Interaction Diagrams

Curt Clifton

Rose-Hulman Institute of Technology

Anonymous Feedback

- " There is not enough information in the **partial use cases, scope and requirements** to go into the type of depth an operational contract with **any type of certainty**. This is taking too much time having to "come up" with details that should have already been laid out by the requirements document and **easily** accessible."
  - OCs are part of the requirements
  - You have the domain model, which is most pertinent
  - This course is all about dealing with uncertainty and resolving it
  - You won't be penalized for answers that are different than mine so long as they follow the guidelines

What Matters Most?

- Principles of assigning responsibilities to objects
- Design patterns

But we need some notation to communicate these ideas

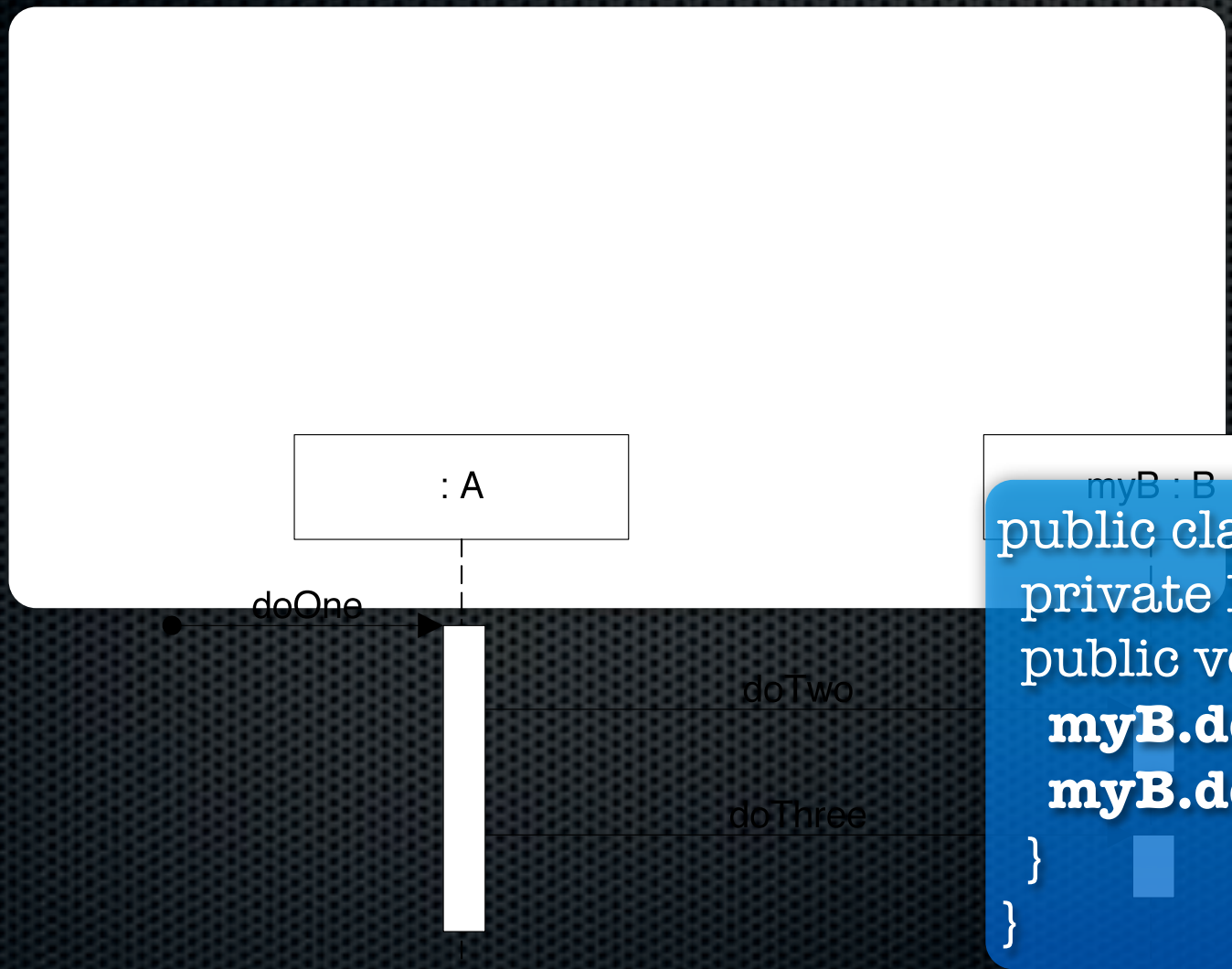
Interaction Diagrams

- For **dynamic** object modeling
- Two common types:
  - Sequence diagrams
  - Communication diagrams

Don't confuse with System Sequence Diagrams (SSDs), which use a subset of the notation

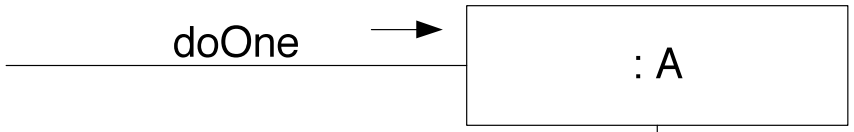
Q1,2

Sequence Diagram Example




```
public class A {
 private B myB = new B();
 public void doOne() {
  myB.doTwo();
  myB.doThree();
 }
}
```

Communication Diagram Example



1: doTwo

2: doThree

myB : B

```
public class A {
  private B myB = new B();
  public void doOne() {
    myB.doTwo();
    myB.doThree();
  }
}
```

Relative Strengths

- Sequence diagrams
  - Clearer notation and semantics
  - Better tool support
  - Easier to follow
  - Excellent for documents
- Communication diagrams
  - Much more space efficient
  - Easier to modify quickly
  - Excellent for UML as sketch

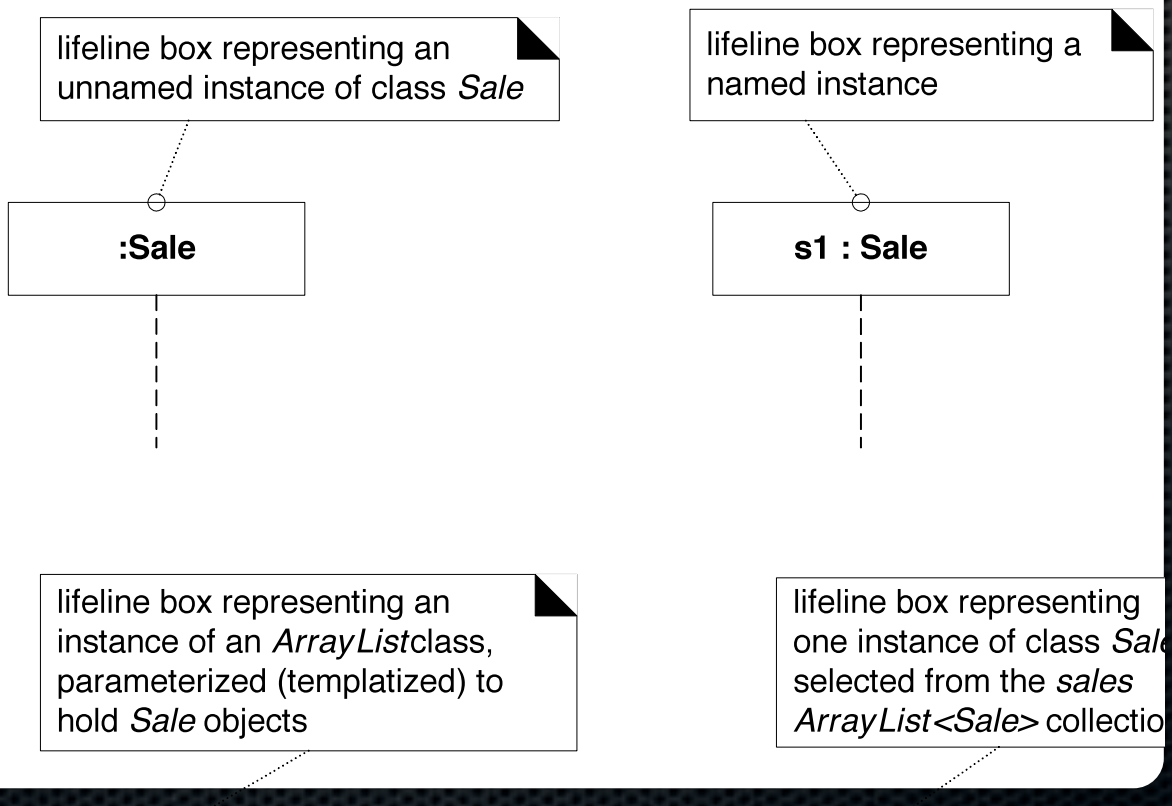
Why Bother with Interaction Diagrams?

- Keep us from getting bogged down in syntax
- Can allocate responsibilities with minimal commitment

But don't get bogged down

Common Notation

Lifeline Boxes



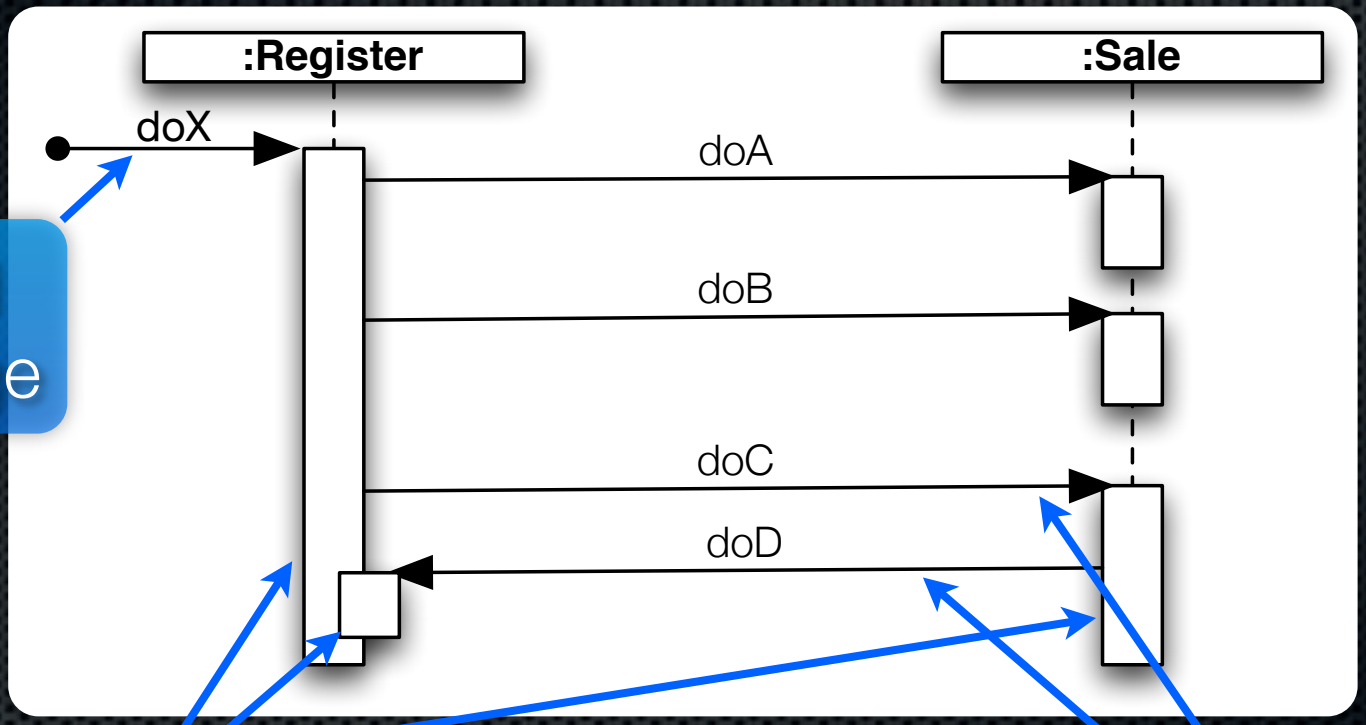
Q3

Message Syntax

- id = message(parameter : parameterType) : returnType
- Much is optional, for example:
  - initialize(register)
  - initialize
  - d = getProductDescription(id)
  - d = getProductDescription(id:ItemID)
  - d = getProductDescription(id:ItemID) : ProductDesc

Sequence Diagrams

Terminology



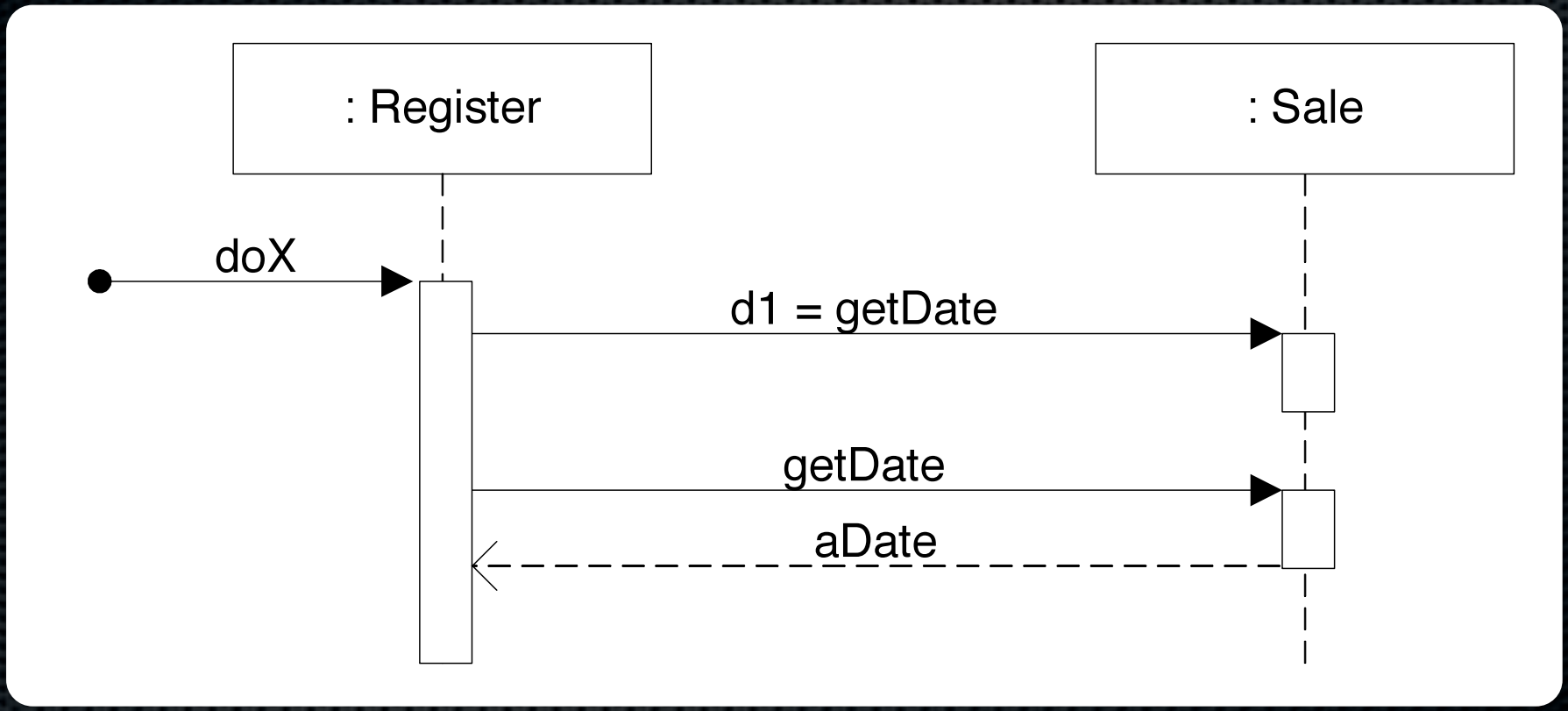
Found message

Execution specification bars

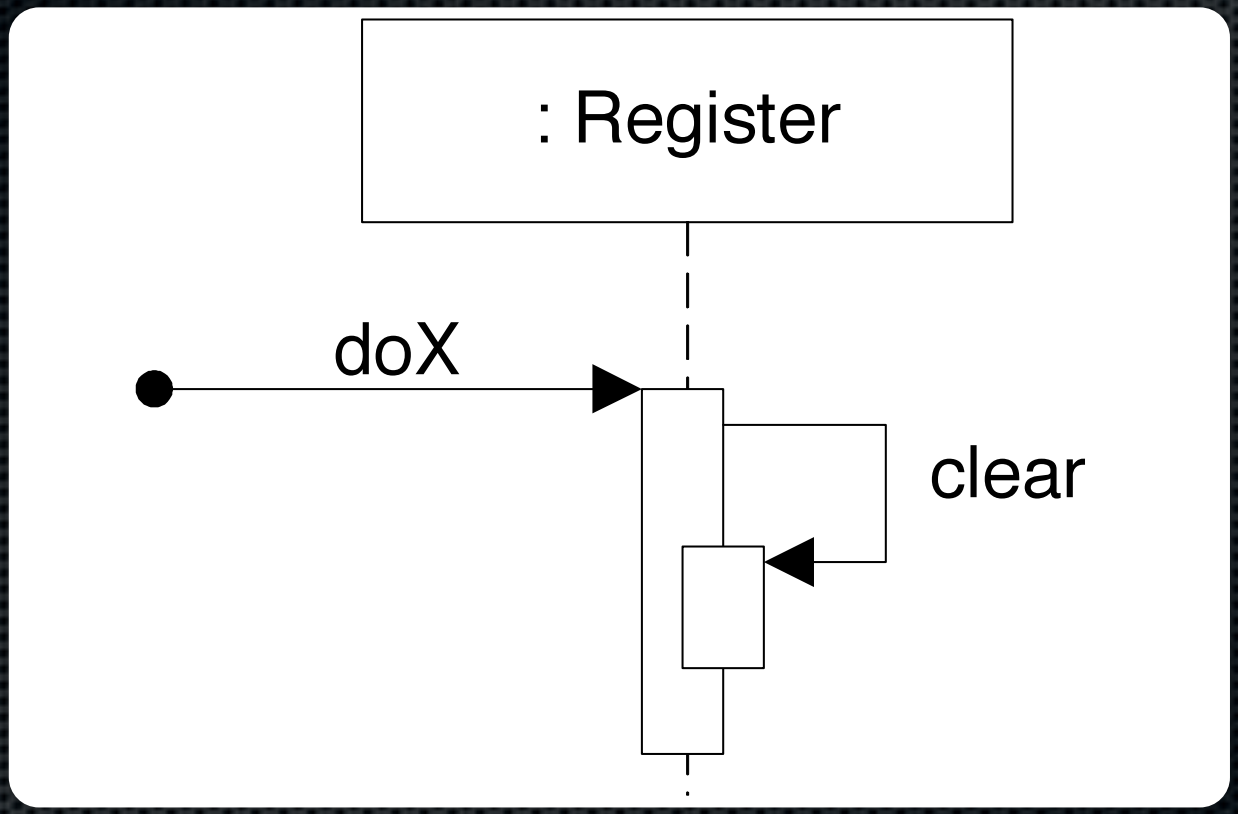
Synchronous messages

Q4

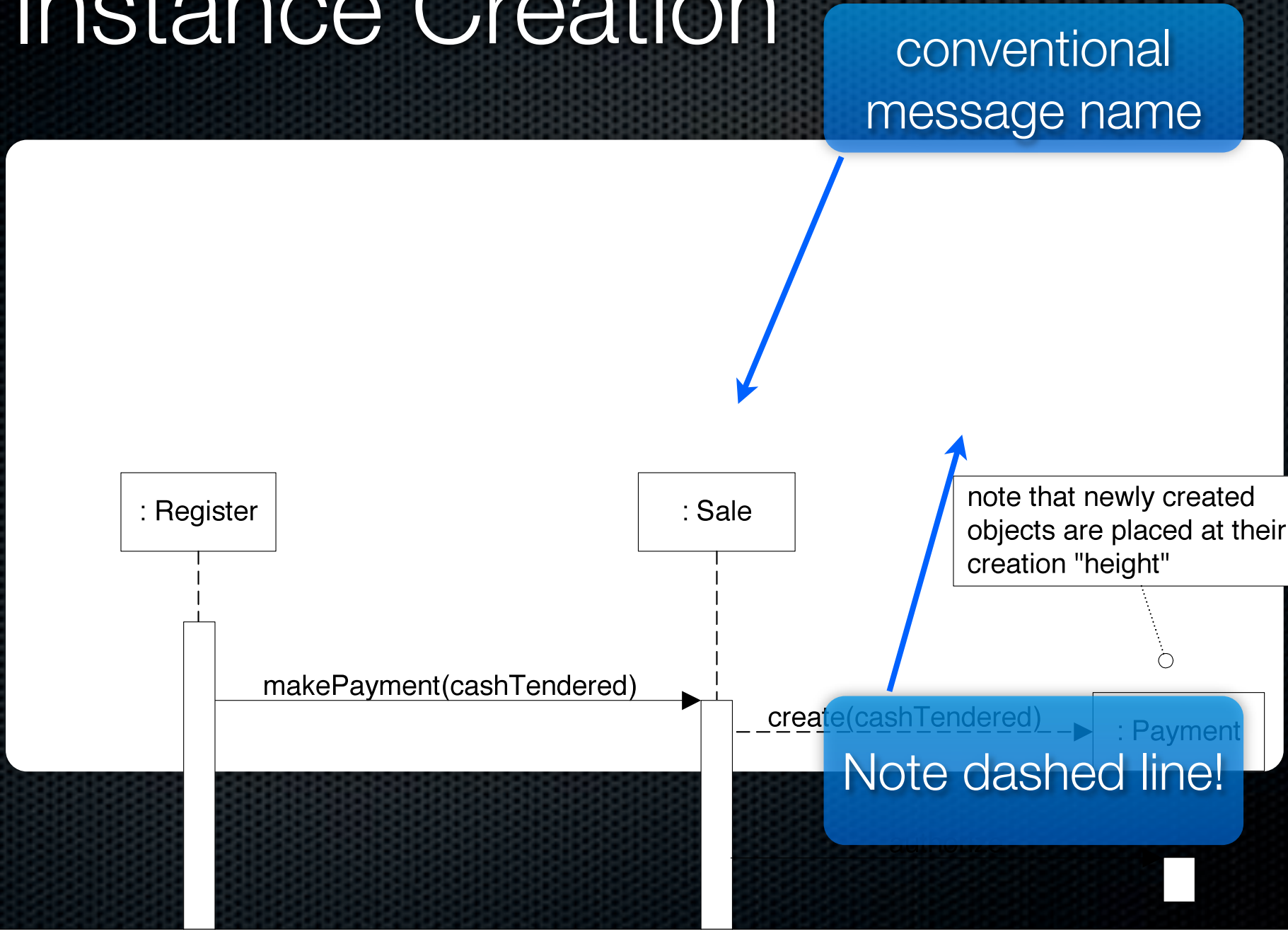
Two Ways of Illustrating Return Values



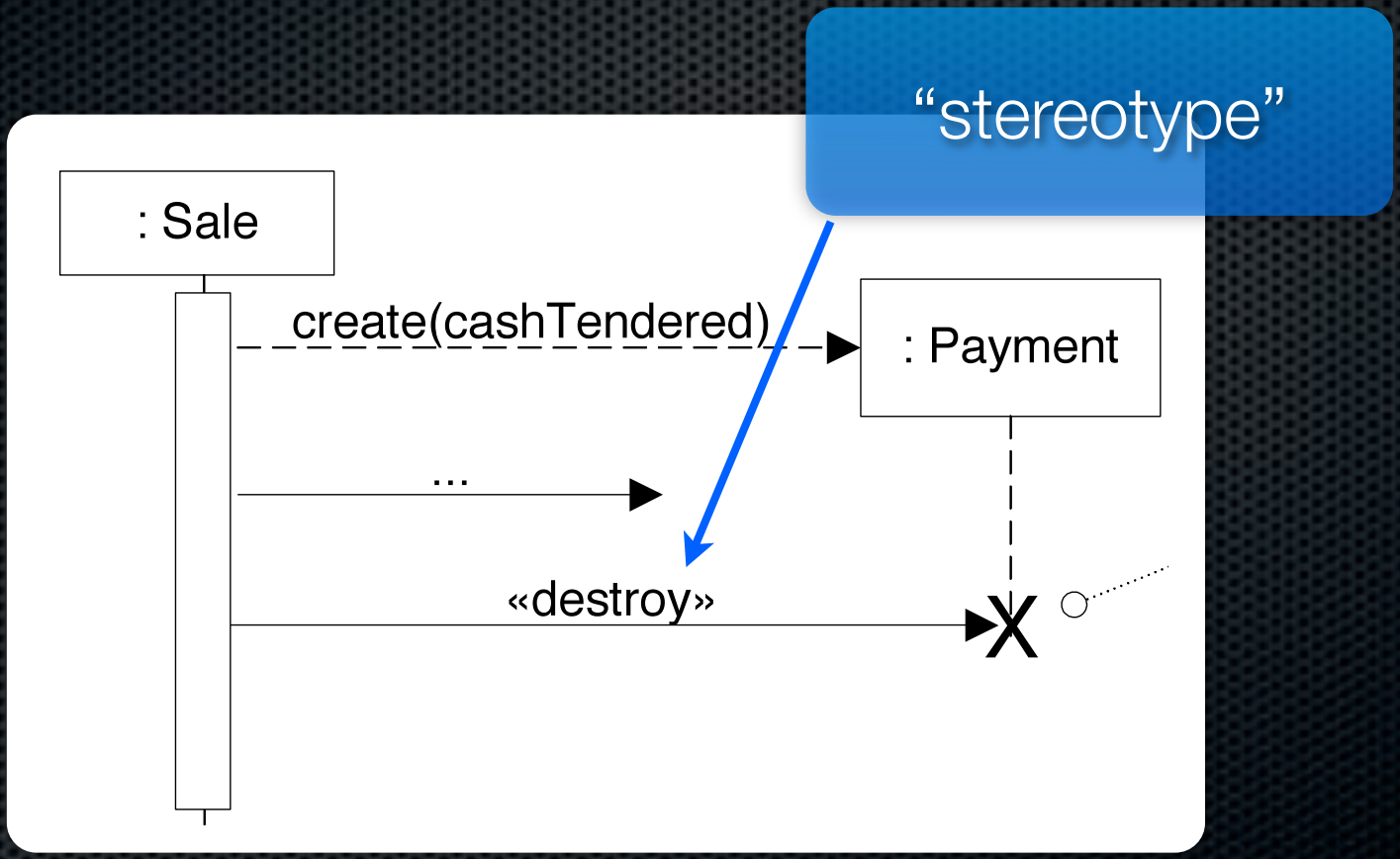
Messages to Self



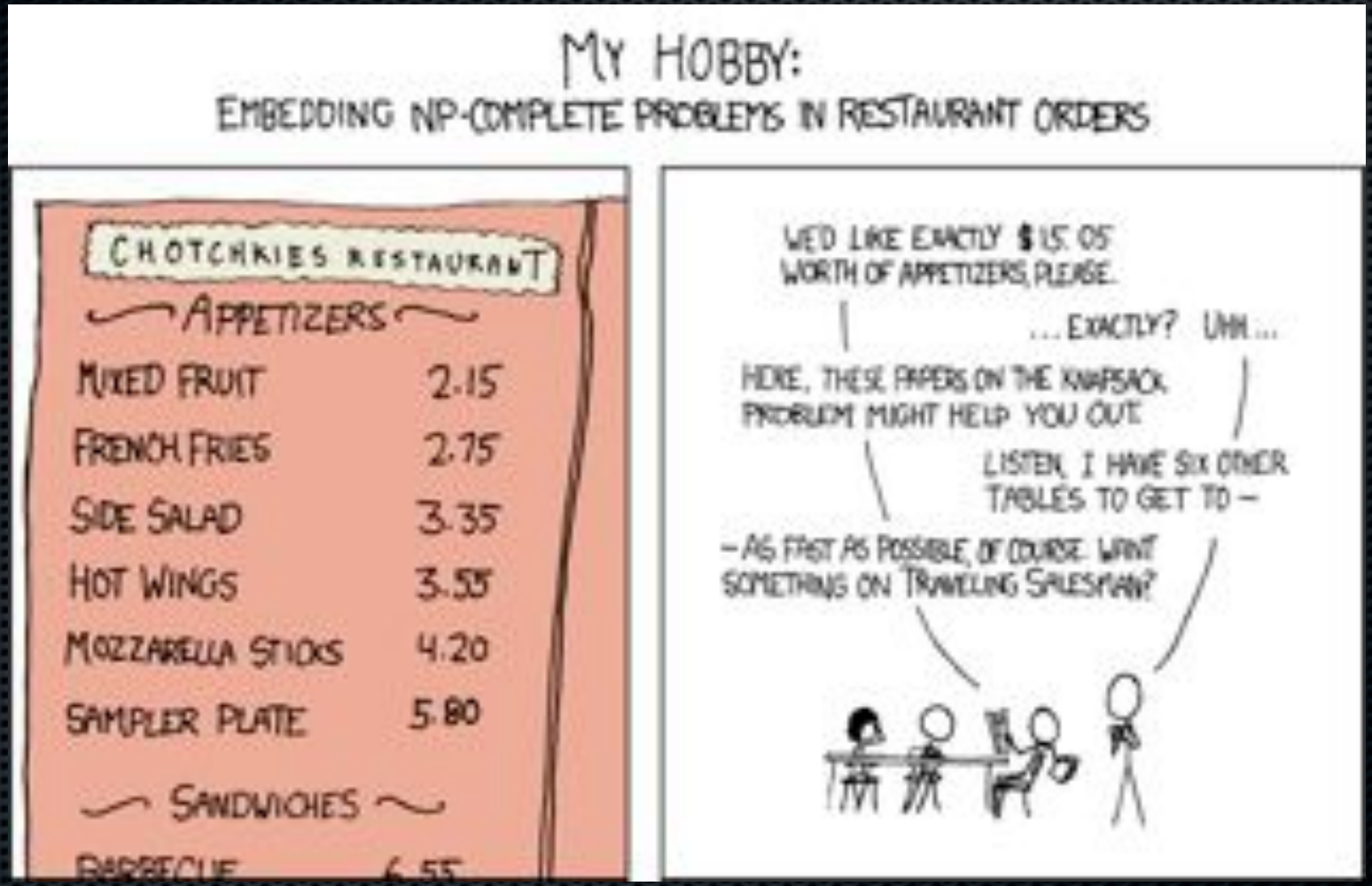
Instance Creation



Instance Destruction



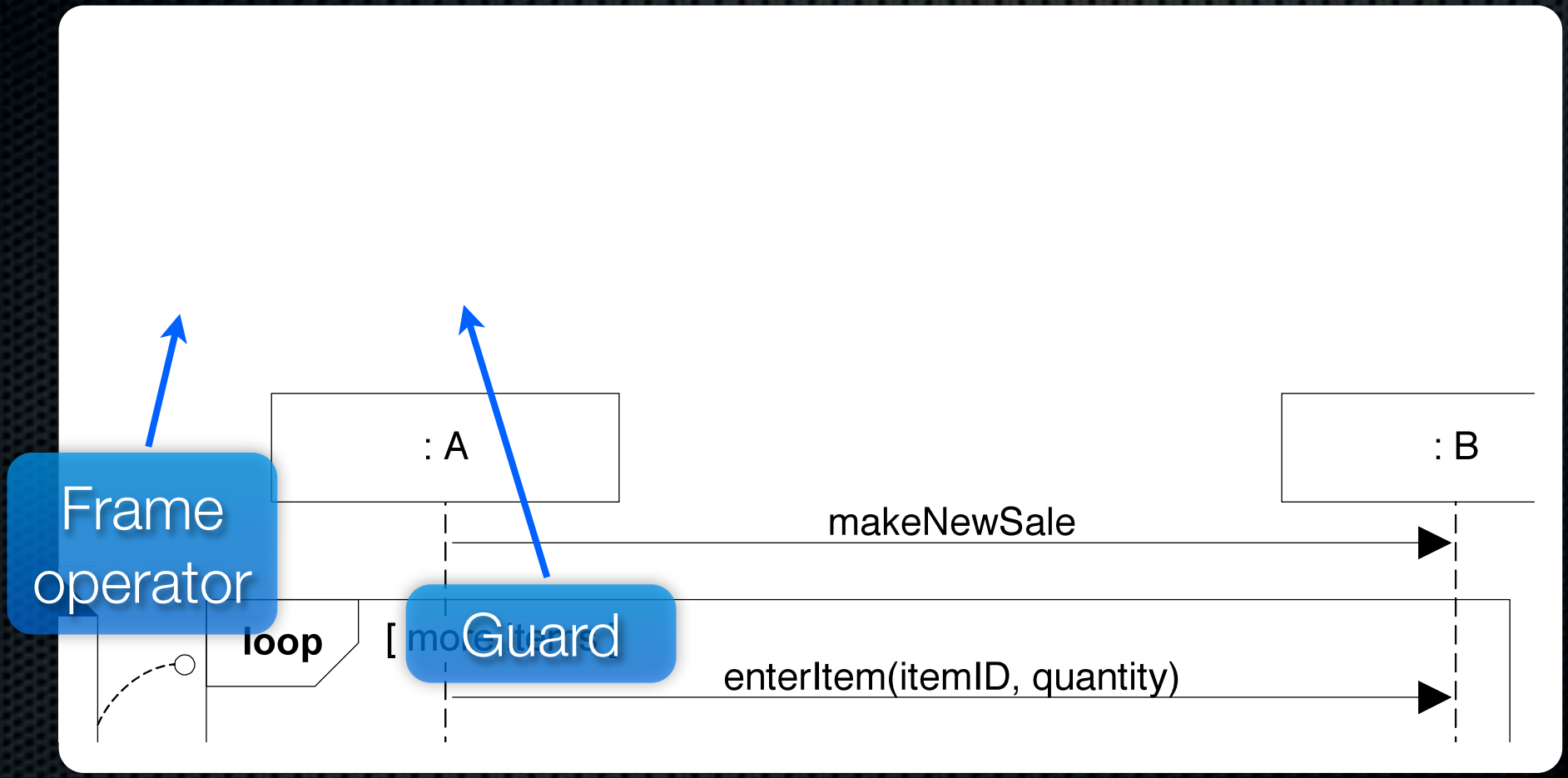
Cartoon of the Day



http://xkcd.com/287/

General solutions get you a 50% tip

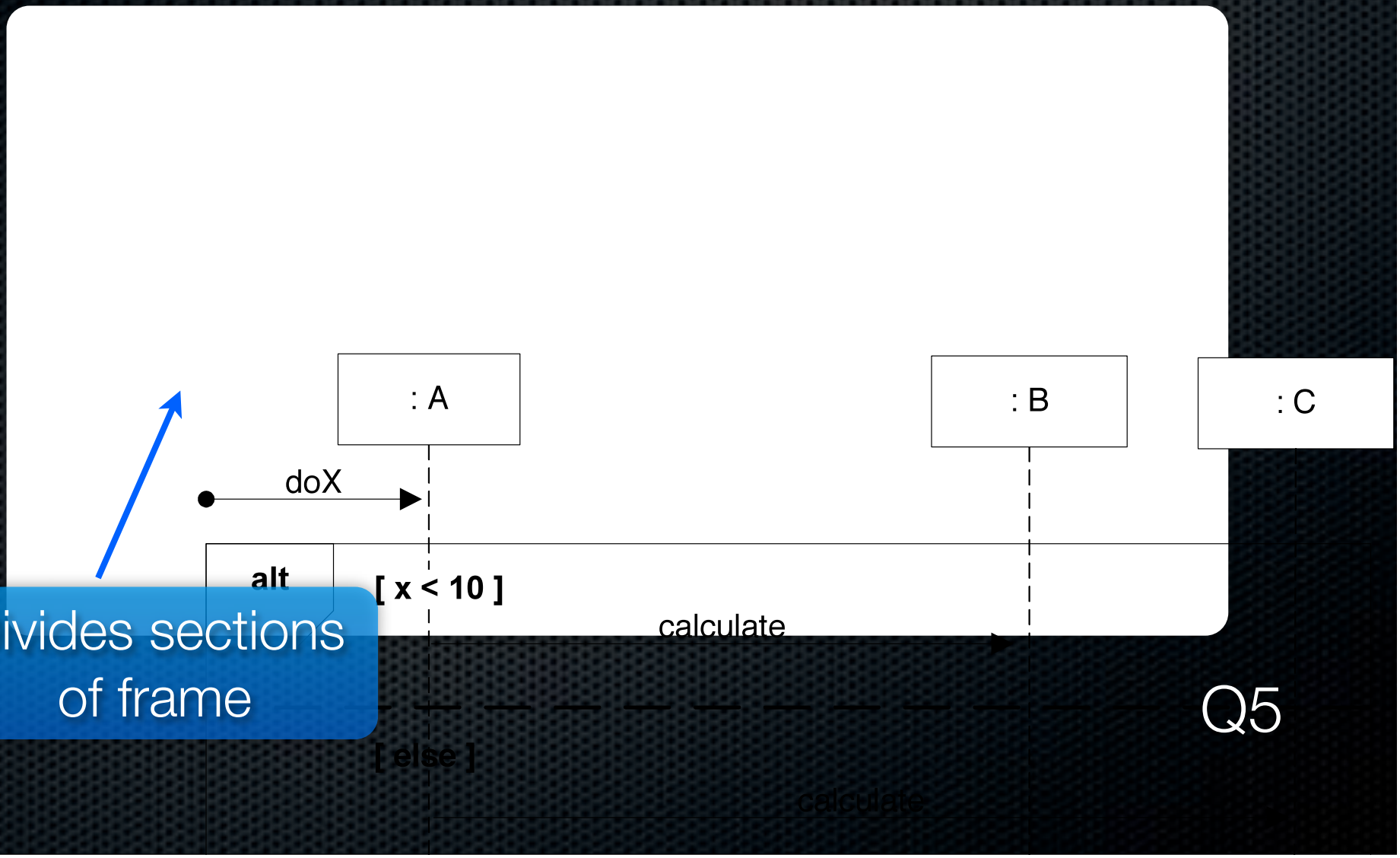
Speaking of Sales…
Recall Interaction Frames



Common Frame Operators

Operator	Meaning
alt	“alternative”, if-then-else or switch
loop	loop while guard is true, or *loop(n)* times
opt	optional fragment executes if guard is true
par	parallel fragments
region	critical region (single threaded)
ref	a “call” to another sequence diagram
sd	a sequence diagram that can be “called”

Mutual Exclusion “alt” Frame



Divides sections of frame

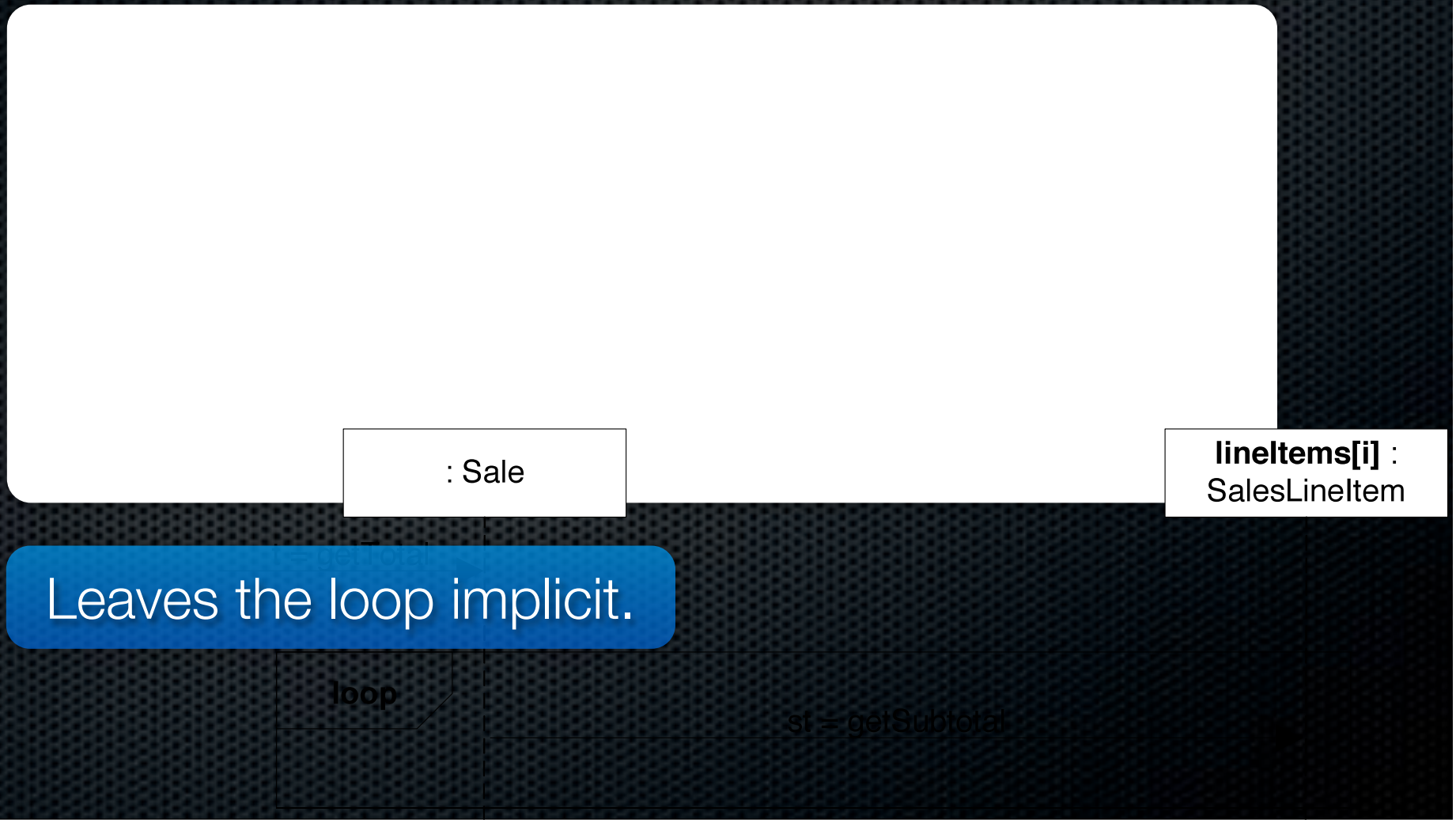
Q5

Iterating Over a Collection —Version 1

One instance from a collection

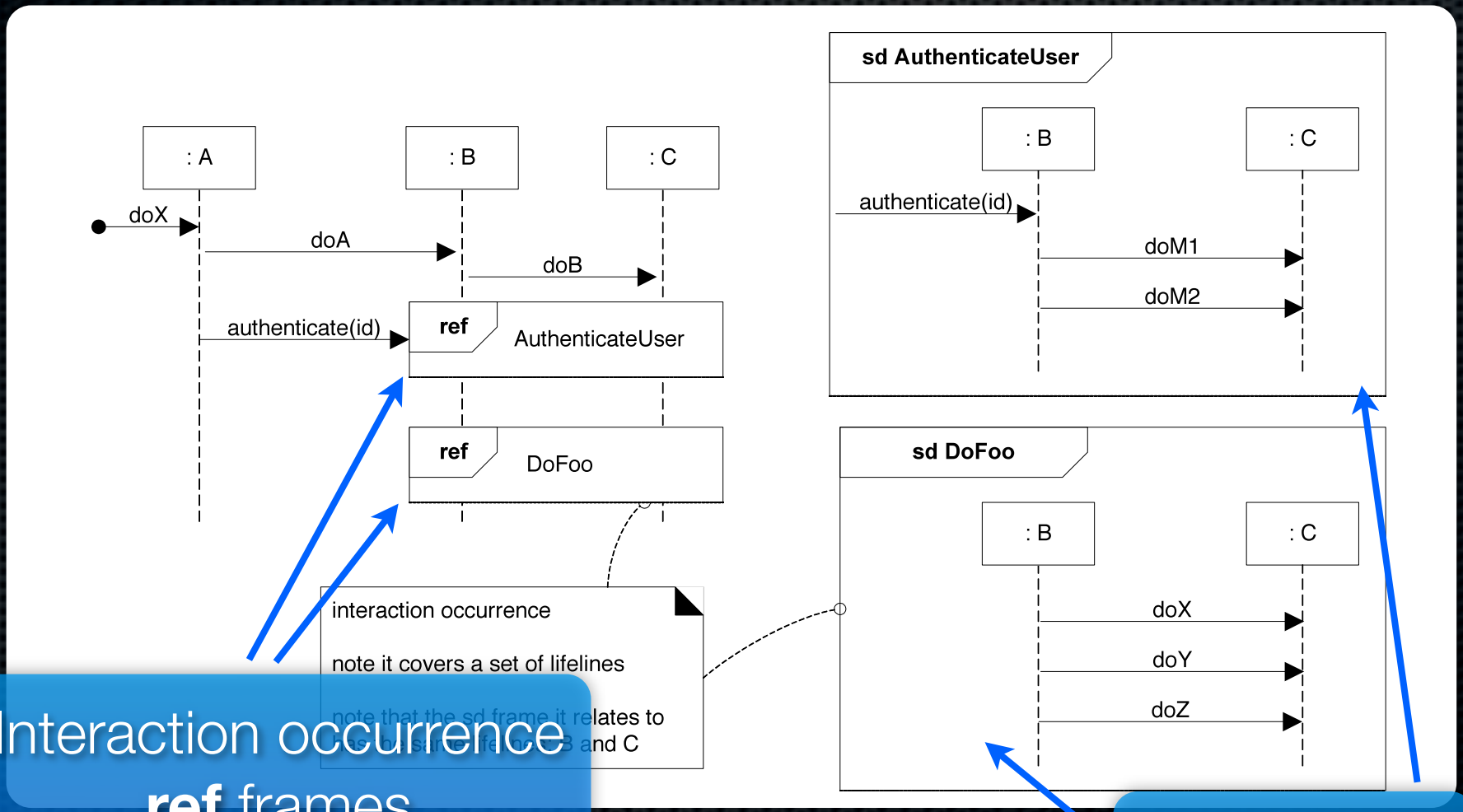
Action box contains arbitrary statements from implementation language

Iterating Over a Collection —Version 2



Leaves the loop implicit.

Abstracting Interaction



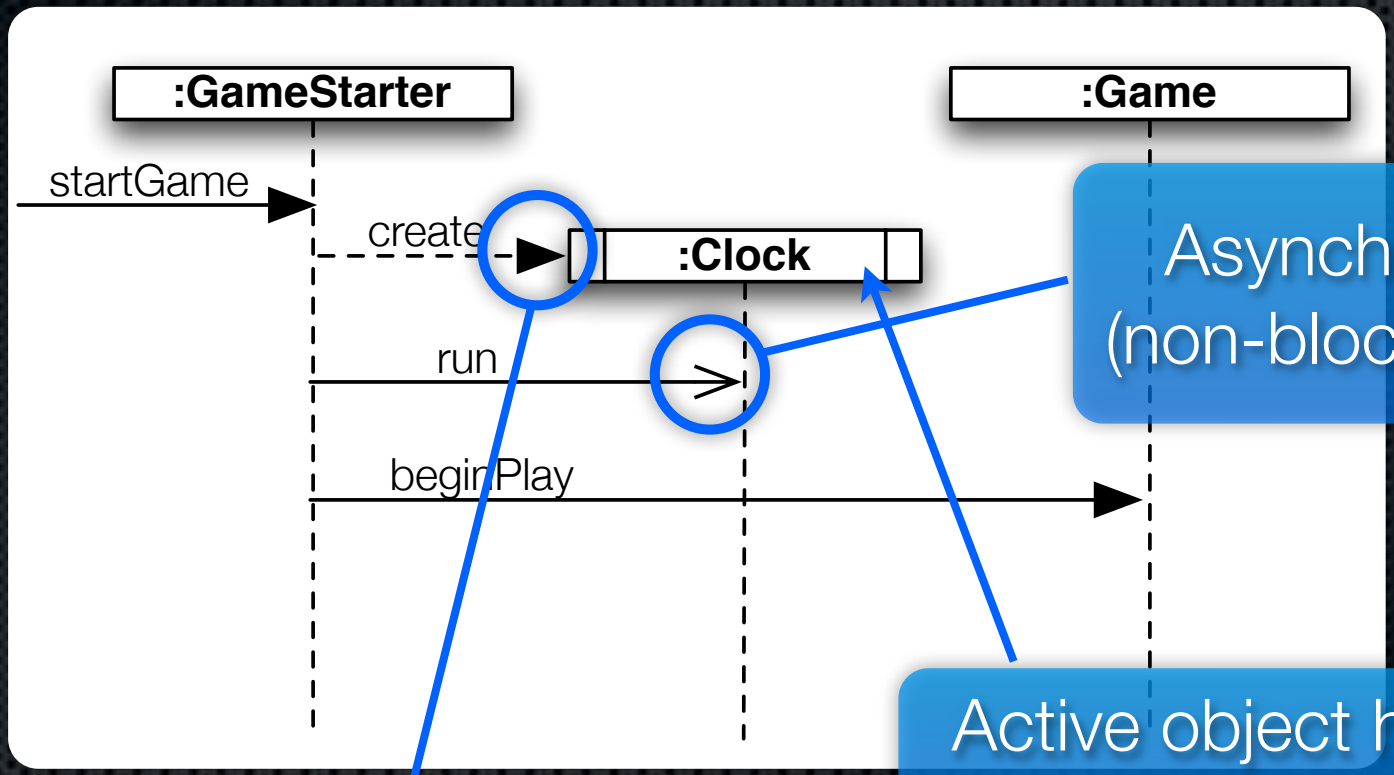
Interaction occurrence

ref frames

Q6

sd frames

Asynchronous Calls



Asynchronous (non-blocking) call

Active object has its own thread

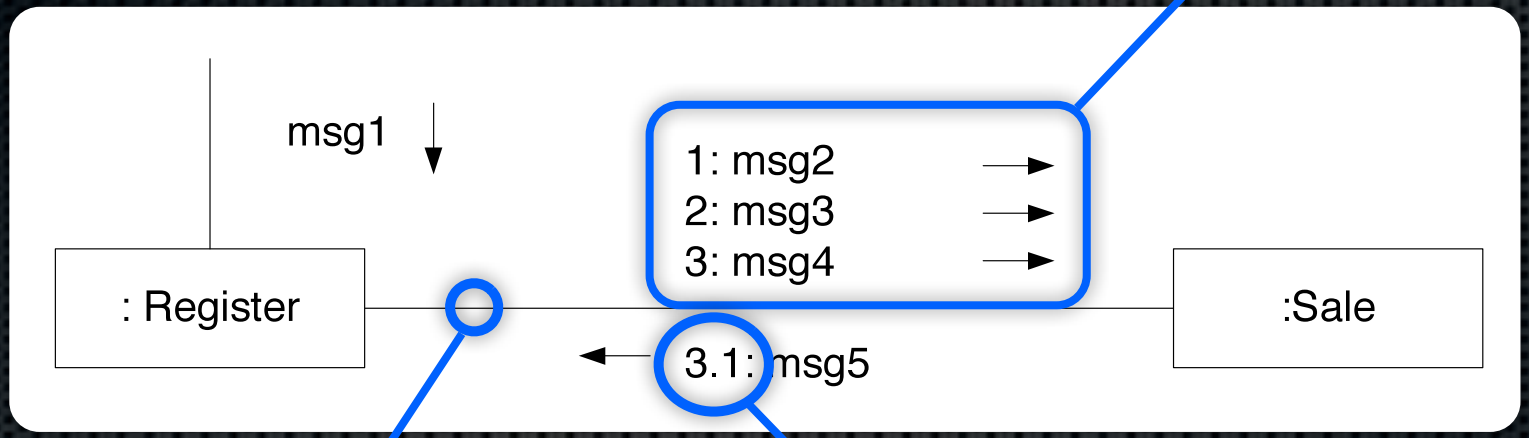
Synchronous (blocking) call

And now for something completely different

Communication Diagrams

Links vs. Messages

Multiple **messages** traverse links

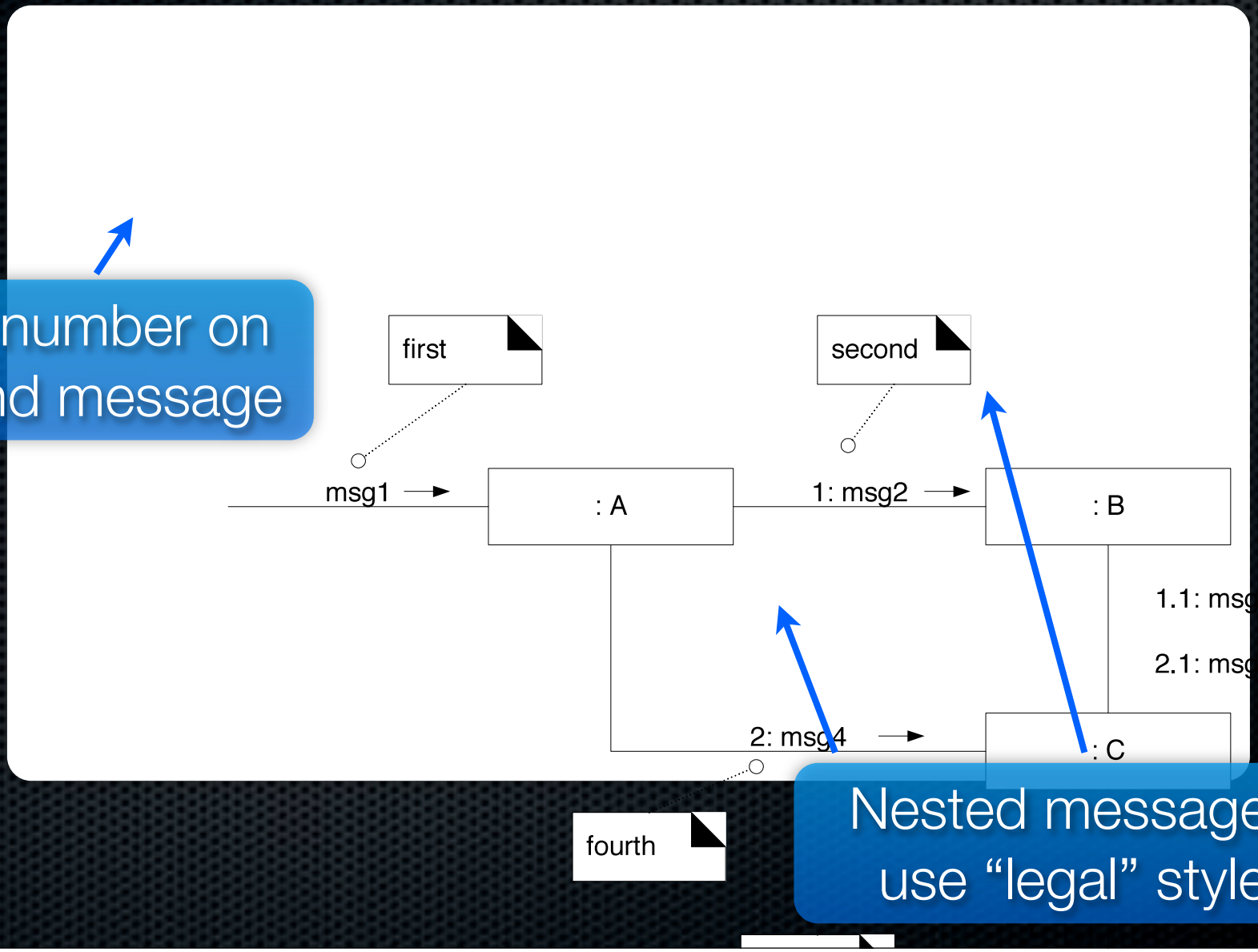


Single **link** connects two objects

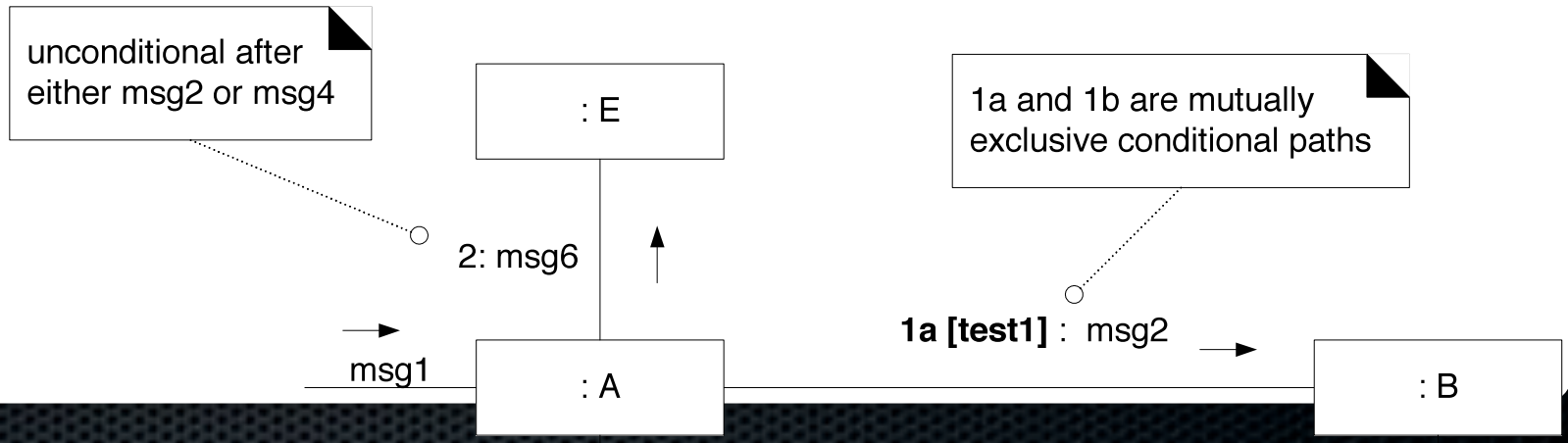
Sequence number gives ordering

Q8

Sequence Numbering

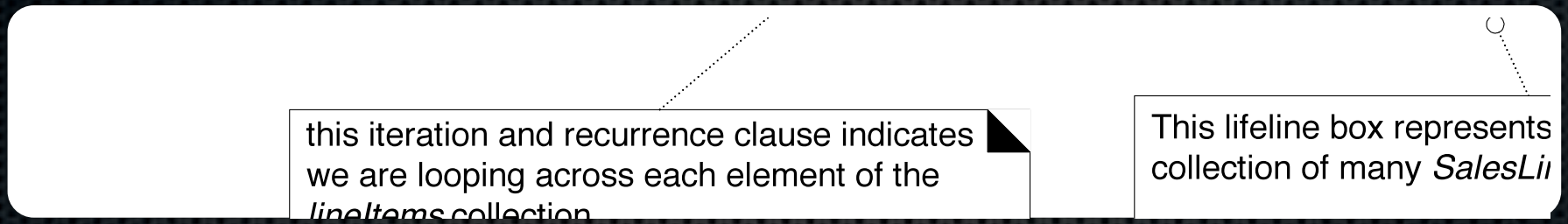


Conditional Messages Use Guards



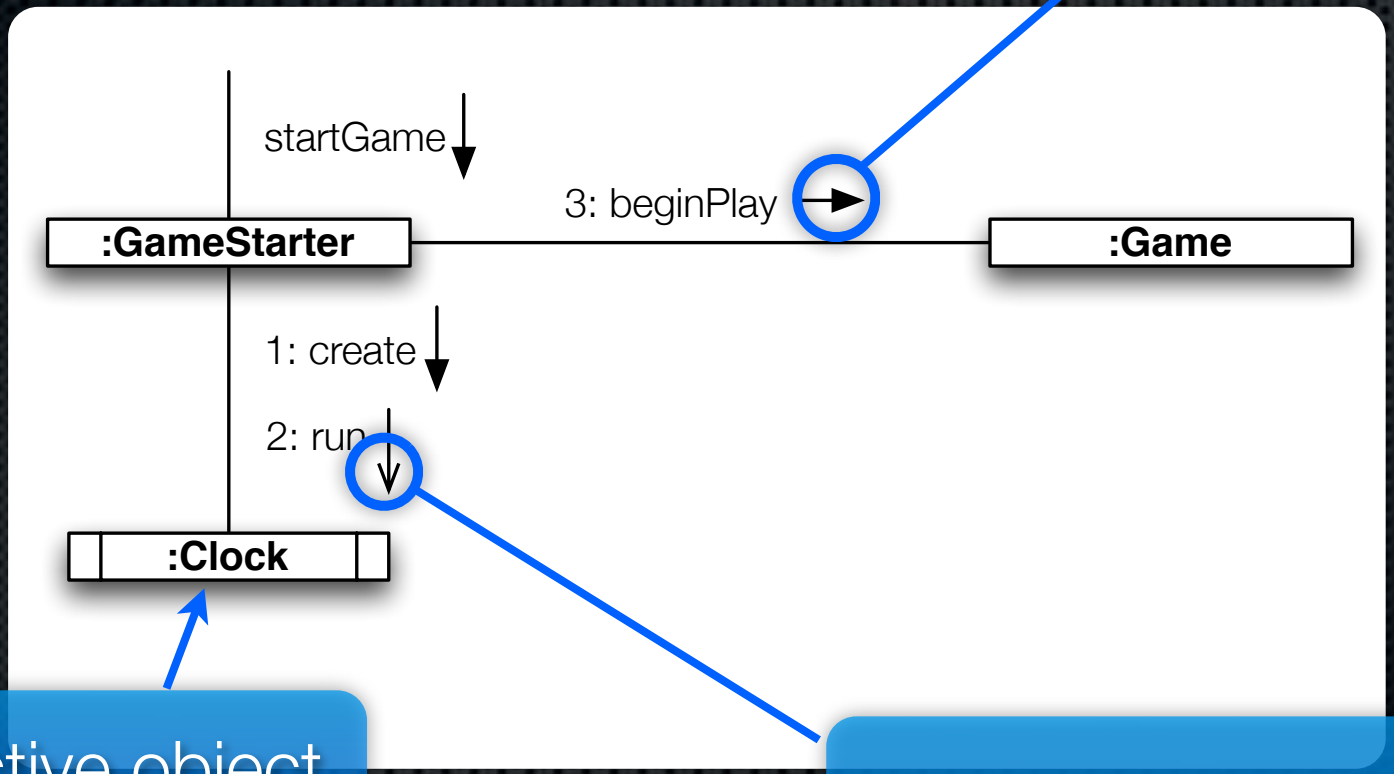
Q9a

Iteration Uses Stars



Q9b

Asynchronous Calls



Synchronous (blocking) call

Active object

Asynchronous (non-blocking) call

Q10