1. Develop the client, by having the servant running in a specific container (based on an image distributed by the professor). In order to set-up the servant running in a specific container, use the following commands

```
% docker pull mecellone/course-softengineering:soapws
% docker run -p 8080:8080 mecellone/course-softengineering:soapws

164 [main] INFO org.apache.cxf.wsdl.service.factory.ReflectionServiceFactoryBean -
Creating Service {http://soapws.softeng.sapienza.it/}WSImplService from class
it.sapienza.softeng.soapws.WSInterface
652 [main] INFO org.apache.cxf.endpoint.ServerImpl - Setting the server's publish
address to be http://0.0.0.0:8080/WSInterface
705 [main] INFO org.eclipse.jetty.util.log - Logging initialized @1328ms to
org.eclipse.jetty.util.log.Slf4jLog
802 [main] INFO org.eclipse.jetty.server.Server - jetty-9.4.z-SNAPSHOT; built:
2018-06-05T18:24:03.829Z; git: d5fc0523cfa96bfebfbda19606cad384d772f04c; jvm
1.8.0_282-b08
880 [main] INFO org.eclipse.jetty.server.AbstractConnector - Started
ServerConnector@5eeb6b21{HTTP/1.1,[http/1.1]}{0.0.0.0:8080}
881 [main] INFO org.eclipse.jetty.server.Server - Started @1521ms
891 [main] WARN org.eclipse.jetty.server.handler.ContextHandler - Empty contextPath
937 [main] INFO org.eclipse.jetty.server.handler.ContextHandler - Started
o.e.j.s.h.ContextHandler@7dcf94f8{/,null,AVAILABLE}
```

2. Then develop the servant. If you want to make it a standalone `.jar`, use the `.pom` distributed by the professor, and if you want to deliver as your own image, see the givem instructions.

IMPORTANT: *all the course, given the enterprise nature of the presented technologies and the current status of development (cf. the provided links), is based on a fully working Java 8 runtime and development kit (i.e., jdk8). If you have different versions, please install also JDK 1.8 (see the provided links for common OSs). All Docker images provided by the professor will be on Java 8.*