Introduction to the Jolie Programming Language

Stefano Pio Zingaro
stefanopio.zingaro@unibo.it
cs.unibo.it/~stefanopio.zingaro

What is Jolie?

A **Service-Oriented** Programming Language

Why SOC and Jolie?

Jolie is perfect for fast prototyping. In little time a small team of developers can build up a full-fledged distributed system.

But I already know Java!
Why shall I use Jolie?

Why SOC and Jolie?

```
SocketChannel socketChannel = SocketChannel.open();
  socketChannel.connect(
new InetSocketAddress("http://someurl.com", 80));
  Buffer buffer = . . .; // byte buffer
  while( buffer.hasRemaining() ) {
     channel.write( buffer );
}
```

Happy?

Ok, but you did not even close the channel or handled exceptions

Why SOC and Jolie?

```
SocketChannel socketChannel = SocketChannel.open();
try {
  socketChannel.connect(new InetSocketAddress("http://someurl.com",
80));
  Buffer buffer = . . .; // byte buffer
  while( buffer.hasRemaining() ) {
    channel.write( buffer );
} }
catch( UnresolvedAddressException e ) { . . . }
catch( SecurityException e ) { . . . }
/* . . . many catches later . . . */
catch( IOException e ) { . . . }
finally { channel.close(); }
```

Happier now?

Yes, but what about the **server**?

Why SOC and Jolie?

```
Selector selector = Selector.open();
channel.configureBlocking(false);
SelectionKey key = channel.register(selector, SelectionKey.OP_READ);
while(true) {
  int readyChannels = selector.select();
  if(readyChannels == 0) continue;
  Set<SelectionKey> selectedKeys = selector.selectedKeys();
  Iterator<SelectionKey> keyIterator = selectedKeys.iterator();
  while(keyIterator.hasNext()) {
    SelectionKey key = keyIterator.next();
    if(key.isAcceptable()) {
        // a connection was accepted by a ServerSocketChannel.
    } else if (key.isConnectable()) {
        // a connection was established with a remote server.
    } else if (key.isReadable()) {
        // a channel is ready for reading
    } else if (key.isWritable()) {
        // a channel is ready for writing
    }
    keyIterator.remove();
  }
}
```

Here you are

Why SOC and Jolie?

Well, ok, but again, you are not **handling exceptions**. And what about if **different operations** use the **same channel**?

And if we wanted to use **RMIs** instead of **Sockets**?

In what **format** are you transmitting data? And if we need to **change** the **format** after we wrote the application? Do you **check** the **type of data** you receive/send?

Why SOC and Jolie?

Programming distributed systems is usually harder than programming non distributed ones.

Concerns of **concurrent** programming.

Plus (not exhaustive):

- handling **communications**;
- handling **heterogeneity**;
- handling **faults**;
- handling the **evolution** of systems.

Why SOC and Jolie?

Applications in a distributed system can perform a **distributed transaction**.

Example:

- a client asks a store to buy some music;
- the store opens a request for handling a payment on a bank;
- the client sends his credentials to the bank for closing the payment;
- the store sends the goods to the client.

Looks good, but a lot of things **may go wrong**, for instance:

- the store (or the bank) could be offline;
- the client may not have enough money in his bank account;
- the store may encounter a problem in sending the goods.

Why SOC and Jolie?

Things can be made easier by **hiding the low-level** details.

Two main approaches:

- make a library/tool/framework for an existing programming language;
- make a new programming language.

Can you tell the difference between the two approaches?

Service-Oriented Programming

3 Commandments

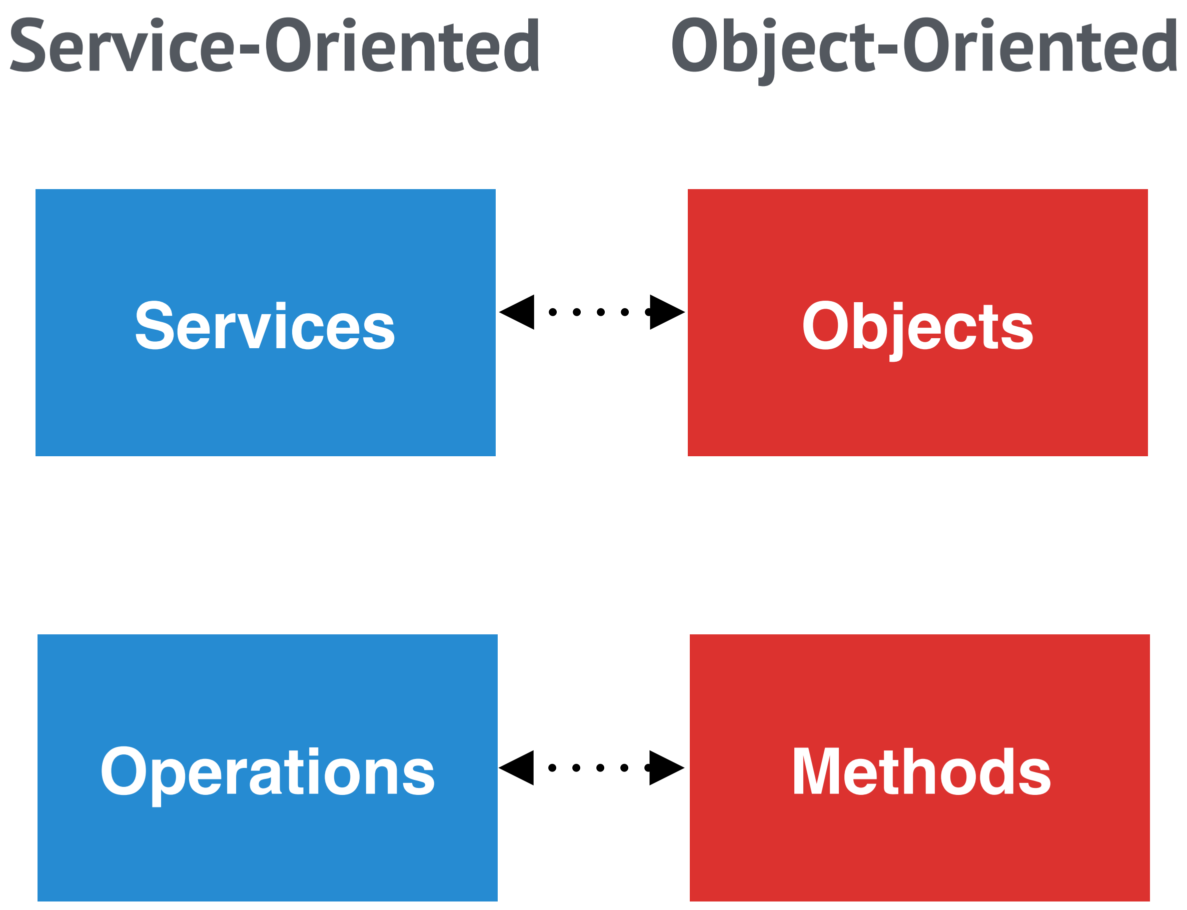
- Everything is a **service**;
- A service is an application that offers **operations**;
- A service can **invoke** another service by calling one of its operations.

Service-Oriented Programming

3 Commandments

- Everything is a **service**;
- A service is an application that offers **operations**;
- A service can invoke another service by calling one of its operations.

Recalling the Object-Oriented creed



Why Jolie?

A **Service-Oriented** Programming Language

Service-Oriented	**Object-Oriented**
Services	**Objects**
Operations	**Methods**

Why Jolie?

Because it is "Full Stack"

Formal foundations from Academia

IT University of Copenhagen

Taught also in:

DTU

Innopolis University

Why Jolie?

Because it is “Full Stack”

Tested and used in the **Real World**

Why Jolie?

Because it is "Full Stack"

It is a live **open source** project with continuous updates and a well documented codebase

https://github.com/jolie/jolie

"This *is* the programming language you are looking for"

Why Jolie?

Because it is “Full Stack”

Comprehensive and ever-growing **documentation** and **Standard Library**.

http://docs.jolie-lang.org

Why Jolie?

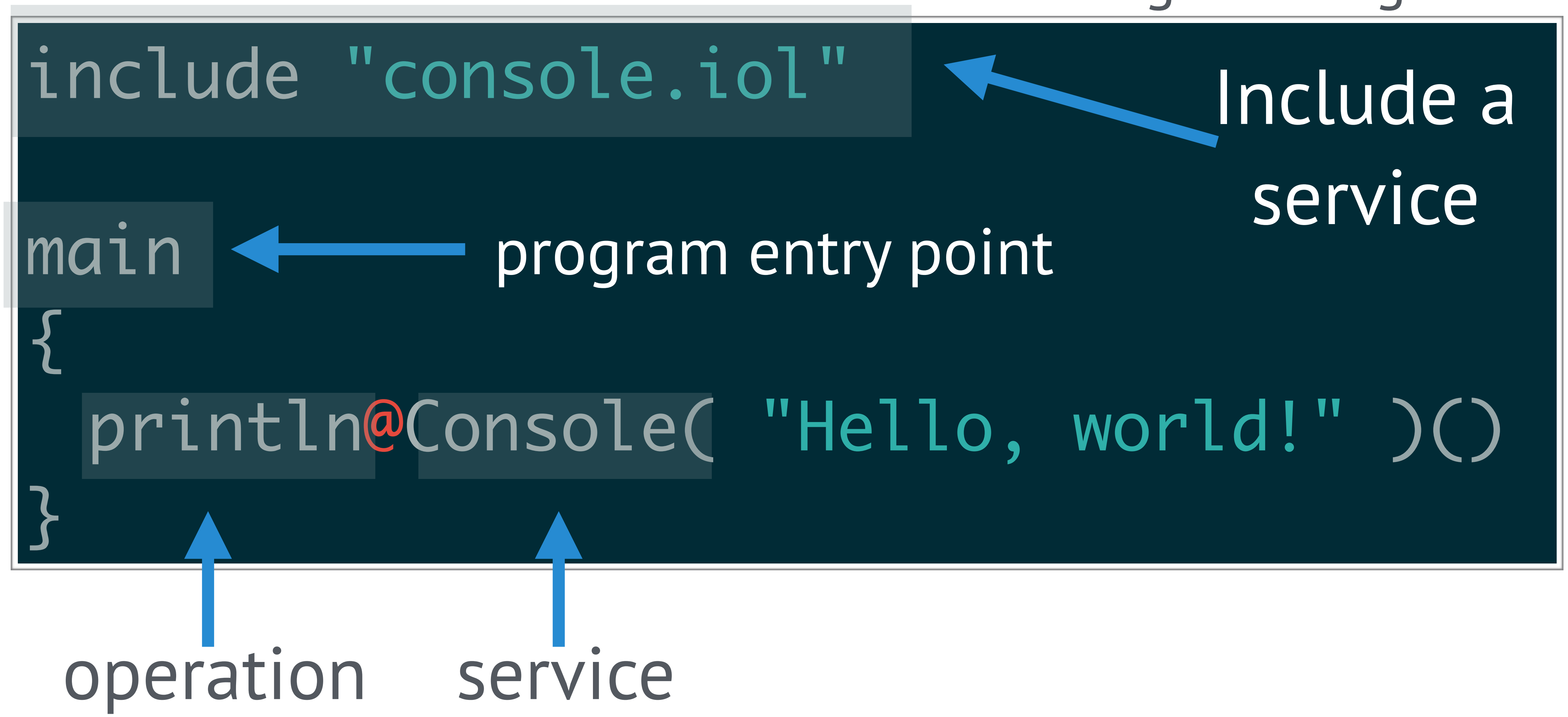
Because it is “Full Stack”

Cool Logo

Hello World! in Jolie

Let us get our hands dirty.

"Hello World!" is enough to let you see some of the main features of Jolie and Service-Oriented Programming.



```
include "console.iol"

main
{
  println@Console( "Hello, world!" )()
}
```

Hello World! in Jolie

Let us get our hands dirty.

“Hello World!” is enough to let you see some of the main features of Jolie and Service-Oriented Programming.

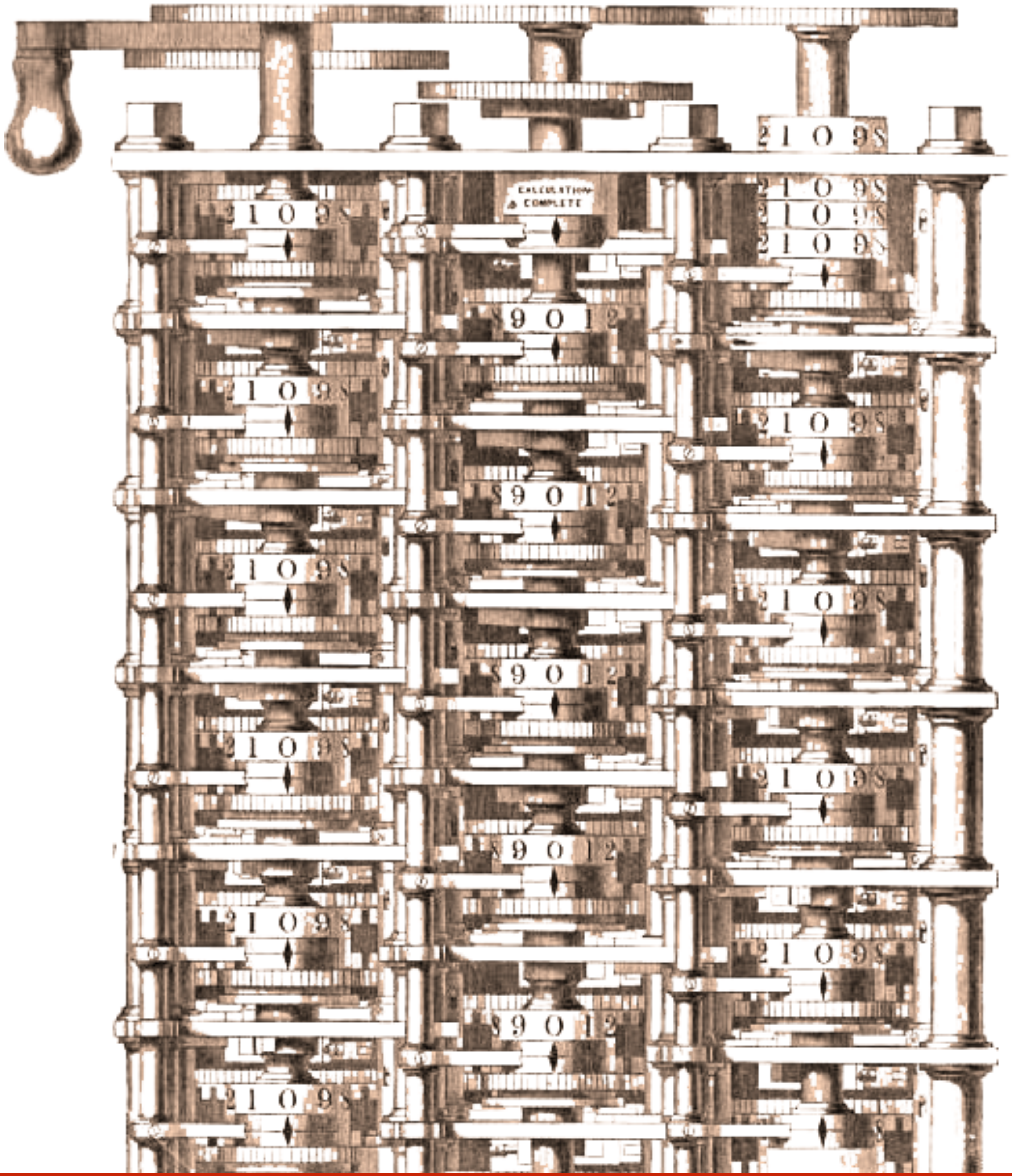
```
include "console.iol"

main
{
  println@Console( "Hello, world!" )()
}
```

```
$ jolie hello_world.ol
```

hello_world.ol

Let us see some **Jolie in Action**

Everything starts with a **calculator**…



Resources | **Online**

- Official Website:
  - **http://www.jolie-lang.org**
- Official Docs:
  - **http://docs.jolie-lang.org**
- Official Codebase:
  - **https://github.com/jolie/jolie**

Resources | **The Jolie Interpreter**

Last release

http://www.jolie-lang.org/downloads.html

- Requires JRE 1.6+
- Download jolie-installer.jar
- open a console and run

```
java -jar jolie-installer.jar
```

Resources | **The Jolie Interpreter**

Compile the last version from the repository (requires JDK1.6+ and ant)

```
$ git clone https://github.com/
jolie/jolie.git
$ cd jolie
$ ant && sudo ant install
```

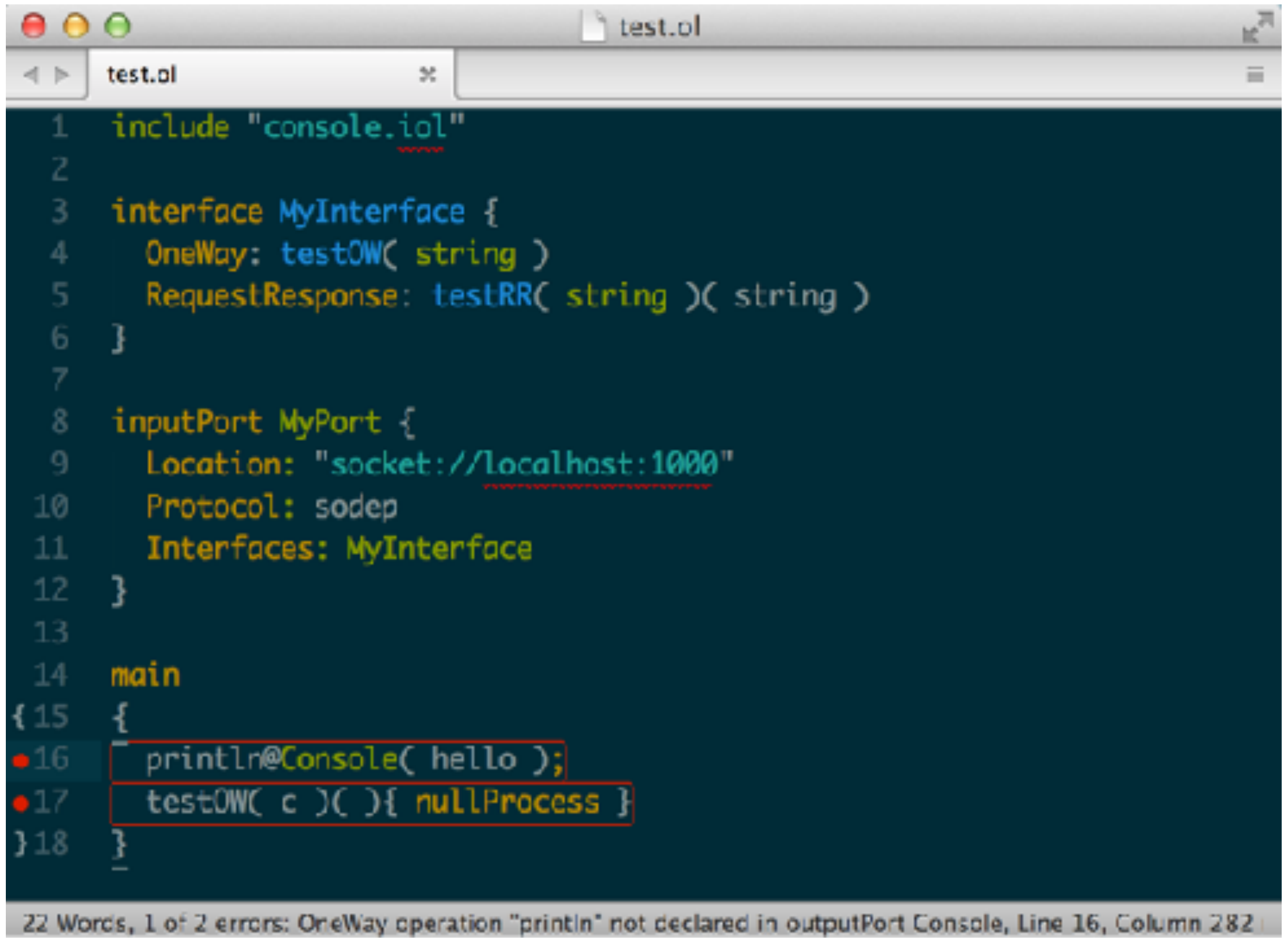
Resources | **Editors**

Sublime Text

https://github.com/thesave/
sublime-Jolie

+ =

https://github.com/thesave/
SublimeLinter-jolint



```
include "console.iol"

interface MyInterface {
  OneWay: testOW( string )
  RequestResponse: testRR( string )( string )
}

inputPort MyPort {
  Location: "socket://localhost:1000"
  Protocol: sodep
  Interfaces: MyInterface
}

main
{
  println@Console( hello );
  testOW( c )( ){ nullProcess }
}
```

22 Words, 1 of 2 errors: OneWay operation "println" not declared in outputPort Console, Line 16, Column 282