University of Westminster
Faculty of Science and Technology

Module code and title: 5COSC004W-Client Service Architecture Tutorial Manual	
Tutorial title	**Getting acquainted with the concepts and procedures of the Coding Exercise One (Building a Client from a Server)**
Tutorial type	**Guided and indepenent and non-marked**
Week 07	**14/03/19**

Contents

Learning Goals

This tutorial focuses on two main learning goals:

- to get acquainted with the the concepts of the first Coding Exercise (create a client to connect to an exhisting server)

It is divided into two separate sections, the student will perform the first task (1-X) following the instructions of the tutor, and then, will complete the other tasks independently.

TASKS to be Performed under the instruction of the Tutor (from Task 1 to Task 8)

1) Start Netbeans in your system. If Netbeans is not present in your system, use AppsAnywhere to launch it: (https://www.westminster.ac.uk/sites/default/public-files/general-documents/Using%20AppsAnywhere.pdf)
2) Download the Server Project from Blackboard (https://learning.westminster.ac.uk/bbcswebdav/pid-2439314-dt-content-rid-16393746_1/xid-16393746_1) and import it into your Netbeans (Figure 1)

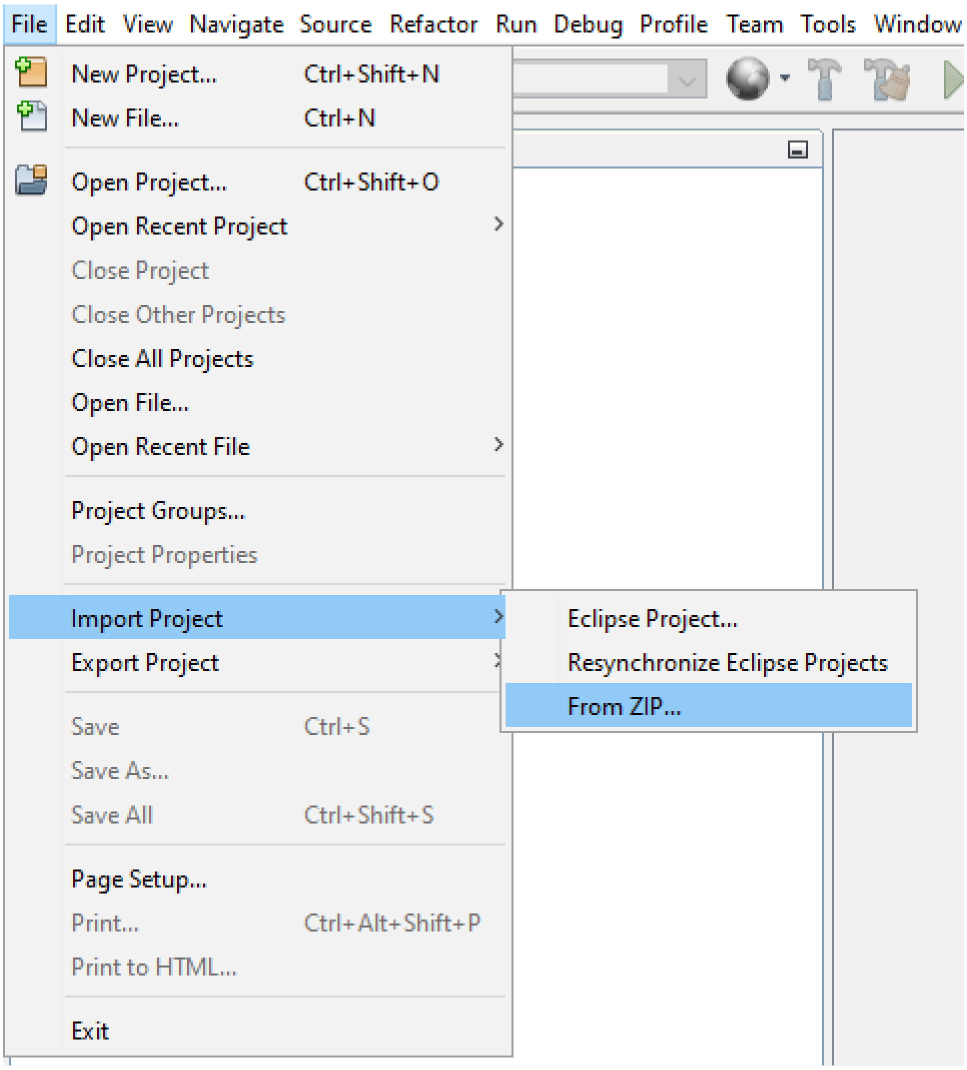
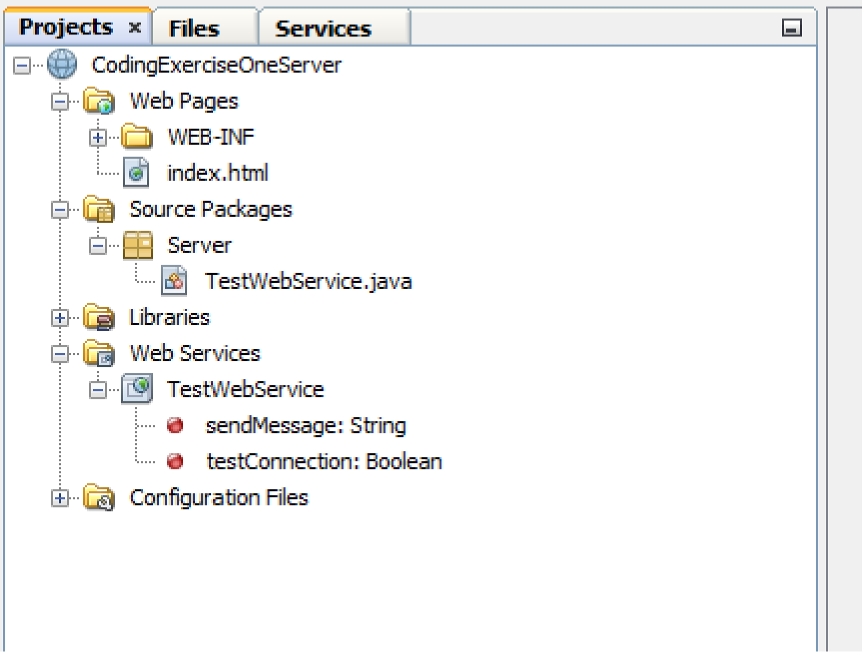


Figure 1, Importing the server project

3) As you can observe in Figure 2, the server you have imported has two methods:
    a. String sendMessage(String arg) which returns a String that contains arg
    b. Boolean testConnection() which returns true.



```
CodingExerciseOneServer
    Web Pages
        WEB-INF
        index.html
    Source Packages
        Server
            TestWebService.java
    Libraries
    Web Services
        TestWebService
            sendMessage: String
            testConnection: Boolean
    Configuration Files
```

Figure 2, Structure of the Server Project

4) Test if the Web Service works. (Figure 3 and Figure 4)

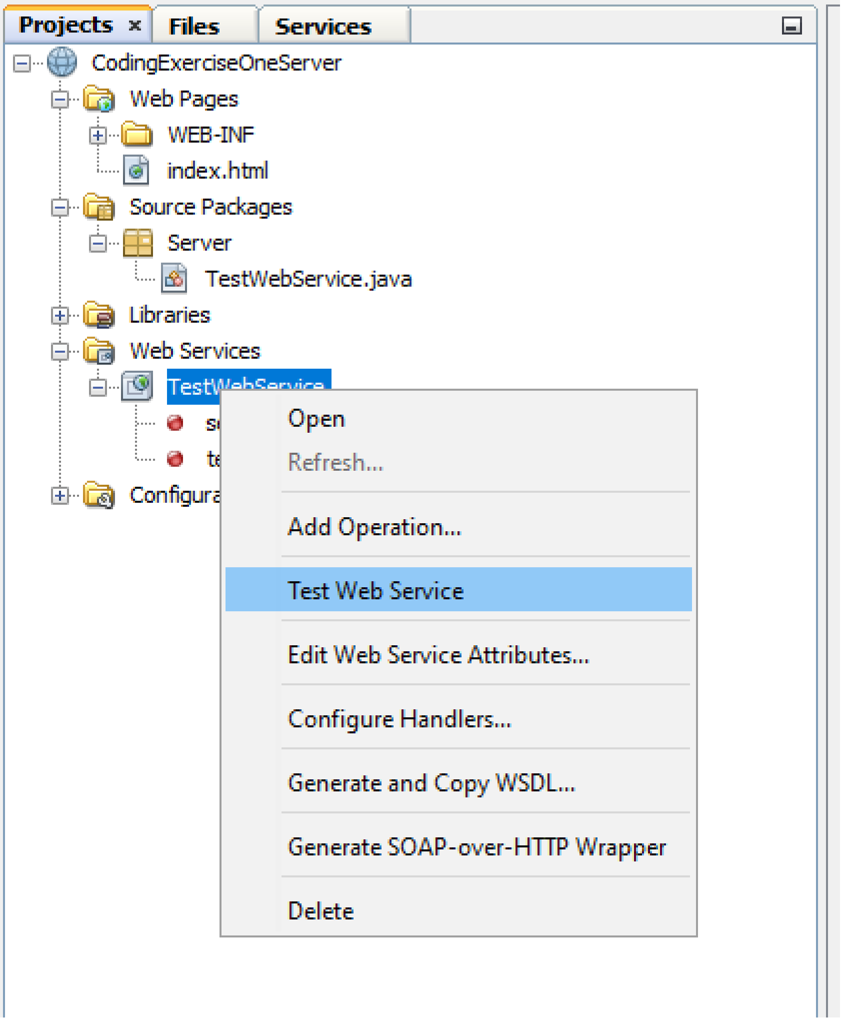
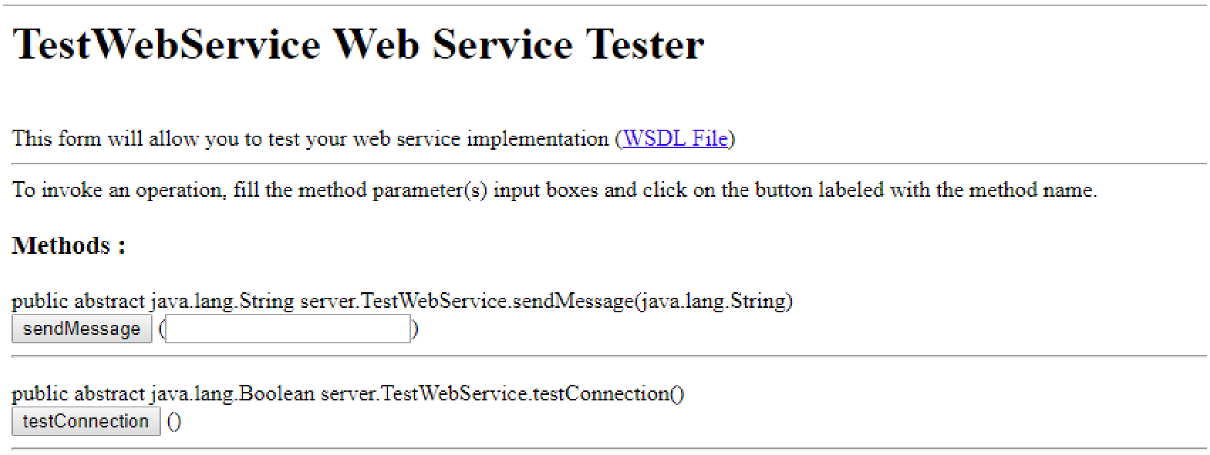


Figure 3, Testing the Web Service A



TestWebService Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String server.TestWebService.sendMessage(java.lang.String)
sendMessage ( )

public abstract java.lang.Boolean server.TestWebService.testConnection()
testConnection ()

Figure 4, Testing the Web Service B

5) Now we have to create a new Java Project to implement the Client.

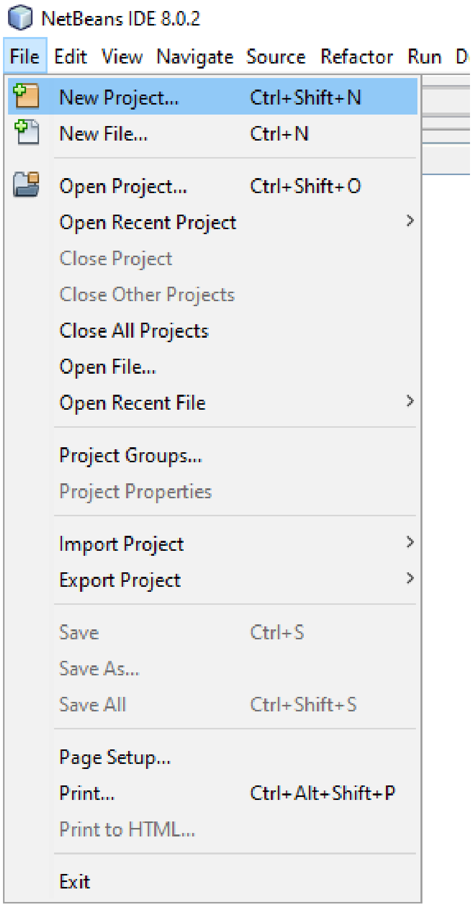
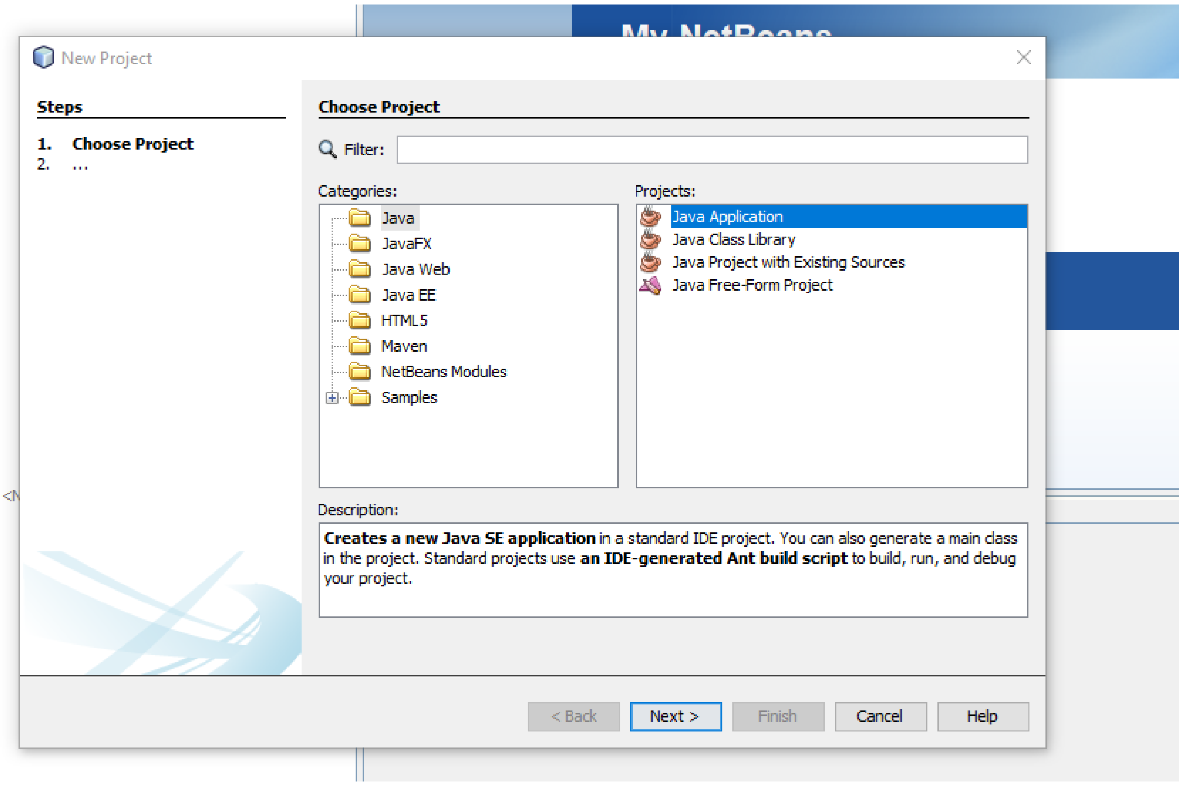
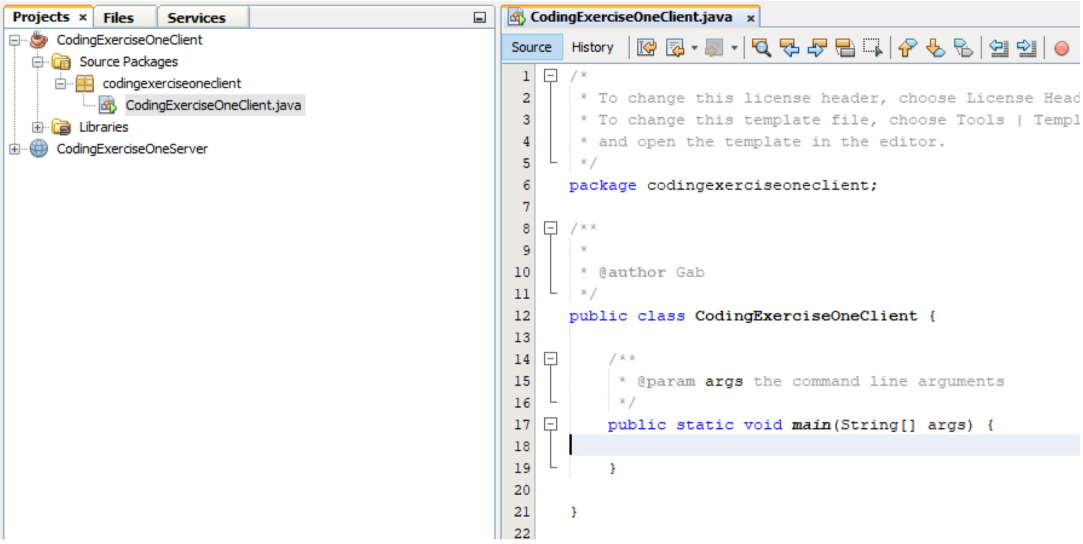


Figure 5, Create Client Project A



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

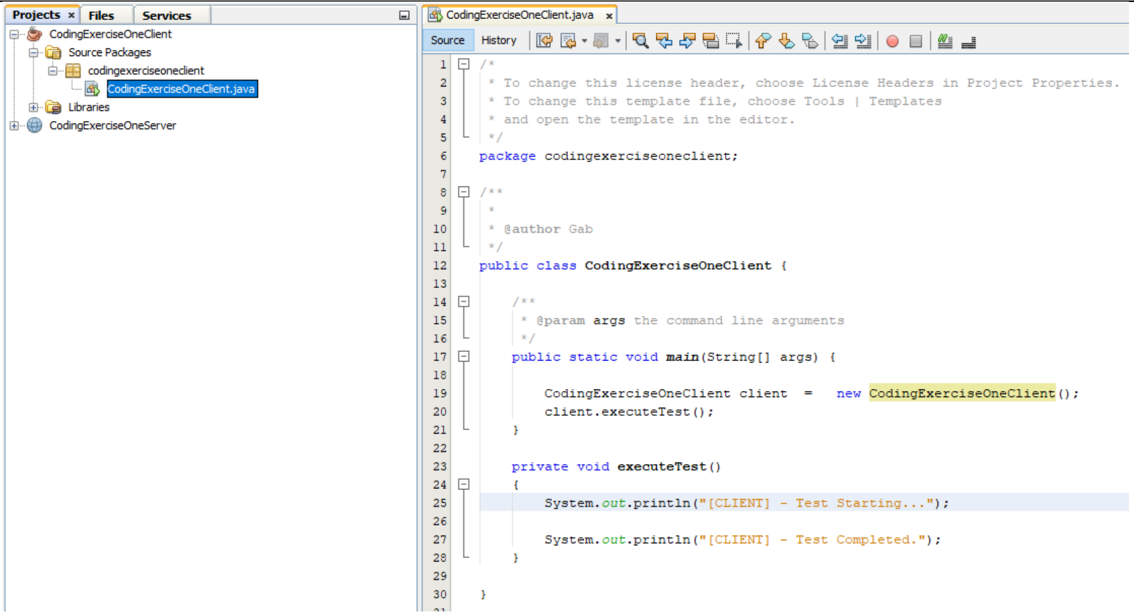
Figure 6, Create Client Project B



```
/*
 * To change this license header, choose License Head
 * To change this template file, choose Tools | Templ
 * and open the template in the editor.
 */
package codingexerciseoneclient;

/**
 *
 * @author Gab
 */
public class CodingExerciseOneClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

    }

}
```

Figure 7, Create Client Project C

6) Create the usual executeTest method and add the usual logging lines (Figure 8)



```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package codingexerciseoneclient;

/**
 *
 * @author Gab
 */
public class CodingExerciseOneClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        CodingExerciseOneClient client  =   new CodingExerciseOneClient();
        client.executeTest();
    }

    private void executeTest()
    {
        System.out.println("[CLIENT] - Test Starting...");

        System.out.println("[CLIENT] - Test Completed.");
    }

}
```

Figure 8, Client Structure

7) Now we want to add the client stubs: the methods that will handle the connection with the server. (Figure 9 and Figure 10)

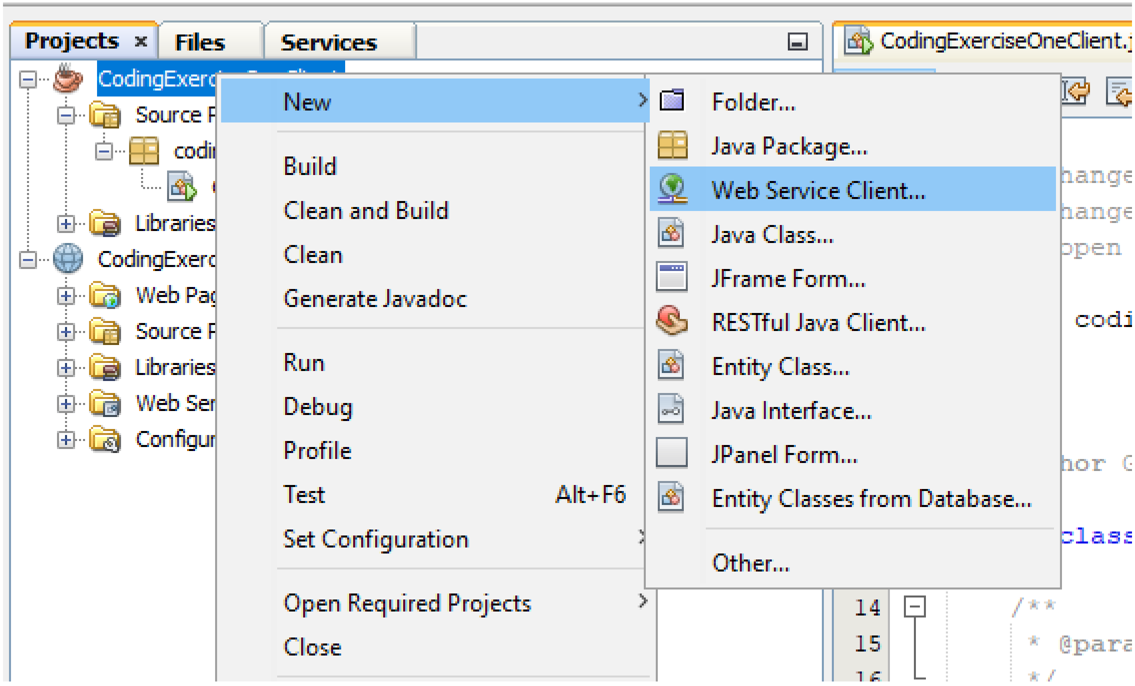
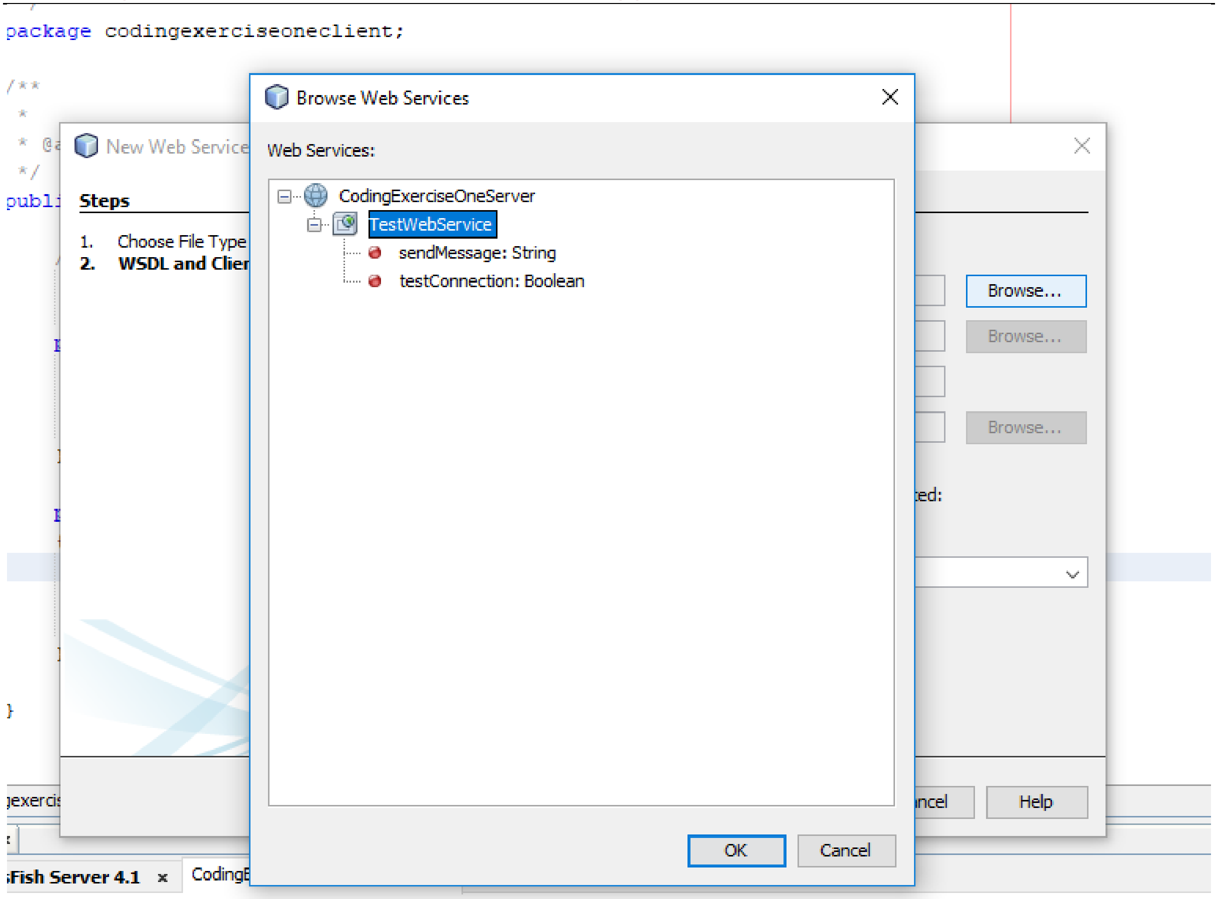


Figure 9, Create Client Side Stubs (A)



```
package codingexerciseoneclient;
```

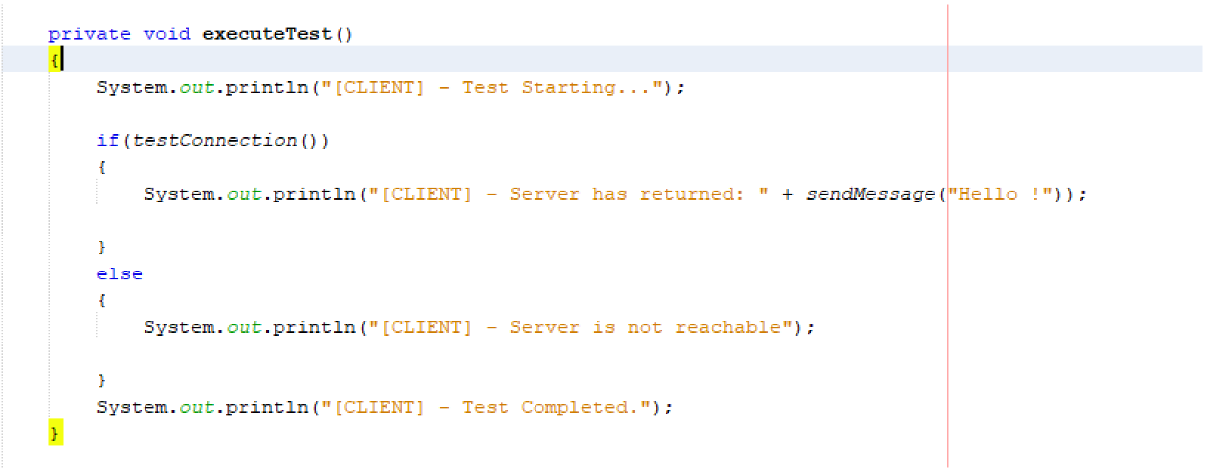
Browse Web Services

Web Services:

- CodingExerciseOneServer
  - TestWebService
    - sendMessage: String
    - testConnection: Boolean

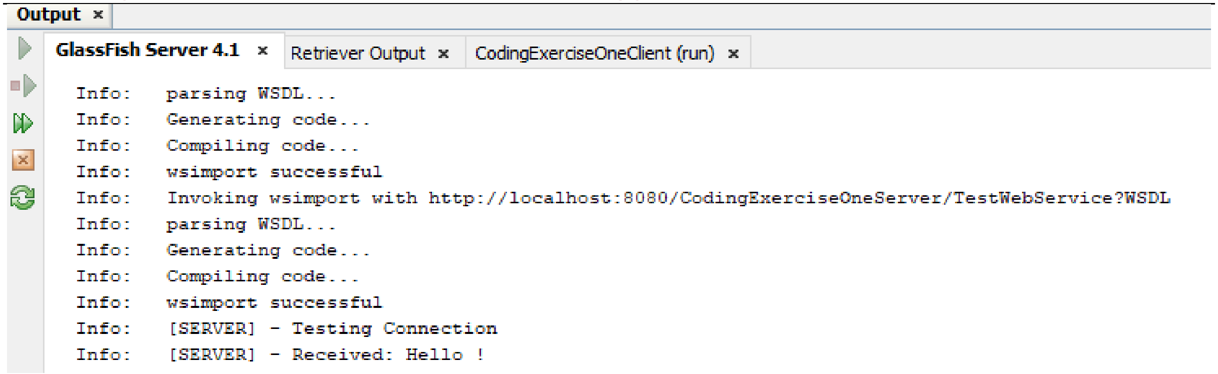
Figure 10, Create Client Side Stubs (B)

8) Drag and drop the client stubs into the code and invoke them (Figure 11 and Figure 12).



```
private void executeTest()
{
    System.out.println("[CLIENT] - Test Starting...");

    if(testConnection())
    {
        System.out.println("[CLIENT] - Server has returned: " + sendMessage("Hello !"));

    }
    else
    {
        System.out.println("[CLIENT] - Server is not reachable");

    }
    System.out.println("[CLIENT] - Test Completed.");
}
```

Figure 11, Complete the Client and Test it (A)



```
Info:   parsing WSDL...
Info:   Generating code...
Info:   Compiling code...
Info:   wsimport successful
Info:   Invoking wsimport with http://localhost:8080/CodingExerciseOneServer/TestWebService?WSDL
Info:   parsing WSDL...
Info:   Generating code...
Info:   Compiling code...
Info:   wsimport successful
Info:   [SERVER] - Testing Connection
Info:   [SERVER] - Received: Hello !
```

Figure 12, Complete the Client and Test it (B)

TASKS to BE PERFORMED Independently be the student (from Task 9 to Task 11) (Formative Assessment)

9) Modify the testConnection method so that the client can send its id (as a string) and that is returned from the method (e.g. Connection from client ….. succeded)
10) Modify the Server Class so that the server has a name (as a stgring) so that the returned string from the method testConnection is (e.g. Server….. : Connection from client ….. succeded)
11) Modify the Server Class so that the server can add a time stamp to method testConnection (e.g. [Date and Time] - Server….. : Connection from client ….. succeded)