SWEN30006
Software Modelling and Design

WORKSHOP 3 (WEEK 4)

Beware of bugs in the above code;
I have only proved it correct, not tried it.

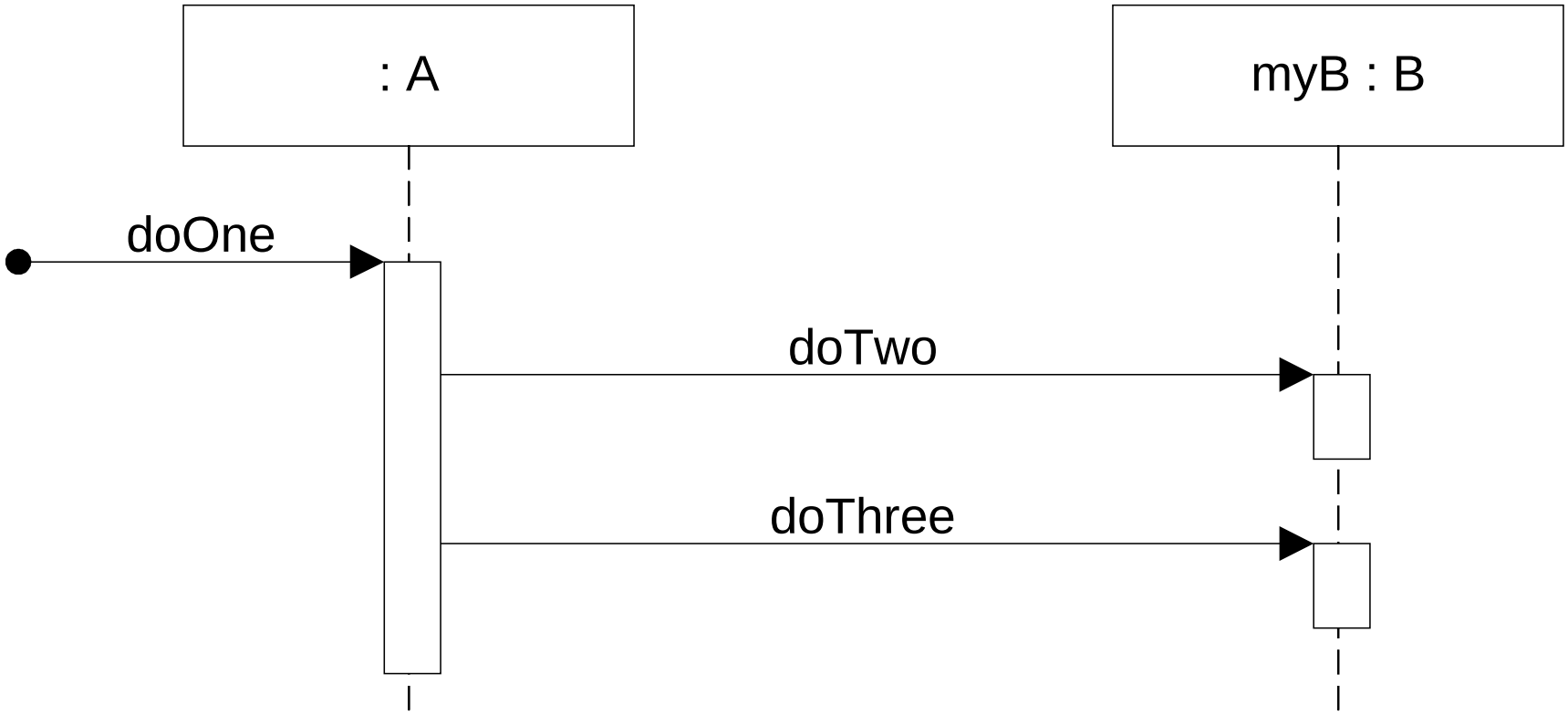
—Donald Knuth

SWEN30006
Software Modelling and Design

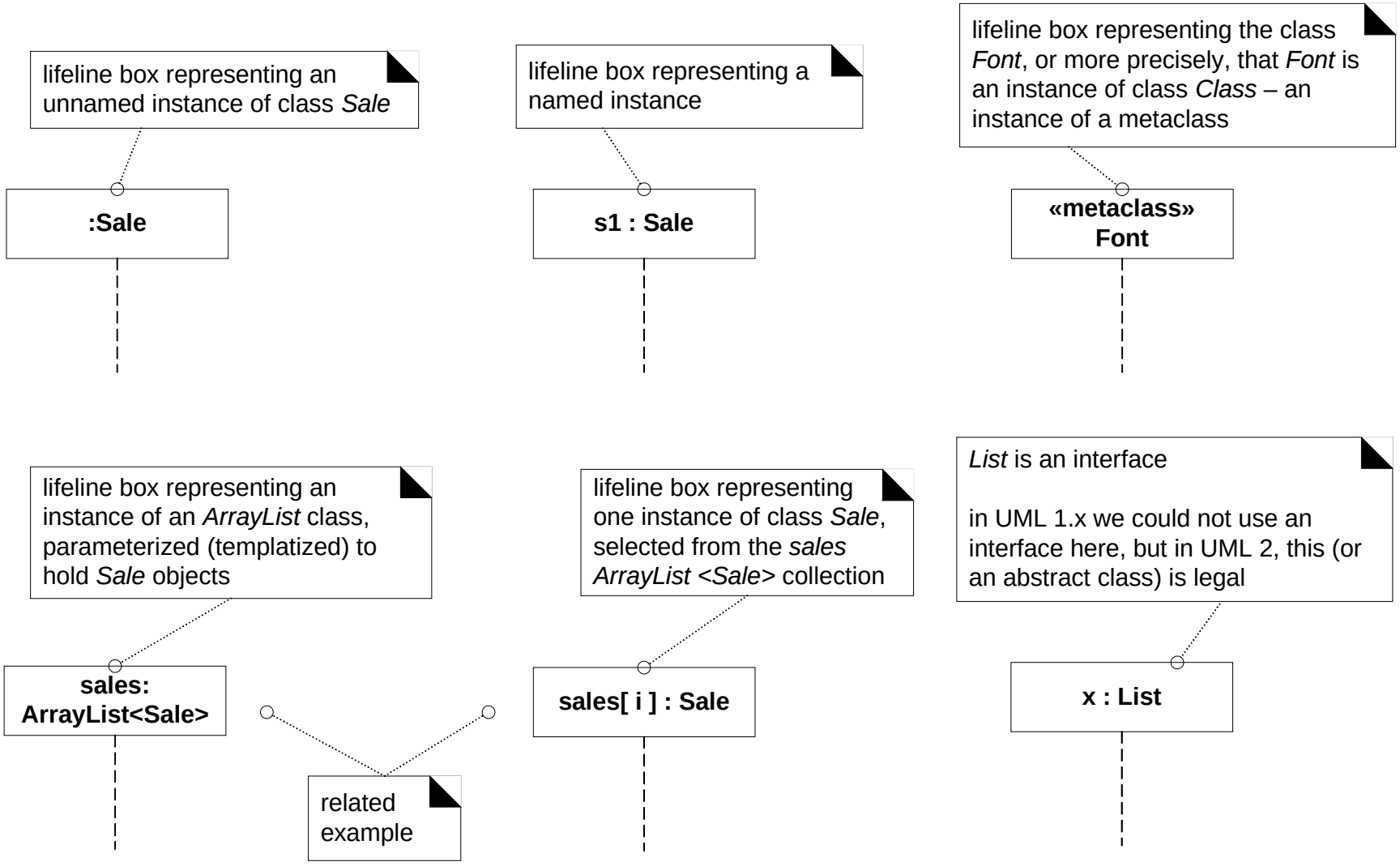
UML Sequence Diagrams

Larman Chapter 15

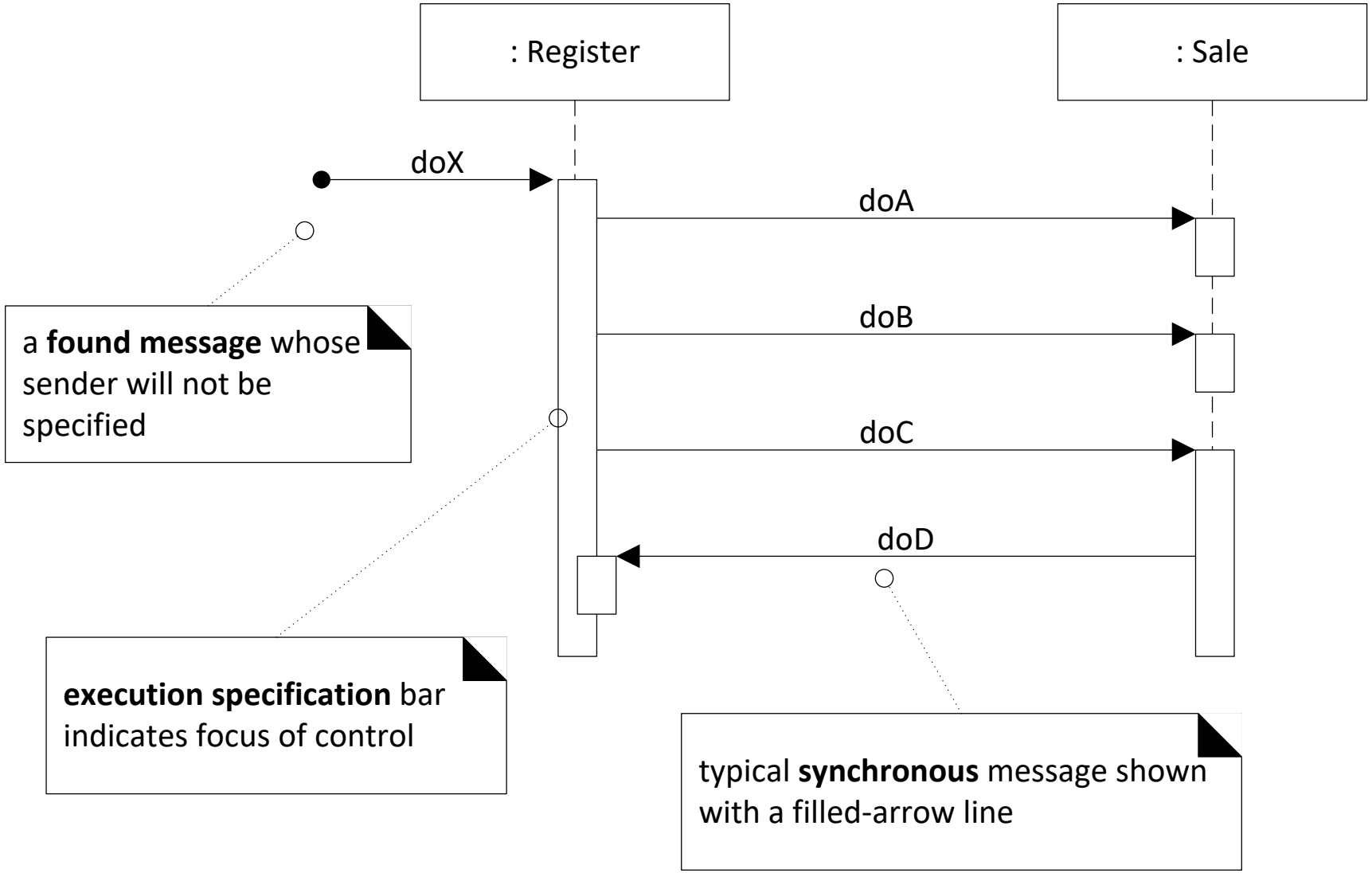
Sequence Diagram



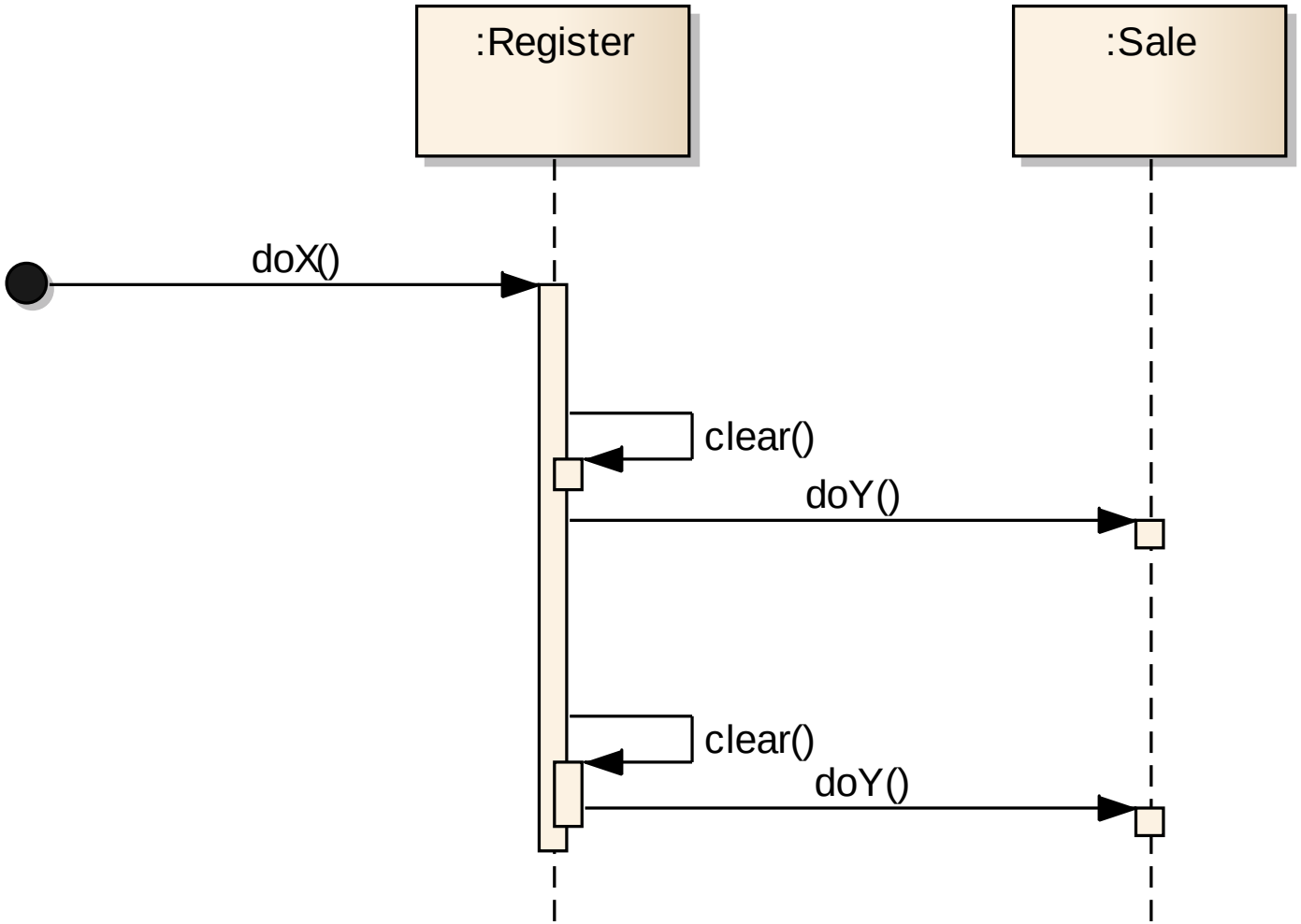
Lifelines: Different Participants



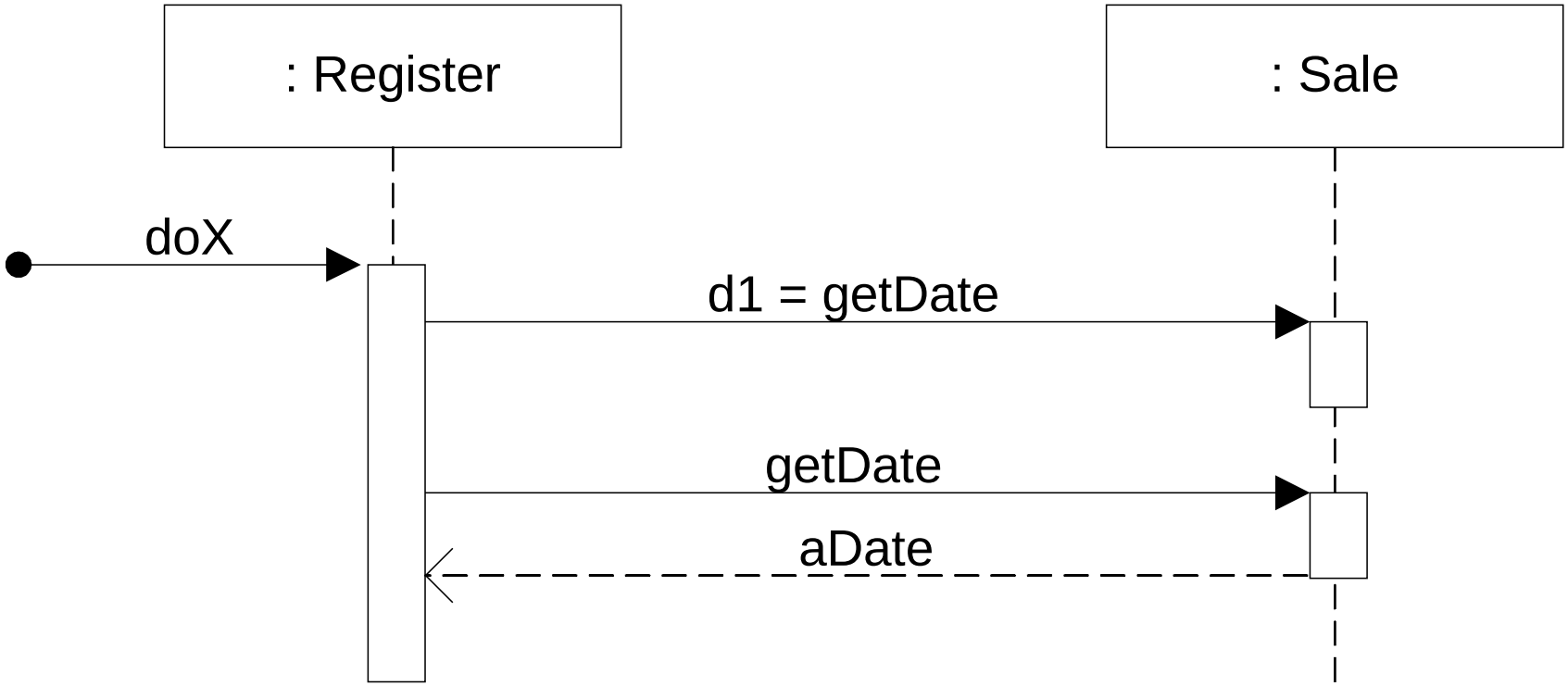
Messages and the Exec. Spec. Bar



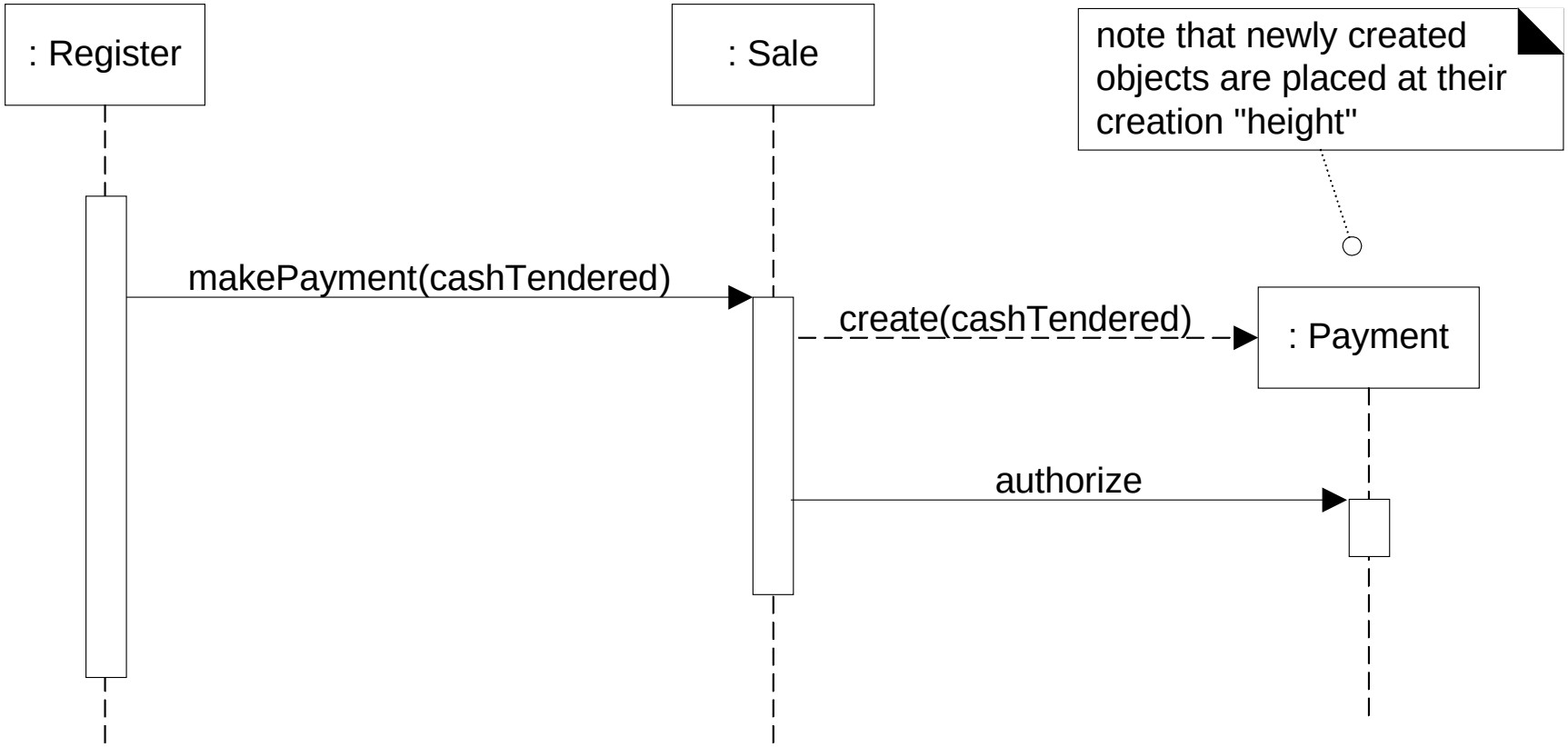
Messages to Self (*this*)



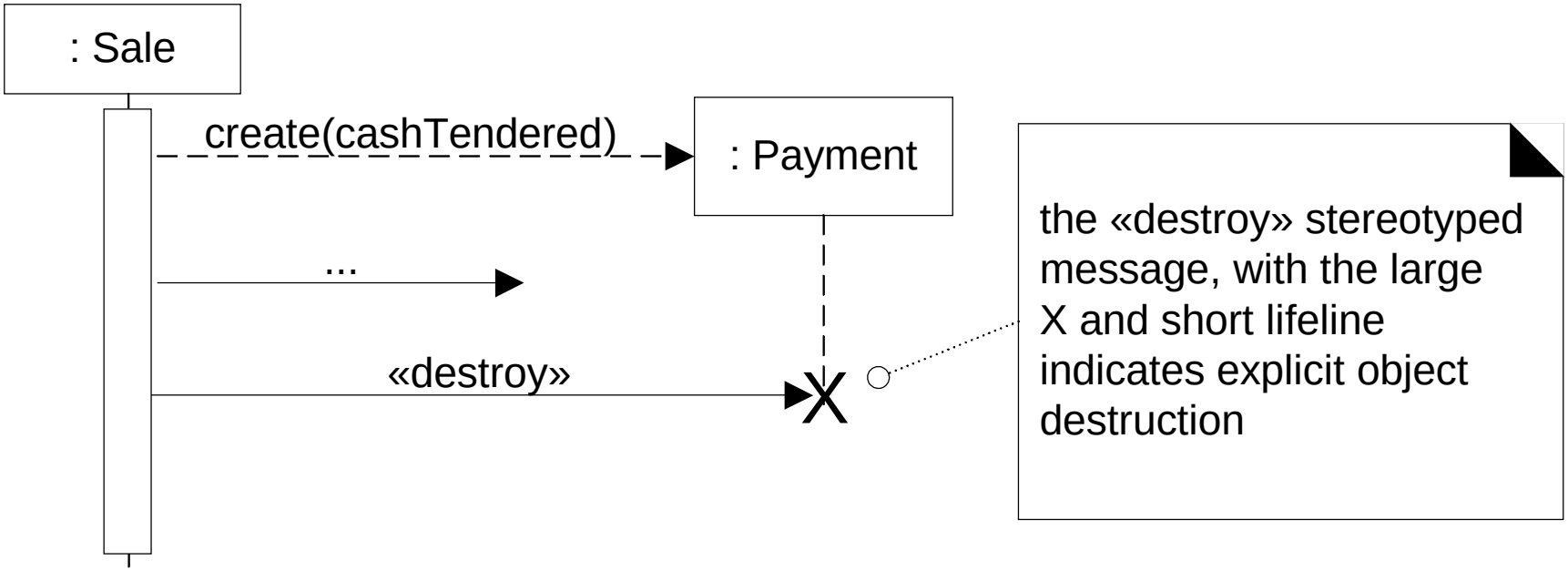
Showing Return Results



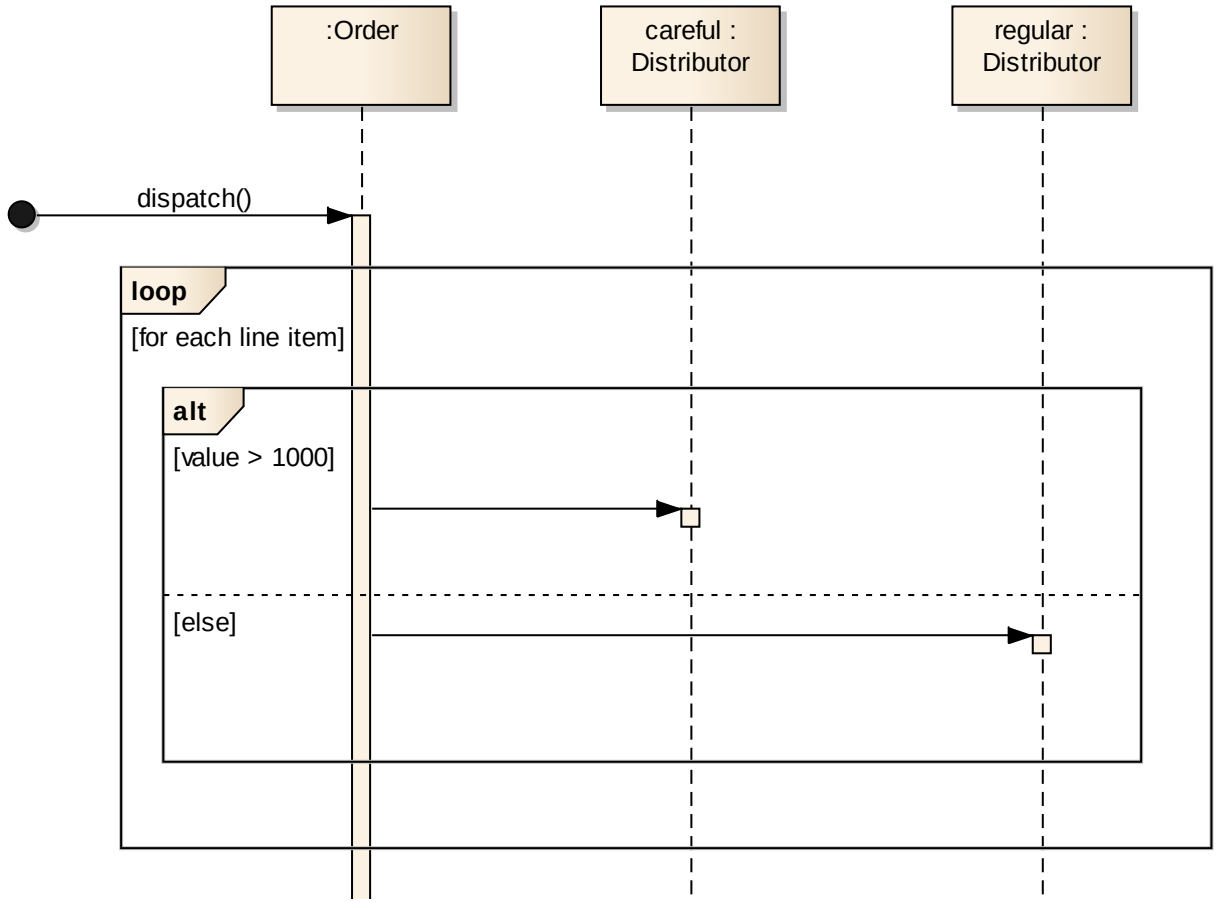
Object Creation and Lifelines



Object Destruction



UML Frames



- loop

  (loop) [condition or items]

- if/then

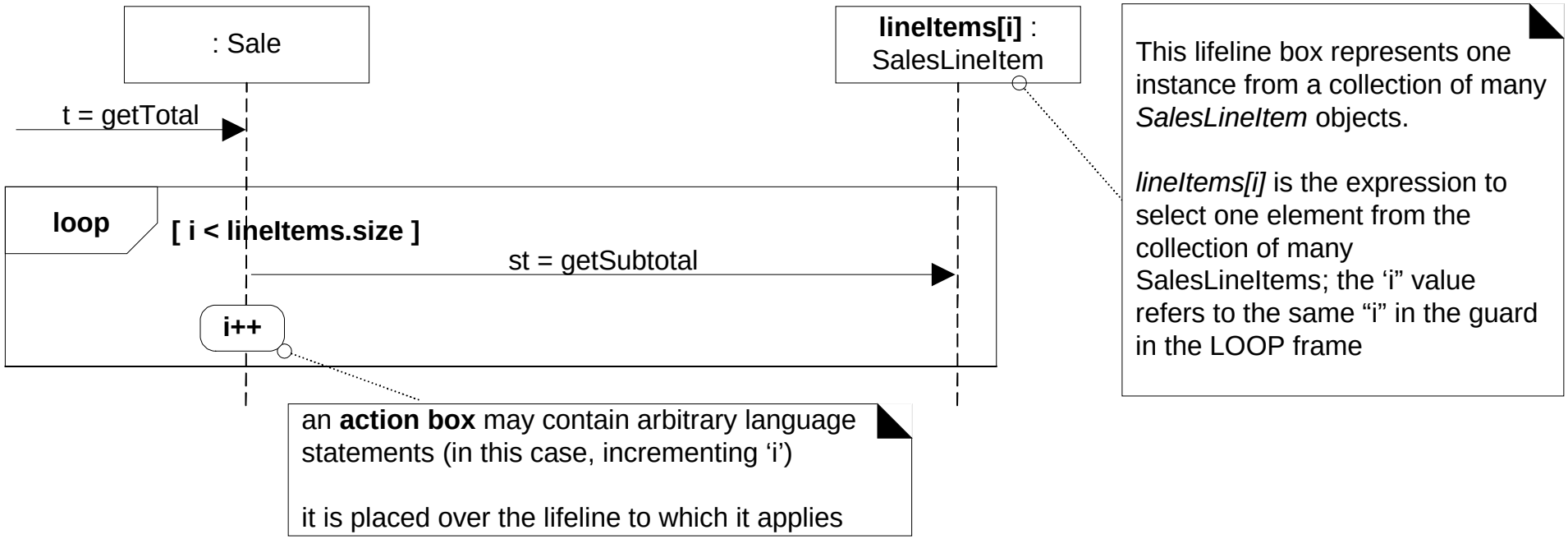
  (opt) [condition]

- if/then/else

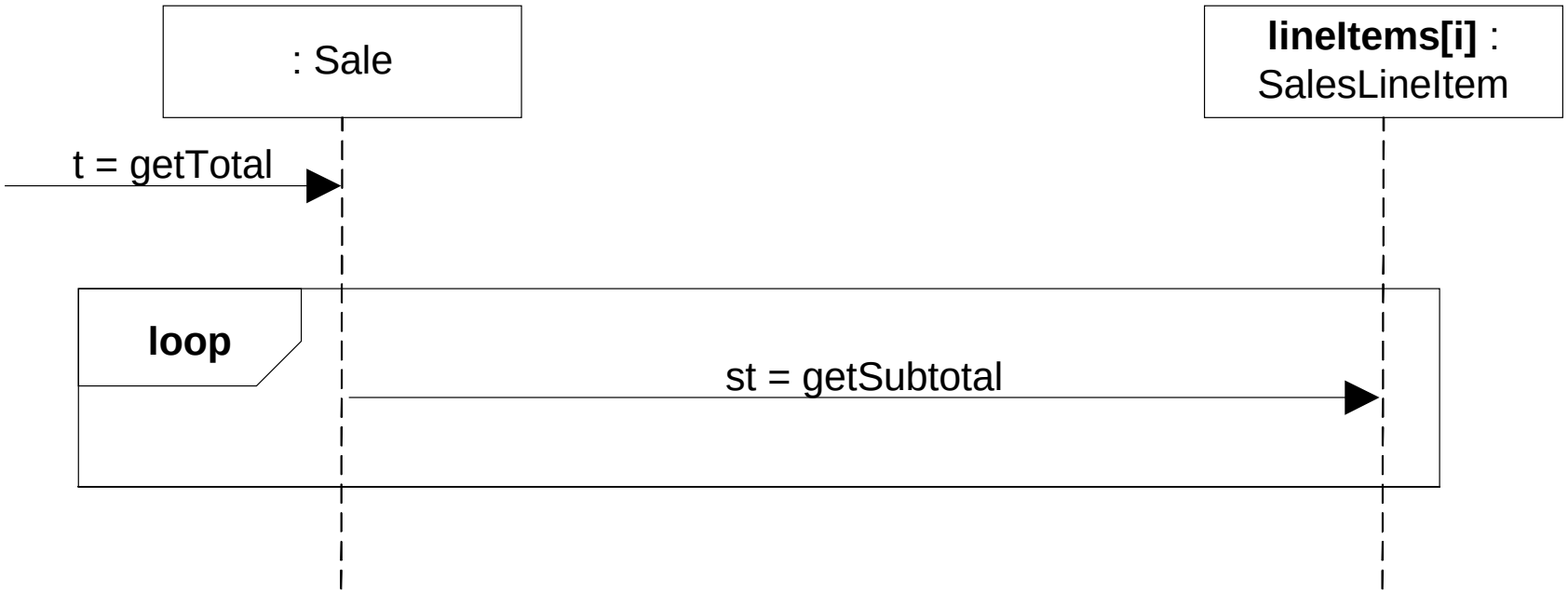
  *(alt) [condition or **else**]*

  with dashed separator

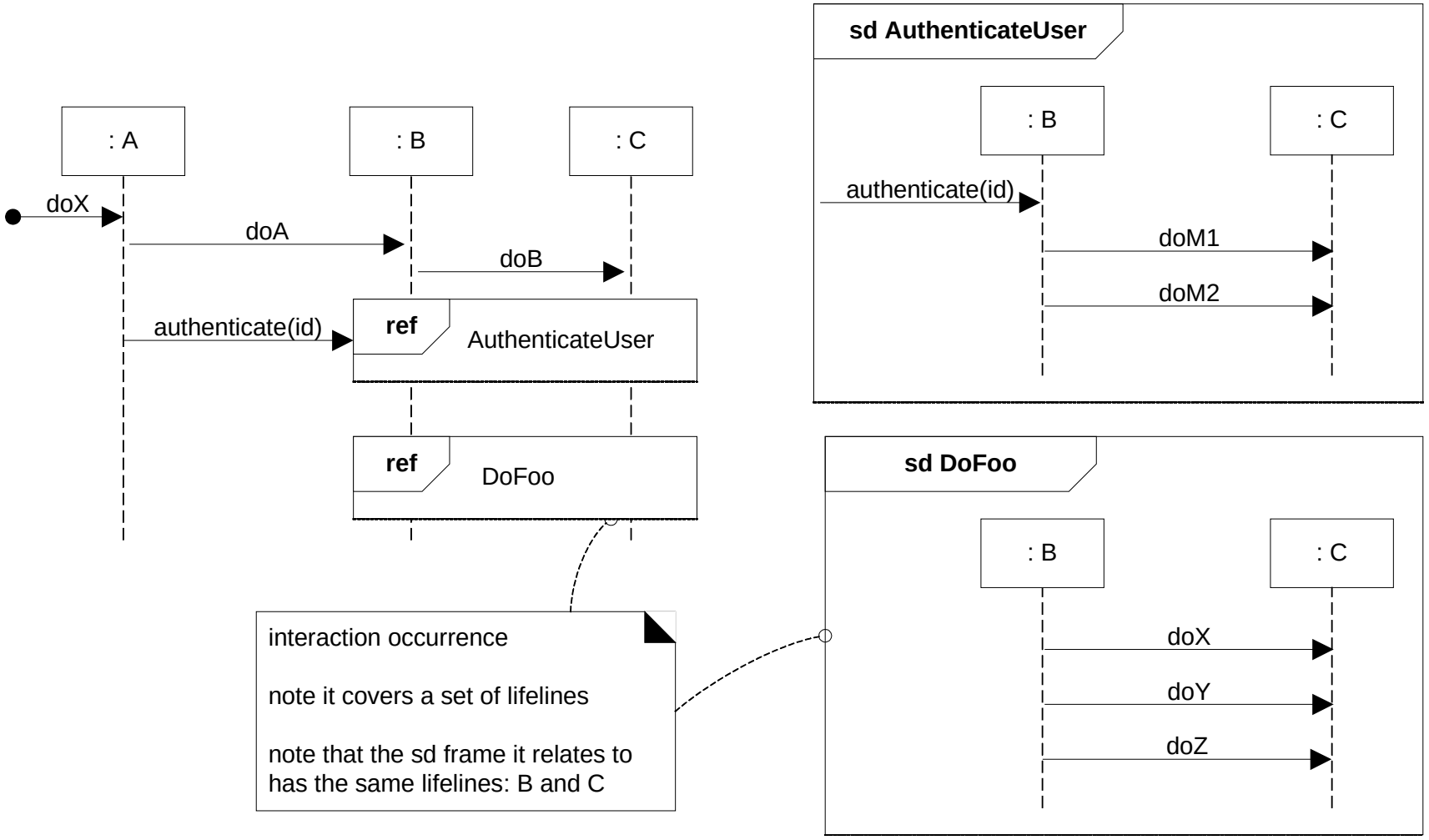
UML Frames: loop–Collections (1)



UML Frames: loop–Collections (2)



UML Frames: sd/ref (define/refer)



SWEN30006
Software Modelling and Design

DESIGNING FOR VISIBILITY

Larman Chapter 19

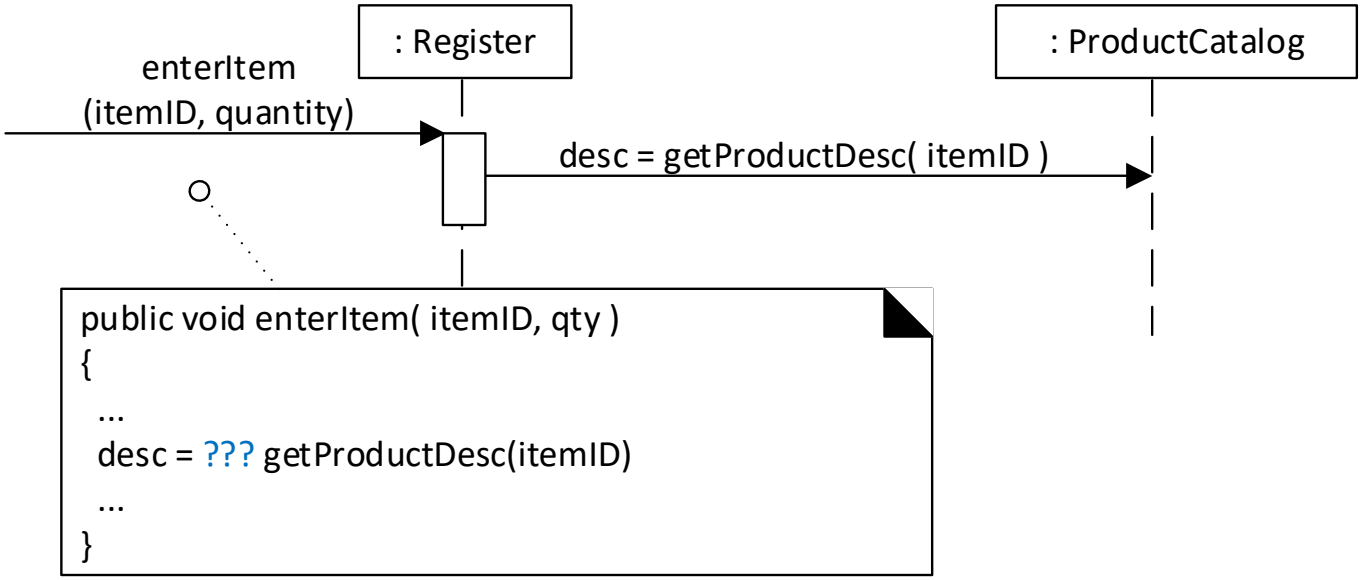
Objectives

On completion of this topic you should be able to:

- Identify four types of visibility.
- Design to establish visibility where required.
- Recognise that different forms of visibility/usage represent different levels of coupling

Visibility from Register to ProductCatalog

How does Register gain visibility of Product catalog?



Objects require visibility of each other to cooperate

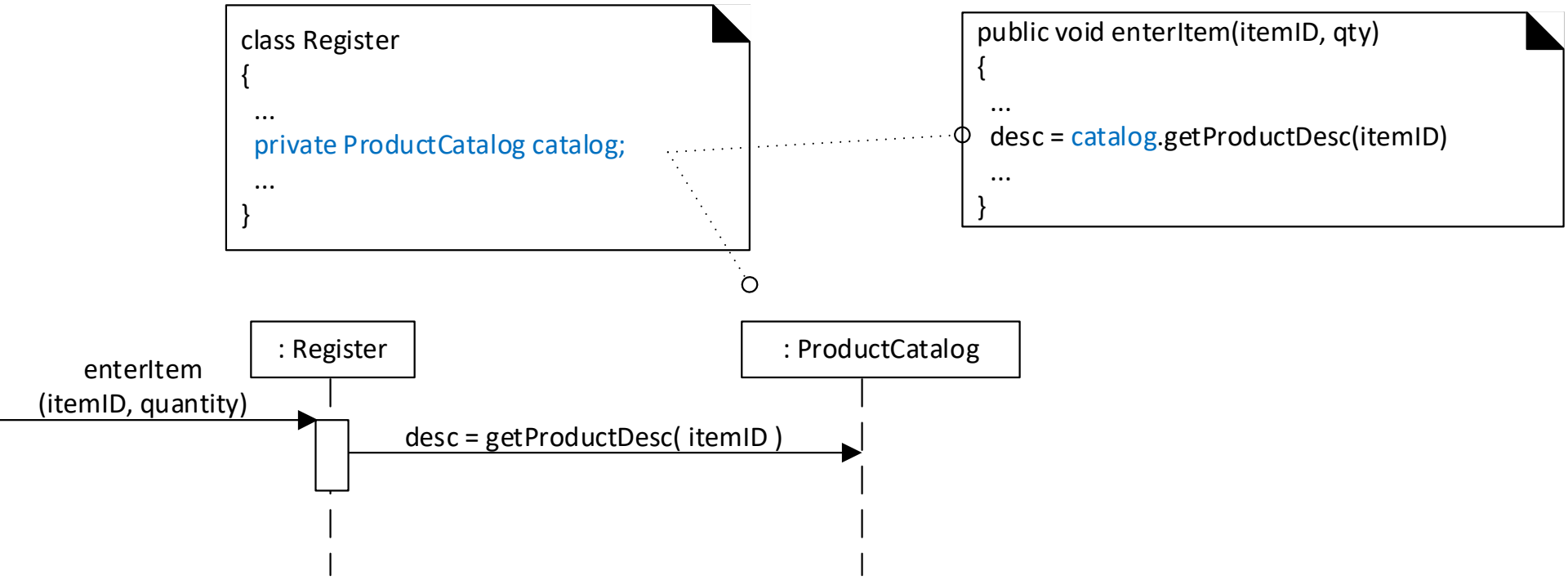
What is Visibility?

- Ability of an object to “see” or refer to another object
- For A to send a message to B, B must be visible to A

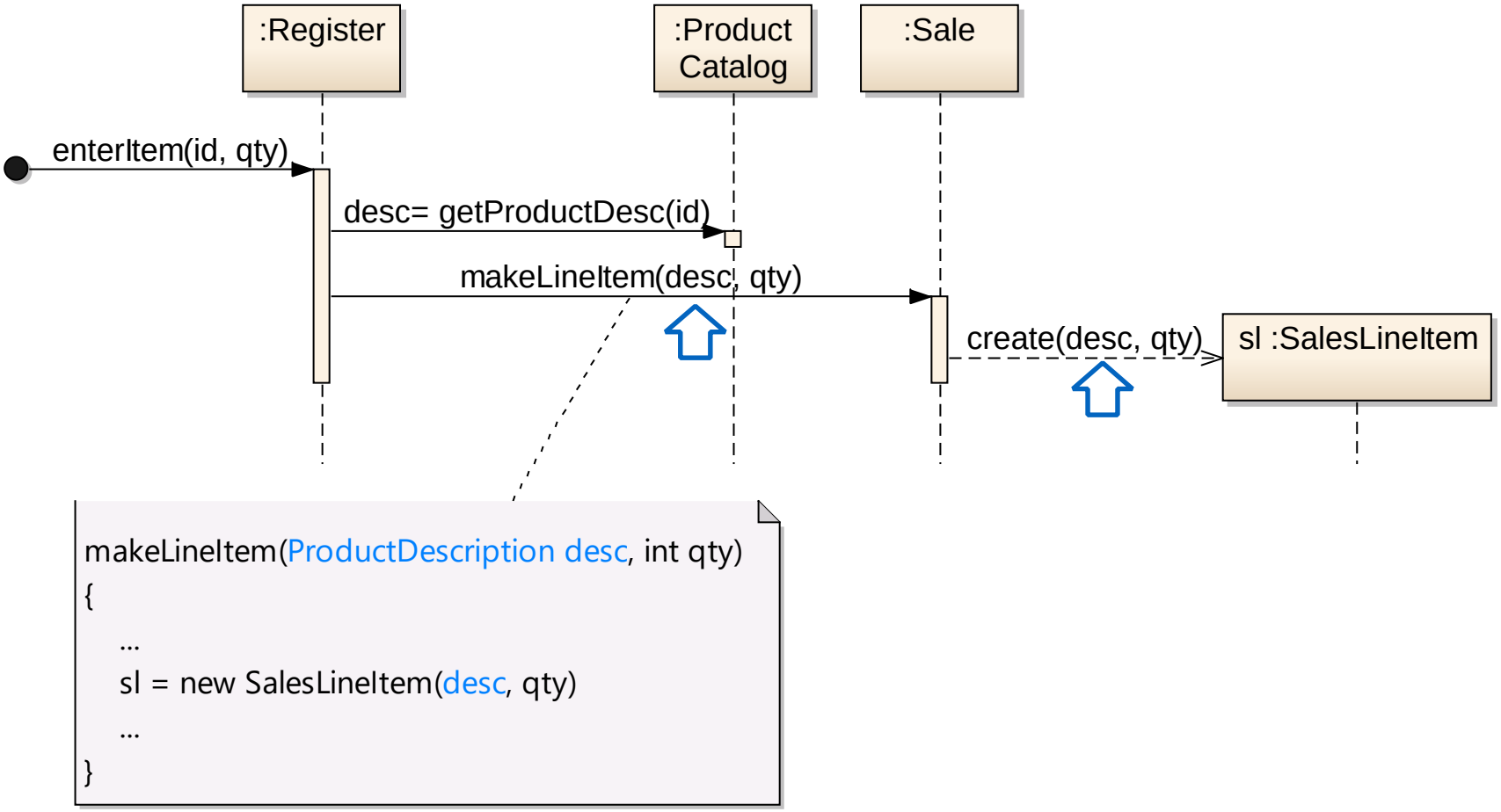
Is it in scope? Four common ways:

1. B is an *attribute* of A.
2. B is a *parameter* of a method of A.
3. B is a (non-parameter) *local* object in a method of A.
4. B has in some way *global* visibility.

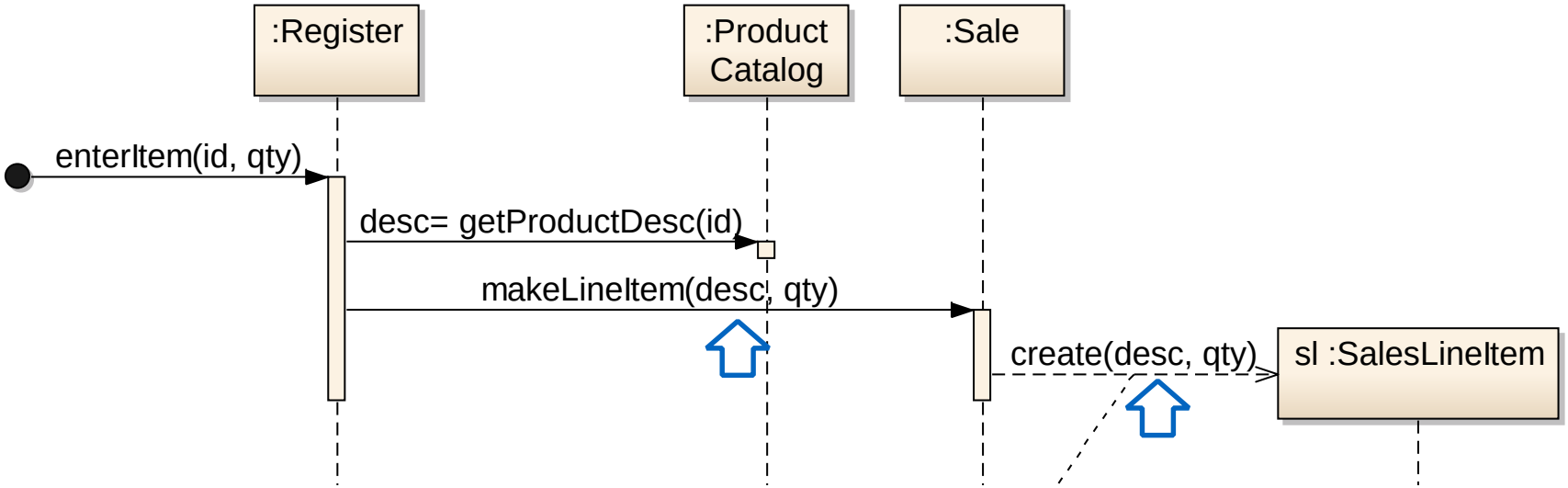
1. Attribute Visibility



2. Parameter Visibility

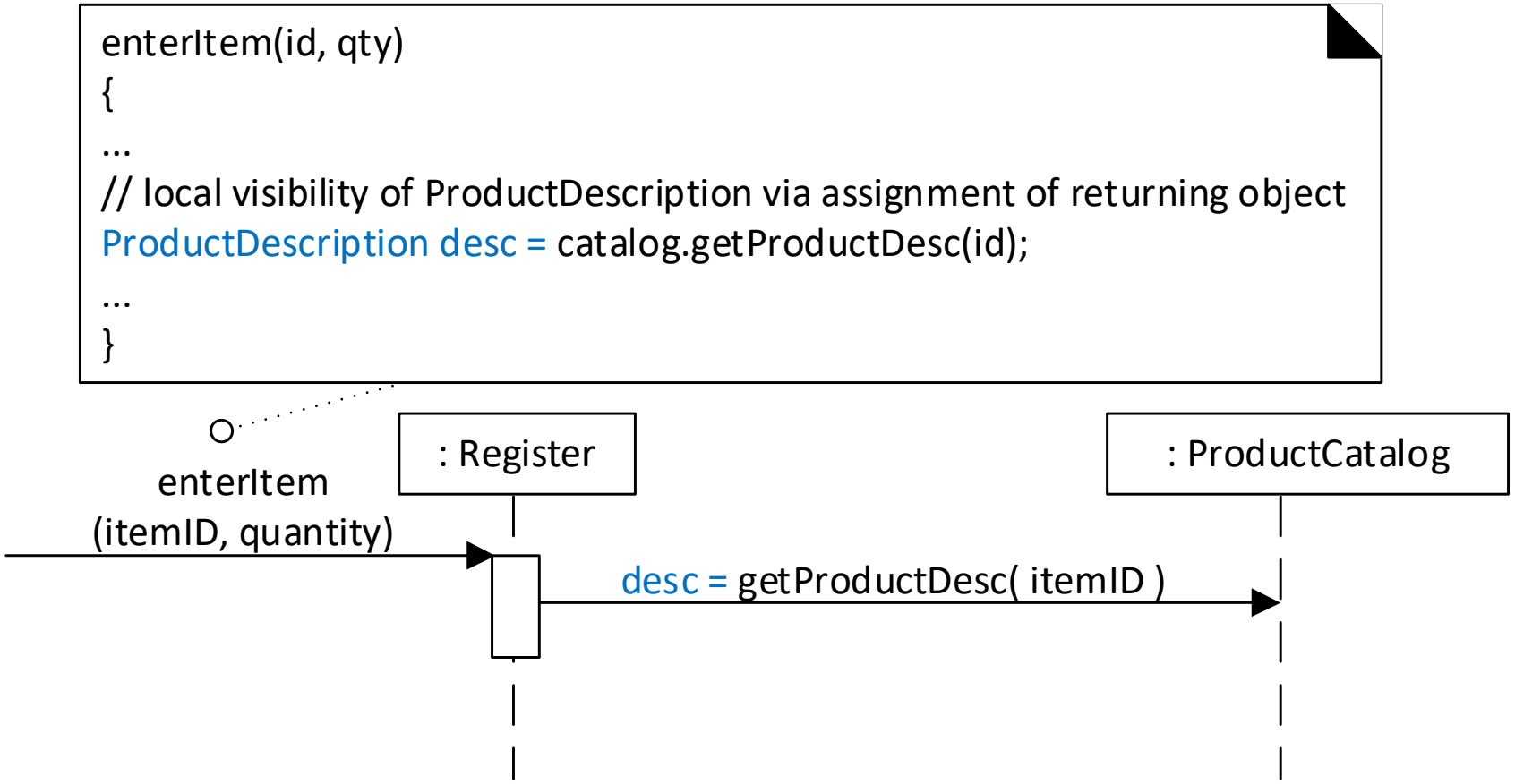


Parameter to Attribute Visibility




```
// initializing method (e.g., a Java constructor)
SalesLineItem(ProductDescription desc, int qty)
{
    ...
    description = desc; // parameter to attribute visibility
    ...
}
```

3. Local Visibility




```
// Cf. implicit local Catalog (not explicitly assigned)
ProductDescription desc = (other.getCatalog()).getProductDesc(id);
```

4. Global Visibility

	C++	Java

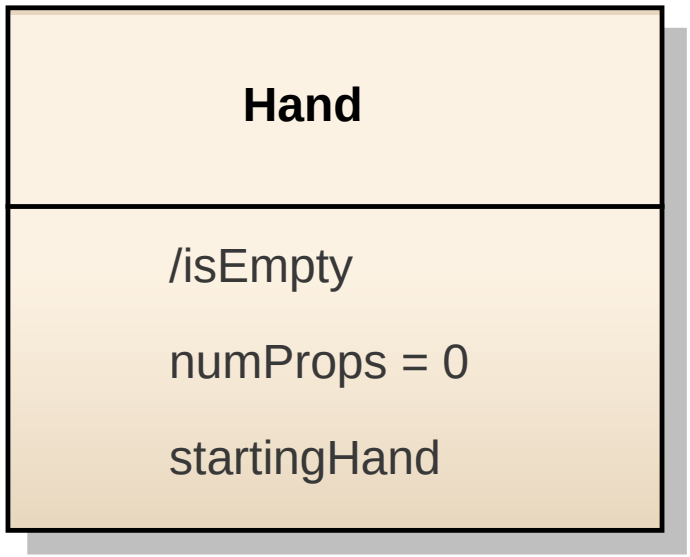
Declaration	int v = 1;	public class Global {
		  public static int v = 1;
		}

Usage	… main(…) {
	
	  int n1 = v;
	  int n2 = ::v;
	}
		… main(…) {
		  Global g = new Global();
		  int n1 = g.v;
		  int n2 = Global.v;
		}

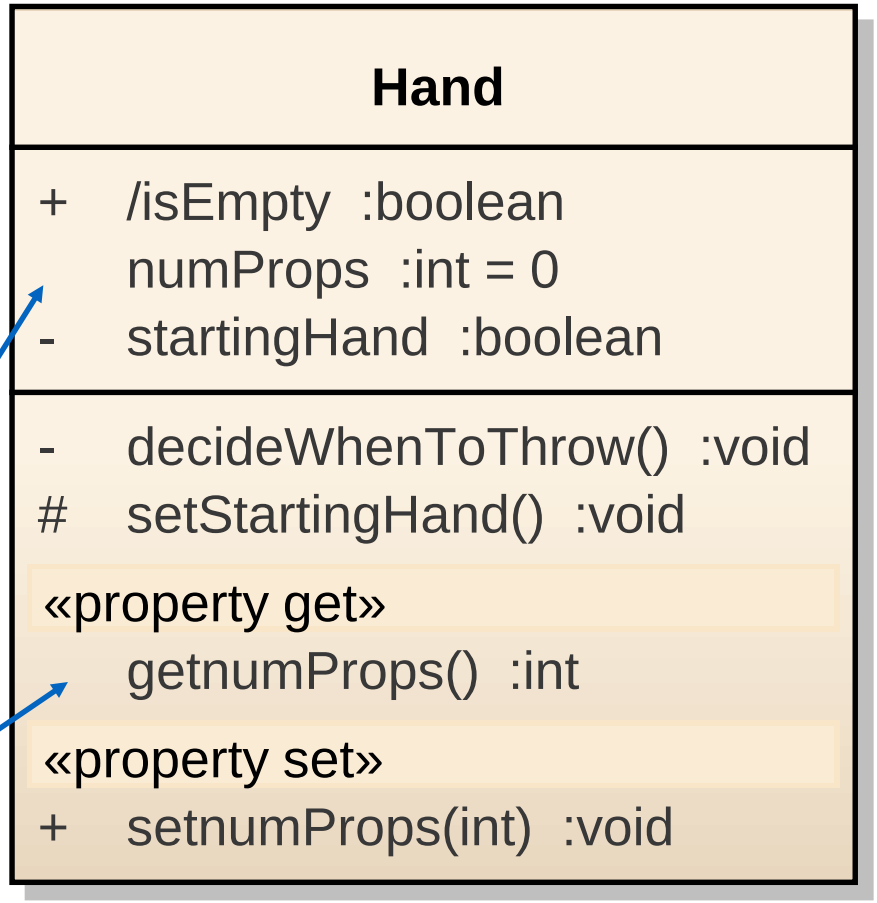
- Strictly, Java does not have global visibility
- Effect is achievable, but use Singleton (Ch. 26)

UML Visibility Marks

Domain (no visibility)



Design (visibility)



Convention:

Attr. default: private (-)

Method default: public (+)

SWEN30006
Software Modelling and Design

MAPPING DESIGNS TO CODE

Larman Chapter 20

Objectives

On completion of this topic you should be able to:

- Map design artefacts to code in an object-oriented language.

Mapping Designs to Code

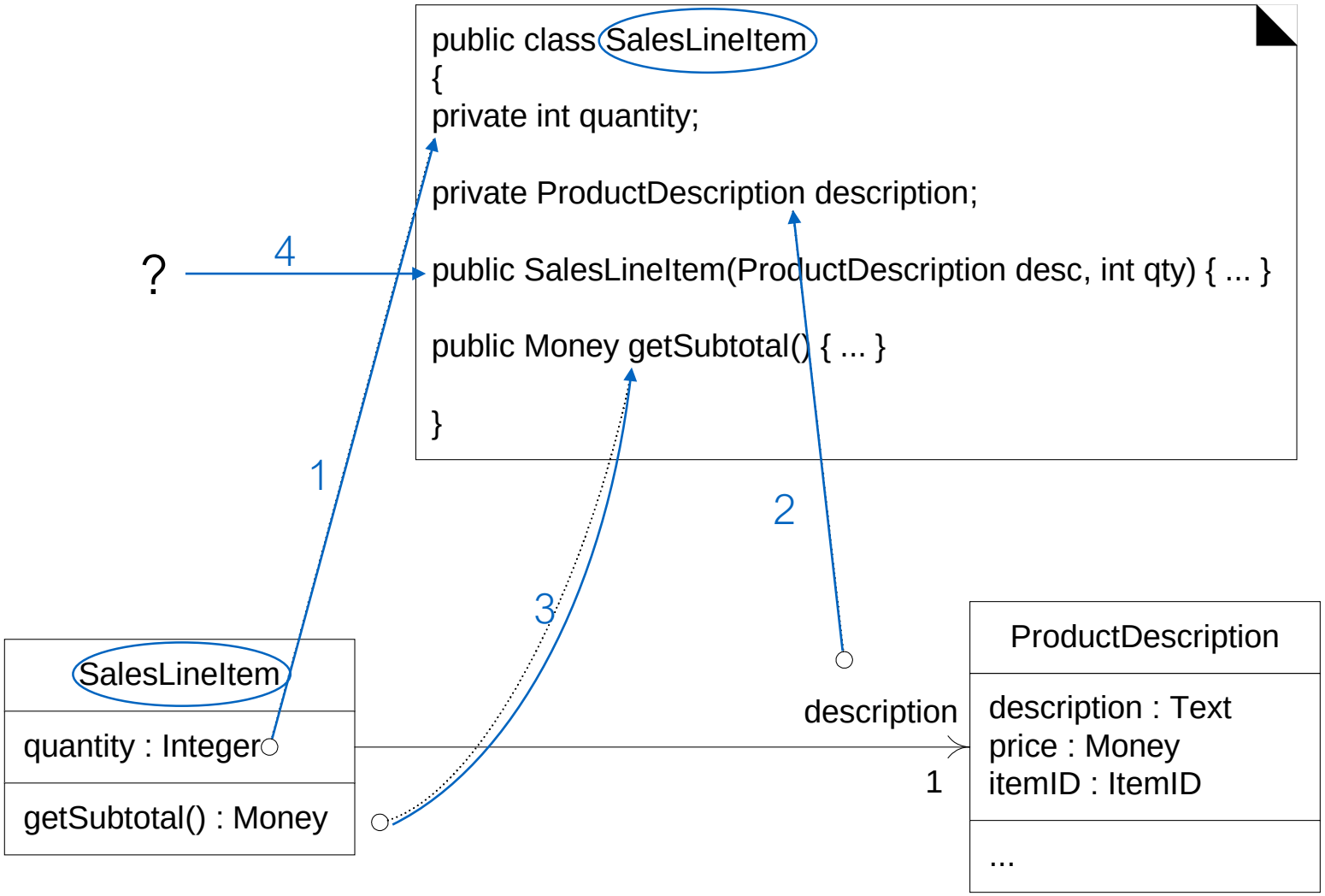
OO implementation requires writing:

- Class and interface definitions
- Method definitions

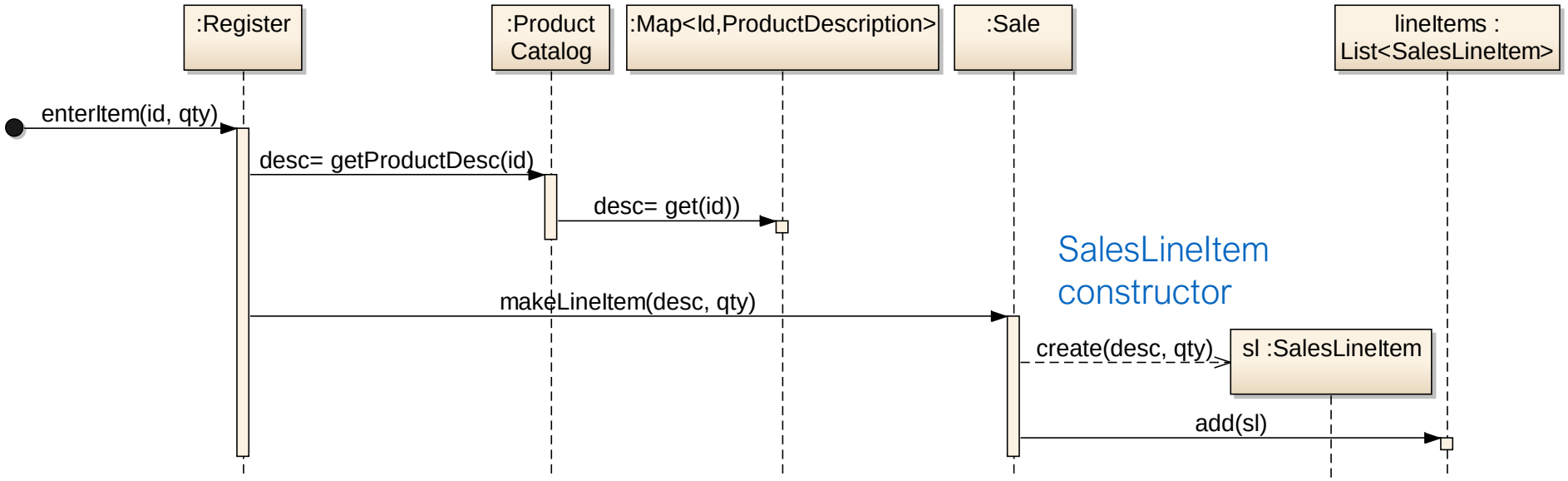
Mappings covered here are relatively mechanical.

- Design should be a great basis; nonetheless,
- Programming in general is iterative and creative.

Creating Class Definitions from DCDs



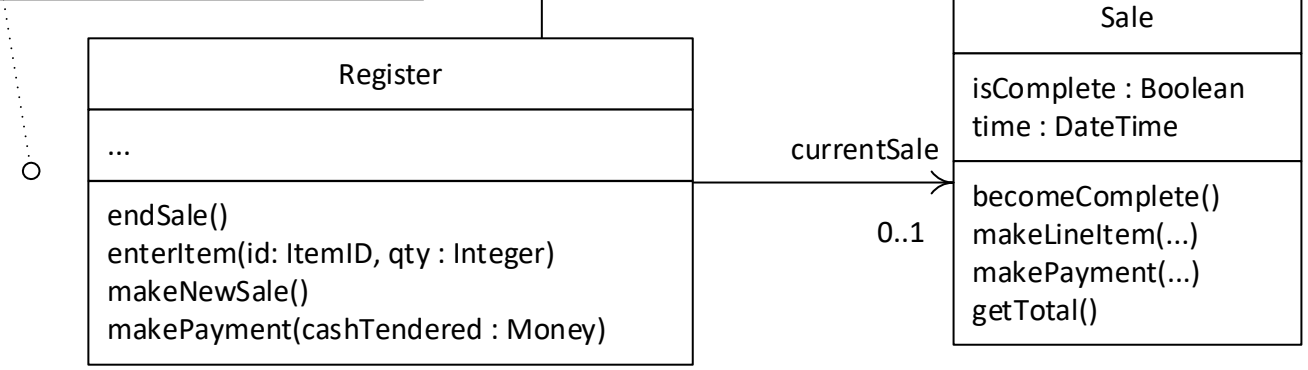
Creating Methods from Sequence Diagrams



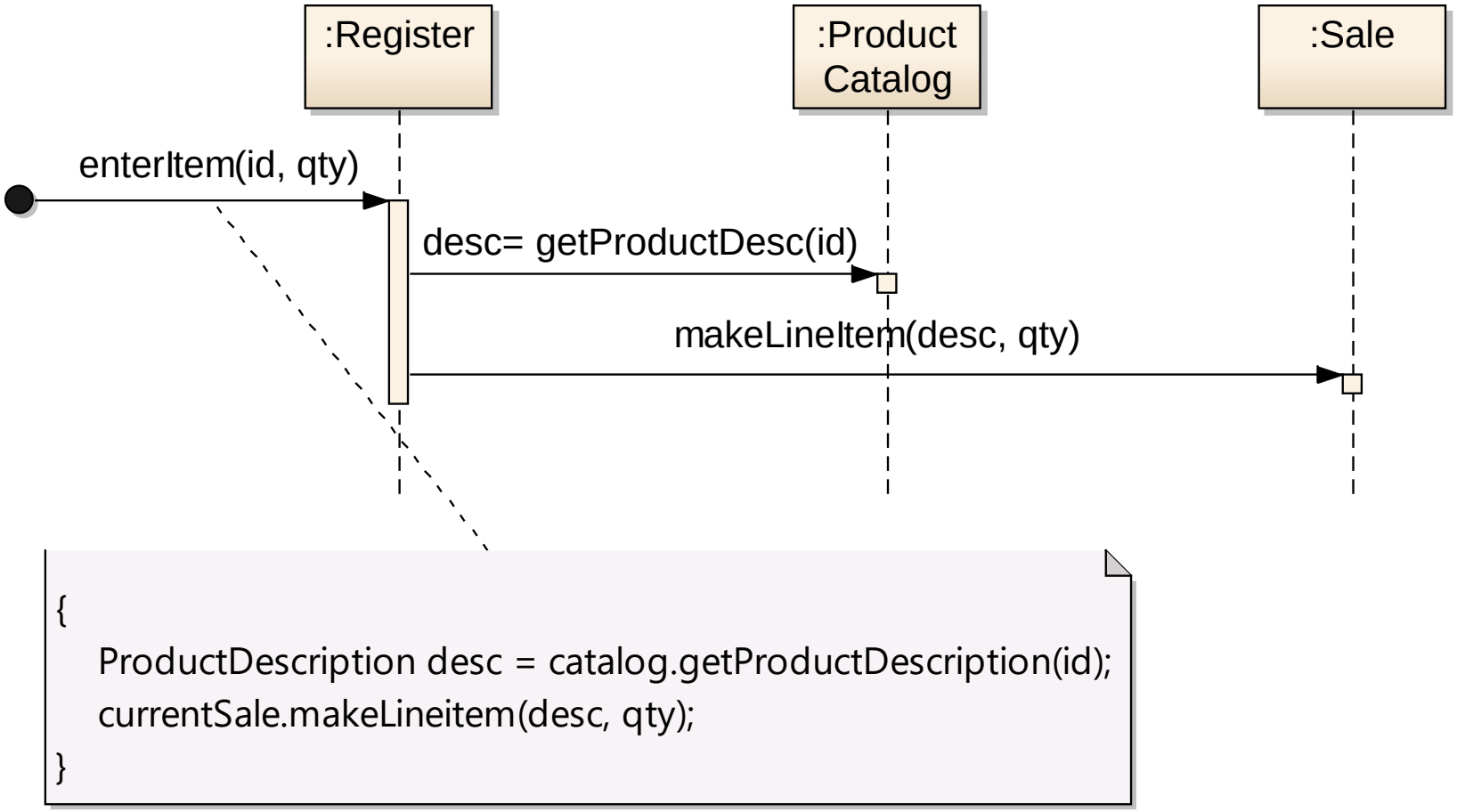
enterItem Sequence Diagram

Create Register Class from Design SD

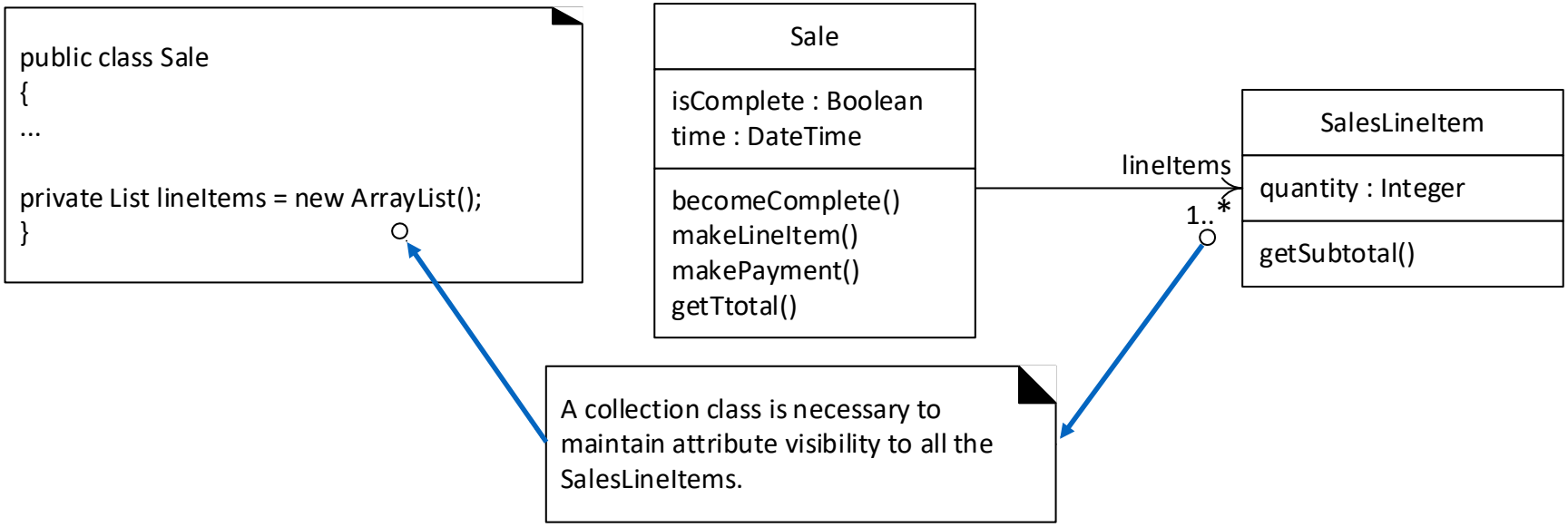
```
public class Register
{
private ProductCatalog catalog;
private Sale currentSale;

public Register(ProductCatalog pc) {...}

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



Create Register.enterItem Method



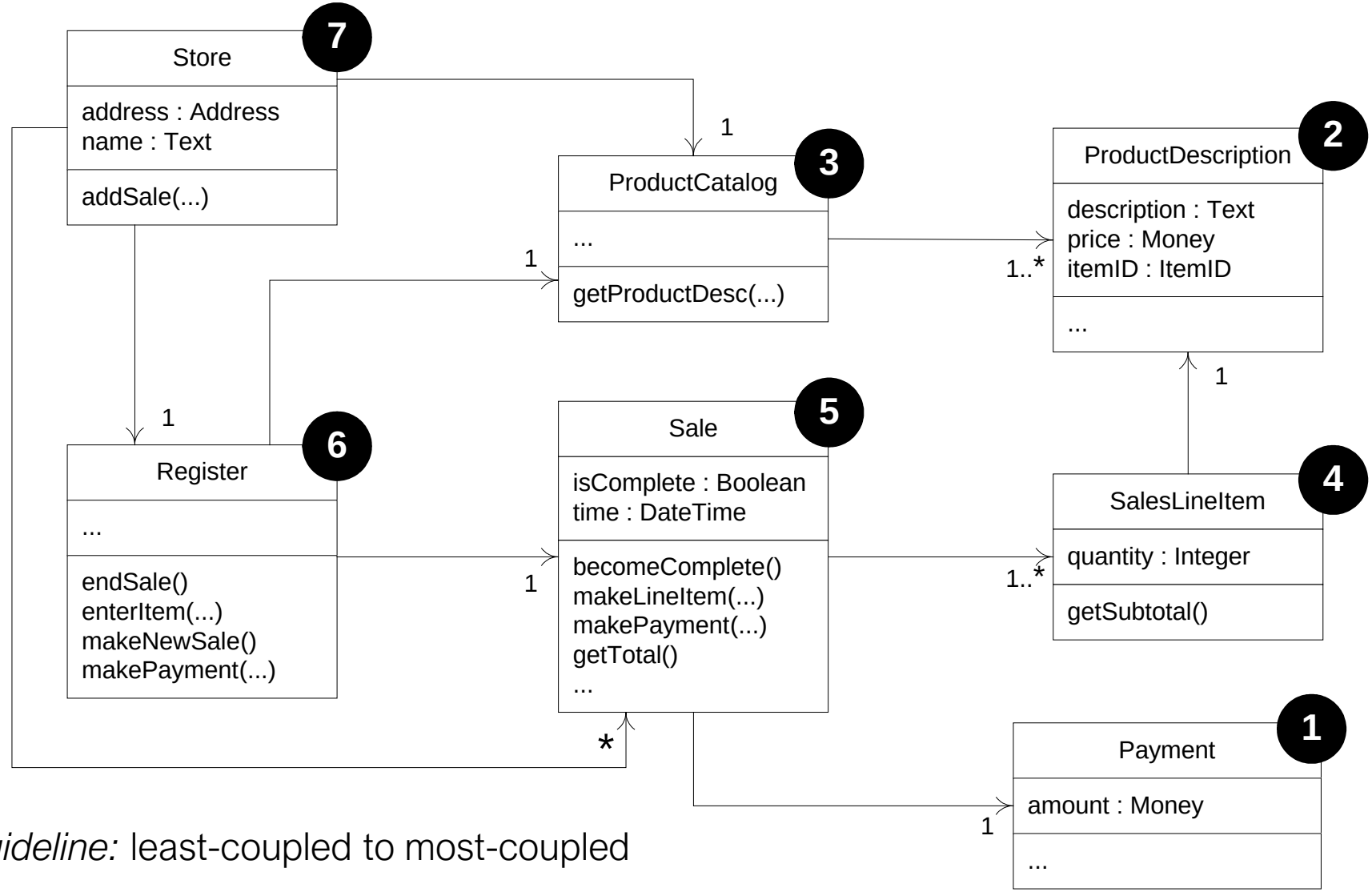
Multiplicity *: Adding a Collection



Guidelines:

- Chose collection class supporting required operations
  - E.g. Key-based lookup -> *Map*; Growing ordered list -> *List*
- If it implements an interface, declare in terms of the interface
  - E.g. *Map<String, Integer> m = new HashMap<String, Integer>();*

Possible Order: Implement/Test



Guideline: least-coupled to most-coupled

Lecture Identification

Coordinator/Lecturer: Philip Dart

Semester: S2 2018

These slides include materials from:

Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition, by Craig Larman, Pearson Education Inc., 2005.