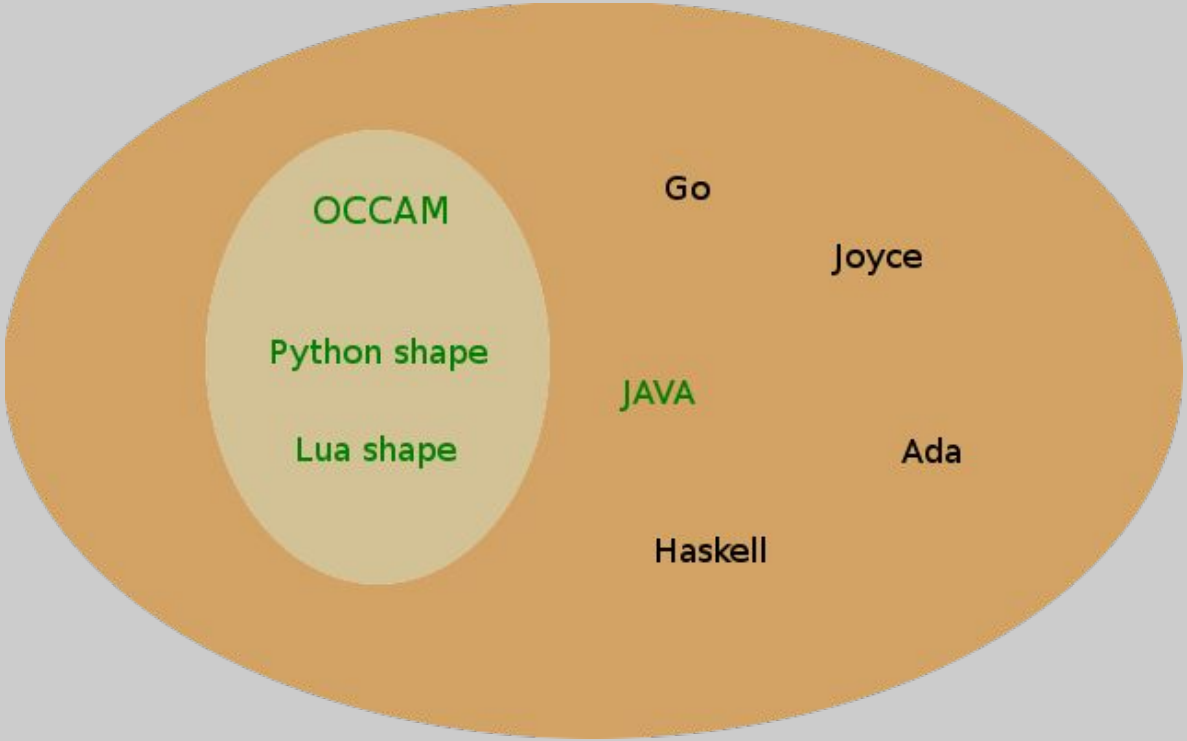
Occam
DSL shapes

Producer/Consumer use case

Occam DSL What's that??

- Communicating sequential processes programming language (CSP algebra implementation)
- Imperative language
- First release in 1983
- Current version is **occam-π**
- Developed by David May at INMOS

CSP algebra implementations



Internal shape

Occam DSL Internal shape

```
PROC producer (CHAN CHAR out!)
       CHAR c:
       SEQ
              c := "a"
              out ! c
              print("send value : ", c)
:
PROC consumer (CHAN CHAR in?)
       CHAR c:
       SEQ
              in ? c
              print("get value : ", c)
:
PROC network ()
       CHAN CHAR c:
       PAR
              producer (c!)
              consumer (c?)
:
```

External shapes

Occam DSL Python shape

```
#!/usr/bin/env python
from csp.csp import *

@process
def producer(n, channel):
        channel.write(n)
        print “send value : “ + n
        channel.poison()
        return

@process
def consumer(channel):
        print “get value : “ + channel.read()

channel = Channel()
Par(consumer(channel), producer(“a”, channel)).start()
```

Occam DSL Lua shape

```
require "lua-channel"

function producer(char, channel)
    PAR(
        function()
            KEYSTATE( channel, char )
            print(“send value : “ + char)
        end
    )
end
function consumer(channel)
    PAR(
        function()
            local char = channel:IN()
            print(“get value : “ + char)
        end
    )
end
local channel = Channel:new()
producer("a", channel)
consumer(channel)
```

Other implementation

Occam DSL Java equivalent

```
import org.jcsp.lang.*;
import org.jcsp.plugNplay.ints.ParaplexInt;
class external {
  public static void main(String[] args) {
    final One2OneChannelInt[] a = Channel.one2oneIntArray(1);
    final One2OneChannel b = Channel.one2one();
    new Parallel( new CSProcess[]{
            new CSProcess() {
                public void run() {
                  a[0].out().write(1);
                  System.out.println("send value : " + 1);
                }
            }, new CSProcess() {
                public void run() {
                  int[] data = (int[]) b.in().read();
                  System.out.println("get value : " + data[0]);
                }
            }, new ParaplexInt(Channel.getInputArray(a), b.out())
        }
     ).run();
  }
}
```

Any questions ?

Thank you

References :

- http://pop-users.org/occam-pi/
- http://www.cs.kent.ac.uk/projects/ofa/jcsp/
- http://frmb.org/occtutor.html
- https://en.wikipedia.org/wiki/Occam_%28programming_language%29