SOEN 423, Fall 2024, Assignment 2

Ties Schasfoort

7th November 2024



## Overall design:

For this assignment I implemented the distributed emergency response management system from Assignment 2 as a web service, using JAX-WS. All of the methods are direct copies from A2, however the setup of the client-server connection is different since a different approach is used. Because the methods are the same as in A2, I won't discuss them in this documentation, for that I would refer to the documentation of A2. Instead, I will focus on the adjustments that were made to the code of A2 such that it uses JAX-WS instead of CORBA.

## Differences:

### Client setup:

Below is the code which would connect the client and ClientIDListServer in CORBA:

```
String[] orbArgs = {"-ORBInitialHost", "localhost", "-ORBInitialPort", "1050"};
ORB orb = ORB.init(orbArgs, null);


org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);


// Access ClientIDList
ClientIDListInterface clientIDList = ClientIDListInterfaceHelper.narrow(ncRef.resolve_str("ClientIDList"));
```

However, in JAX-WS this would connect the client to the ClientIDListServer:

```
URL clientIDListURL = new URL("http://localhost:8080/ClientIDListServer?wsdl");
QName clientIDListQName = new QName("http://example.com/ClientIDList", "ClientIDListServerService");
//specify the unique identifier for the ClientIDListServiceImplService service to be referenced by the Service object
Service clientIDListService = Service.create(clientIDListURL, clientIDListQName); //here the Service object acts as a factory for creating service stubs that can communicate with the actual web service
ClientIDListInterface clientIDListStub = clientIDListService.getPort(ClientIDListInterface.class);
```

Here we first create a URL which points to WSDL document for the web service at the specified location. This is so that the client can understand (from the document) how to interact with the web service/server.

Then we create a Qualified Name as global unique identifier for the service.

After this we create the clientIDListService object which can be used to interact with the web service, in this case we use it to create a stub as is described next.

Finally, we create the clientIDListStub to act as a client to the web service to invoke methods that were defined in the ClientIDListInterface. GetPort returns a stub that implements ClientIDListInterface.

And this is how we set up the ClientIDListServer (leaving out the overriding of the interface's methods):

```


@WebService(endpointInterface = "ClientIDListInterface", targetNamespace = "http://example.com/ClientIDList")
public class ClientIDListServer implements ClientIDListInterface {
```

```
    private List<String> clientIDs;

    public ClientIDListServer() {
        this.clientIDs = new ArrayList<>();

    public static void main(String[] args) {
        try {
            System.out.println("Initializing the service...");
            ClientIDListServer server = new ClientIDListServer();
            System.out.println("Service initialized, publishing...");
            Endpoint.publish("http://localhost:8080/ClientIDListServer", server);
            System.out.println("ClientIDListServer Service is published!");

        } catch (Exception e) {
            System.err.println("Error occurred: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Here we first have the ClientIDListServer implement the interface by setting it as its endpoint. In that line we also define the namespace for the service. This is used by the client to identify the service and connect to it (the QName).

In the try block we then create an instance of the server and publish it at the specified URL.

This is how the client connects to its respective server (MTL, SHE, or QUE) in CORBA:

```
ServerInterface stub = ServerInterfaceHelper.narrow(ncRef.resolve_str(clientId.substring(0, 3).toUpperCase()));
```

This is how it is done in JAX-WS:

```
String cityPrefix = clientId.substring(0, 3).toUpperCase();
String serverURL = getServerURL(cityPrefix);

//Connect to the appropriate server
QName serverQName = new QName("http://example.com/Server", "ServerService");
Service serverService = Service.create(new URL(serverURL), serverQName);
ServerInterface serverStub = serverService.getPort(ServerInterface.class);
```

We use the prefix of the client id to get the correct server URL, which is done in the getServerURL method which contains a switch case on the prefix.

Then we give a general QName to the server, note that using the same QName for the 3 servers doesn't matter since the unique URL is used for identifying each individual server.

### Main server setup:

In CORBA, we initialize the server's threads for client-server communication and inter-server communication like this:

```
Thread serverMTL = new Thread(() -> {
   String[] orbArgs = {"-ORBInitialHost", "localhost", "-ORBInitialPort", "1050"};
   ORB orb = ORB.init(orbArgs, null);
   new ServerThread(7777, "MTL", 7877, orb).run();
});
serverThreads.add(serverMTL);
serverMTL.start();
```

In JAX-WS I implemented it like this for each server:

```
Thread serverMTL = new Thread(new ServerThread(7777, "MTL",7877));
serverThreads.add(serverMTL);
serverMTL.start();
```

This is the same as what I did in RMI, the difference is seen in the section below.

This is the code that is executed when calling new ServerThread, which is different from the previous two assignments:

```
Server server = new Server(this.serverName,this.udpPort);


String url = "http://localhost:" + port + "/ServerService";
Endpoint.publish(url, server);
System.out.println("Server: " + this.serverName + " is running on port: " + this.port + " and UDP port: " +
this.udpPort);


//Start the UDP listener in a separate thread
server.startUDPListener();
```

As shown above, the server is started at the port number which was passed as an argument to the ServerThread object. The web service is started and available for clients to interact with after the publish() method is called.

The line below is inserted at the start of the file, just like we did for the ClientIDListServer. Its use was described earlier.

```
@WebService(endpointInterface = "ServerInterface", targetNamespace = "http://example.com/Server")
```

## Interfaces:

In this implementation I did not have to consider idl files which made the structure of the interfaces quite similar to those of the first assignment.

The only things I did for the 2 interfaces respectively were:

```
@WebService(targetNamespace = "http://example.com/Server")
```

Which marks the class as a web service endpoint, making it available for communication between the client and the server. The same reason as to why this is done in the implementation itself.

And I also had to add @WebMethod, before every method to make it accessible for implementation.

## Running the code:

When I ran the code in CORBA it was more complicated since it had to run all of these commands:

1. Run “idlj -fall ServerInterface.idl” to compile the interfaces.
2. Run “tnameserv -ORBInitialPort 1050” to start the Name Service such that it can be connected to on port 1050.
3. Run ServerBase.java and ClientIDListServer.java. Order doesn’t matter.
4. Run Client.java.

For JAX-WS it was just like for RMI, namely:

1. Run ClientIDListServer.java and Server.java. The order doesn’t matter.
2. Then run Client.java.

## Concurrency:

I made no changes to the implementation of concurrency itself. However, the ConcurrencyTest.java file was changed to adjust to JAX-WS. Below the results of running this test can be seen. They are the same as they were for A2, meaning they are still correct. In A2 it was:

```
Client MTLR1005remove: Resource couldn't be removed
Client MTLR1005 add1: Resource added successfully
Client MTLR1005 add2: Resource added successfully
Client MTLR1005 req: Resource was given
Client MTLR1005 swap: Successfully swapped resources
Client MTLR1001remove: Resource couldn't be removed
Client MTLR1001 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1001 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1001 req: Resource was given
Client MTLR1001 swap: Successfully swapped resources
Client MTLR1002remove: Resource couldn't be removed
Client MTLR1002 add1: Duration increased successfully
Client MTLR1002 add2: Duration increased successfully
Client MTLR1002 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1002 swap: No success in swapping resources
Client MTLR1008remove: Resource couldn't be removed
Client MTLR1008 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1008 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1008 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1008 swap: No success in swapping resources
Client MTLR1004remove: Resource couldn't be removed
Client MTLR1004 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1004 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1004 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1004 swap: No success in swapping resources
Client MTLR1007remove: Resource couldn't be removed
Client MTLR1007 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1007 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1007 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1007 swap: No success in swapping resources
Client MTLR1000remove: Resource couldn't be removed
Client MTLR1000 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1000 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1000 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1000 swap: No success in swapping resources
Client MTLR1006remove: Resource couldn't be removed
Client MTLR1006 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1006 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1006 req: Resource was given
Client MTLR1006 swap: Successfully swapped resources
Client MTLR1009remove: Resource couldn't be removed
Client MTLR1009 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1009 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1009 req: Resource was given
Client MTLR1009 swap: Successfully swapped resources
Client MTLR1003remove: Resource couldn't be removed
Client MTLR1003 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1003 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1003 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1003 swap: No success in swapping resources
All clients finished.
```

Now it is still:

```
Client MTLR1000 remove: Resource couldn't be removed
Client MTLR1000 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1000 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1000 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1000 swap: No success in swapping resources
Client MTLR1008 remove: Resource couldn't be removed
Client MTLR1008 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1008 add2: Resource added successfully
Client MTLR1008 req: Resource was given
Client MTLR1008 swap: Successfully swapped resources
Client MTLR1002 remove: Resource couldn't be removed
Client MTLR1002 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1002 add2: Duration increased successfully
Client MTLR1002 req: Resource was given
Client MTLR1002 swap: Successfully swapped resources
Client MTLR1003 remove: Resource couldn't be removed
Client MTLR1003 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1003 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1003 req: Resource was given
Client MTLR1003 swap: Successfully swapped resources
Client MTLR1009 remove: Resource couldn't be removed
Client MTLR1009 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1009 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1009 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1009 swap: No success in swapping resources
Client MTLR1004 remove: Resource couldn't be removed
Client MTLR1004 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1004 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1004 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1004 swap: No success in swapping resources
Client MTLR1006 remove: Resource removed
Client MTLR1006 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1006 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1006 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1006 swap: No success in swapping resources
Client MTLR1007 remove: Resource couldn't be removed
Client MTLR1007 add1: Duration increased successfully
Client MTLR1001 remove: Resource couldn't be removed
Client MTLR1001 add1: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1001 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1001 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1001 swap: No success in swapping resources
Client MTLR1007 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1007 req: Resource not available. Would you like to be added to the queue? (yes/no)
Client MTLR1007 swap: No success in swapping resources
Client MTLR1005 remove: Resource couldn't be removed
Client MTLR1005 add1: Resource added successfully
Client MTLR1005 add2: Resource couldn't be added since given duration was lower than duration for already present resource
Client MTLR1005 req: Resource was given
Client MTLR1005 swap: Successfully swapped resources
All clients finished.
```

To clarify, in both cases there were:

2 cases where adding the resource was successful, 0 where removing the resource was successful, 4 where requesting a resource was successful, and 4 where swapping the requested resource was successful.