

UML - Sequence Diagrams

Linda Marshall and Vreda Pieterse

Department of Computer Science
University of Pretoria

(2014)

UML 2.0 Interaction diagrams:

- sequence diagrams
- communication diagrams
- interaction overview diagrams
- timing diagrams

All interaction diagrams model how objects interact with one another in terms of the messages they pass to one another.

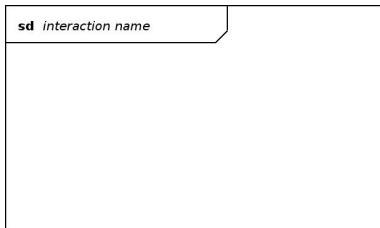
Sequence diagrams:

- are a type of interaction diagram and models behaviour
- are used to model the interaction between object instances showing the sequence of messages that are exchanged by the objects
- emphasise the order of the messages over time.

show lifelines that represent objects over time.

- **time** is represented on the vertical axis
- **lifelines** are represented on the horizontal axis

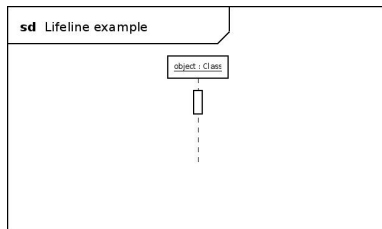
Frames:



Comprises of a *heading* and a *content area*.

- A frame may be used to delineate scope
- A heading give a descriptive name to the interaction being modelled

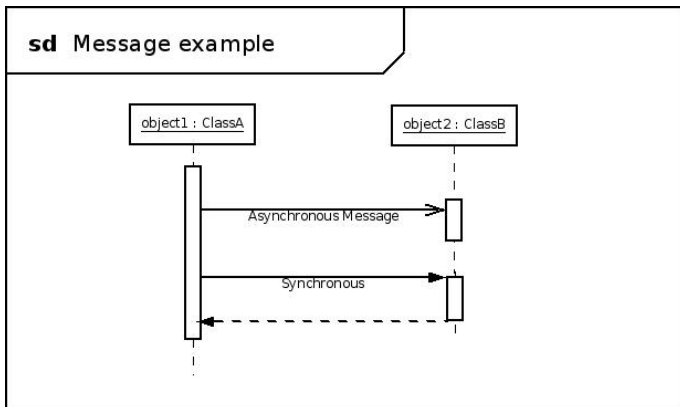
Lifelines:



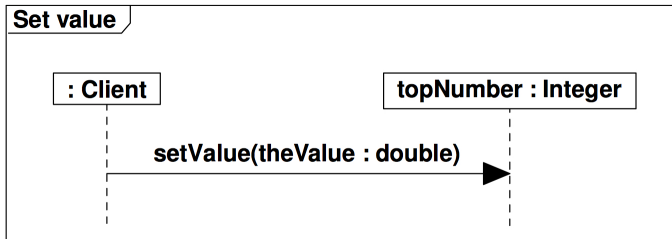
A sequence diagram model objects

The syntax is the same as in object diagrams

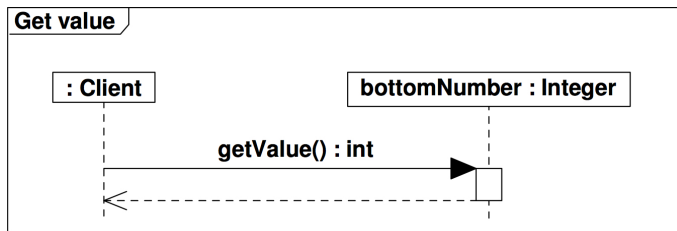
Messages:



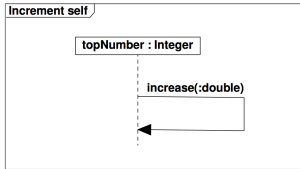
An asynchronous message is a message call without a return value



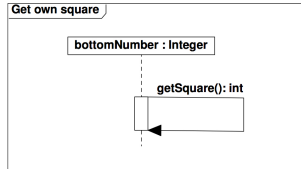
A synchronous message is a message call with a return value



A reflexive message

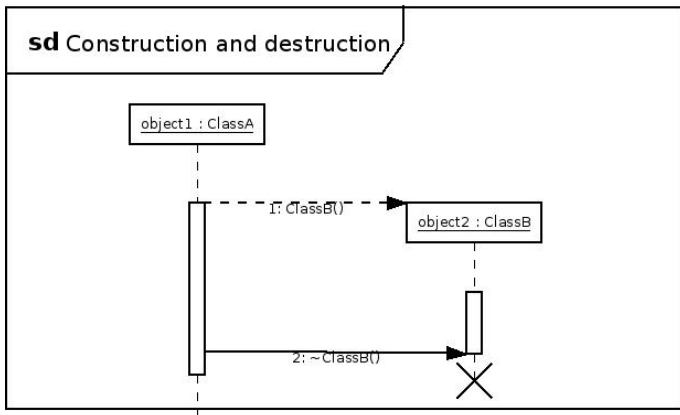


without a return value



with a return value

Creation and Destruction:



Message Arrows:



Synchronous or call



Asynchronous



Creation



Reply



Destruction

Message detail:

[attribute=]

signal_or_operation_name

[(arguments)] [:return_value] |

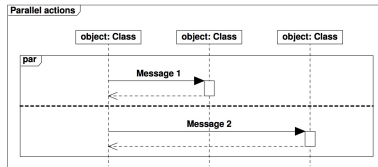
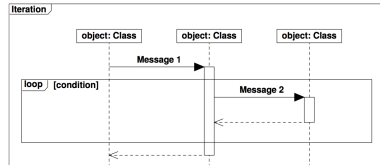
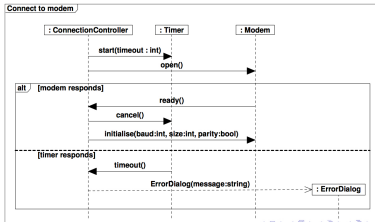
'*'

Argument detail:

```
[parameter_name=] argument_value |  
attribute=out_parameter_name [:  
argument_value] | -
```

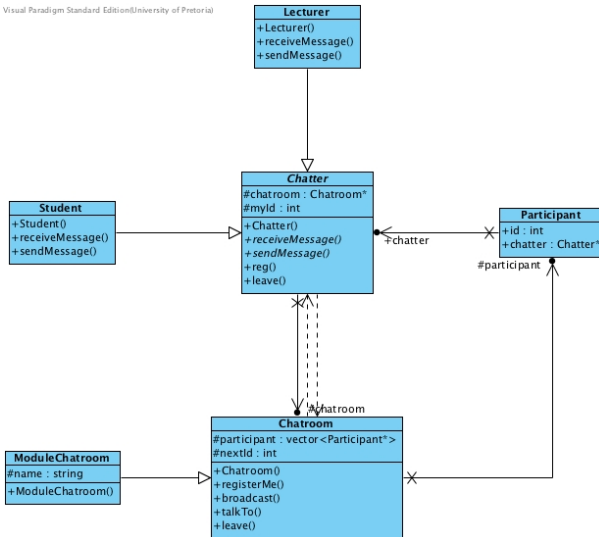
Combined fragment keywords:

- alt - Alternative - like an if statement
- loop - Loop - like a while statement
- par - Parallel actions



Introduction Classification Structure Example

Visual Paradigm Standard Edition (University of Pretoria)



```
Chatroom* cr =  
    new ModuleChatroom("COS121");  
Chatter* student = new Student();  
student->reg(cr);  
student->sendMessage();  
student->leave();  
delete student;  
delete cr;
```

