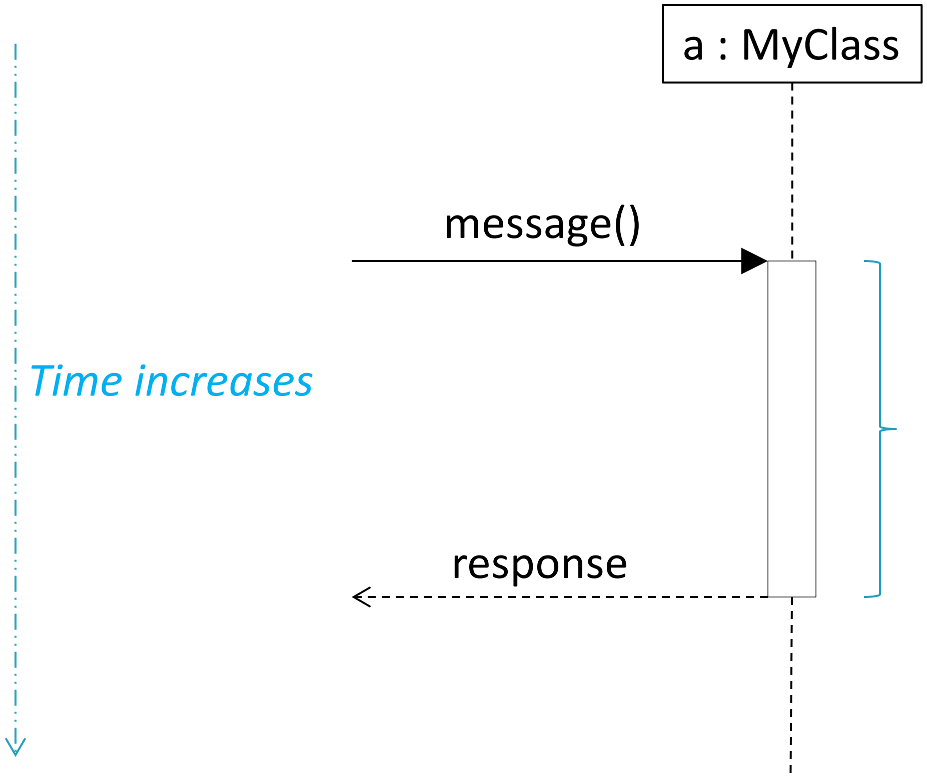
Software Engineering 301:
Software Analysis and Design

Behavioural modelling: Sequence diagrams

Sequence diagrams

- Show sequences of interaction between objects
  - Basic structured programming constructs supported: sequence, choice, loop
- Show messages being passed between objects

Simplest diagram

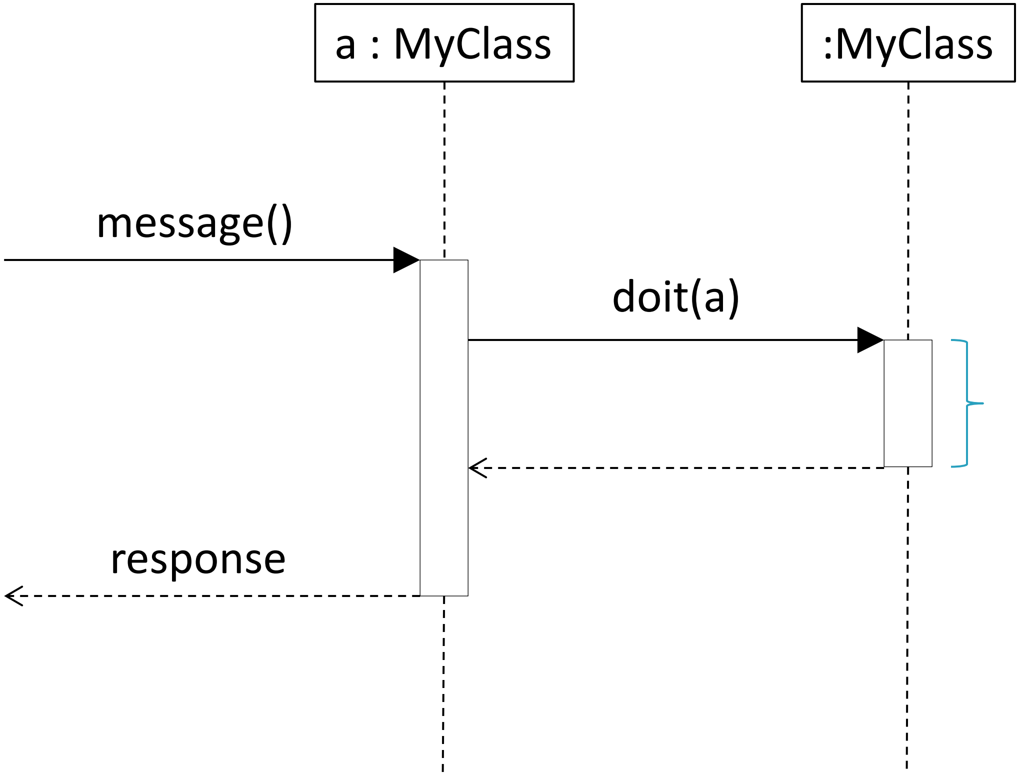


“a” exists at the start of this diagram (it is not explicitly created)

“a” is “active” here, meaning it is doing something (possibly waiting, in general cases)

“a” still exists at the end of this diagram (it is not explicitly destroyed)

Simple diagram

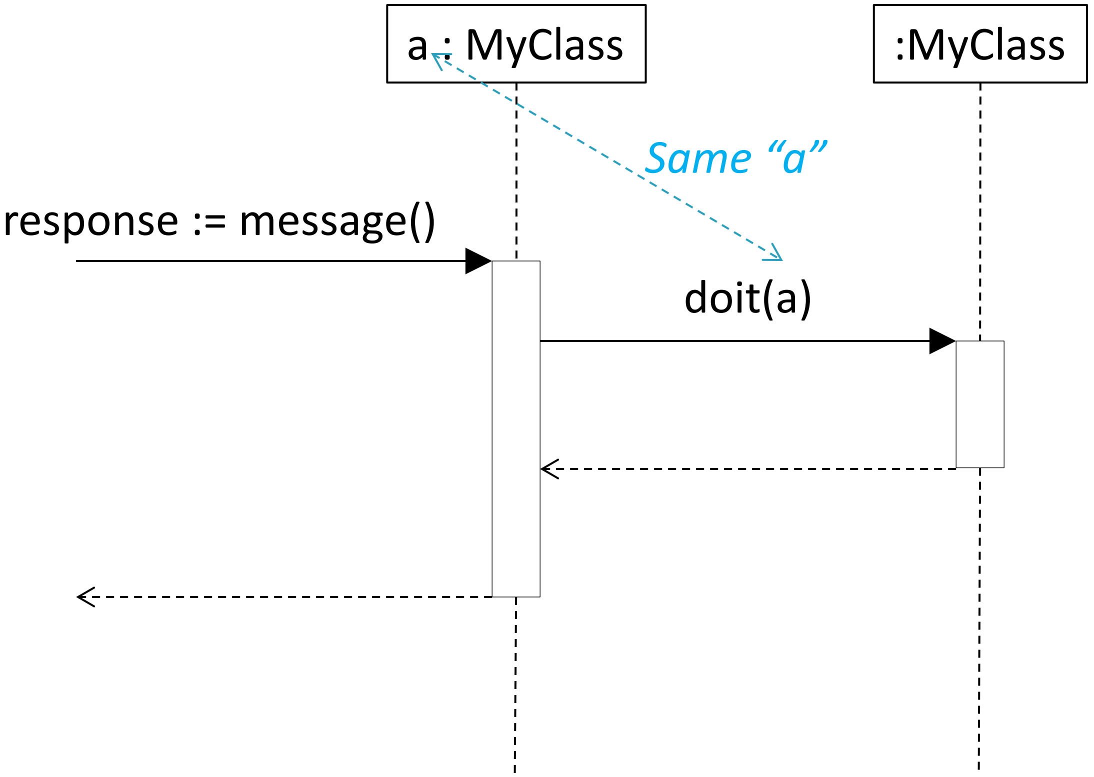


This role is anonymous (no name).

The anonymous role is active here; so is a, but a is waiting for the return. a is pushed down the stack.

The return causes the end of the activation.

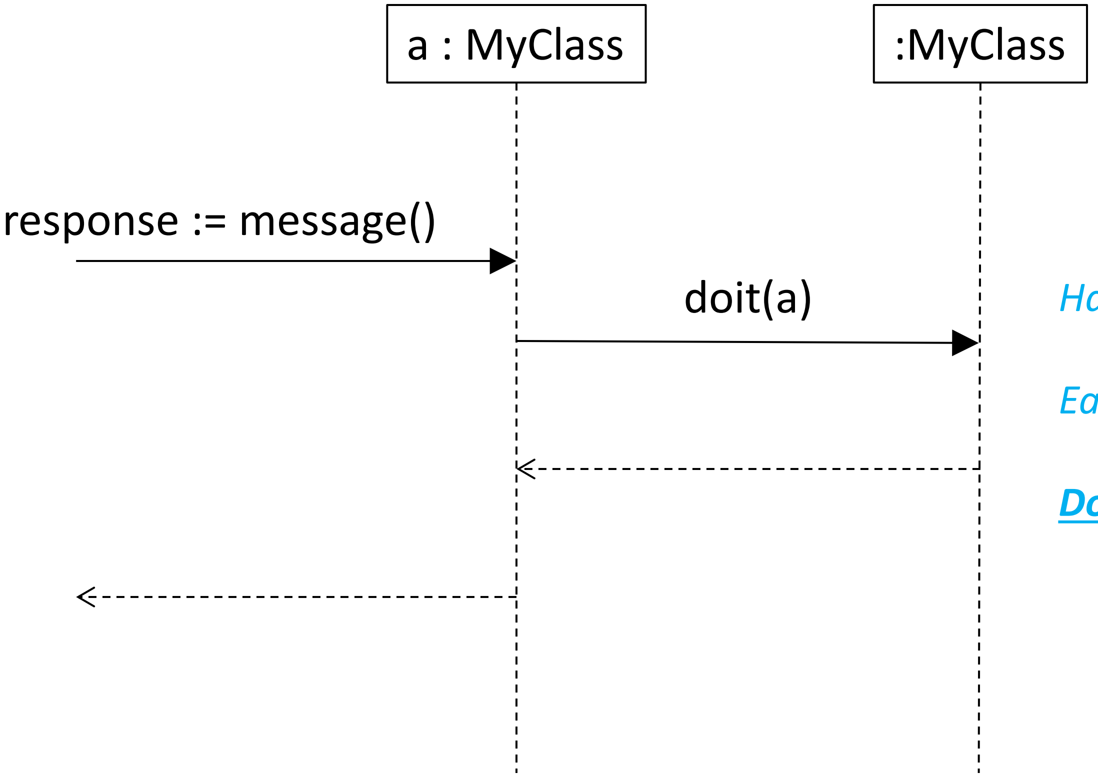
Simple diagram



Same "a"

Slightly different syntax for the return value.

Alternative syntax



Harder to reason about.

Easier to make mistakes.

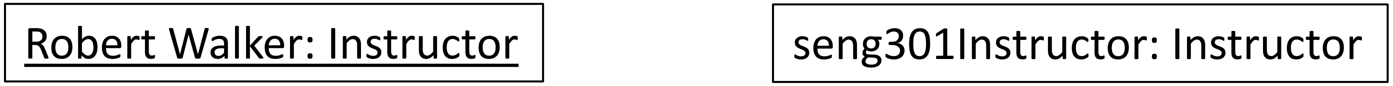
Don't use this in the course.

Objects versus roles

- An object in a diagram (any UML diagram) represents a specific object
- If we need to model a generic object, we can use a role

Object:

Role:

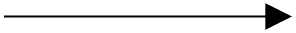
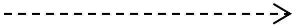
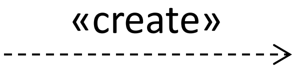


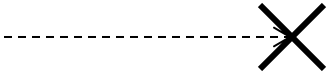
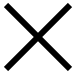
Robert Walker: Instructor

seng301Instructor: Instructor

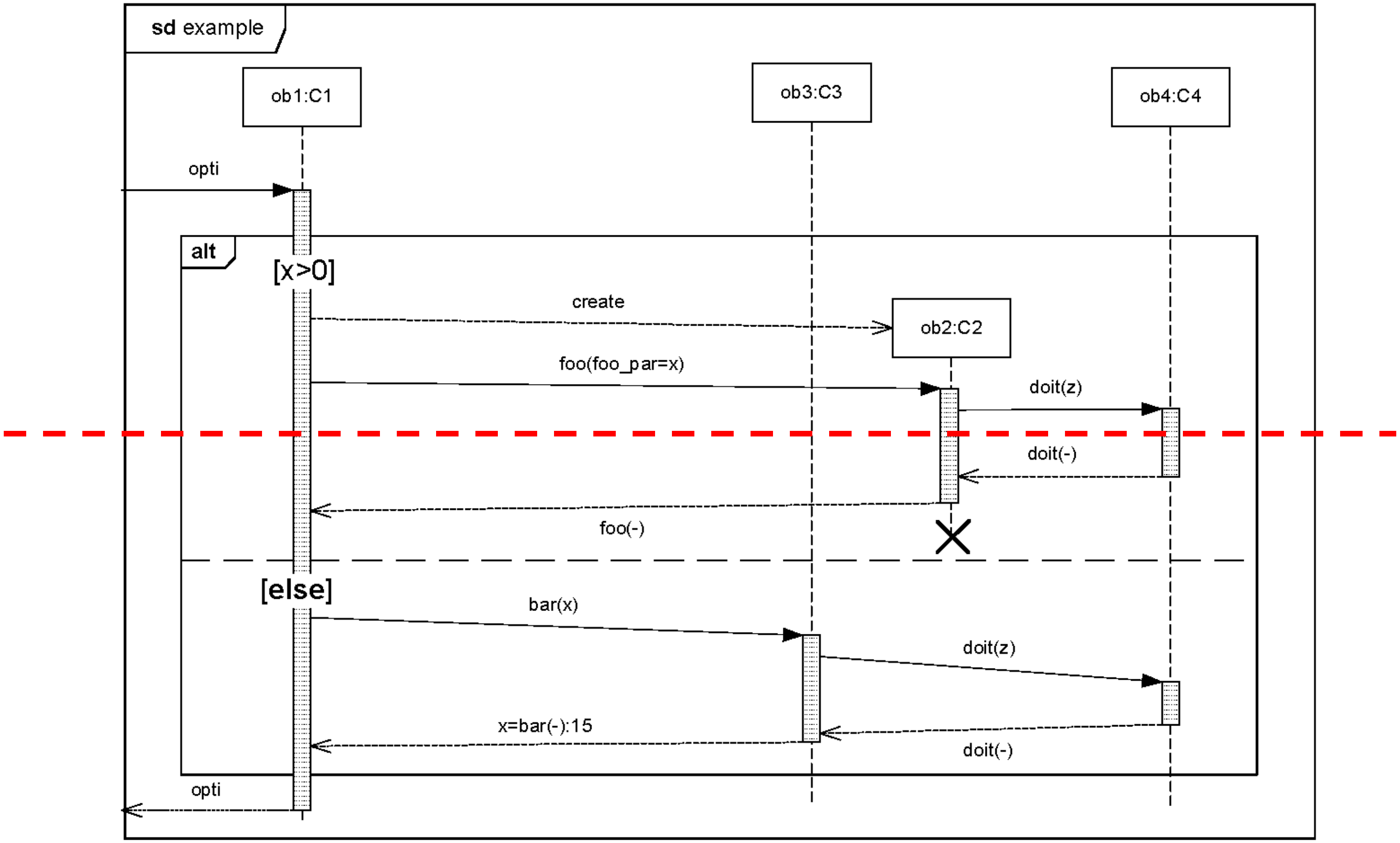
- Note that the colon is always present, so these are not classes

Message kinds

- Synchronous call 
  - The normal case
- Return 
  - Also normal, for use in conjunction with synchronous
- Object creation 

- Object destruction
  - explicit 
  - implicit (no message is sent) 
- Other kinds exist, but beyond scope of course

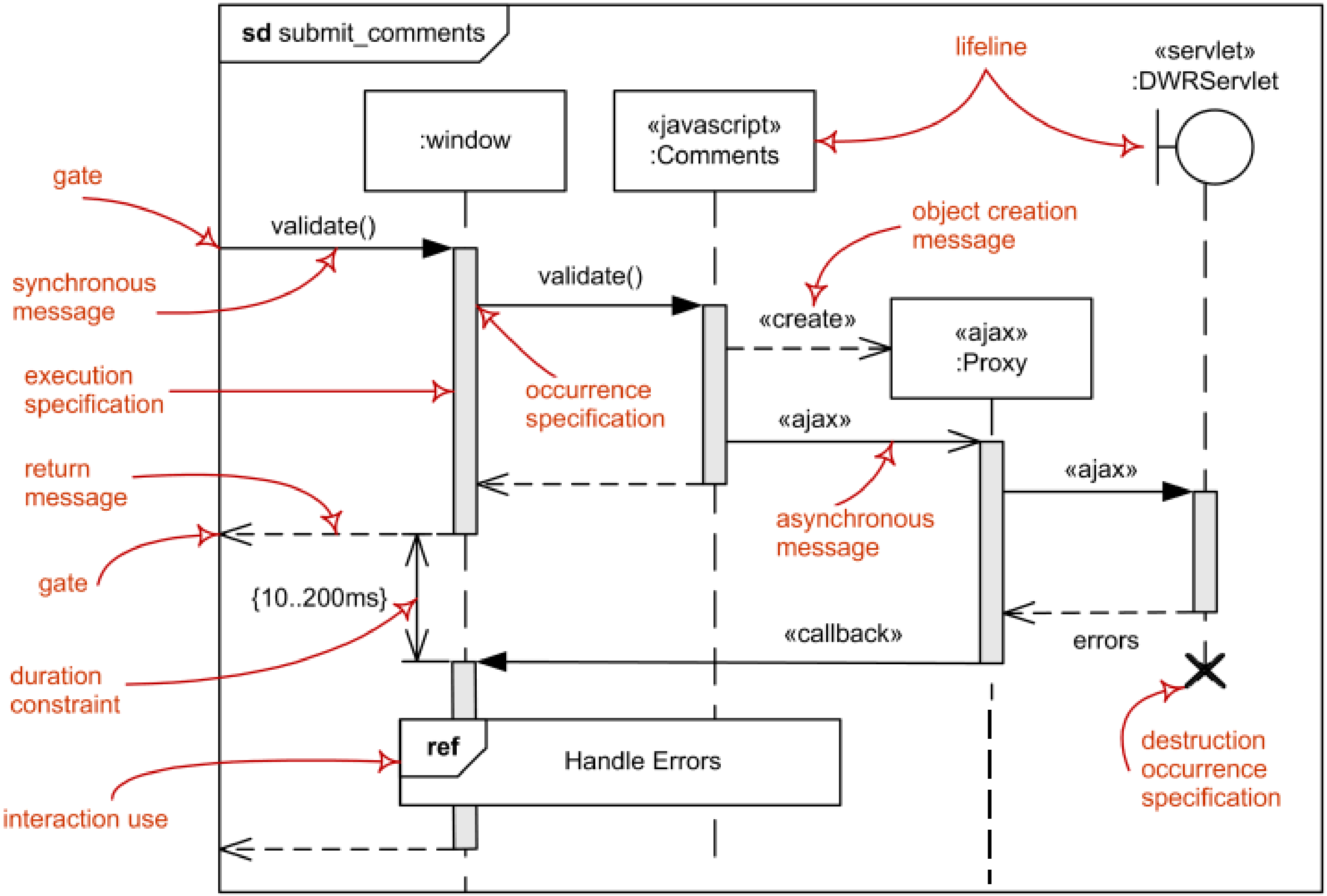
Sequence diagram example



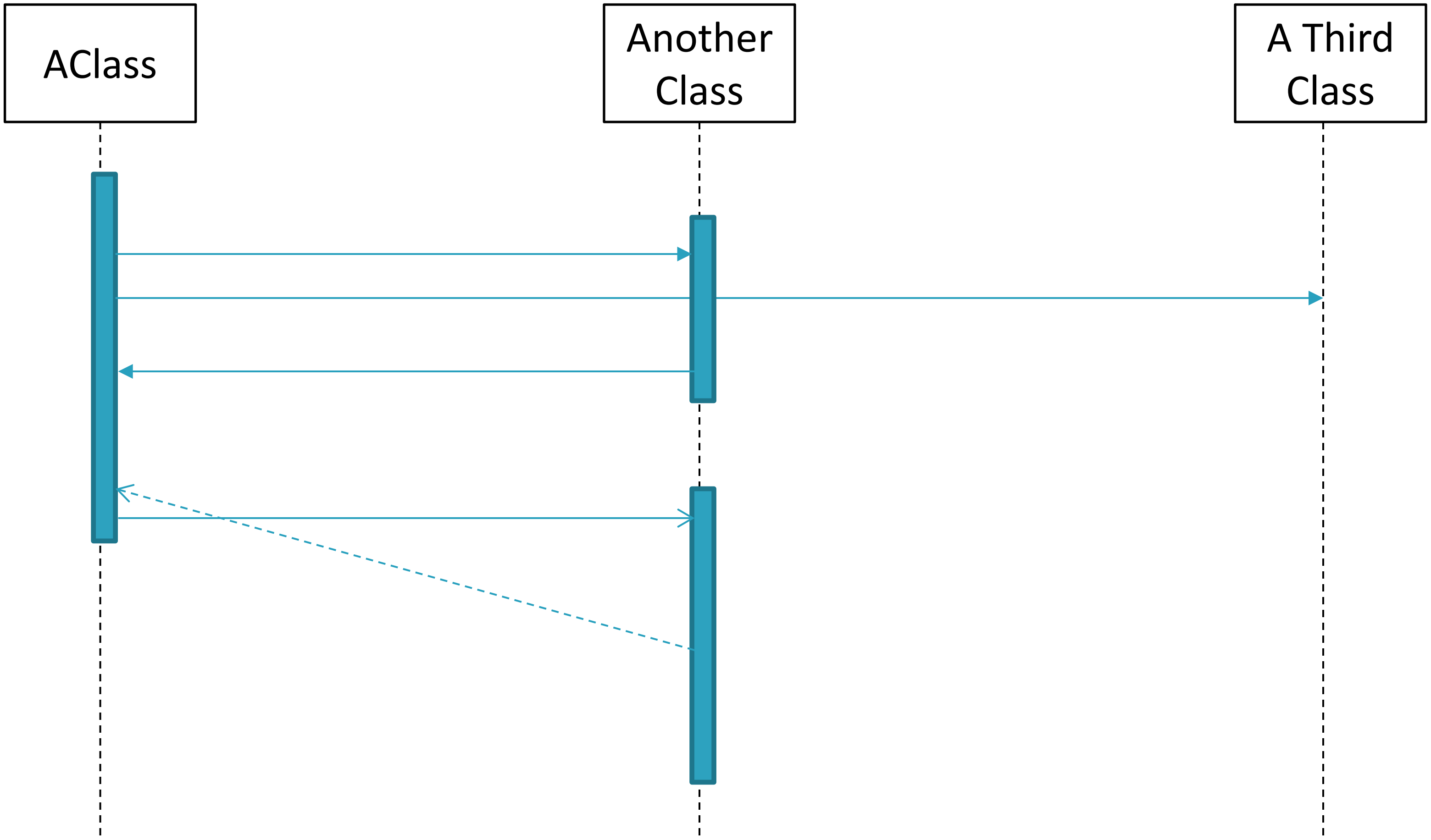
Key elements of sequence diagrams

- Time
  - Relative time (not absolute!) increases downwards
- Lifelines
  - Objects present at start of sequence shown at the top
  - Objects “live” in real time
    - Objects can be created, at which point they appear (ob2)
- Interaction frames
  - Used to model iteration (loop), conditionals (alt), and “subroutines” (ref)
- Activation bars (or “execution specifications”)
  - Indicate that the object is “active” at that time (equivalently, it is on the execution stack as the target object of an invocation)
  - Red dotted line: ob4 (top of stack), ob2, ob1 (bottom); “x>0” has to be true

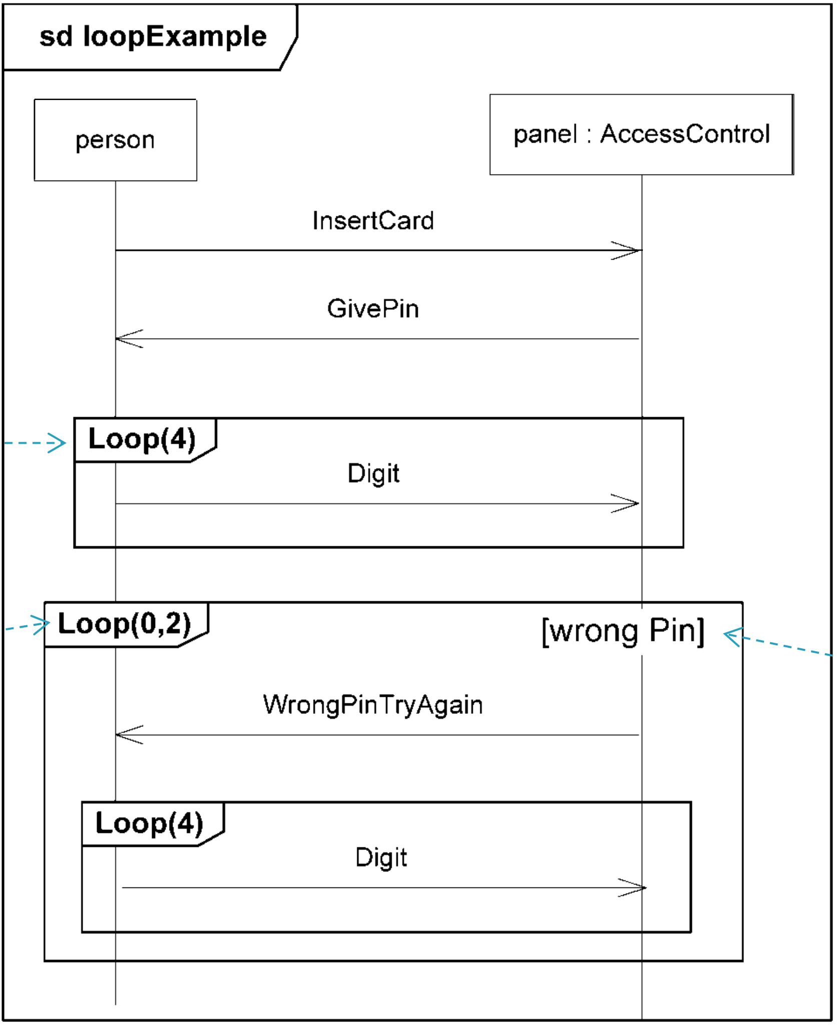
An annotated example with lots of details



Some common errors



Iteration



loop exactly 4 times

loop at least 0 times, at most 2 times

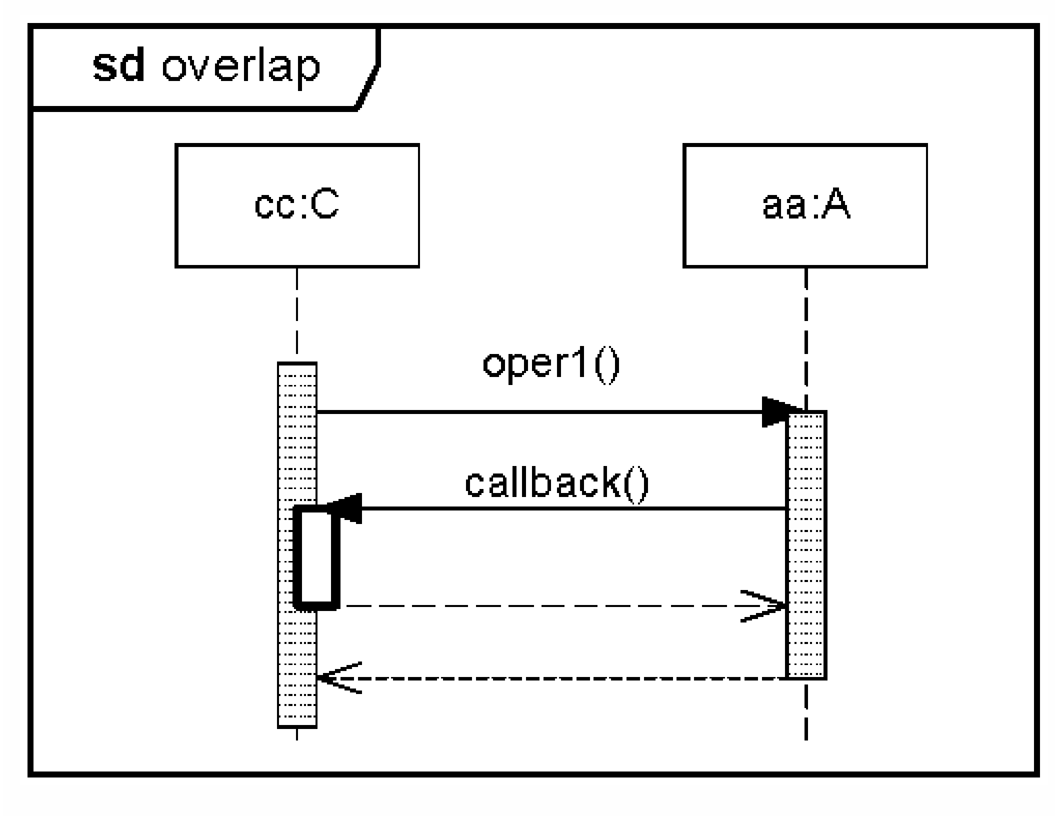
loop only if the guard is true

If you want "loop while true", just use "loop" with no bounds, plus a guard

Note: All asynchronous messages.

No activations are shown, no returns are shown, which will be marked as wrong in the context of the course

Overlapping activations



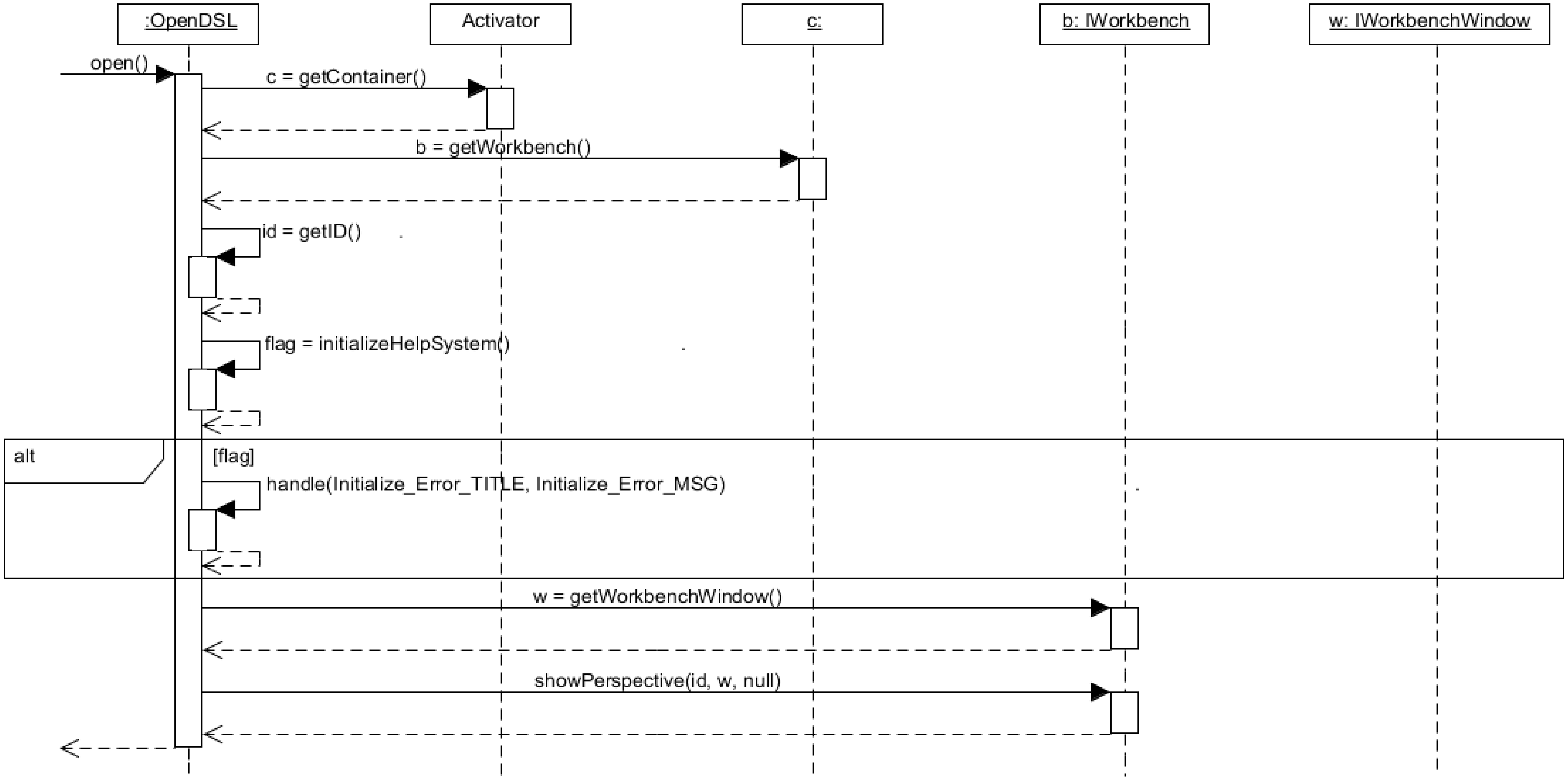
Exercise

- Model this:

```
public class OpenDSL {
   public void open() {
      IWorkbench bench = Activator.getContainer().getWorkbench();
      String id = getID();

      if( initializeDSLHelpSystem() ) {
           handle(Initialize_Error_TITLE, Initialize_Error_MSG);
      }

      IWorkbenchWindow window = bench.getWindow();
      IAdapter adapter = null;
      bench.showPerspective(id, window, adapter);
   }
}
```



Roles would probably have been a better choice than explicit objects.

Next time

- Behavioural Models: State Machine Diagrams