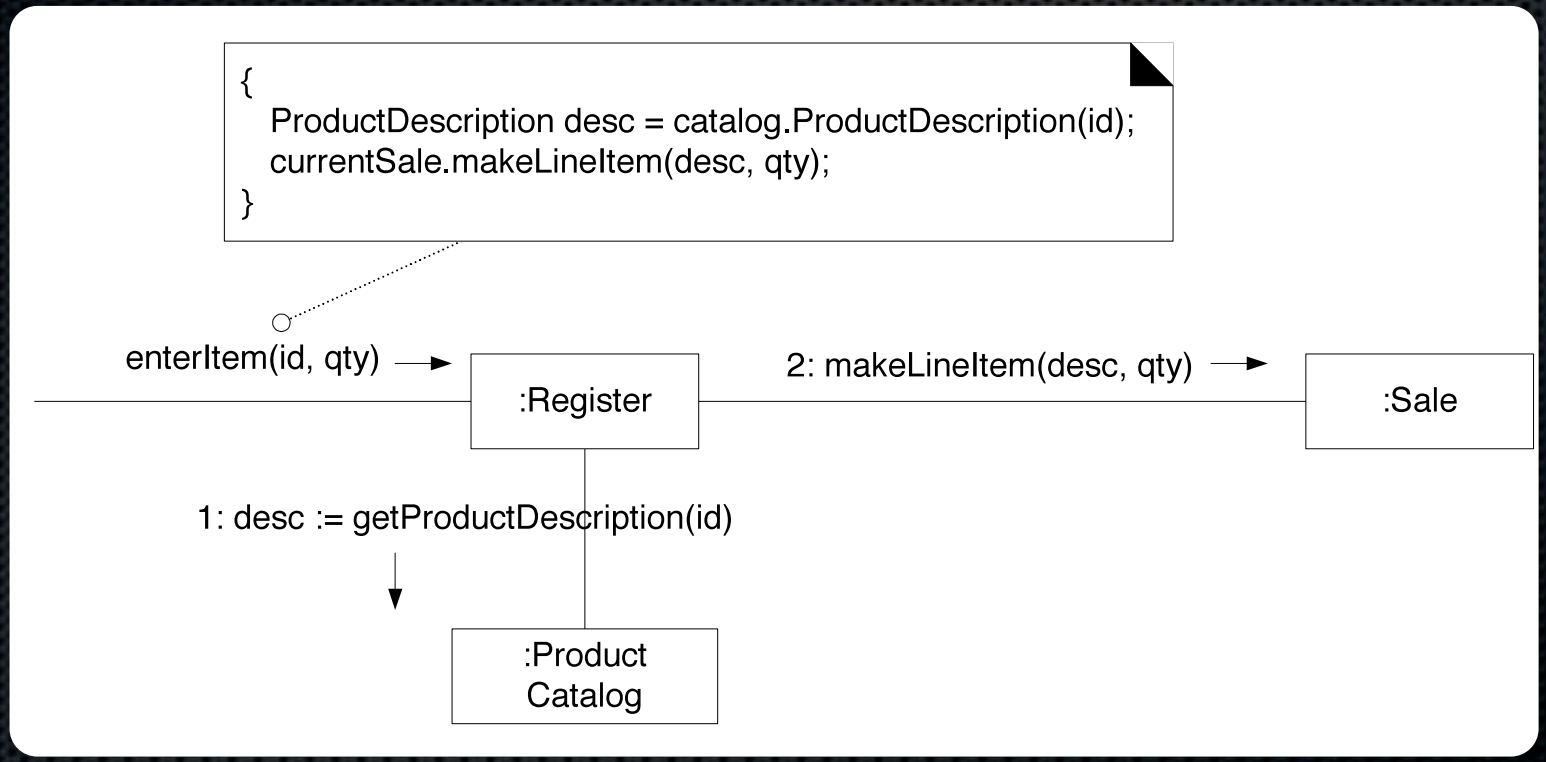
Create Class Definitions from DCDs



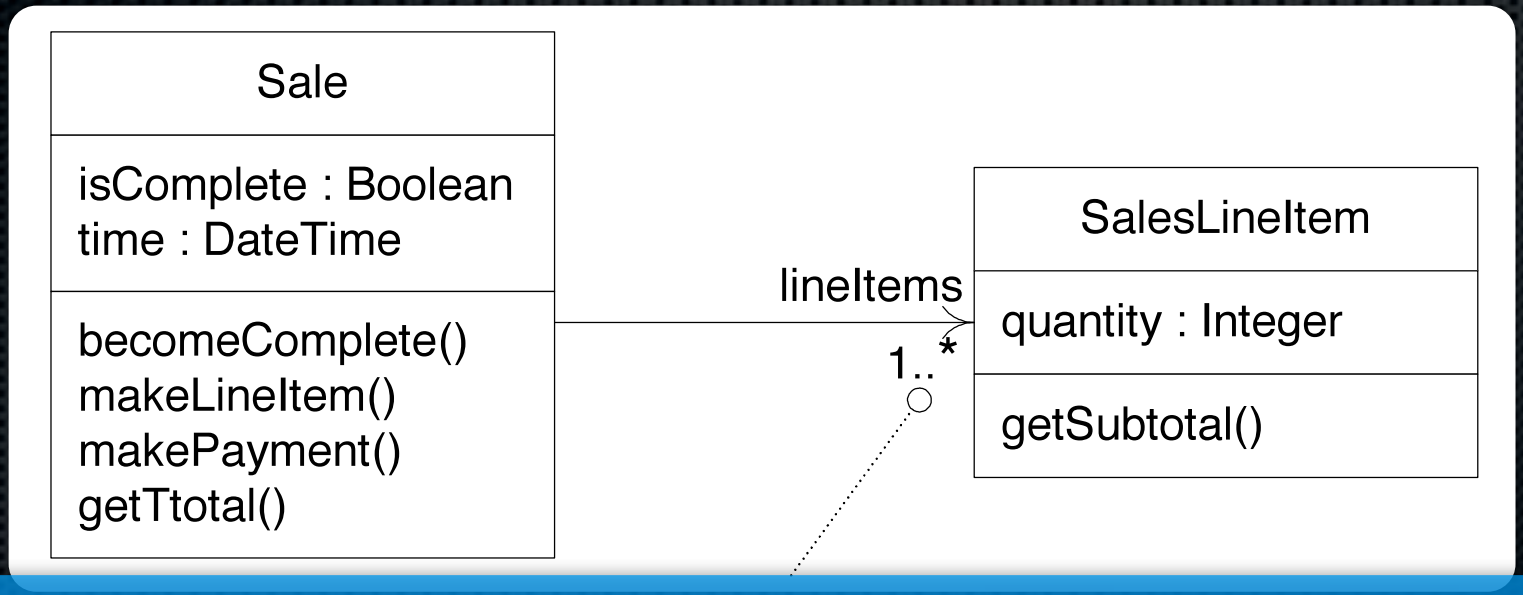
Create Methods from Interaction Diagrams



Implementation of *enterItem(ItemID id, int qty)*



Collections

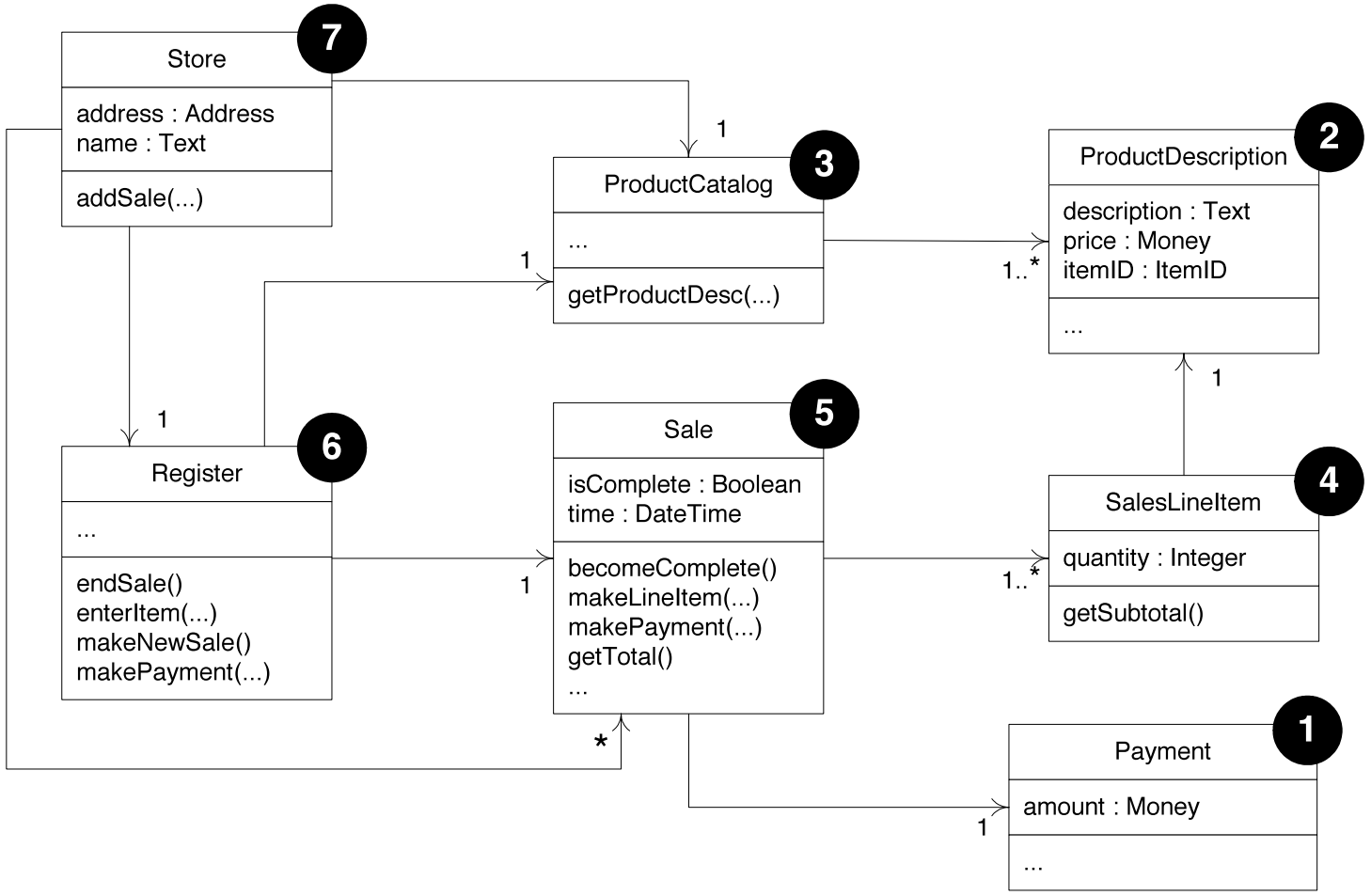



```
public class Sale {
    …
    private List<SalesLineItem> lineItems = new ArrayList<SalesLineItem>();
    …
}
```

Guideline: If an object implements an interface, use the interface type for the variable.

What Order?

Typically, least coupled to most coupled. Why?



Q8,9