Designing for Visibility & Mapping to Code

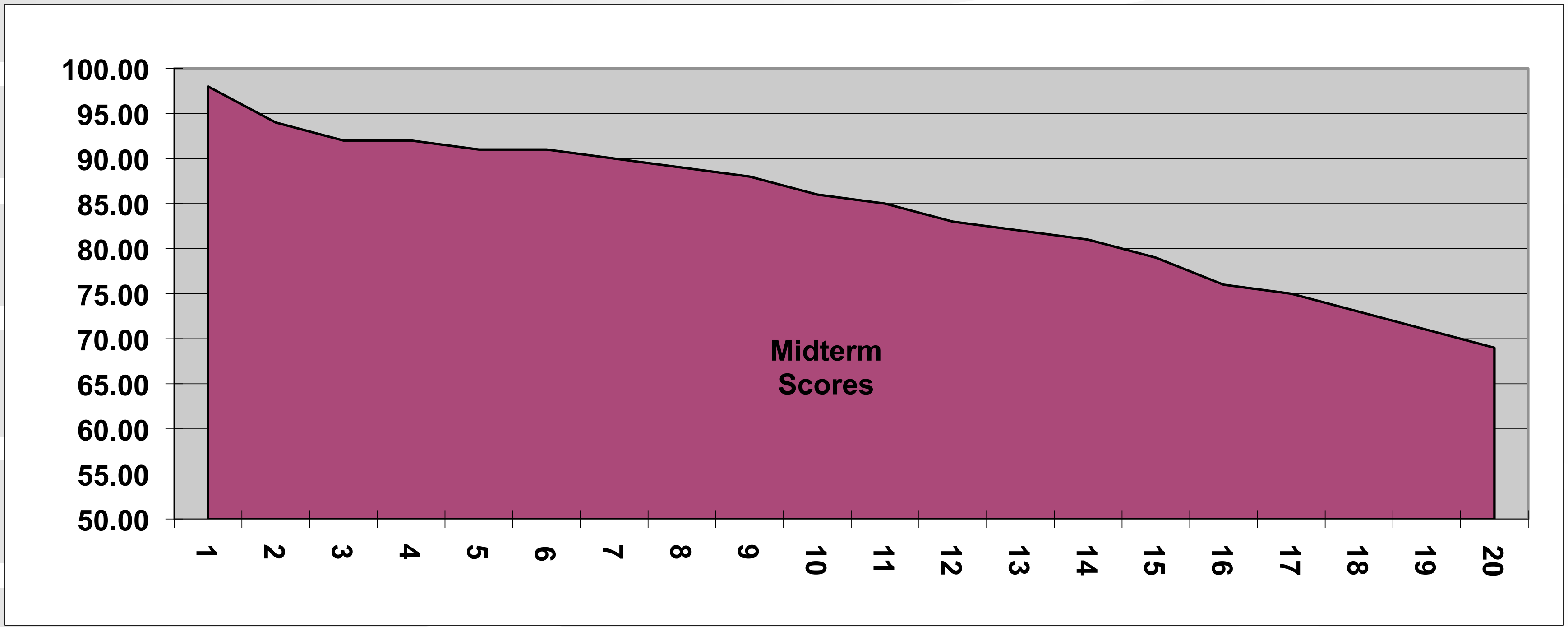
Shawn Bohner
Office: Moench Room F212
Phone: (812) 877-8685
Email: bohner@rose-hulman.edu

Q1

Agenda

- Exam Results
- Designing for Visibility
- Mapping Designs to Code

Examination #1 Results



Average Score	84.25%
Median Score	85.50%
Lowest Score	69.00%
Highest Score	98.00%

Exam 1 Stats (Comparative only – course grades will be determined from composite number grades)

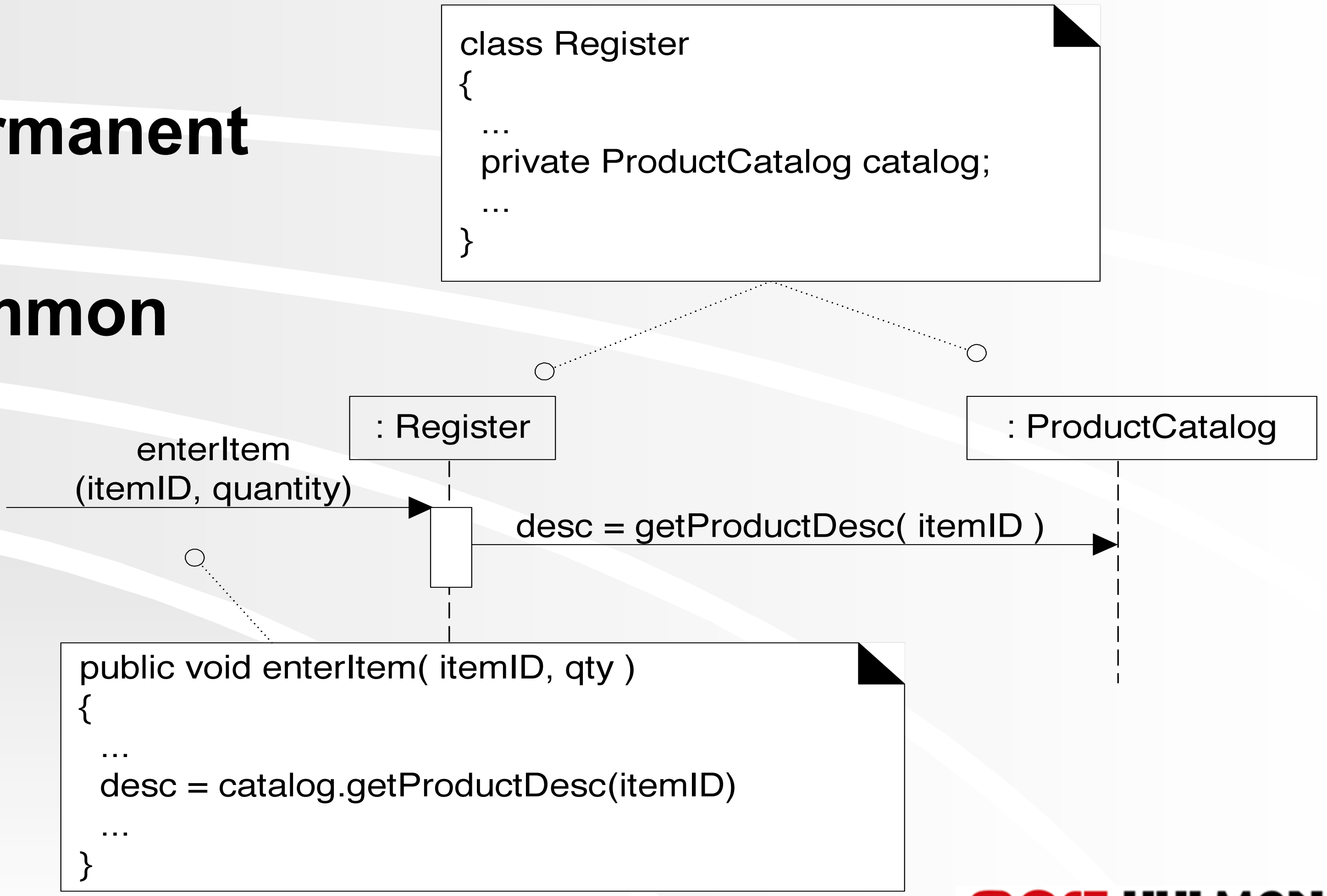
Cutoffs	Grade	# of Grade
90.0%	A	7
85.0%	B+	4
80.0%	B	3
75.0%	C+	3
70.0%	C	2
65.0%	D+	1
60.0%	D	0
0.0%	F	0

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



```
class Register
{
  …
  private ProductCatalog catalog;
  …
}
```

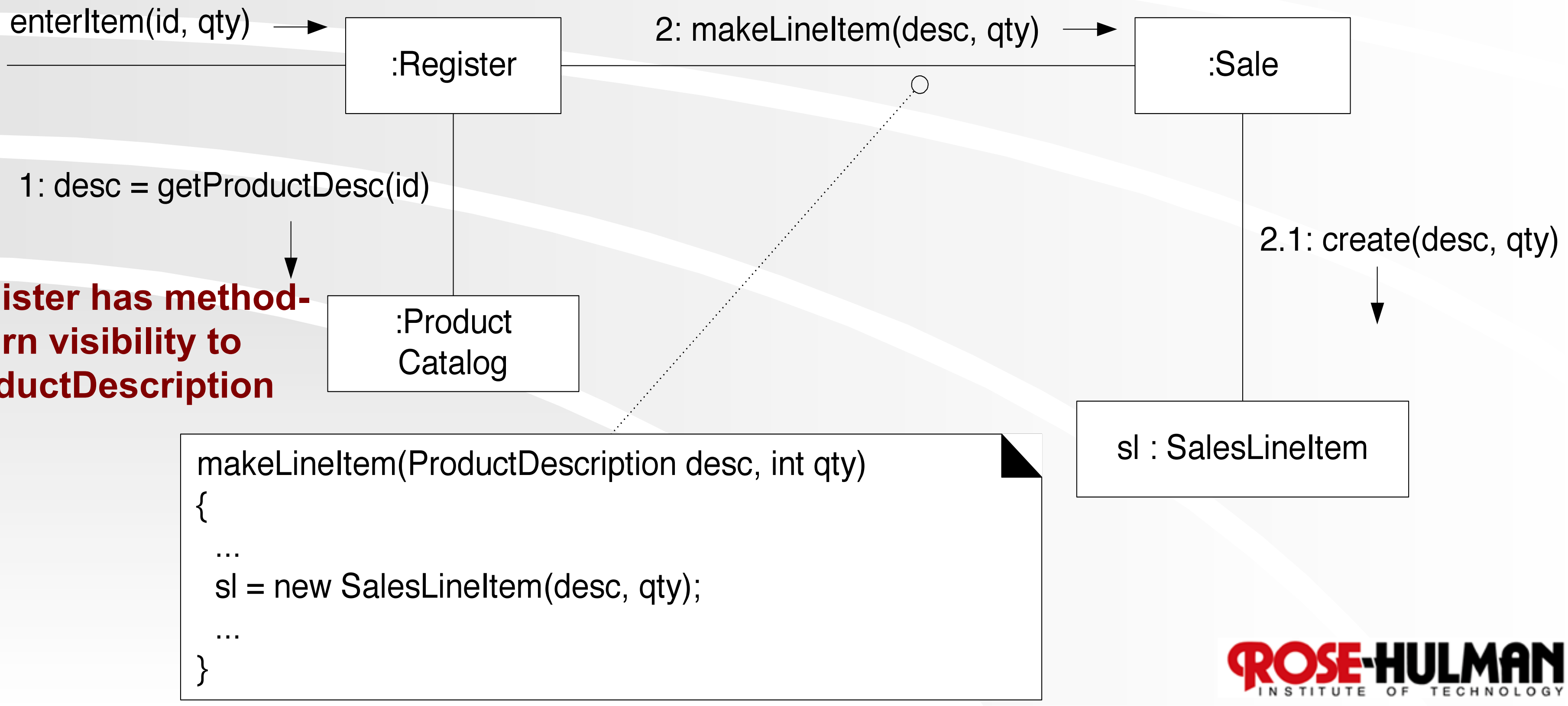
```
public void enterItem( itemID, qty )
{
  …
  desc = catalog.getProductDesc(itemID)
  …
}
```

Q2

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**



Register has method-return visibility to ProductDescription

```
makeLineItem(ProductDescription desc, int qty)
{
  ...
  sl = new SalesLineItem(desc, qty);
  ...
}
```

Q3

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

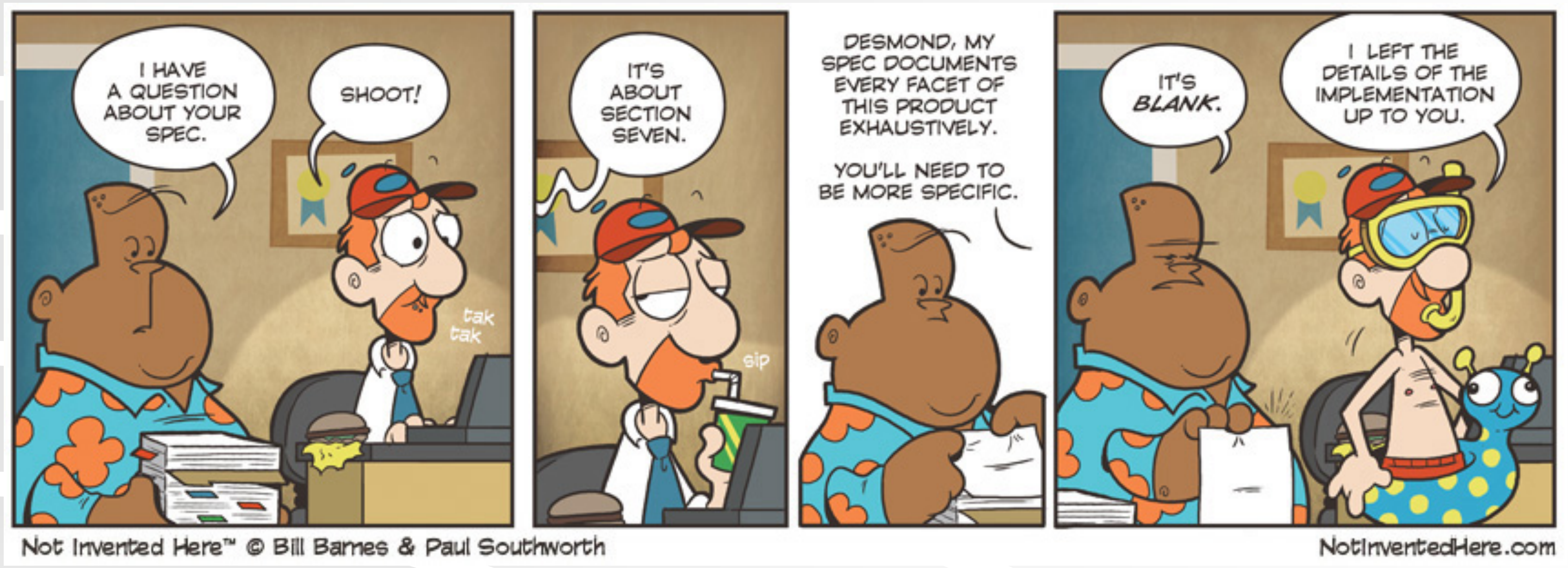
Q4

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)

Q5

Cartoon of the Day


I HAVE A QUESTION ABOUT YOUR SPEC.
SHOOT!
tak tak
IT'S ABOUT SECTION SEVEN.
sip
DESMOND, MY SPEC DOCUMENTS EVERY FACET OF THIS PRODUCT EXHAUSTIVELY.
YOU'LL NEED TO BE MORE SPECIFIC.
IT'S *BLANK*.
I LEFT THE DETAILS OF THE IMPLEMENTATION UP TO YOU.
Not Invented Here™ © Bill Barnes & Paul Southworth
NotInventedHere.com

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

Before we get into C

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

- Created Domain Mod 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

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

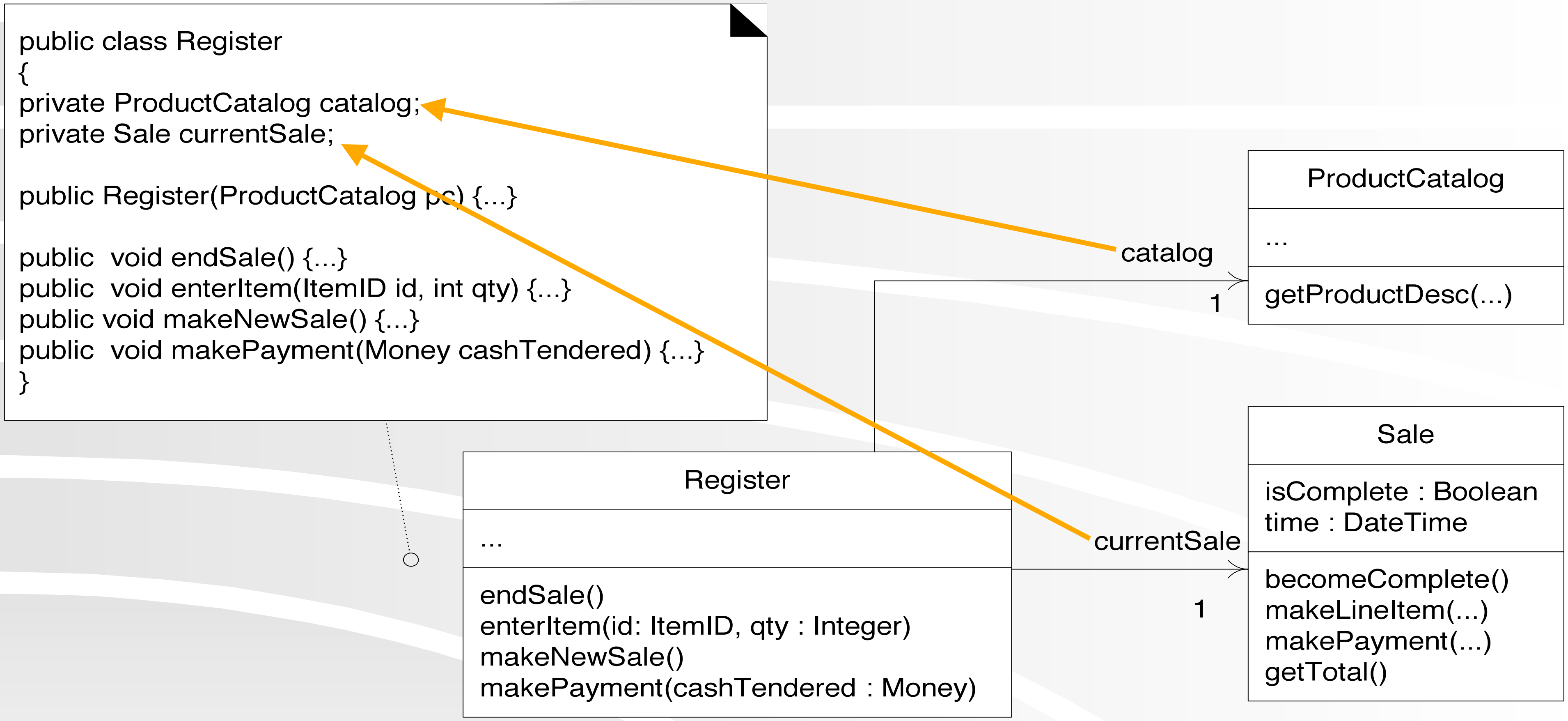
Moving from Design to Code

- Design provides starting point for Coding
  - DCDs contain class or interface names, superclasses, method signatures, and simple attributes

- Two primary tasks
  1. Define classes & interfaces
  2. Define methods

- Elaborate from associations to add reference attributes

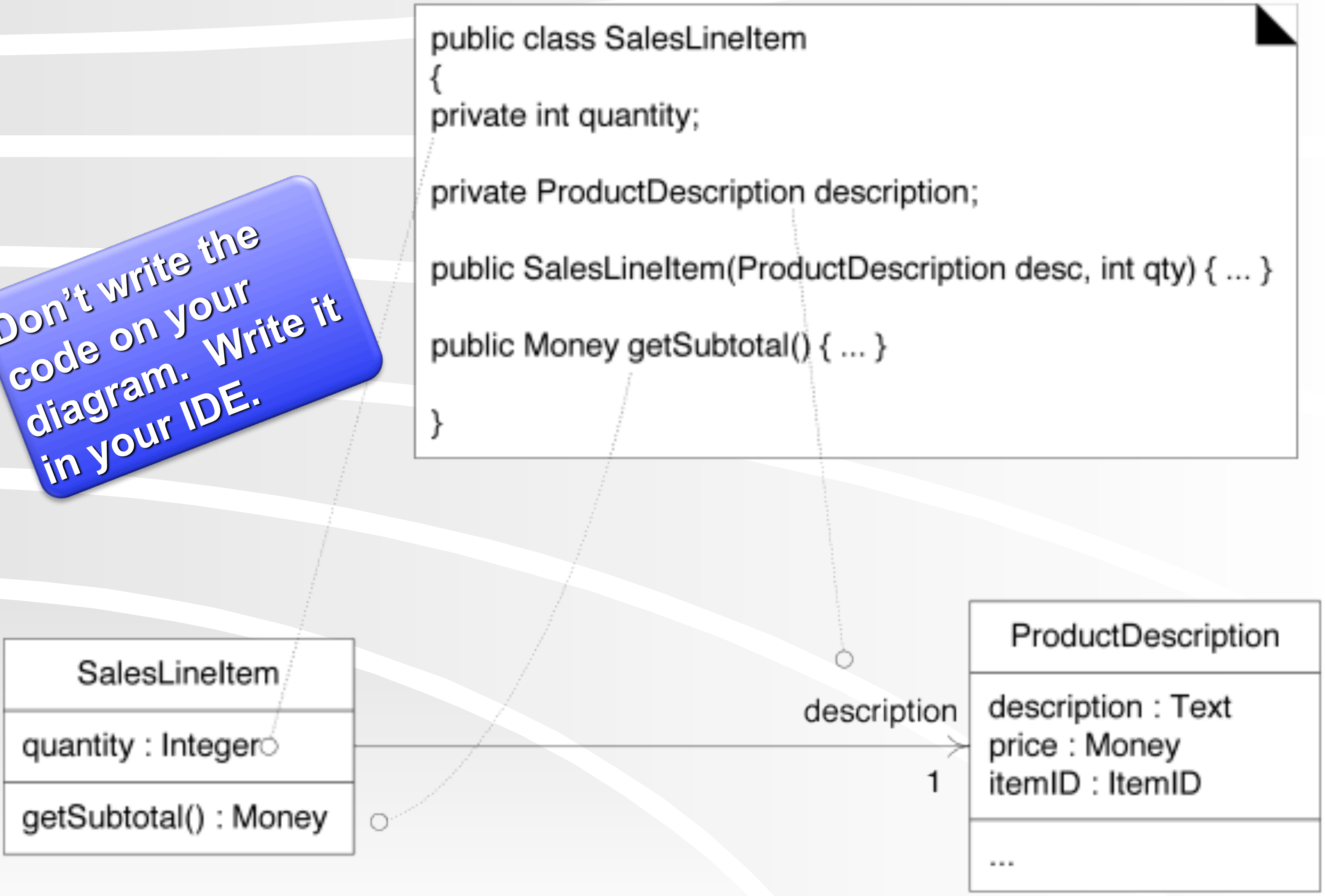
Example: Defining Register Class



```
public class Register
{
private ProductCatalog catalog;
private Sale currentSale;

public Register(ProductCatalog pc) {...}

public  void endSale() {...}
public  void enterItem(ItemID id, int qty) {...}
public void makeNewSale() {...}
public  void makePayment(Money cashTendered) {...}
}
```

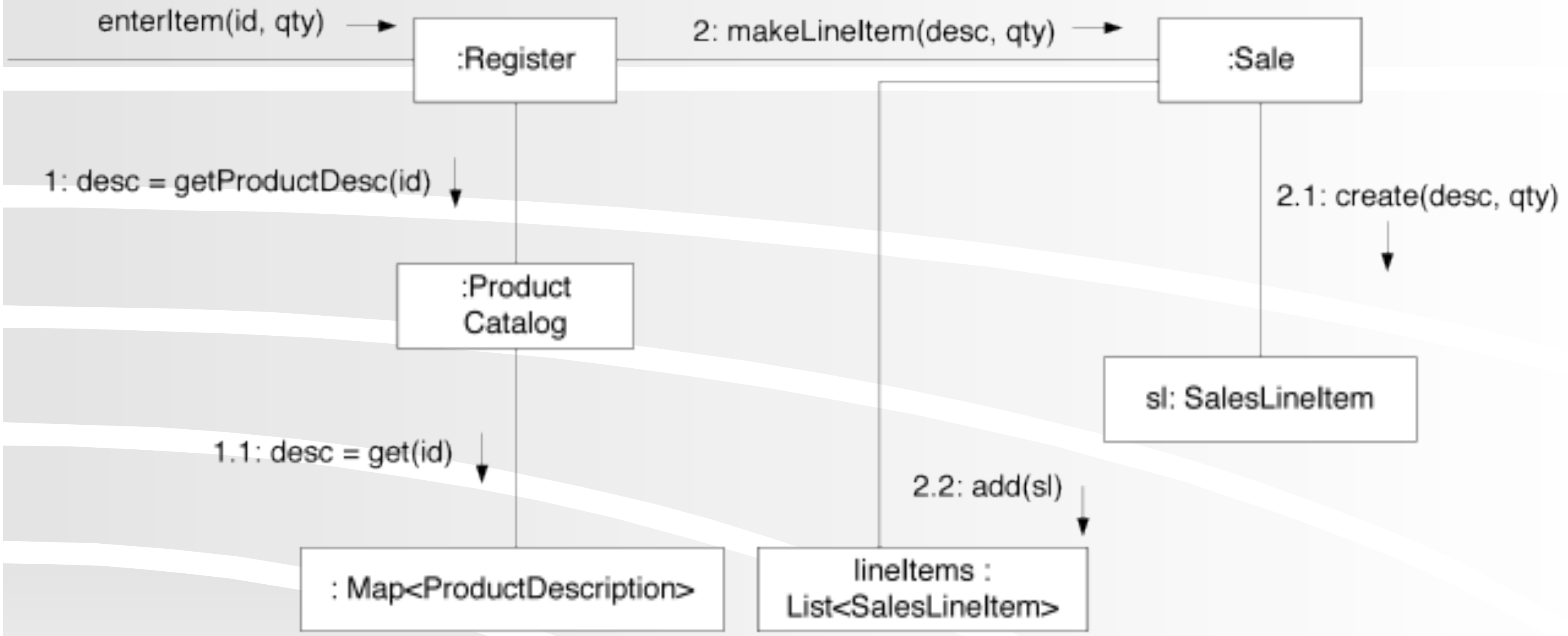
Create Class Definitions from DCDs

Don't write the code on your diagram. Write it in your IDE.

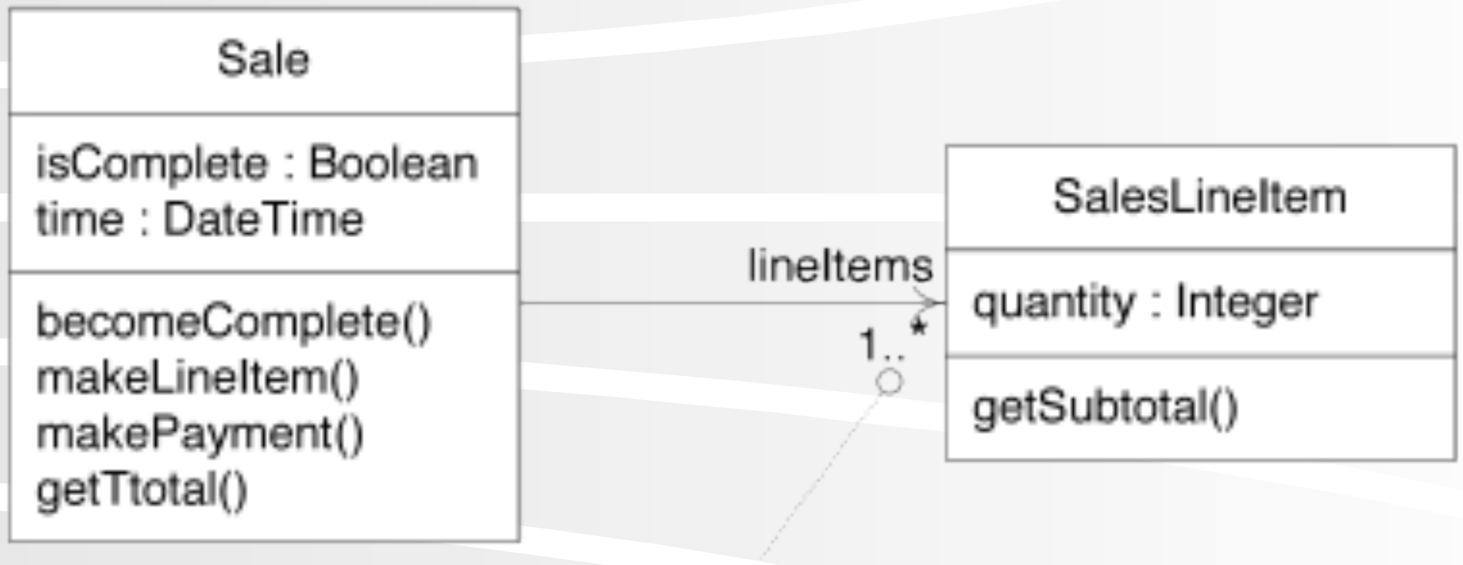


```
public class SalesLineItem
{
private int quantity;

private ProductDescription description;

public SalesLineItem(ProductDescription desc, int qty) { ... }

public Money getSubtotal() { ... }

}
```

Create Methods from Interaction Diagrams



Collections

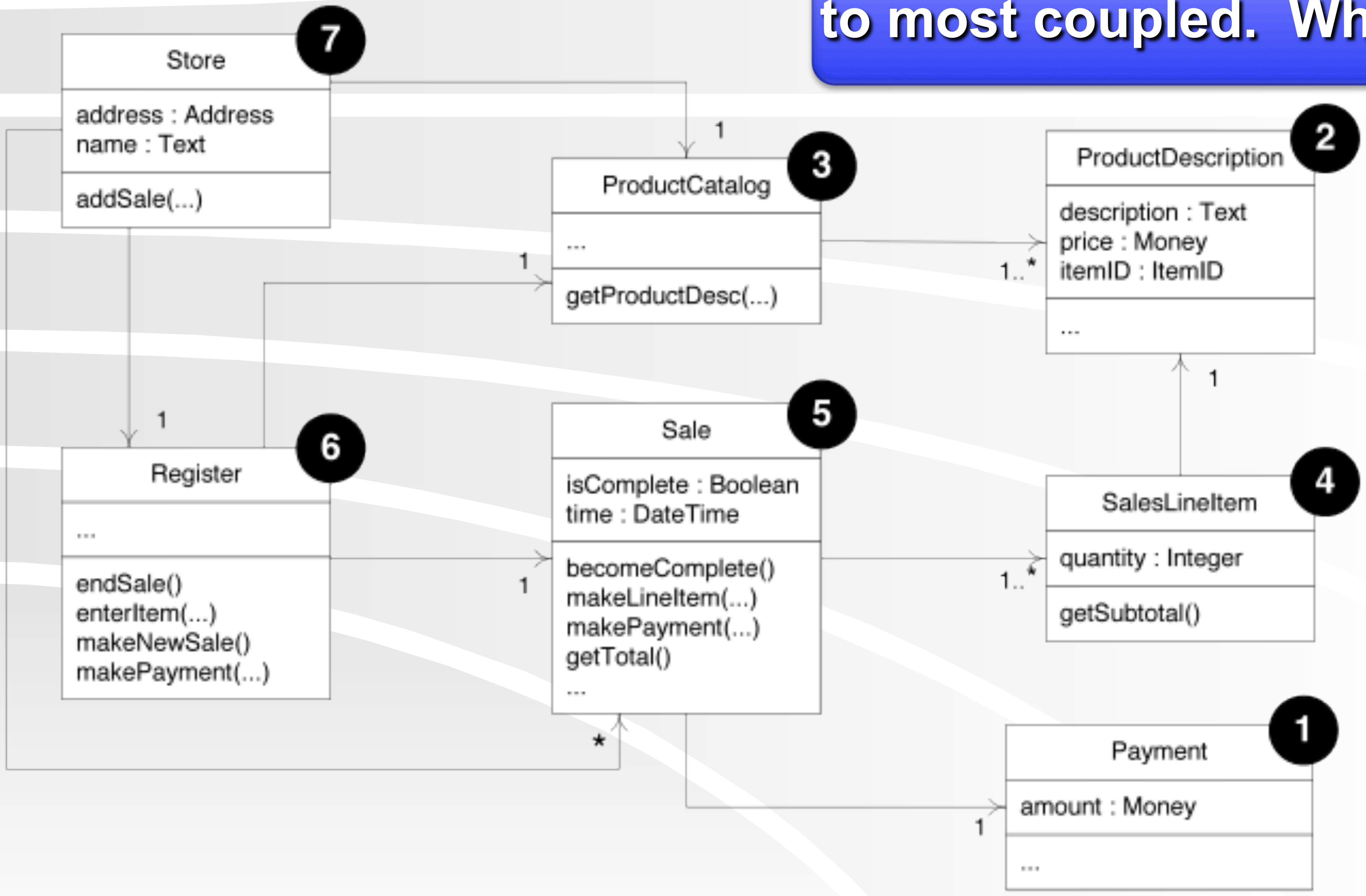


```
public class Sale {
          …
  private List<SalesLineItem> lineItems = new ArrayList<SalesLineItem>();
          …
}
```

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

Homework and Milestone Reminders

- Read Chapters 21, 23, and 24
- Homework 6 – More GRASP on Video Store Design
  - Due by 5:00pm Tuesday, January 26th, 2010
- Milestone 4: Patterns and Detailed Design, with some Iteration 2 on the Side
  - Due by 11:59pm Friday, January 29th, 2010