Chapter 16
UML Class Diagrams

Larman, C. “Applying UML and Patterns”. 3rd Ed.
Ed. Prentice-Hall: 2005.

Fig. 16.1

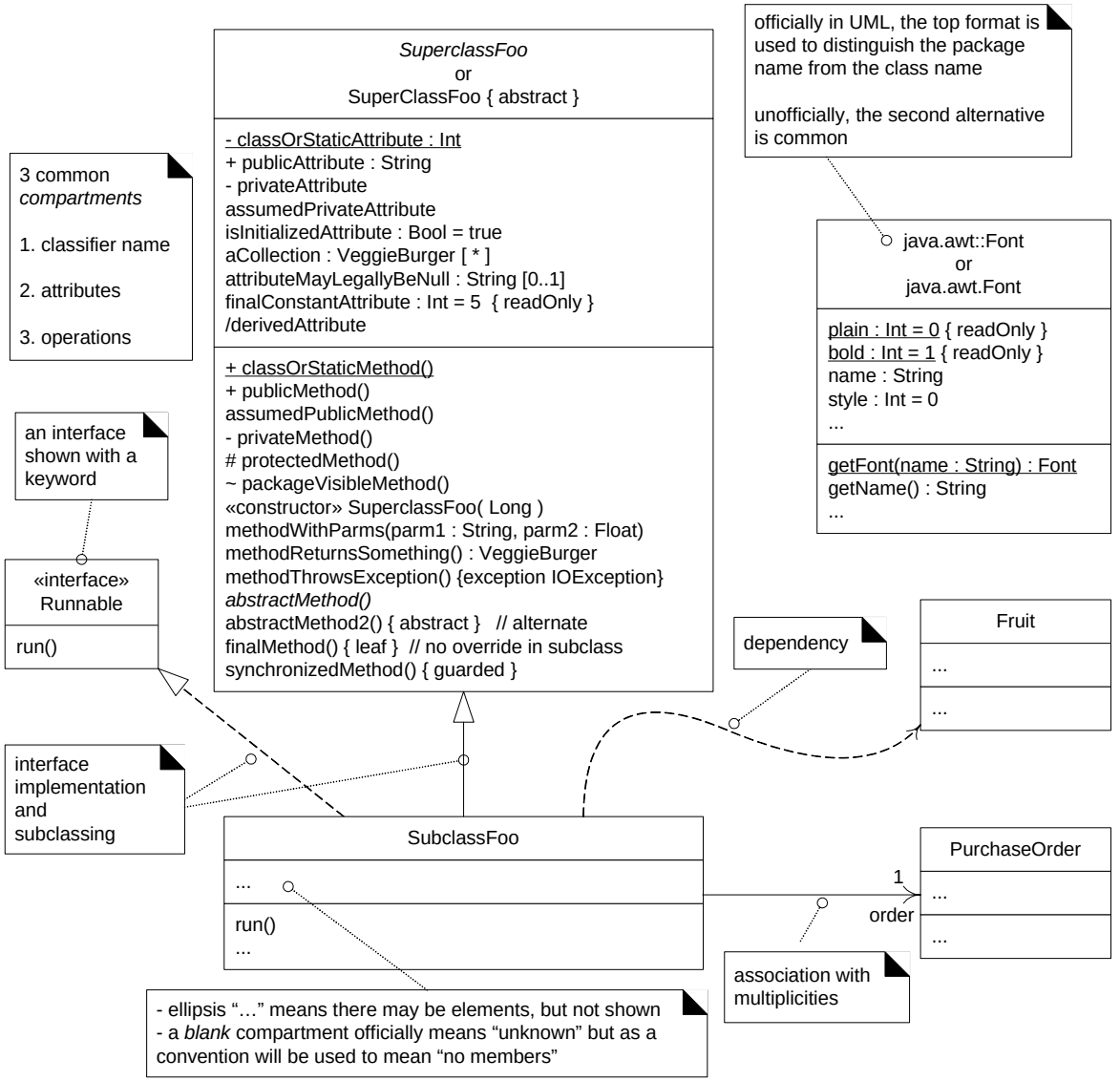

SuperclassFoo
or
SuperClassFoo { abstract }
- classOrStaticAttribute : Int
+ publicAttribute : String
- privateAttribute
assumedPrivateAttribute
isInitializedAttribute : Bool = true
aCollection : VeggieBurger [*]
attributeMayLegallyBeNull : String [0..1]
finalConstantAttribute : Int = 5 { readOnly }
/derivedAttribute
+ classOrStaticMethod()
+ publicMethod()
assumedPublicMethod()
- privateMethod()
protectedMethod()
~ packageVisibleMethod()
«constructor» SuperclassFoo(Long)
methodWithParms(parm1 : String, parm2 : Float)
methodReturnsSomething() : VeggieBurger
methodThrowsException() {exception IOException}
abstractMethod()
abstractMethod2() { abstract } // alternate
finalMethod() { leaf } // no override in subclass
synchronizedMethod() { guarded }
3 common compartments
1. classifier name
2. attributes
3. operations
officially in UML, the top format is used to distinguish the package name from the class name
unofficially, the second alternative is common
java.awt::Font
or
java.awt.Font
plain : Int = 0 { readOnly }
bold : Int = 1 { readOnly }
name : String
style : Int = 0
...
getFont(name : String) : Font
getName() : String
...
an interface shown with a keyword
«interface»
Runnable
run()
dependency
Fruit
...
...
interface implementation and subclassing
SubclassFoo
...
run()
...
PurchaseOrder
...
...
1
order
association with multiplicities
- ellipsis "…" means there may be elements, but not shown
- a blank compartment officially means "unknown" but as a convention will be used to mean "no members"

Fig. 16.2

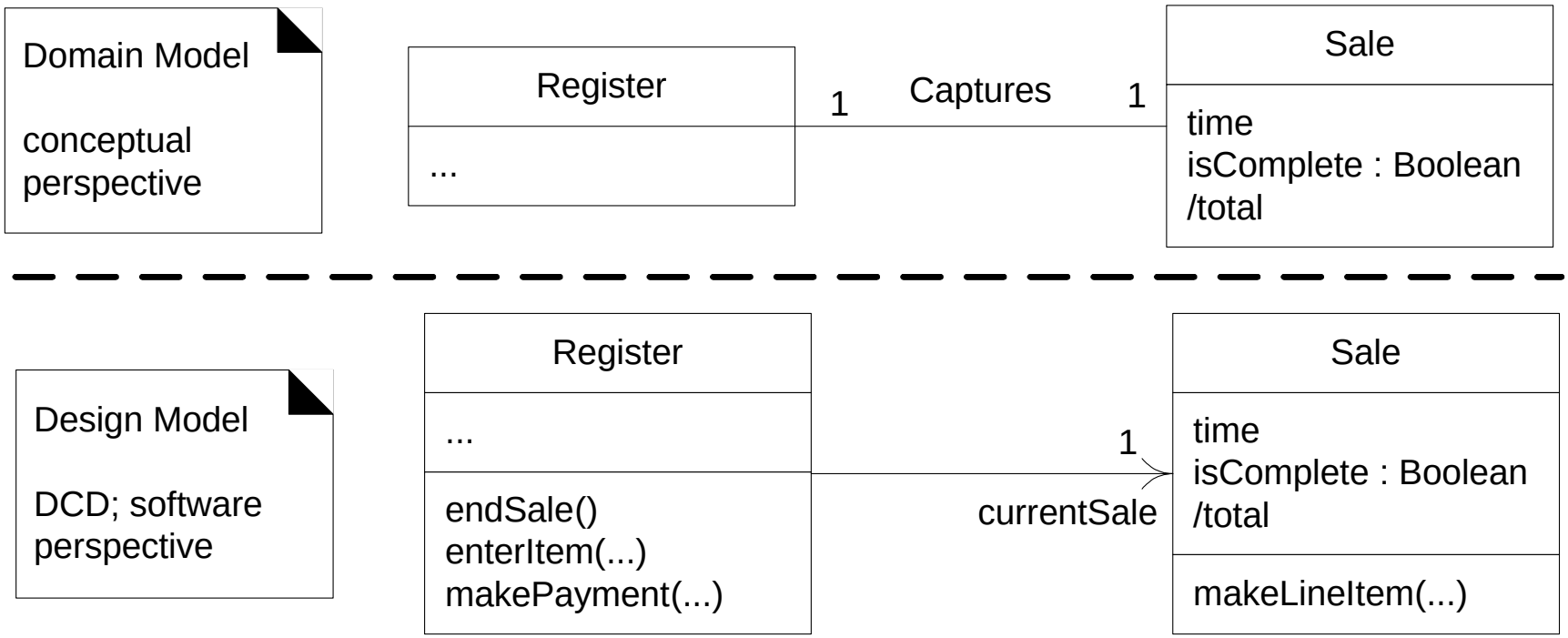

Domain Model
conceptual perspective
Register
...
1
Captures
1
Sale
time
isComplete : Boolean
/total
Design Model
DCD; software perspective
Register
...
endSale()
enterItem(...)
makePayment(...)
1
currentSale
Sale
time
isComplete : Boolean
/total
makeLineItem(...)

Fig. 16.3

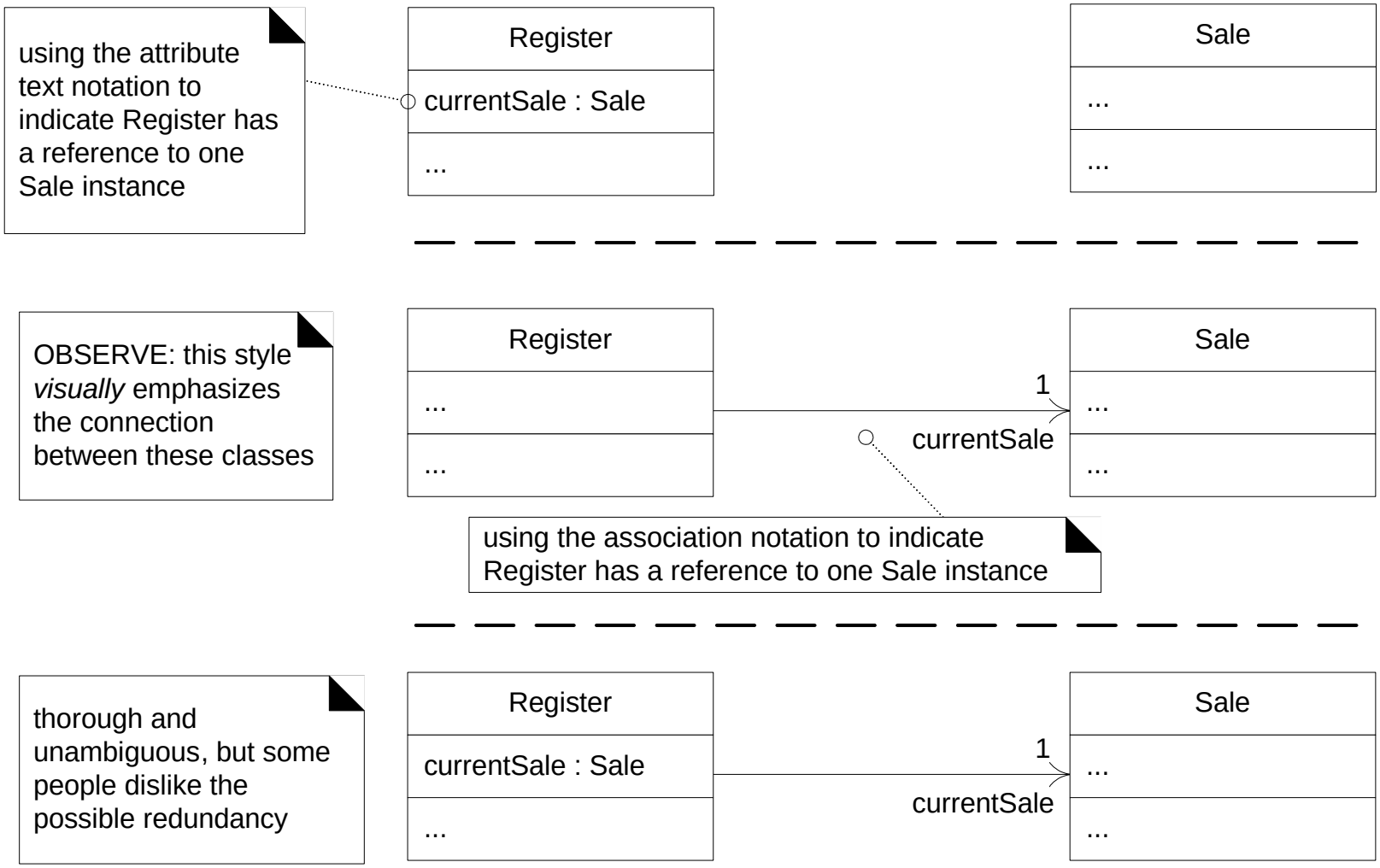

using the attribute text notation to indicate Register has a reference to one Sale instance
Register
currentSale : Sale
...
Sale
...
...
OBSERVE: this style *visually* emphasizes the connection between these classes
Register
...
...
1
currentSale
Sale
...
...
using the association notation to indicate Register has a reference to one Sale instance
thorough and unambiguous, but some people dislike the possible redundancy
Register
currentSale : Sale
...
1
currentSale
Sale
...
...

Fig. 16.4

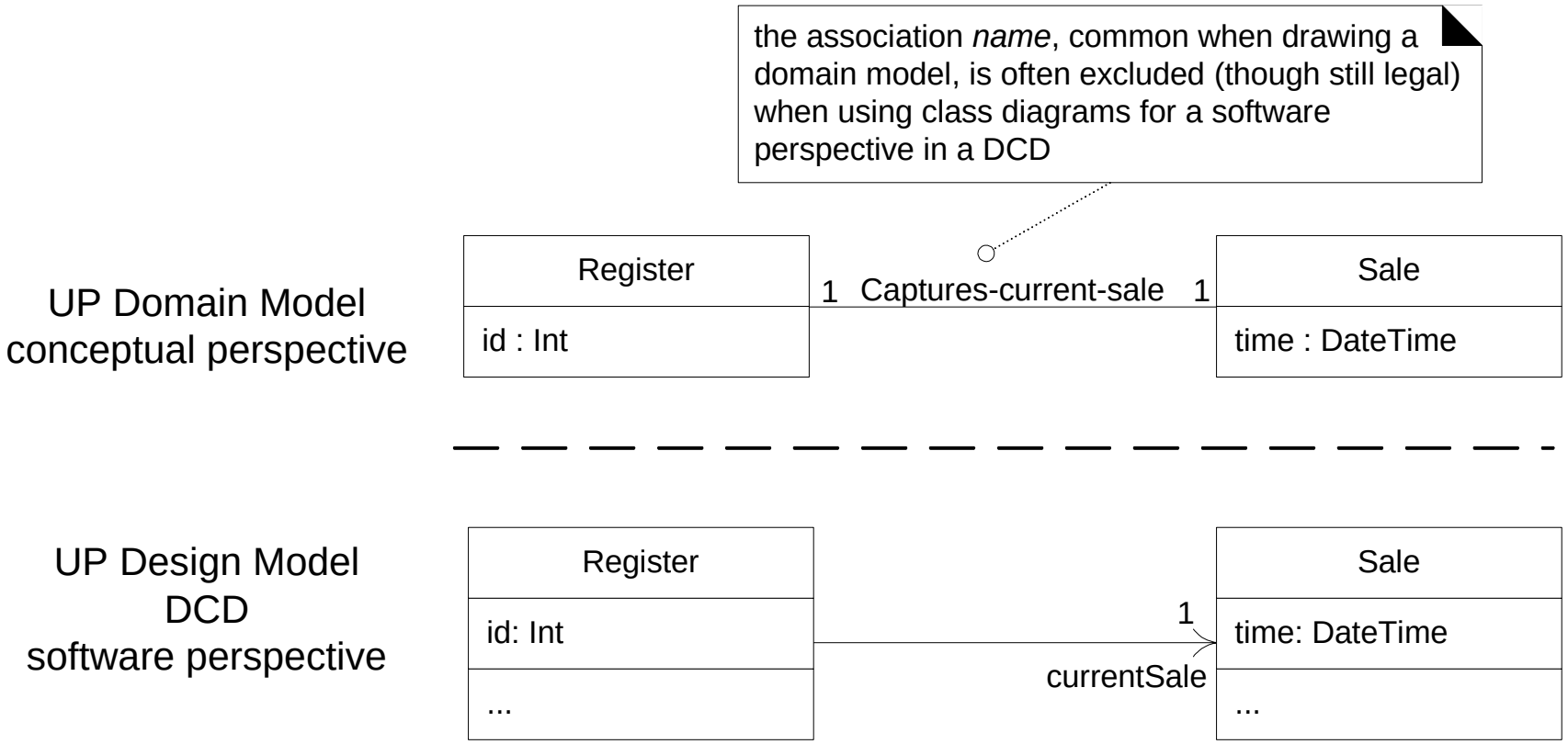

the association *name*, common when drawing a domain model, is often excluded (though still legal) when using class diagrams for a software perspective in a DCD
UP Domain Model
conceptual perspective
Register
id : Int
1
Captures-current-sale
1
Sale
time : DateTime
UP Design Model
DCD
software perspective
Register
id: Int
...
1
currentSale
Sale
time: DateTime
...

Fig. 16.5

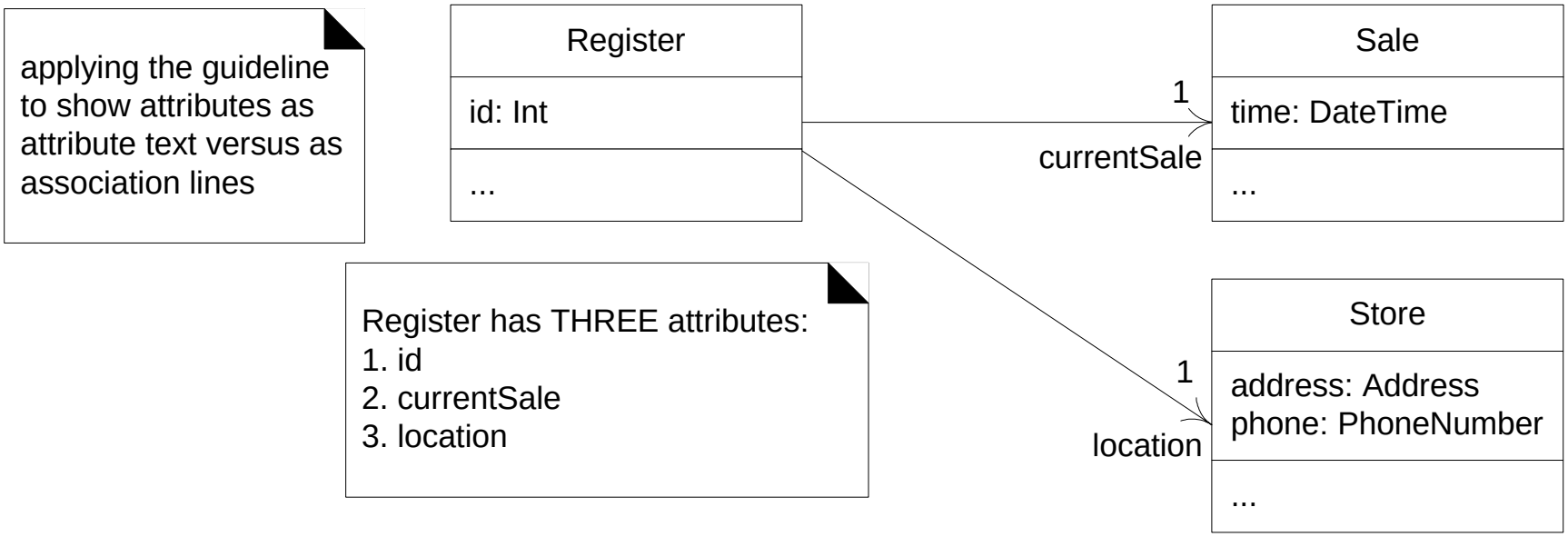

applying the guideline to show attributes as attribute text versus as association lines
Register
id: Int
...
Sale
time: DateTime
...
1
currentSale
Register has THREE attributes:
1. id
2. currentSale
3. location
Store
address: Address
phone: PhoneNumber
...
1
location

Fig. 16.6

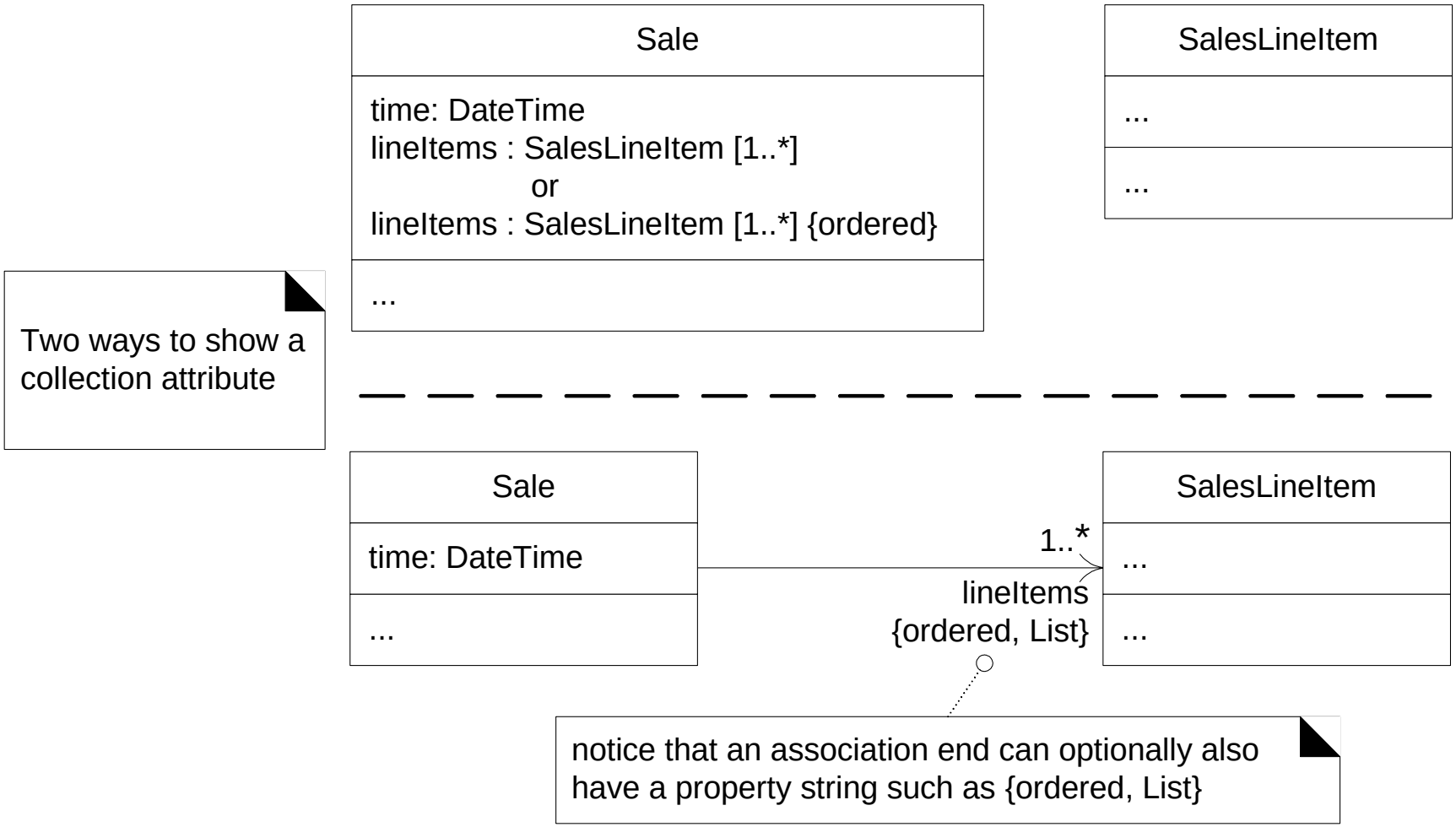

Sale
time: DateTime
lineItems : SalesLineItem [1..*]
or
lineItems : SalesLineItem [1..*] {ordered}
...
SalesLineItem
...
...
Two ways to show a collection attribute
Sale
time: DateTime
...
SalesLineItem
...
...
1..*
lineItems
{ordered, List}
notice that an association end can optionally also have a property string such as {ordered, List}

Fig. 16.7


«method»
// pseudo-code or a specific language is OK
public void enterItem(id, qty)
{
ProductDescription desc = catalog.getProductDescription(id);
sale.makeLineItem(desc, qty);
}
Register
...
endSale()
enterItem(id, qty)
makeNewSale()
makePayment(cashTendered)

Fig. 16.8

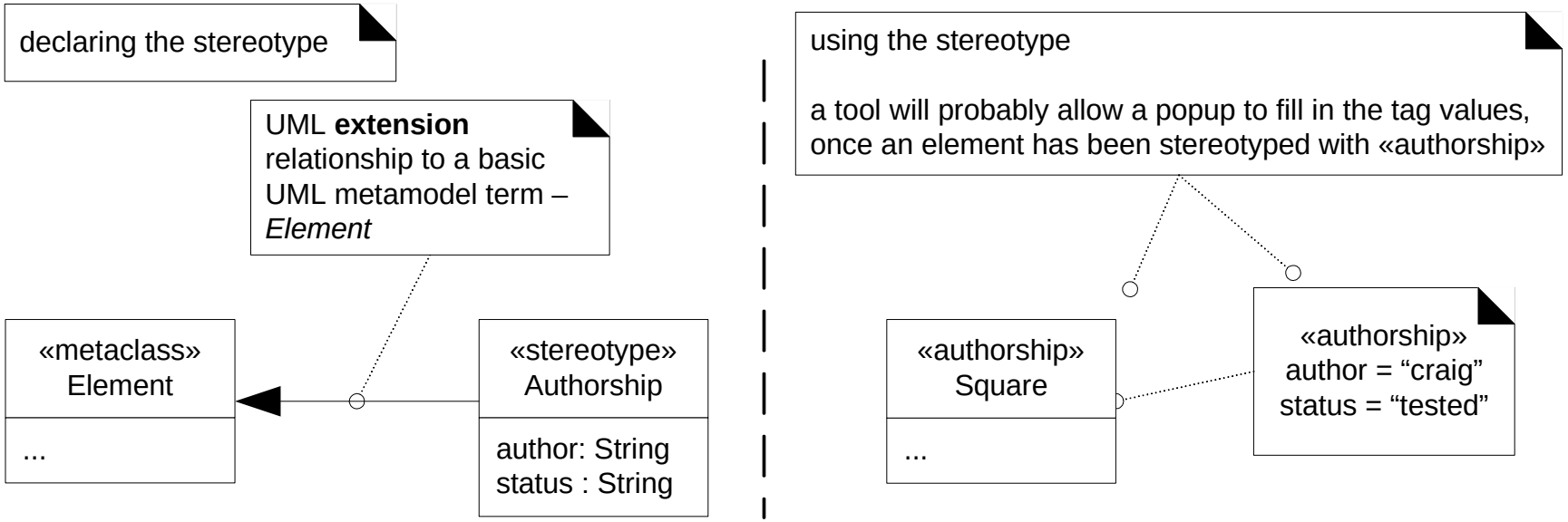

declaring the stereotype
UML **extension** relationship to a basic UML metamodel term – *Element*
«metaclass»
Element
...
«stereotype»
Authorship
author: String
status : String
using the stereotype
a tool will probably allow a popup to fill in the tag values, once an element has been stereotyped with «authorship»
«authorship»
Square
...
«authorship»
author = “craig”
status = “tested”

Fig. 16.9

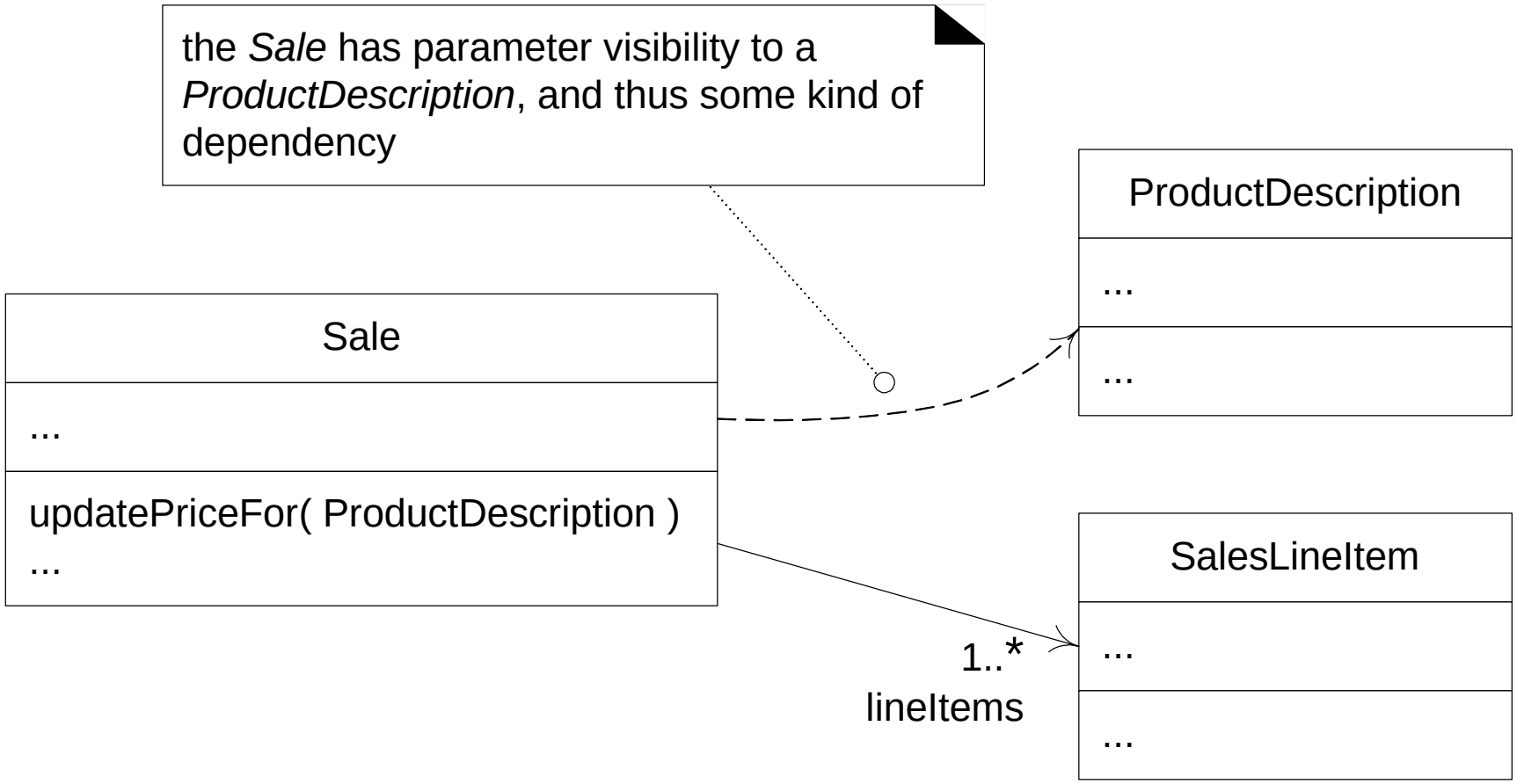

the *Sale* has parameter visibility to a *ProductDescription*, and thus some kind of dependency
ProductDescription
...
...
Sale
...
updatePriceFor(ProductDescription)
...
SalesLineItem
...
...
1..*
lineItems

Fig. 16.10

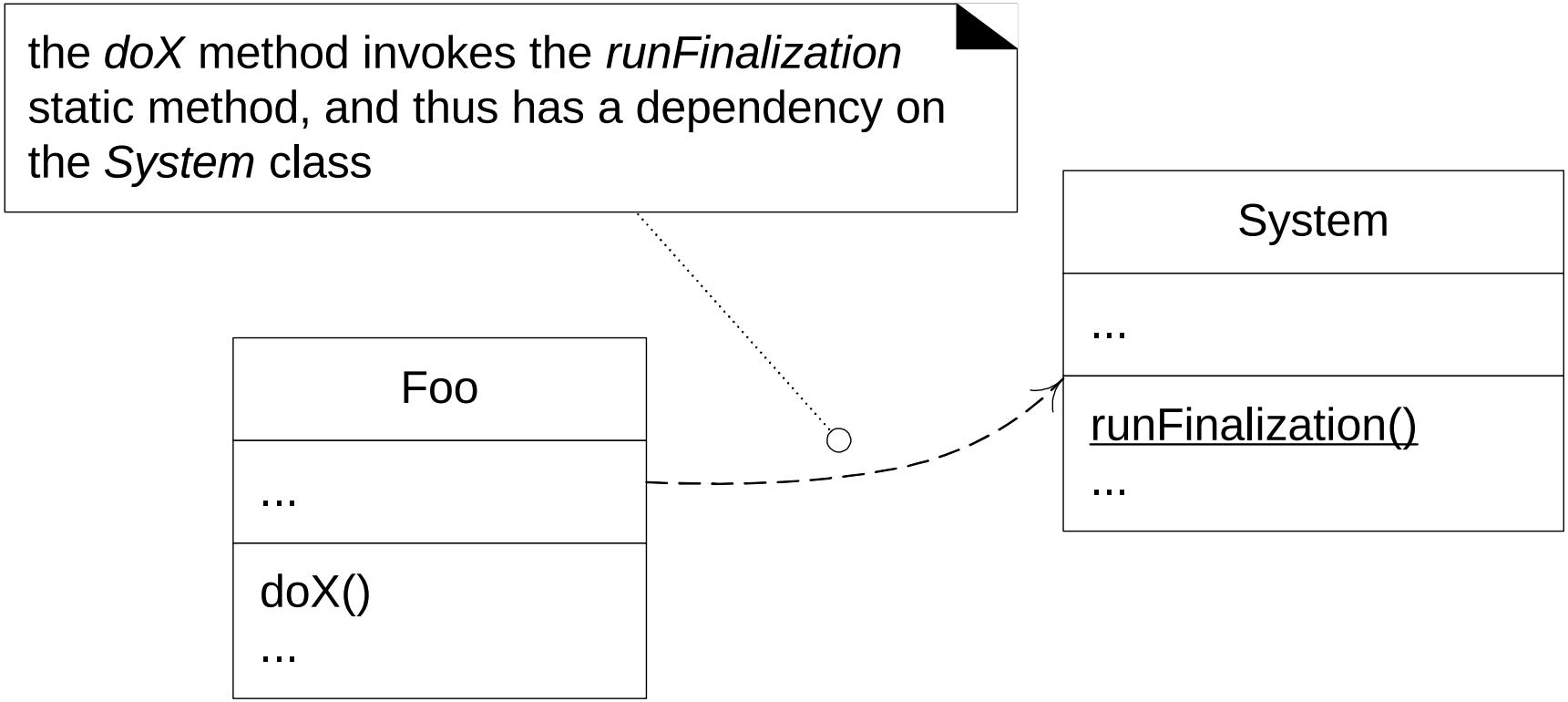

the *doX* method invokes the *runFinalization* static method, and thus has a dependency on the *System* class
Foo
...
doX()
...
System
...
runFinalization()
...

Fig. 16.11

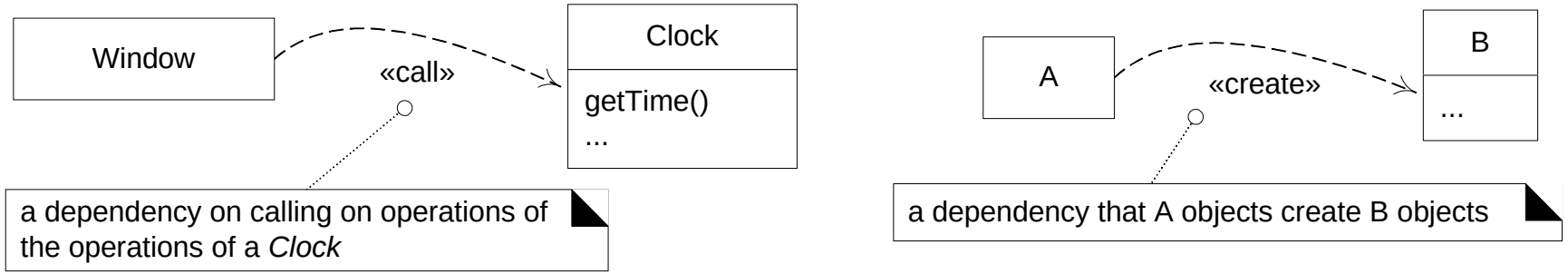

Window
«call»
Clock
getTime()
...
a dependency on calling on operations of the operations of a *Clock*
A
«create»
B
...
a dependency that A objects create B objects

Fig. 16.12

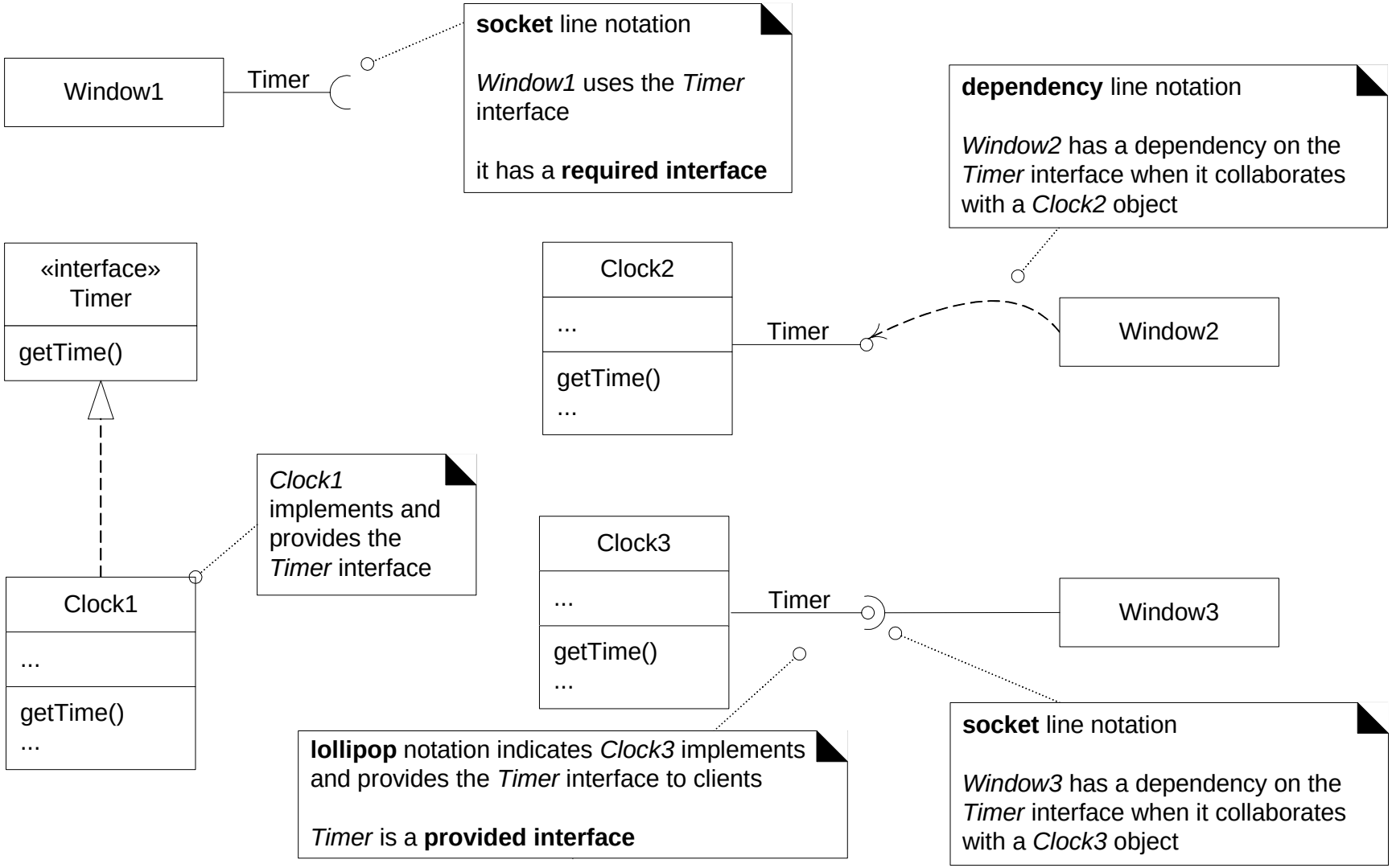
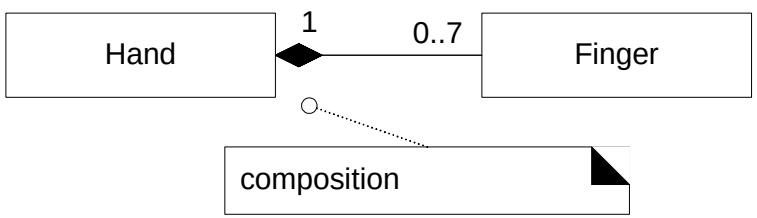
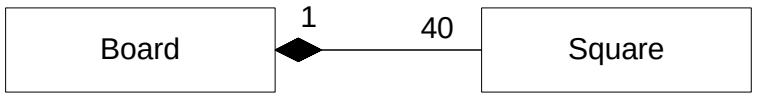

Window1
Timer
socket line notation
Window1 uses the *Timer* interface
it has a **required interface**
dependency line notation
Window2 has a dependency on the *Timer* interface when it collaborates with a *Clock2* object
«interface»
Timer
getTime()
Clock2
...
getTime()
...
Timer
Window2
Clock1 implements and provides the *Timer* interface
Clock1
...
getTime()
...
Clock3
...
getTime()
...
Timer
Window3
lollipop notation indicates *Clock3* implements and provides the *Timer* interface to clients
Timer is a **provided interface**
socket line notation
Window3 has a dependency on the *Timer* interface when it collaborates with a *Clock3* object

Fig. 16.13



composition means
-a part instance (*Square*) can only be part of one composite (*Board*) at a time

-the composite has sole responsibility for management of its parts, especially creation and deletion



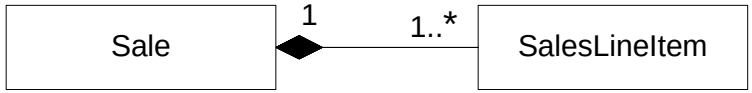


Fig. 16.14

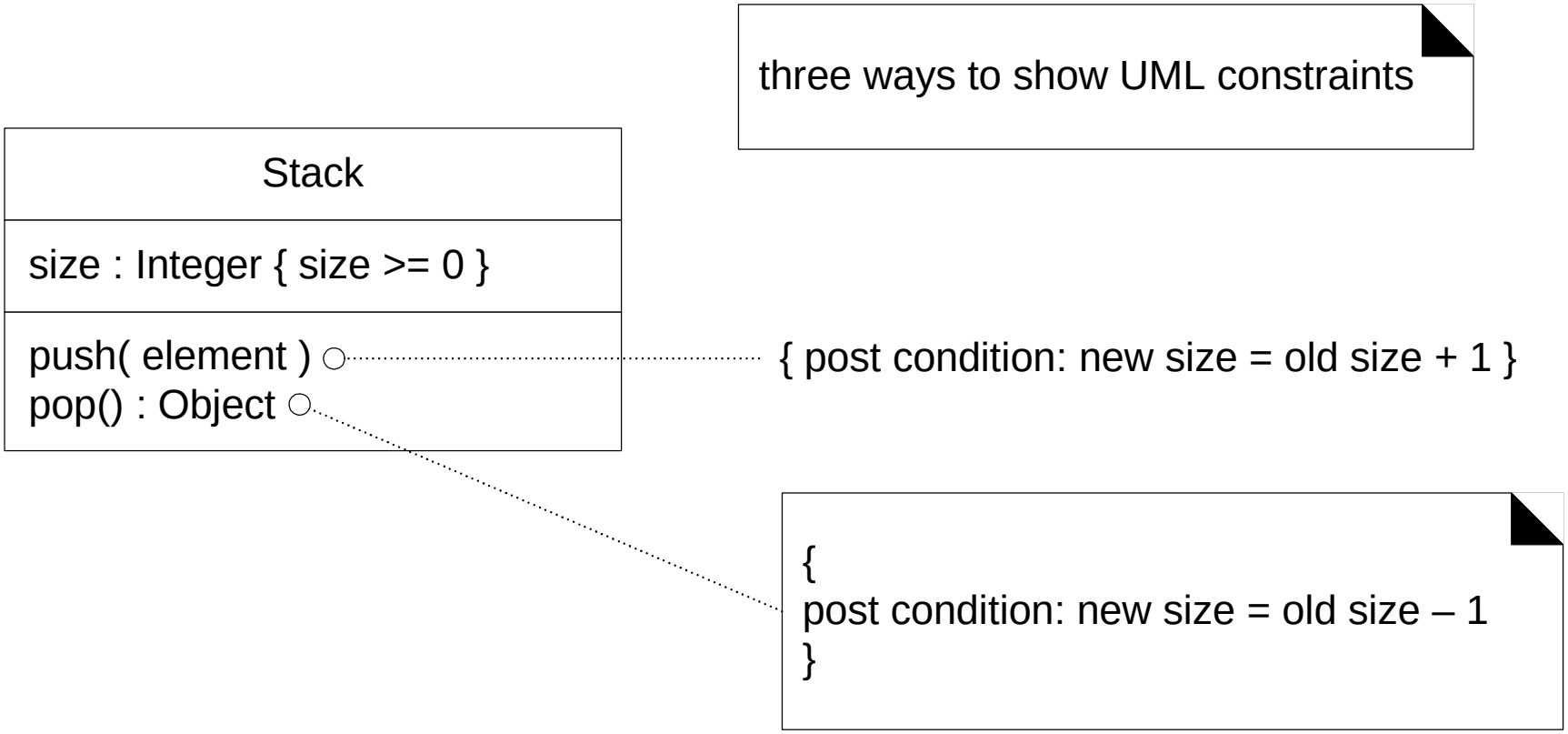

three ways to show UML constraints
Stack
size : Integer { size >= 0 }
push(element)
pop() : Object
{ post condition: new size = old size + 1 }
{
post condition: new size = old size – 1
}

Fig. 16.15

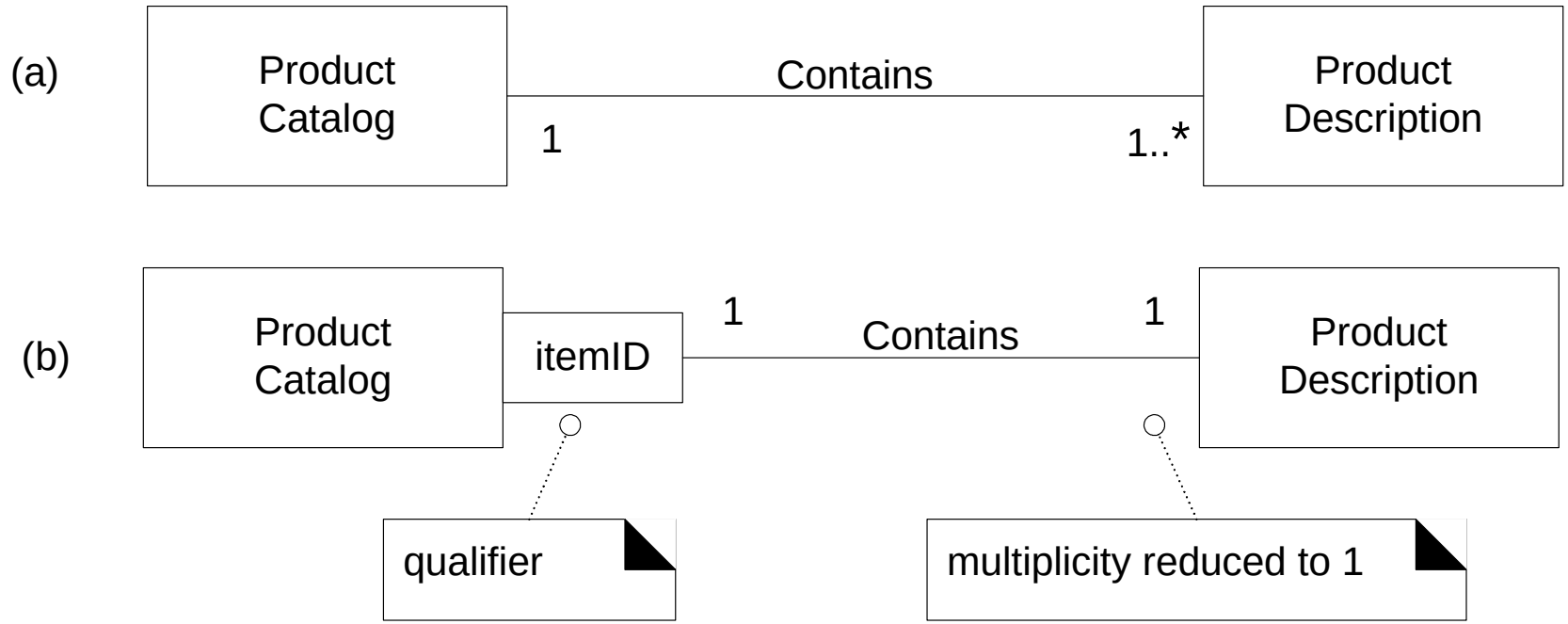

(a)
Product Catalog
Contains
1
1..*
Product Description
(b)
Product Catalog
itemID
1
Contains
1
Product Description
qualifier
multiplicity reduced to 1

Fig. 16.16

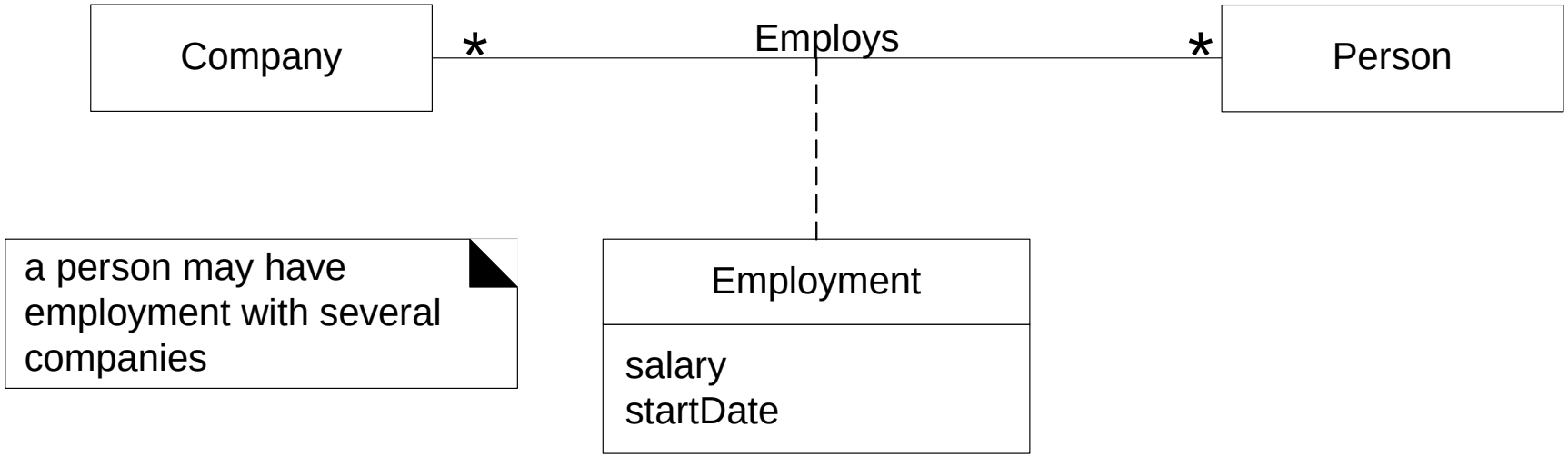

Company
*
Employs
*
Person
a person may have employment with several companies
Employment
salary
startDate

Fig. 16.17

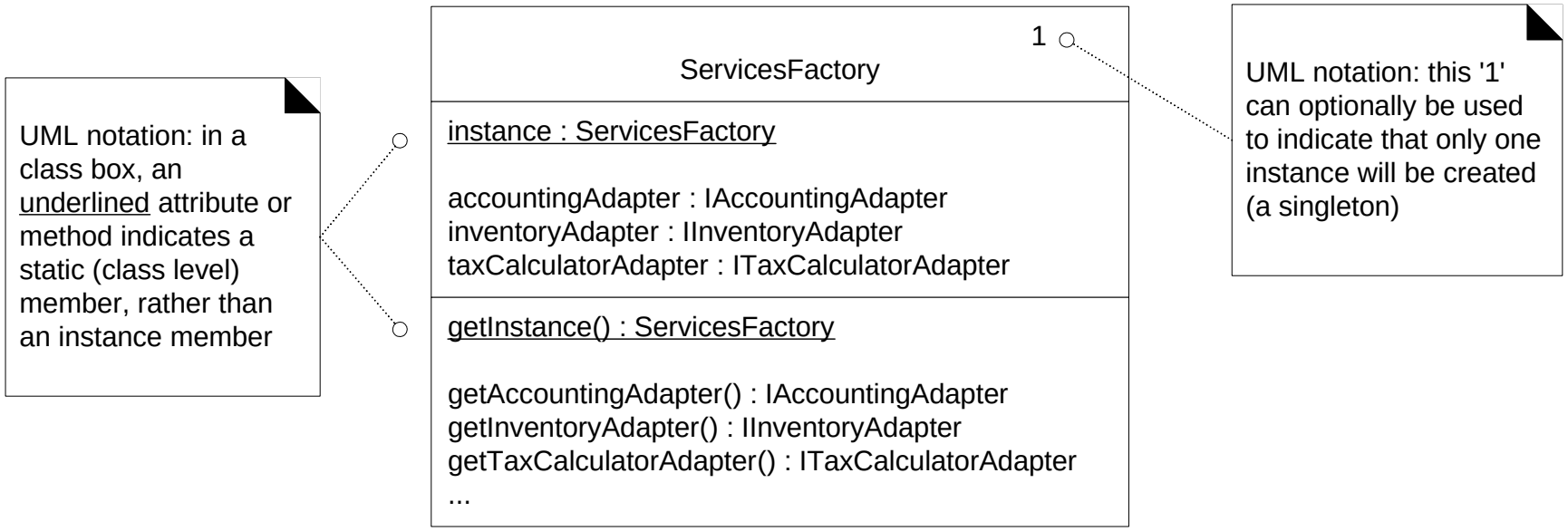

UML notation: in a class box, an underlined attribute or method indicates a static (class level) member, rather than an instance member
ServicesFactory
1
instance : ServicesFactory
accountingAdapter : IAccountingAdapter
inventoryAdapter : IInventoryAdapter
taxCalculatorAdapter : ITaxCalculatorAdapter
getInstance() : ServicesFactory
getAccountingAdapter() : IAccountingAdapter
getInventoryAdapter() : IInventoryAdapter
getTaxCalculatorAdapter() : ITaxCalculatorAdapter
...
UML notation: this '1' can optionally be used to indicate that only one instance will be created (a singleton)

Fig. 16.18

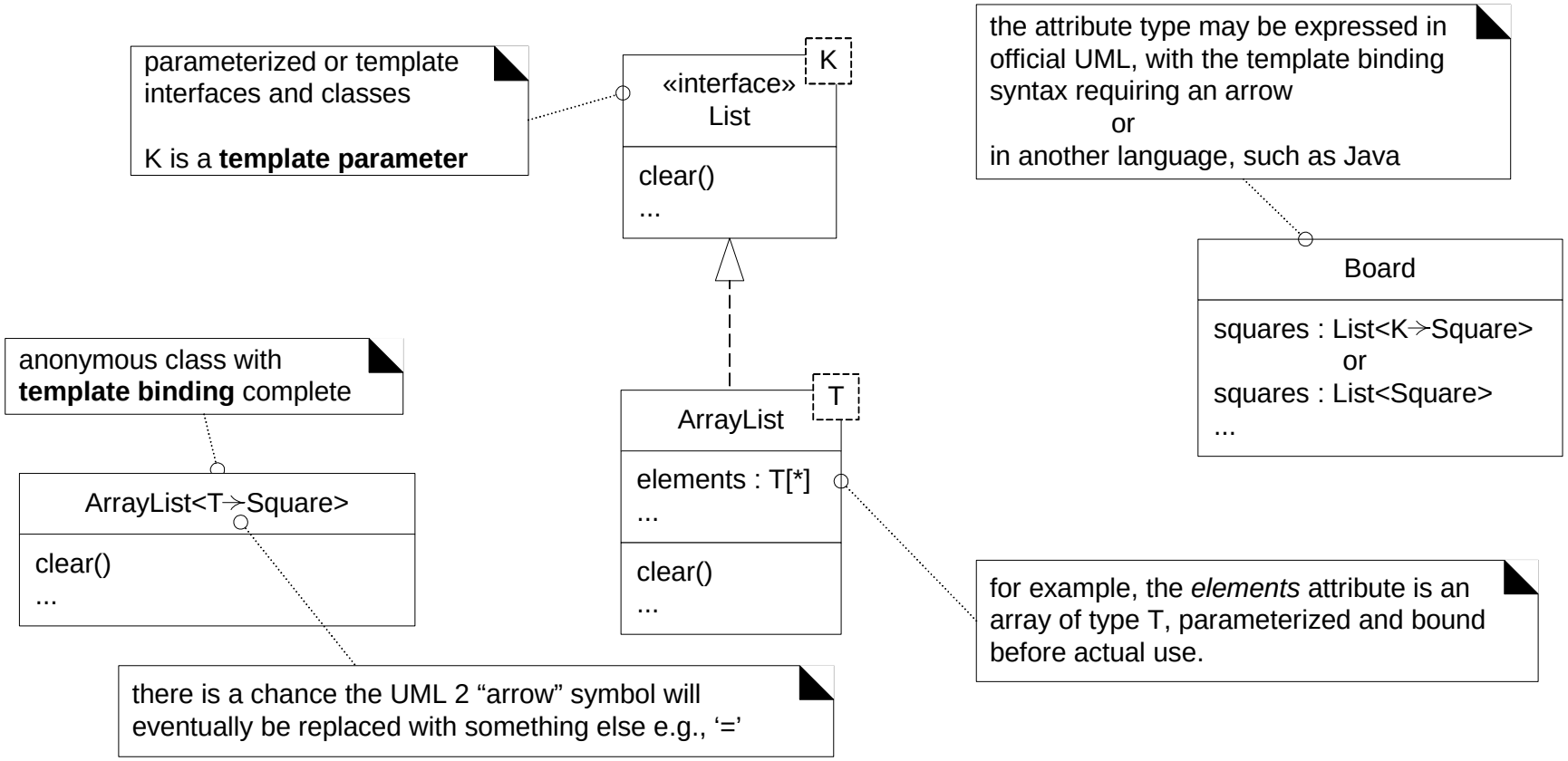

parameterized or template interfaces and classes
K is a **template parameter**
«interface» List
K
clear()
...
the attribute type may be expressed in official UML, with the template binding syntax requiring an arrow
or
in another language, such as Java
Board
squares : List<K→Square>
or
squares : List<Square>
...
anonymous class with **template binding** complete
ArrayList<T→Square>
clear()
...
ArrayList
T
elements : T[*]
...
clear()
...
for example, the *elements* attribute is an array of type T, parameterized and bound before actual use.
there is a chance the UML 2 “arrow” symbol will eventually be replaced with something else e.g., ‘=’

Fig. 16.19

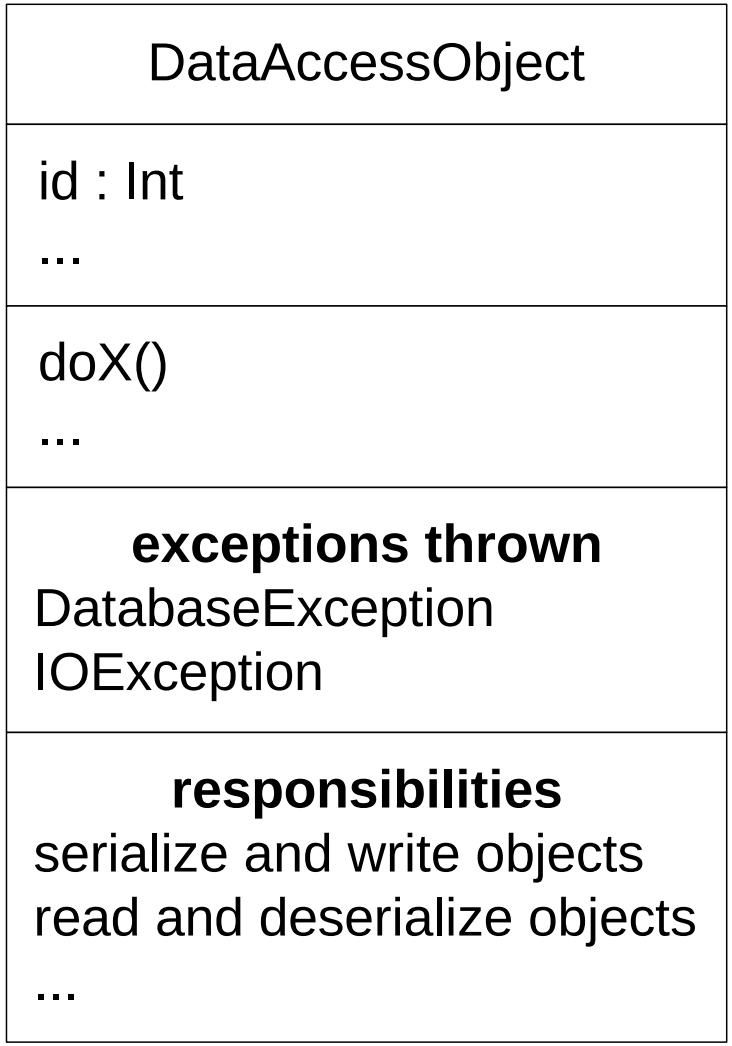

DataAccessObject
id : Int
...
doX()
...
exceptions thrown
DatabaseException
IOException
responsibilities
serialize and write objects
read and deserialize objects
...

Fig. 16.20

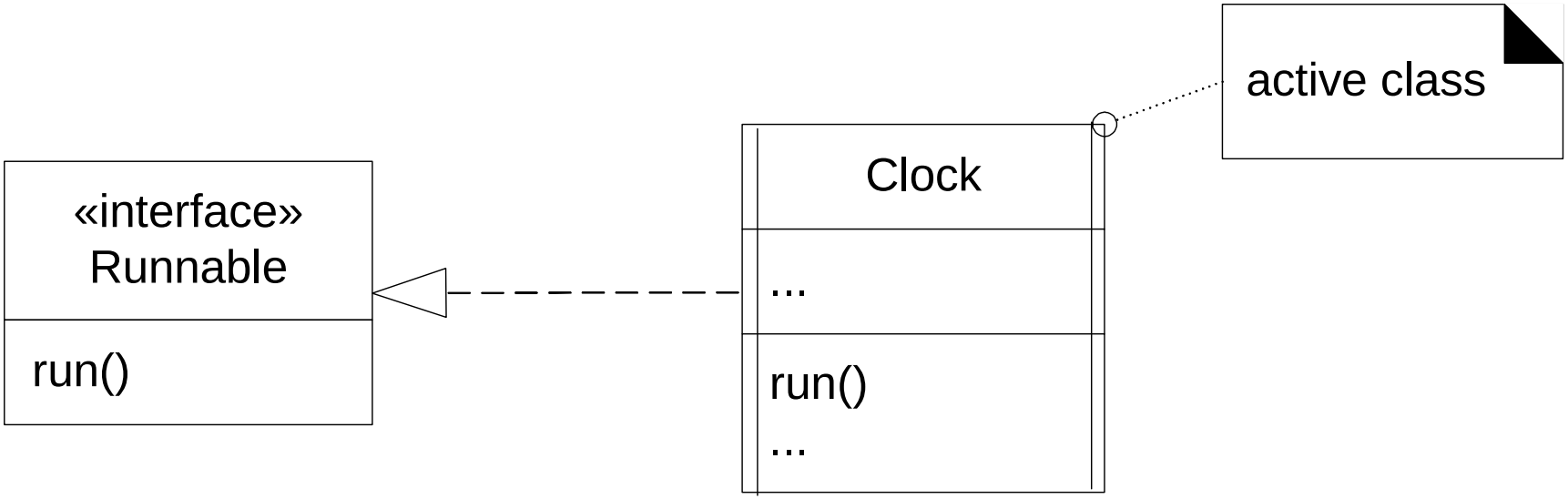
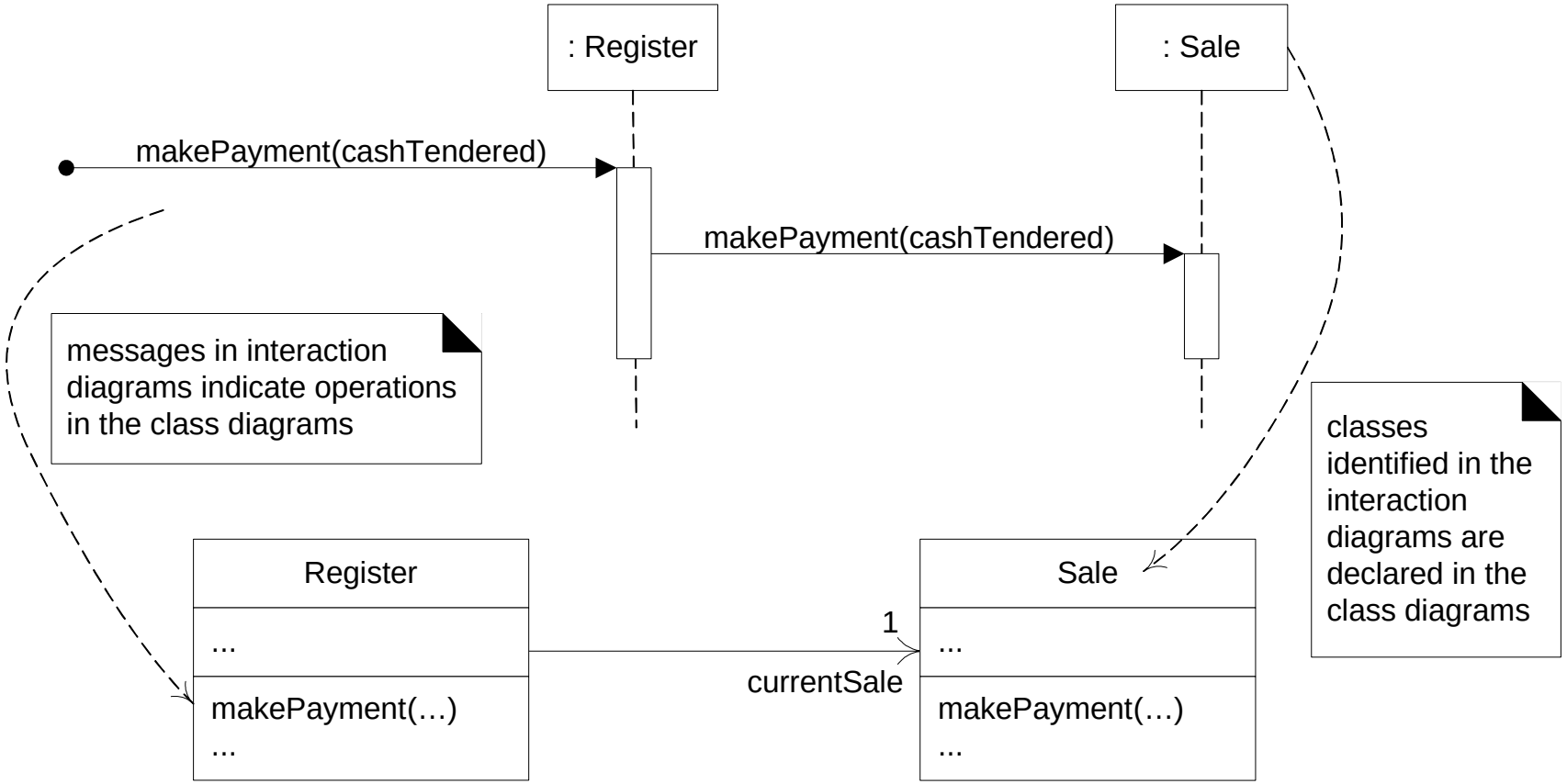


Fig. 16.21


: Register
: Sale
makePayment(cashTendered)
makePayment(cashTendered)
messages in interaction diagrams indicate operations in the class diagrams
classes identified in the interaction diagrams are declared in the class diagrams
Register
...
makePayment(…)
...
Sale
...
makePayment(…)
...
1
currentSale