Jolie

The service-oriented programming language

Course Material

https://jolie-lang.org

```
type HelloRequest {
   name:string
}

interface HelloInterface {
RequestResponse:
   hello( HelloRequest )( string )
}

service HelloService {
   execution: concurrent

   inputPort HelloService {
       location: "socket://localhost:8080"
       protocol: http { format = "json" }
       interfaces: HelloInterface
   }

   main {
       hello( request )( response ) {
           response = "Hello " + request.name + " 😀"
       }
   }
}
```

What is Jolie?

- A service-oriented programming language.
- A collaborative project.
  - Open source.
  - Active collaboration with experts within the Microservices Community.

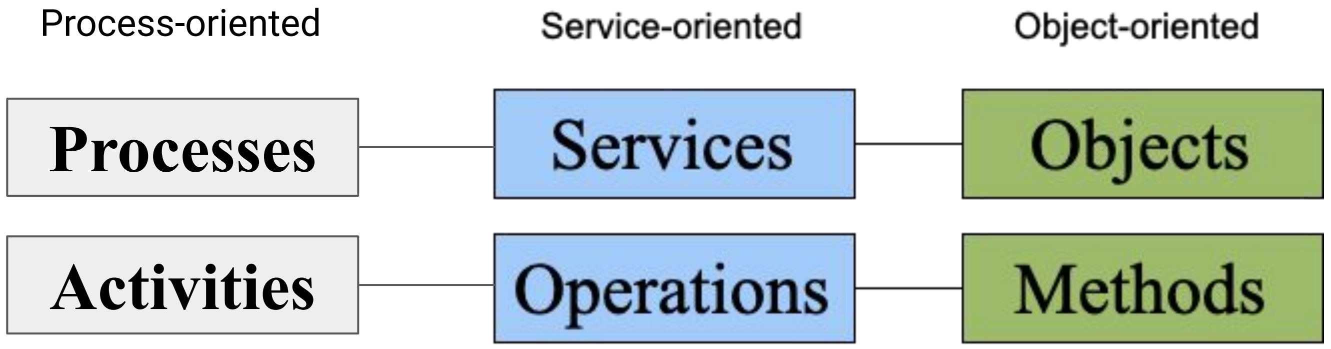
In collaboration with Microservices Community

https://microservices.community

Get in touch!

- Chat with us on DISCORD : https://discord.gg/yQRTMNX
- GitHub: https://github.com/jolie/jolie
- Twitter: https://twitter.com/jolielang
- Mailing list: jolie-devel@googlegroups.com

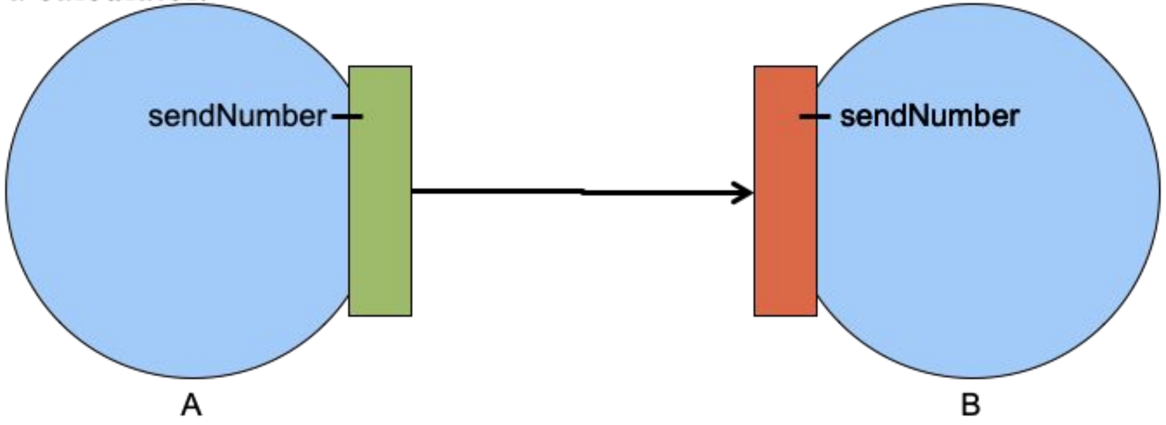
Processes - Services - Objects



Services, ports & interfaces

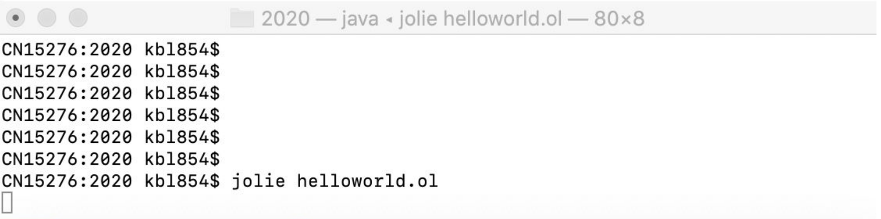
- Services communicate through **ports**.
- **Ports** give access to an **interface**.
- An **interface** is a set of **operations**.
- An **output port** is used to invoke **interfaces** exposed by other services.
- An **input port** is used to expose an **interface**.

- Example: a client has an **output port** connected to an **input port** of
- a calculator.

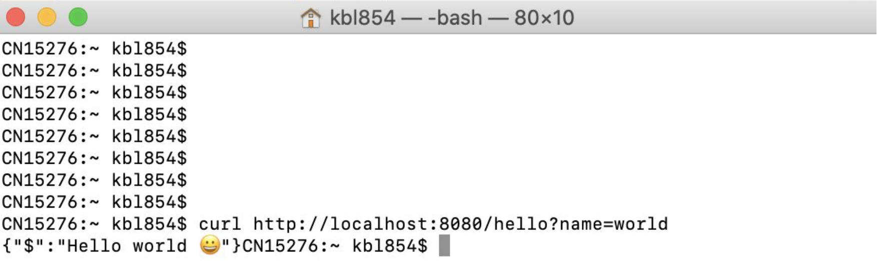


Code Example: Hello World

```
type HelloRequest {
   name:string
}

interface HelloInterface {
RequestResponse:
   hello( HelloRequest )( string )
}
```



```
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$ jolie helloworld.ol
```



```
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$ curl http://localhost:8080/hello?name=world
{"$":"Hello world 😀"}CN15276:~ kbl854$
```

```
service HelloService {
   execution: concurrent

   inputPort HelloService {
      location: "socket://localhost:8080"
      protocol: http { format = "json" }
      interfaces: HelloInterface
   }

   main {
      hello( request )( response ) {
         response = "Hello " + request.name + " 😀"
      }
   }
}
```

Getting Started: A calculator Service

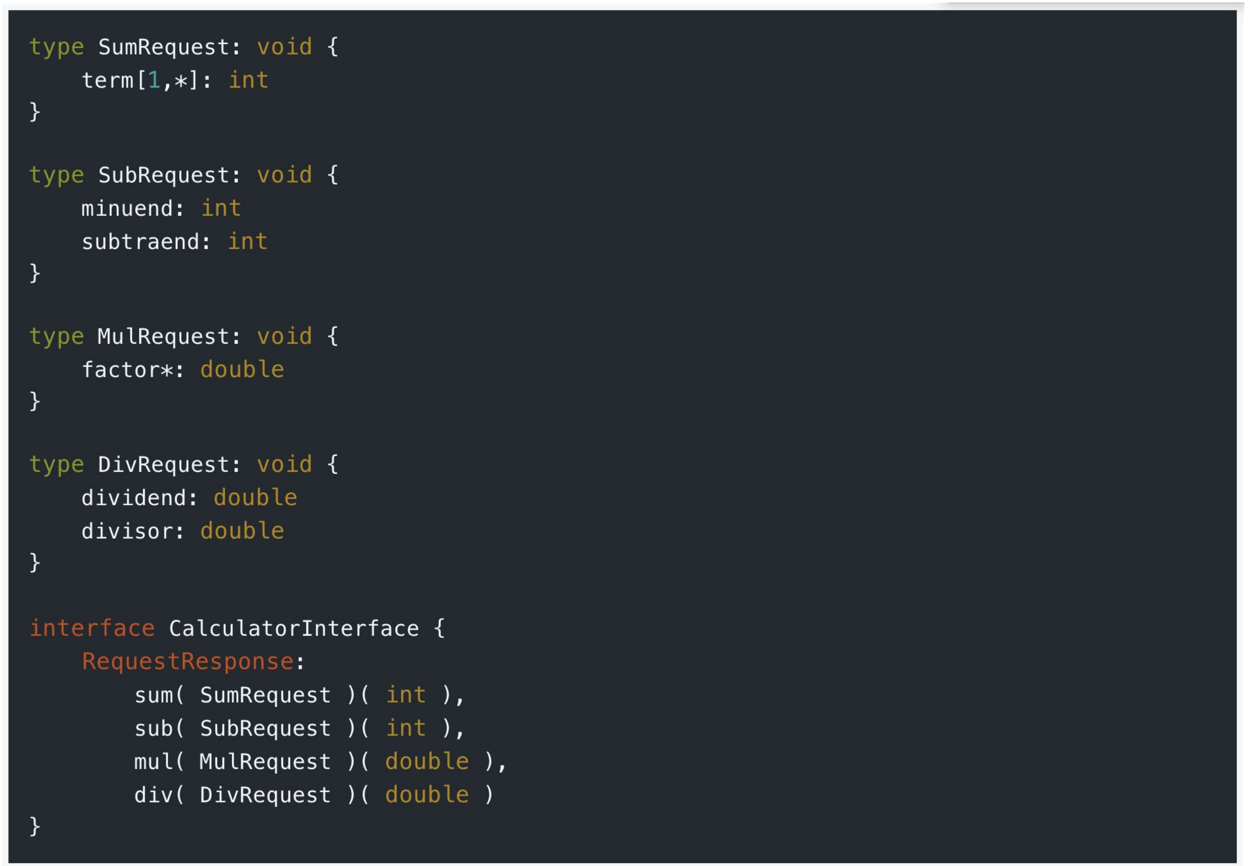
https://docs.jolie-lang.org/v1.10.x/tutorials/getting-started/

```
interface CalculatorInterface {
    RequestResponse:
        sum,
        sub,
        mul,
        div
}
```

https://github.com/jolie/examples/tree/master/v1.10.x/tutorials/getting_started

Detailing the types and interface

CalculatorInterfaceModule.ol



```
type SumRequest: void {
    term[1,*]: int
}

type SubRequest: void {
    minuend: int
    subtraend: int
}

type MulRequest: void {
    factor*: double
}

type DivRequest: void {
    dividend: double
    divisor: double
}

interface CalculatorInterface {
    RequestResponse:
        sum( SumRequest )( int ),
        sub( SubRequest )( int ),
        mul( MulRequest )( double ),
        div( DivRequest )( double )
}
```

Defining the Behaviour I

Deployment: Port, Location, protocol, interface

```
from CalculatorInterfaceModule import CalculatorInterface

service CalculatorService {

  inputPort CalculatorPort {
      location: "socket://localhost:8000"
      protocol: http { format = "json" }
      interfaces: CalculatorInterface
  }
```

Defining the Behaviour II

Parallel composition of operations

[sum(req)(res) {
…
}]

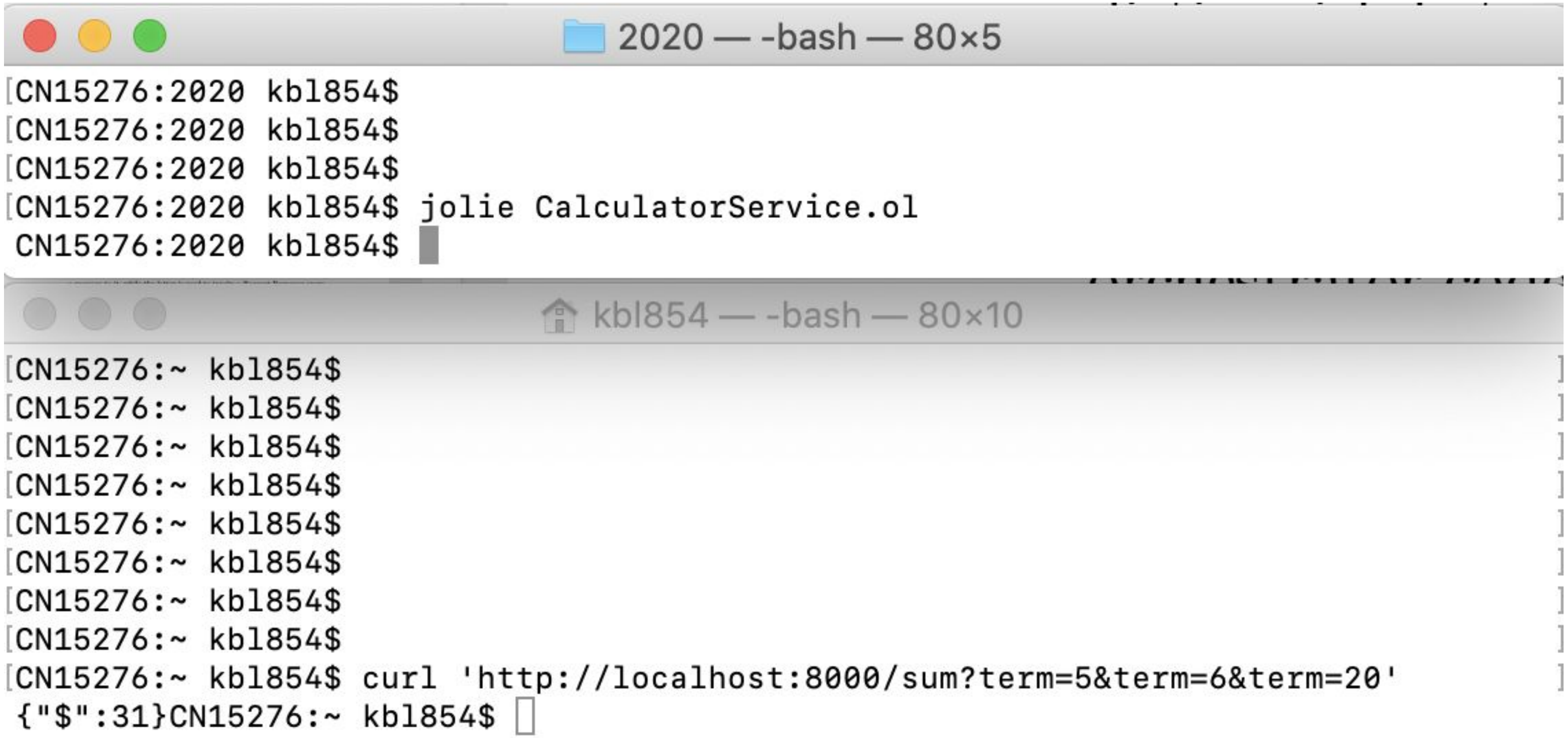
[sub(req)(res) {
…
}]

[mul(req)(res) {
…
}]

[div(req)(res) {
…
}]

```
  main {

      [ sum( request )( response ) {
          for( t in request.term ) {
              response = response + t
          }
      }]

      [ sub( request )( response ) {
          response = request.minuend - request.subtraend
      }]

      [ mul( request )( response ) {
          for ( f in request.factor ) {
              response = response * f
          }
      }]

      [ div( request )( response ) {
          response = request.dividend / request.divisor
      }]
  }

}
```

Calculator Service Example Run



```
2020 — -bash — 80×5
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$ jolie CalculatorService.ol
CN15276:2020 kbl854$
```

```
kbl854 — -bash — 80×10
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$ curl 'http://localhost:8000/sum?term=5&term=6&term=20'
{"$":31}CN15276:~ kbl854$
```

Execution Modality

The execution modality specifies three different way to run a service:

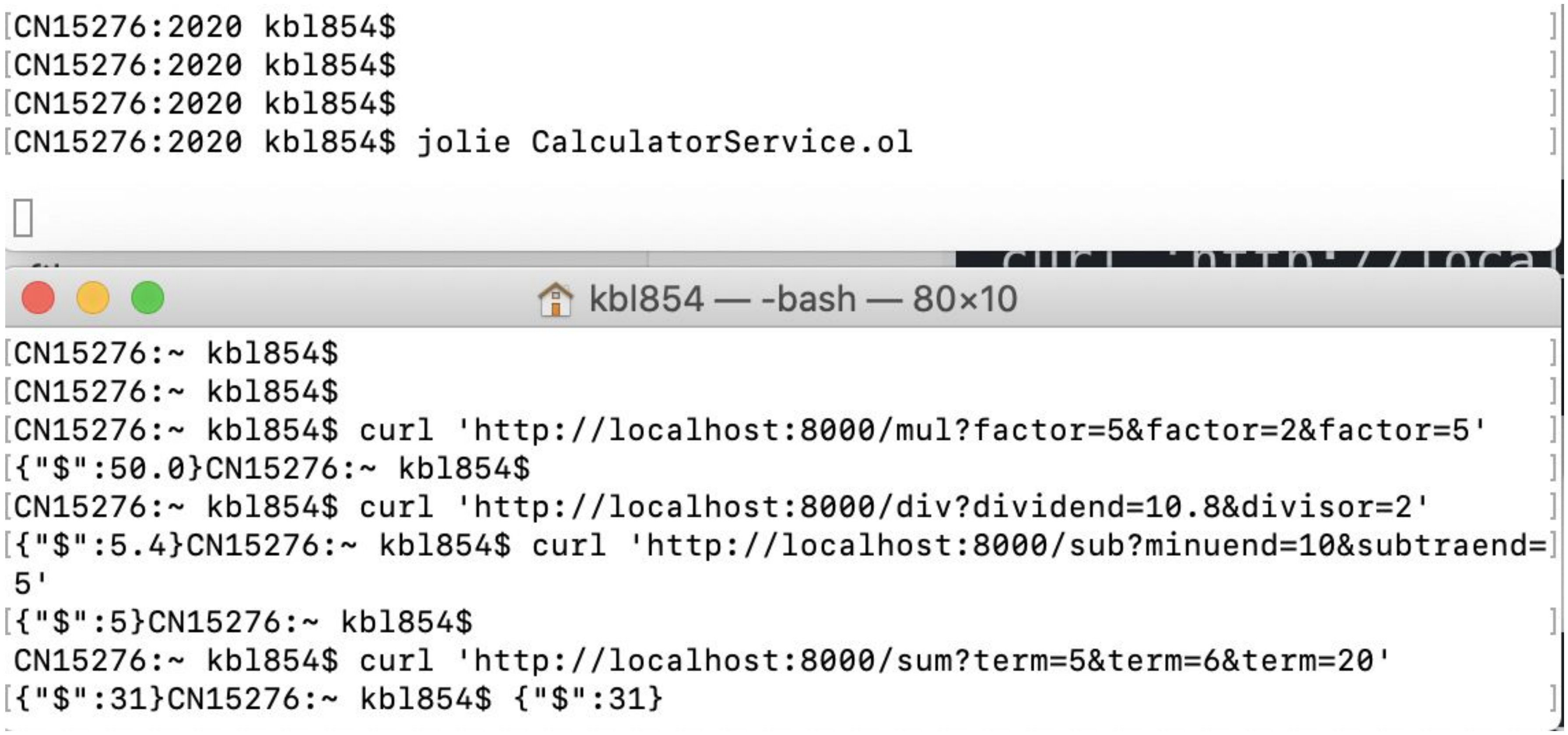
`concurrent`, `sequential` or `single`.

If nothing is specified, modality `single` is set.

This modality means that the service executes its behaviour once, then stops. This is why our service just executed one operation and then stops.

```
from CalculatorInterfaceModule import CalculatorInterface

 service CalculatorService {

     execution{ concurrent }
```

Execution Modality

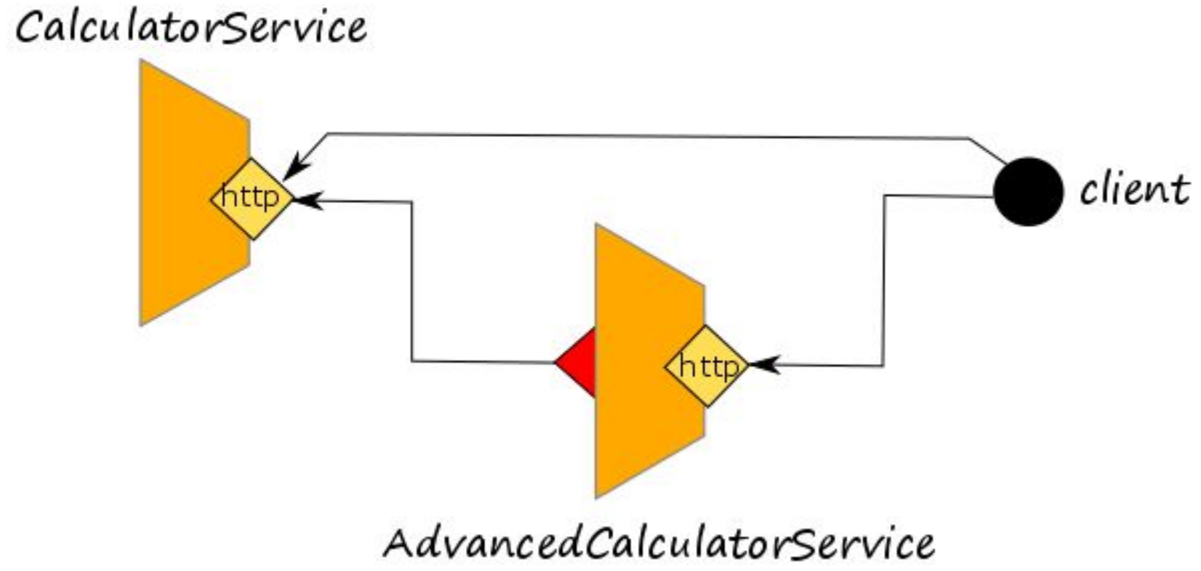


```
[CN15276:2020 kbl854$
[CN15276:2020 kbl854$
[CN15276:2020 kbl854$
[CN15276:2020 kbl854$ jolie CalculatorService.ol

▯
```

```
[CN15276:~ kbl854$
[CN15276:~ kbl854$
[CN15276:~ kbl854$ curl 'http://localhost:8000/mul?factor=5&factor=2&factor=5'
[{"$":50.0}CN15276:~ kbl854$
[CN15276:~ kbl854$ curl 'http://localhost:8000/div?dividend=10.8&divisor=2'
[{"$":5.4}CN15276:~ kbl854$ curl 'http://localhost:8000/sub?minuend=10&subtraend=
5'
[{"$":5}CN15276:~ kbl854$
CN15276:~ kbl854$ curl 'http://localhost:8000/sum?term=5&term=6&term=20'
[{"$":31}CN15276:~ kbl854$ {"$":31}
```

Dependencies: Connecting Services

https://github.com/jolie/examples/tree/master/v1.10.x/tutorials/using_dependencies

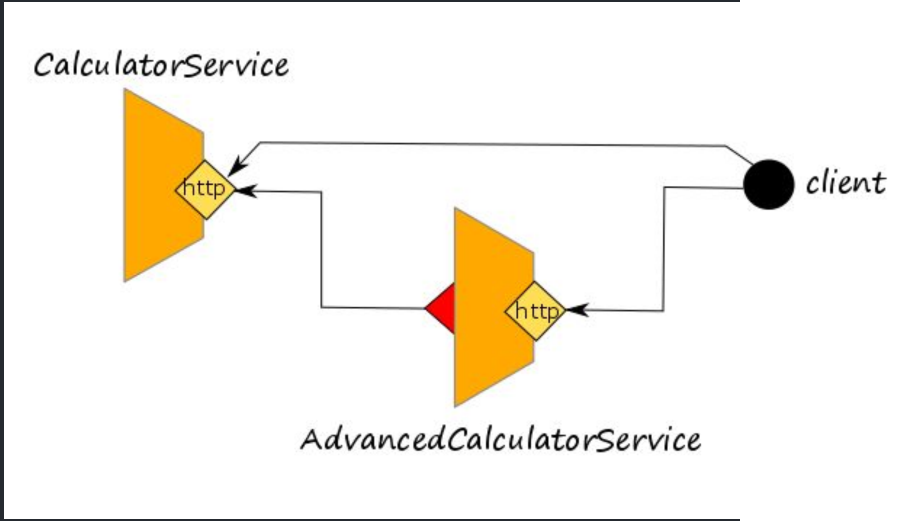


Dependencies: Connecting Services

```
interface AdvancedCalculatorInterface {
    RequestResponse:
        factorial( FactorialRequest )( FactorialResponse ),
        average( AverageRequest )( AverageResponse ),
        percentage( PercentageRequest )( PercentageResponse )
}
```

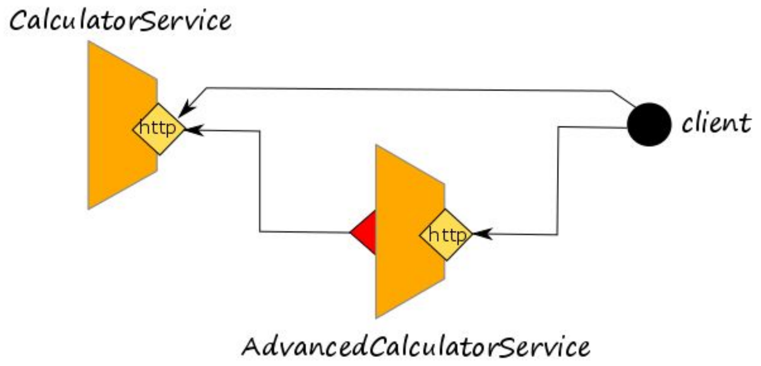
outputPorts

```
from AdvancedCalculatorServiceInterfaceModule import AdvancedCalculatorInterface
from CalculatorInterfaceModule import CalculatorInterface

service AdvancedCalculatorService {

    execution{ concurrent }

    outputPort Calculator {
         location: "socket://localhost:8000"
         protocol: http { format = "json" }
         interfaces: CalculatorInterface
    }

    inputPort AdvancedCalculatorPort {
         location: "socket://localhost:8001"
         protocol: http { format = "json" }
         interfaces: AdvancedCalculatorInterface
    }
```

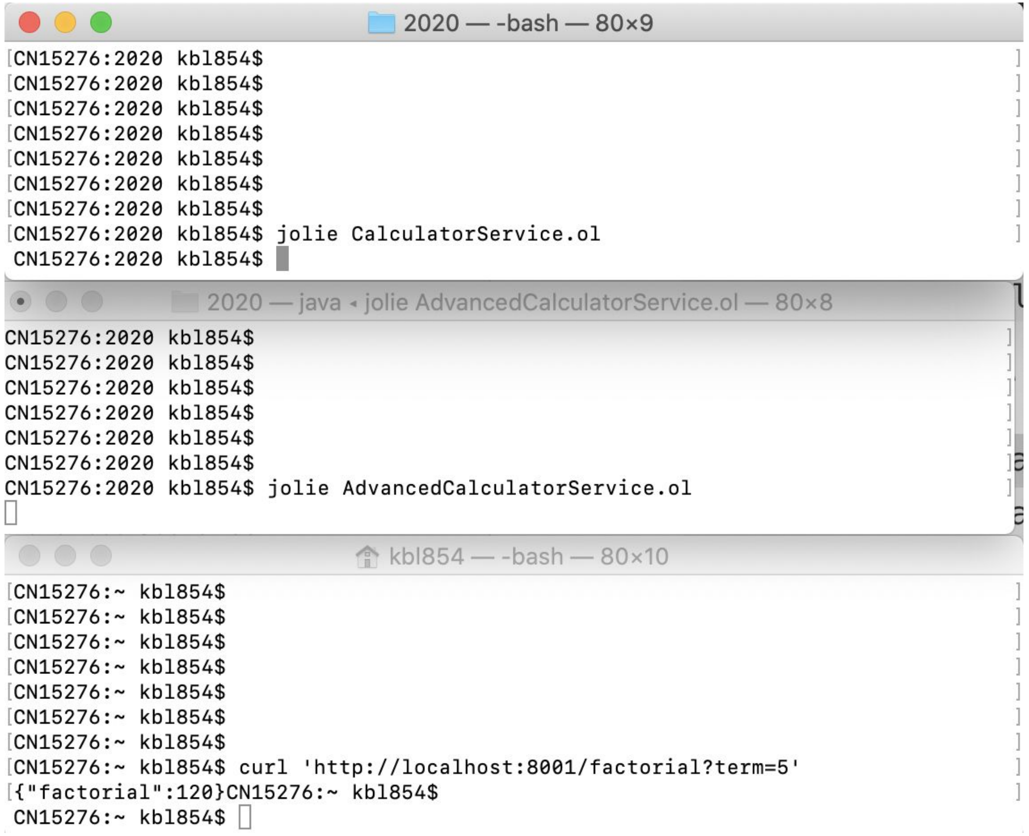


Calling operations at other services

```
main {
    [ factorial( request )( response ) {
        for( i = request.term, i > 0, i-- ) {
            req_mul.factor[ #req_mul.factor ] = i
        }
        mul@Calculator( req_mul )( response.factorial
    }]

    [ average( request )( response ) {
        sum@Calculator( request )( sum_res )
        div@Calculator( { dividend = double( sum_res ), divisor = double( #request.term
    }]

    [ percentage( request )( response ) {
        div@Calculator( { dividend = request.term, divisor = 100.0 })( div_res )
        mul@Calculator( { factor[0] = div_res, factor[1] = request.percentage })( respo
    }]
}
}
```



Connecting Services Example Run



2020 — -bash — 80×9

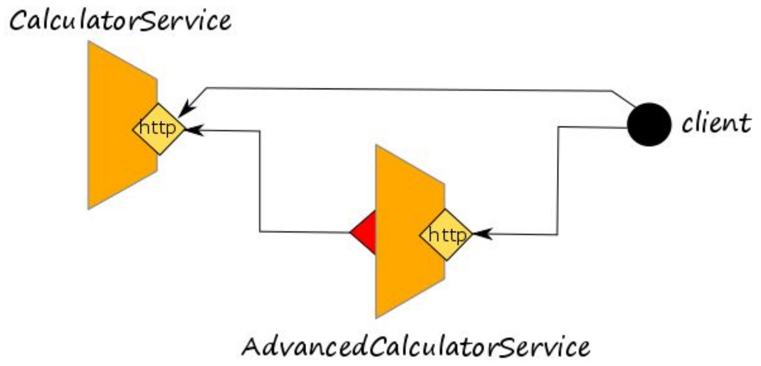
```
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$ jolie CalculatorService.ol
CN15276:2020 kbl854$
```

2020 — java ◂ jolie AdvancedCalculatorService.ol — 80×8

```
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$
CN15276:2020 kbl854$ jolie AdvancedCalculatorService.ol
```

kbl854 — -bash — 80×10

```
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$
CN15276:~ kbl854$ curl 'http://localhost:8001/factorial?term=5'
{"factorial":120}CN15276:~ kbl854$
CN15276:~ kbl854$
```


CalculatorService
http
client
http
AdvancedCalculatorService