

## Excercise Sheet 4

### Solution

## Lecture Distributed Systems

### Winter Term 2025/26

#### Exercise 1: Simulator for Distributed Systems

```
import sys
import json

class Message:
    def __init__(self, src='', dst='', typ='', msg=None, **kwargs):
        if msg:
            self.__dict__ = json.loads(msg)
        else:
            self.src = src
            self.dest = dst
            self.body = { 'type': typ }
            for key, value in kwargs.items():
                self.body[key] = value
    def __getitem__(self, key):
        return self.body[key]
    def __str__(self):
        return json.dumps(self.__dict__)

def receive():
    msg = sys.stdin.readline()
    return Message(msg=msg) if msg else None

def send(msg):
    print(msg, flush=True)

me = receive()['name']
while msg := receive():
    message = msg['message']
    reply = 'World' if msg['message'] == 'Hello' else msg['message']
    send(Message(me, msg.src, 'echo_ok', in_reply_to=msg['id'], message=reply))
```

#### Exercise 3: Programming: Generic proxy objects

You will find the solution to this problem in in the archive [104eFiles.zip](#)<sup>1</sup> on the lecture's web page.

#### Exercise 4: Parameter passing

If *call-by-reference* is used, a pointer to *i* is passed to *incr*. This will increment *i* twice, so the end result in this case is 2.

With *call-by-value* the value is passed directly and no reference to *i*. Therefore *i* remains unchanged for the caller, i.e. 0.

*Call-by-copy/result* means: Copying the parameter values when calling the procedure, copying back and overwriting the call parameters when the procedure is finished. In this case, *i* is passed as a value twice (i.e. 0 each) and incremented so

---

<sup>1</sup><http://www.bs.informatik.uni-siegen.de/web/wismueller/v1/vs/104eFiles.zip>

that both values are now 1. If they are copied back, the second copy overwrites the first and the final value is now also 1.