Inf2C-SEPP:
Lecture 10: Software component interactions. Sequence diagrams

Cristina Adriana Alexandru

School of Informatics
University of Edinburgh

Previous lectures

- Design
  - Architectural design
  - Detailed design
- Class diagrams
  - Notation
  - An approach for identifying objects and classes

This lecture

- Dynamic aspects of design
- Thinking about inter-object behaviour
- Sequence diagrams
- What is a good interaction pattern?

Dynamic aspects of design

Suppose that we have decided what classes should be in our system, provisionally. What next? Well, we have to meet the requirements...

In the end, we need to know what operations they have, and what each operation should do.

Two ways of looking at this:

1. inter-object behaviour: who sends which messages to whom?
2. intra-object behaviour: what state changes does each object undergo as it receives messages, and how do they affect its behaviour?

In this course, we only consider 1.

For 2, UML provides state diagrams, enhanced FSMs

- Covered by SDM course

Thinking about inter-object behaviour

There's no algorithm for constructing a good design. Create one that's good according to the design principles...

1. Your classes should, as far as possible, correspond to domain concepts.
2. The data encapsulated in the classes is usually pretty easy to define using the real world as a model.
3. Then look at the scenarios in the use cases, and work out where to put what operations to get them done.

Can get hard when several objects have to collaborate and it isn't clear which should take overall responsibility.

CRC Cards can help.

Another approach: interaction diagrams

Interaction diagrams

Describe the *dynamic* interactions between objects in the system, i.e. the pattern of message-passing (see next slide).

Good for showing how the system realises [part of] a use case

Particularly useful where the flow of control is complicated, since this can't be deduced from the class model, which is static.

UML has two sorts: sequence and communication diagrams.

In this course, we discuss sequence diagrams.

OO message-passing terminology

```
class A {
   f() {
      B b = ... ;
      ...
      b.g();
      ...
   }
}
```

```
class B {
  g();
}
```

Let a be some object of type A.

- Object a sends a (call) message g to object b
  - An invocation a.f() makes a call b.g()
- Object b sends a reply message to object a
  - An invocation b.g() finishes and control flow returns to a.f()

Developing a sequence diagram

1. Decide exactly what behaviour to model.
2. Check that you know how the system provides the behaviour: are all the necessary classes and relationships in the class model?
3. Name the objects which are involved.
4. Identify the sequence of messages which the objects send to one another.
5. Record this in the syntax of a sequence diagram.

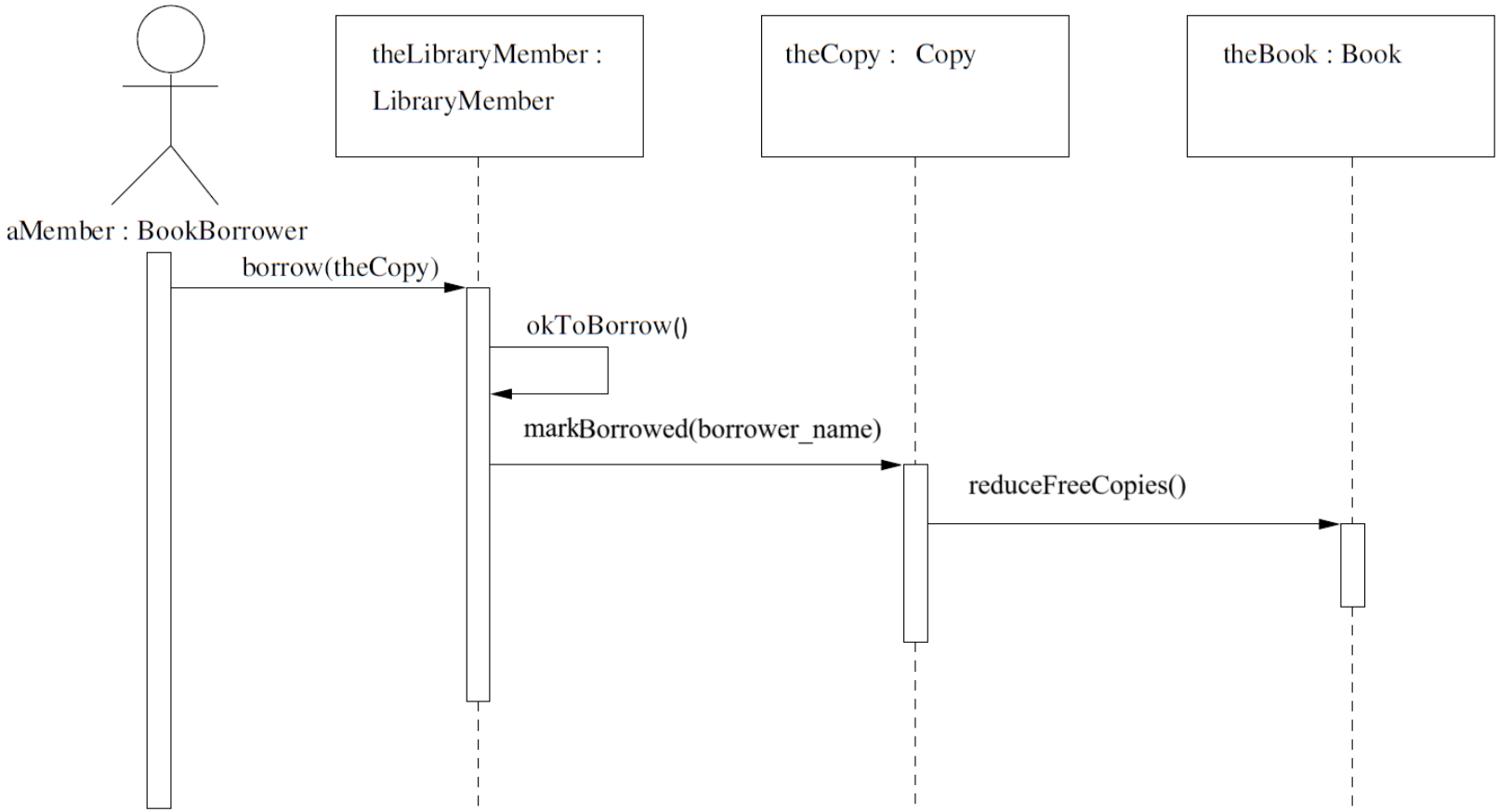
Example: A use case for a library system

Title: Borrow book

Primary Actor: BookBorrower

Description of main success scenario (MSS): A book borrower presents a copy of a book to the system. Assuming the borrower has not already checked out some maximum number of books, the system permits the loan, recording who the book is checked out to, and noting that the number of free copies of the book is reduced.

Sequence diagram for the use case MSS: high level view



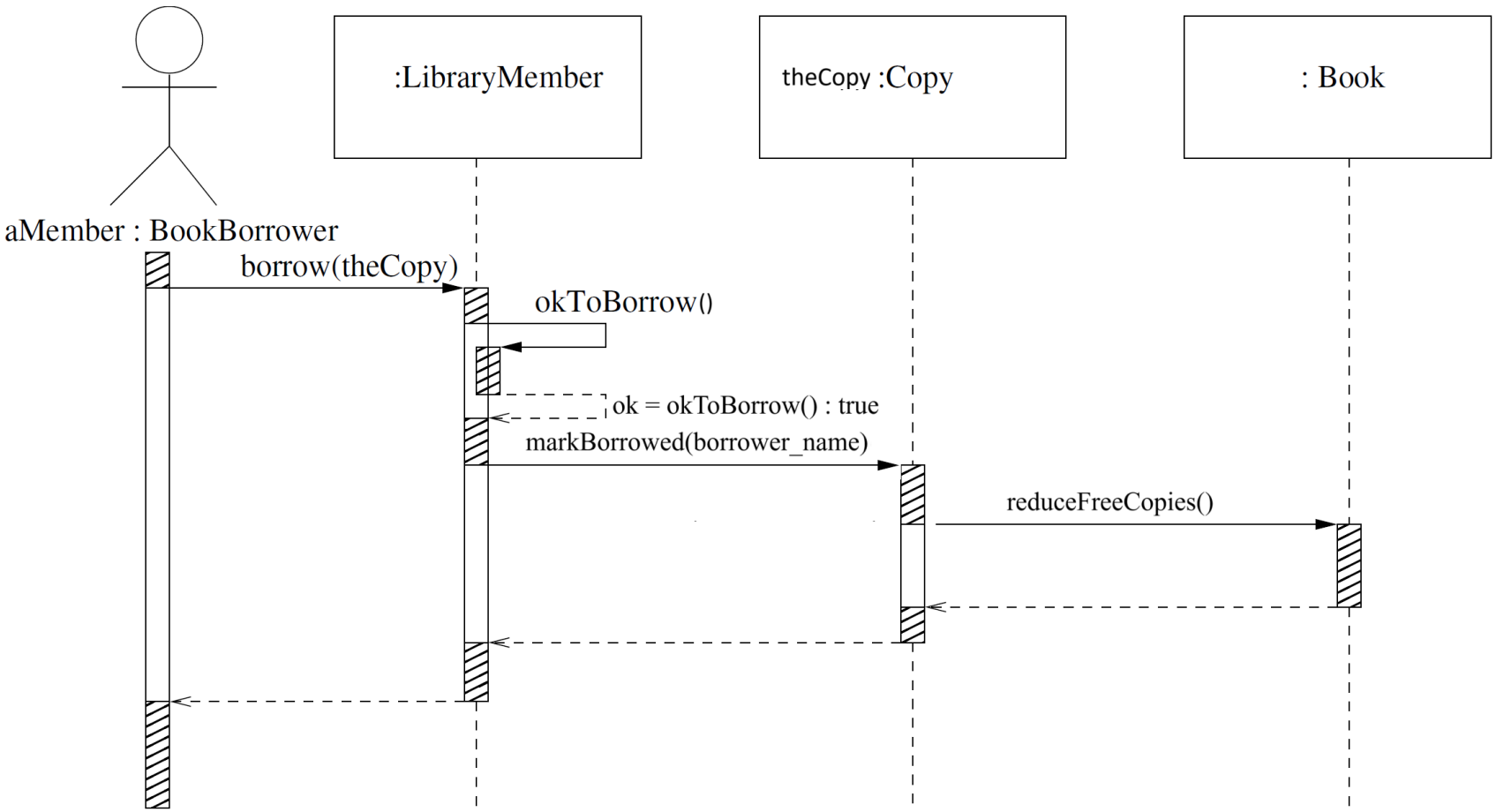
Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

Sequence diagram for the use case MSS: high level view

Notation:

- *Instance of* actor (not actor!) as stick figure with name: rolename
- Objects (not classes!) as name: classname in rectangles
- Timelines as dashed lines from actor instance/object facing downwards, indicating actor instance/ object being alive
- Activation bars as rectangles on top of timelines showing when actor instances/ objects are active (while performing operation)
- Message calls as arrows with filled head and labelled as operationname(parameters) going from the caller (of the message/ operation) to the receiver (of the message/ handler of operation).

Sequence diagram for the use case MSS: Showing more detail



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

Sequence diagram for the use case MSS: Showing more detail

New notation:

- Actor instances/ objects with any name if name left out (:rolename or :classname)
- Shading on activation bars to show when object is computing
- Return messages as dashed arrows with v-shaped arrowheads and including what is returned or nothing if return is void.

More complex sequence diagrams

The examples above showed interaction in the main success scenario.

Sequence diagrams can be used to show all of a use case's scenarios at the same time.

UML provides further notation for e.g.

- conditional behaviour
- iterative behaviour
- including one diagram in another
- concurrent behaviour

In this course, we will look at the first two.

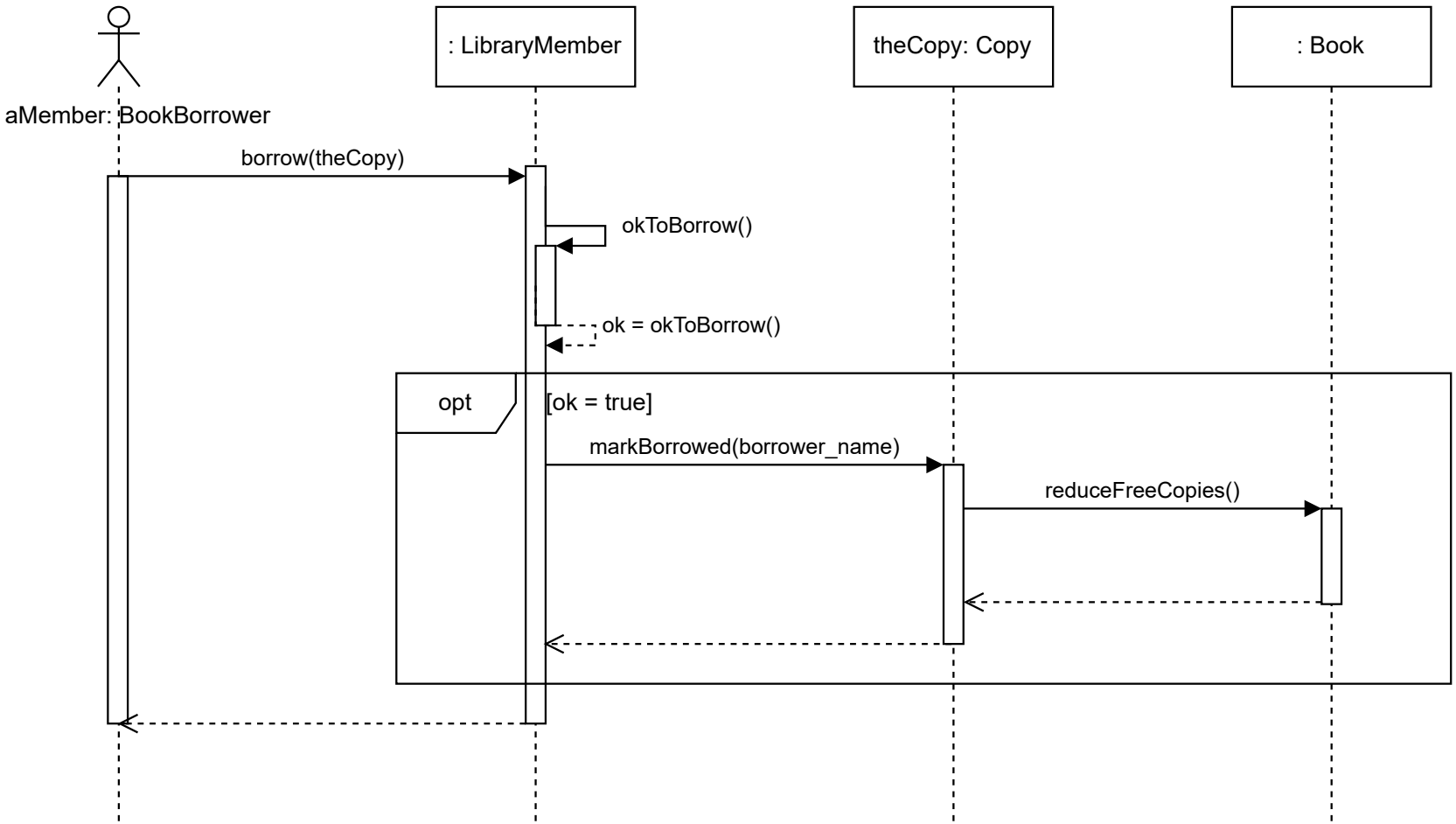
They are both based on the idea of *sequence frames* around involved object timelines.

Conditional behaviour in sequence diagrams

Split into two expressive notations:

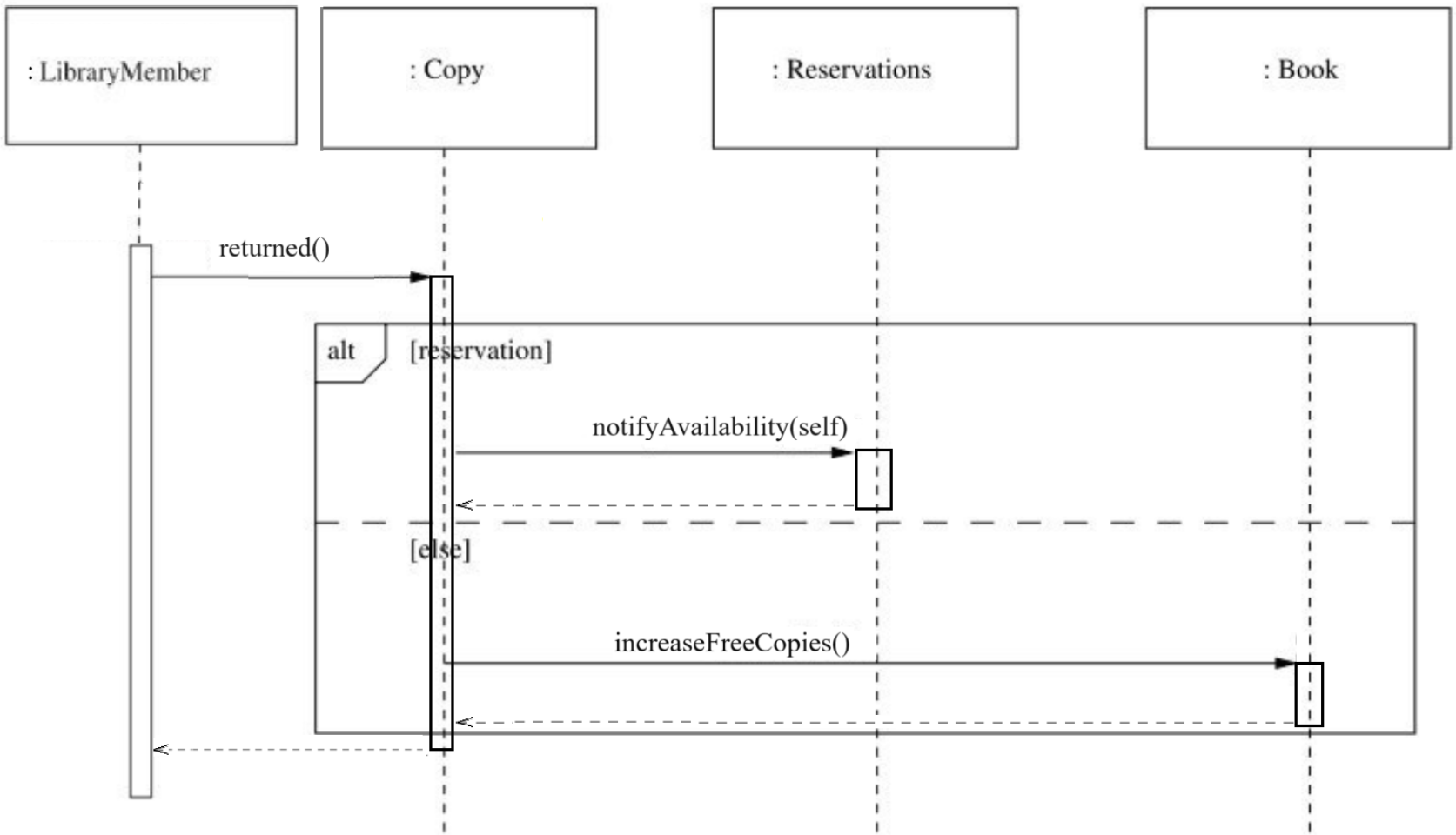
- *Optional behaviour* (used to represent if statements), using an 'opt' frame and a *guard condition*
- *Alternative behaviour* (used to represent if/then/else statements), similar but using an 'alt' frame with different compartments for the different conditions

Optional behaviour in sequence diagrams



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

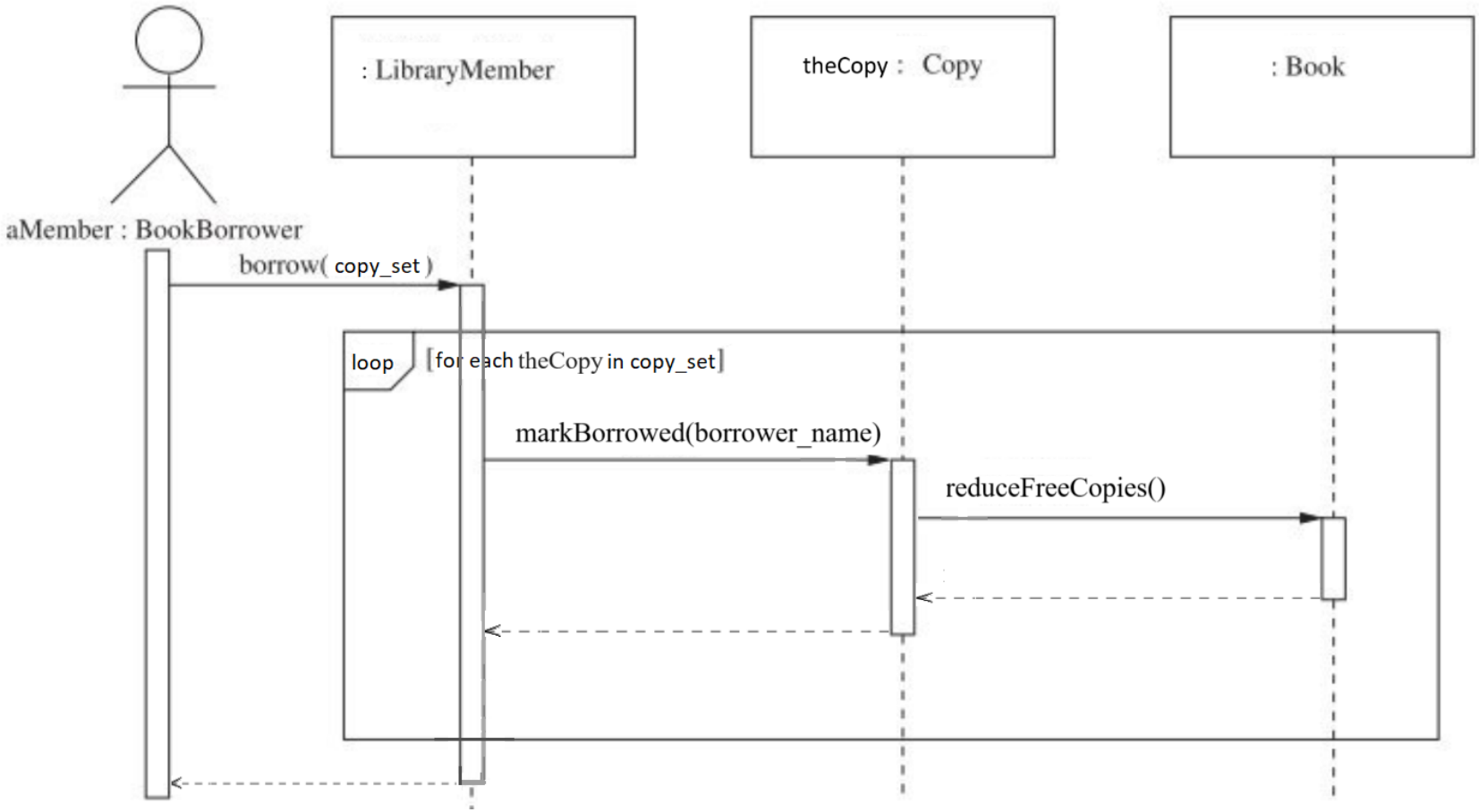
Alternative behaviour in sequence diagrams



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

Iterative behaviour in sequence diagrams

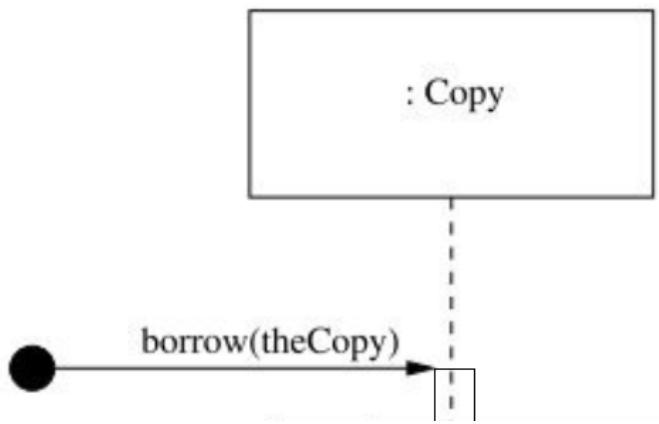
Used to represent for/while/do-while statements, using a 'loop' frame and a *loop condition*



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

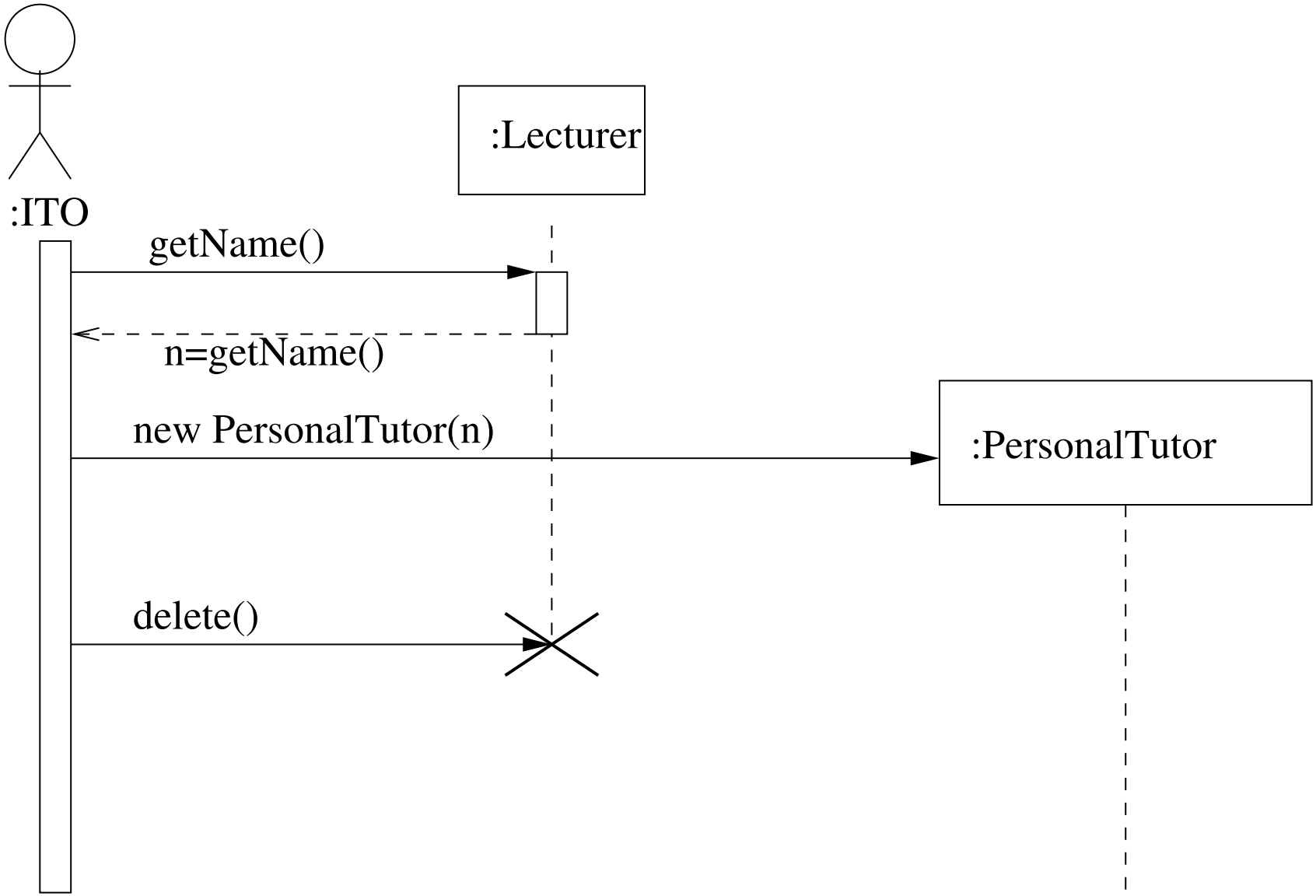
Found messages in sequence diagrams

Notation for a message when who initiates it is not made explicit (i.e. it is not specified as an actor instance or object).



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

Creation/deletion in sequence diagrams



Adapted from: Stevens, P. and Pooley, R.J., 2006. Using UML: software engineering with objects and components. Pearson Education.

What is a good interaction pattern?

In designing an interaction, your first aim is obviously to design *some* collection of operations that can work together to achieve the aim.

Next, consider:

- conceptual coherence: does it make sense for this class to have that operation?
- maintainability: which aspects might change, and how hard will it be to change the interaction accordingly?
- performance: is all the work being done necessary?

Reading

Essential: Stevens

- Ch 9: Essentials of sequence diagrams
- Ch 10: section 10.1, for conditional and iterative behaviour

Suggested: At least one of

- The original paper on CRC cards, a technique for designing interactions: *A laboratory for teaching object oriented thinking*, by Kent Beck and Ward Cunningham.
- Stevens Chapter 5 section 5.6 on CRC cards

CRC cards are covered in detail in SDM.