University of Westminster
Faculty of Science and Technology

Module code and title: 5COSC004W-Client Service Architecture Tutorial Manual

Tutorial title	Example of Coding Exercise (CEX)
Tutorial type	Guided and indepenent and non-marked
Week 08	12/03/2020

Contents

Learning Goals

This tutorial focuses on one main learning goals:

- To acquaint the student with the type of coding exercise at the end of the year. The tutorial does not reflect fully the exam (logic is implemented in the DummyServer while it will not in the CEX and no grading) but it gives an idea of the process. Next tutorial will fully mimic the coding exercise.

It is divided into two separate sections, the student will perform the first task (1-15) following the instructions of the tutor, and then, will complete the other tasks (15 to 16) independently.

TASKS to be Performed under the instruction of the Tutor (from Task 1 to Task 13)

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 file Tutorial8DummyProject.zip in your documents folder.

3) Import the project Tutorial8DummyProject.zip (Figure 1 to Figure 4)

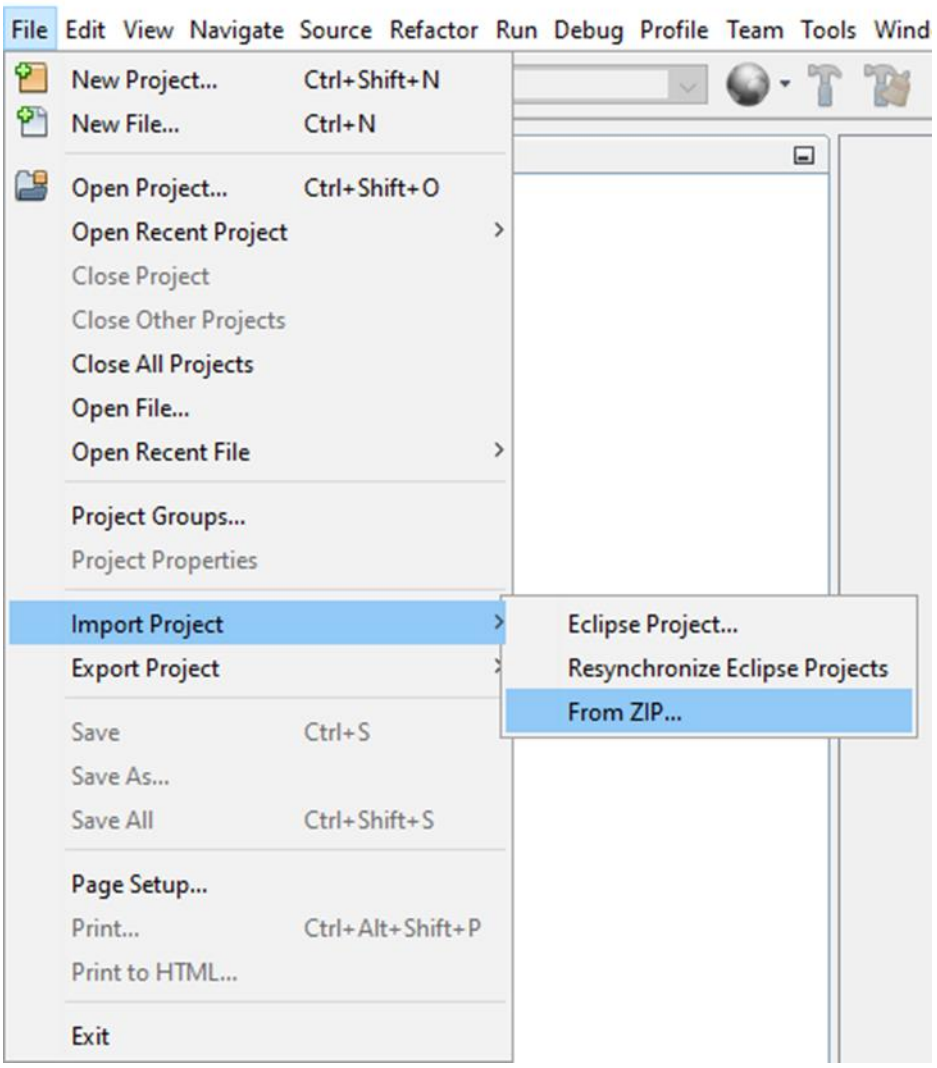
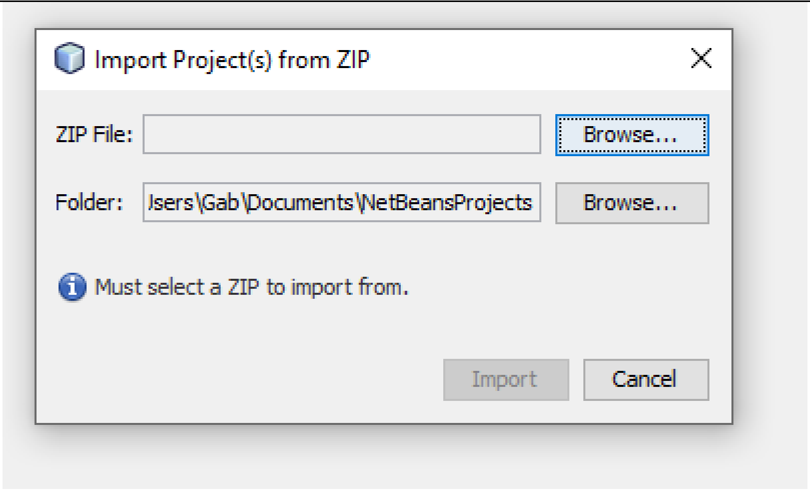


Figure 1, Importing Tutorial8DummyProject (1/4)



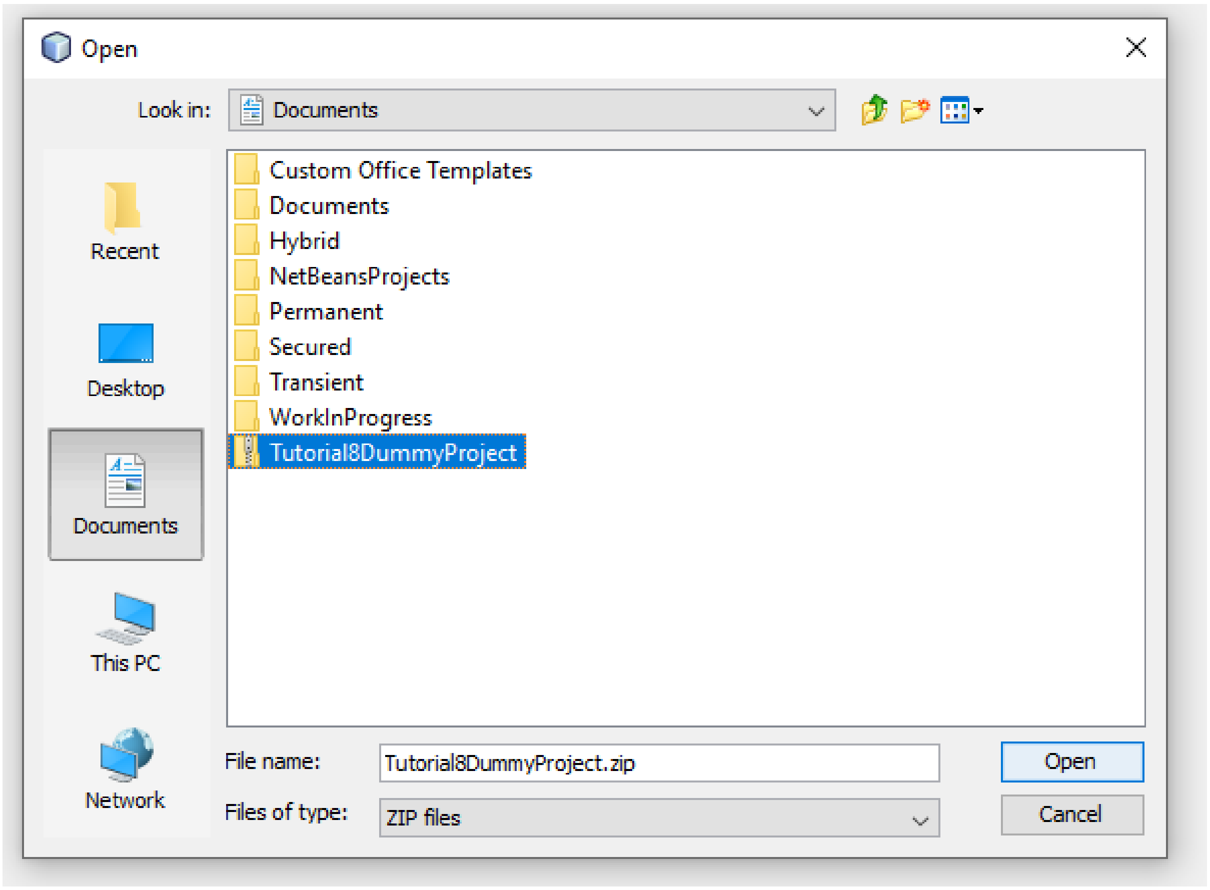
Import Project(s) from ZIP

ZIP File:

Folder: Jsers\Gab\Documents\NetBeansProjects

Must select a ZIP to import from.

Figure 2, Importing Tutorial8DummyProject (2/4)



Open

Look in: Documents

Custom Office Templates
Documents
Hybrid
NetBeansProjects
Permanent
Secured
Transient
WorkInProgress
Tutorial8DummyProject

File name: Tutorial8DummyProject.zip

Files of type: ZIP files

Figure 3, Importing Tutorial8DummyProject (3/4)

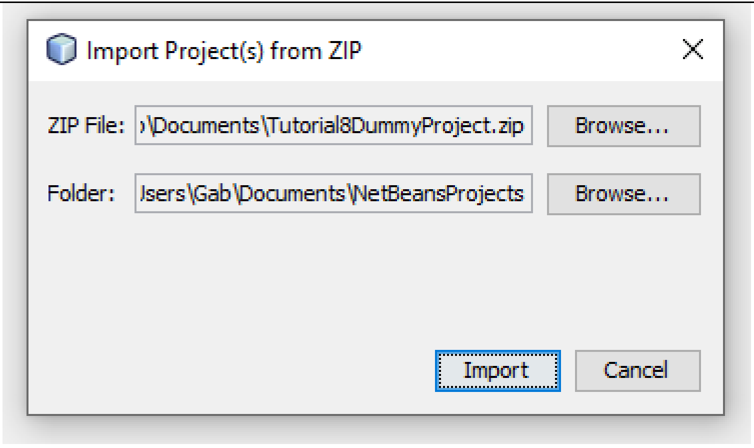

Import Project(s) from ZIP
ZIP File: ›\Documents\Tutorial8DummyProject.zip
Browse...
Folder: Jsers\Gab\Documents\NetBeansProjects
Browse...
Import
Cancel

Figure 4, Importing Tutorial8DummyProject (4/4)

4) This will create a Java project in your Netbeans which contains a DummyClient and a DummyServer. They both contain the logic of the client and the server **(for the real CEX exam, you will have to implement the logic).**
Let's take a look at the project. The project contains the classes listed in Figure 5

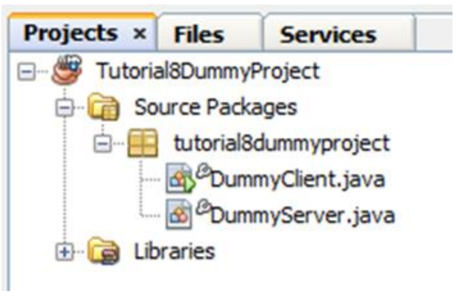


Figure 5, Classes of the Tutorial8DummyProject

5) The DummyServer contains the methods listed in **(for the real CEX exam, you will have to implement the logic).** The full documentation of the methods is available in Appendix A

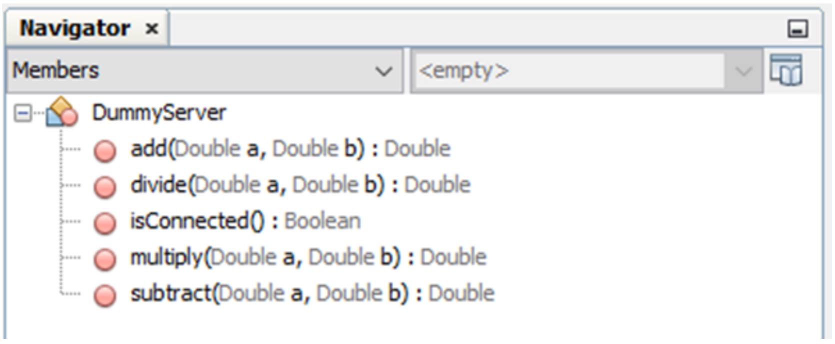

Navigator ×
Members
<empty>
DummyServer
add(Double a, Double b) : Double
divide(Double a, Double b) : Double
isConnected() : Boolean
multiply(Double a, Double b) : Double
subtract(Double a, Double b) : Double

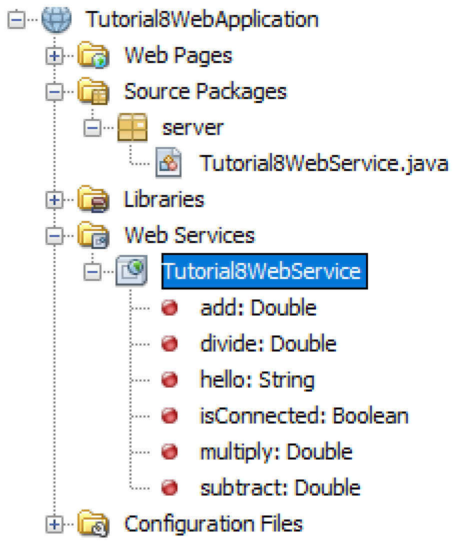
Figure 6, Methods of the DummyServer

6) Observe the methods on the DummyServer and the logic implemented in the DummyClient. Execute the DummyClient to observe its behaviour.

7) Now, this, is what you will receive as input for the Coding Exercise **(with the difference that in the Coding Exercise there will be no implementation of the logic but only the methods signatures)**

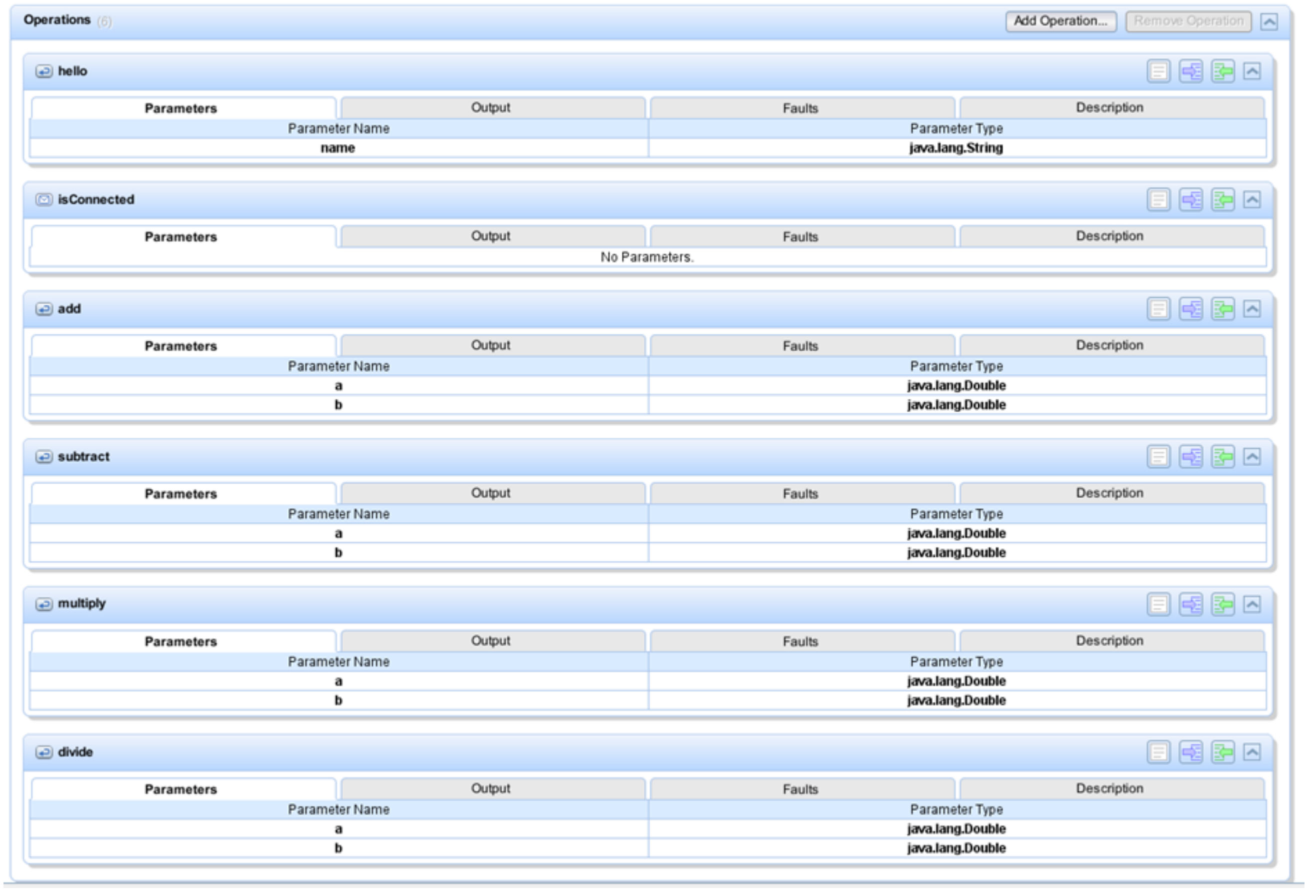
8) The first thing you must do during the coding exercise is create a Web Service Server, let's call it Tutorial8WebService. **(If you do not know how to create a Web Service, re-do Tutorial5.)**

9) The real server will have the structure of Figure 7 and will expose the WebMethods listed in Figure 8.



```
Tutorial8WebApplication
  Web Pages
  Source Packages
    server
      Tutorial8WebService.java
  Libraries
  Web Services
    Tutorial8WebService
      add: Double
      divide: Double
      hello: String
      isConnected: Boolean
      multiply: Double
      subtract: Double
  Configuration Files
```

Figure 7, Structure of the Web Service Server


Operations (6)

Add Operation... | Remove Operation

hello

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
name	java.lang.String

isConnected

Parameters	Output	Faults	Description

No Parameters.

add

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
a	java.lang.Double
b	java.lang.Double

subtract

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
a	java.lang.Double
b	java.lang.Double

multiply

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
a	java.lang.Double
b	java.lang.Double

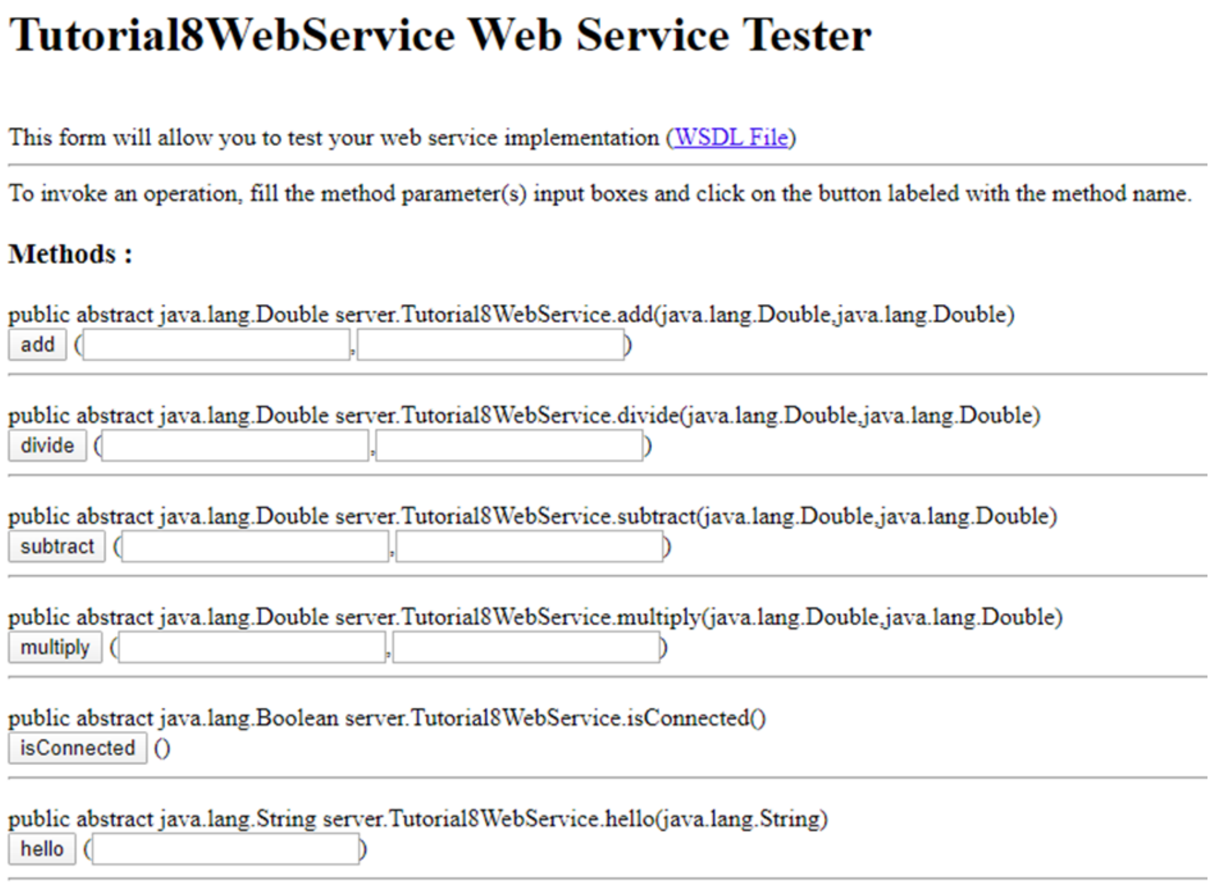
divide

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
a	java.lang.Double
b	java.lang.Double

Figure 8, Web Methods of the Web Service Server

10) You can test the correcteness of the logic of the Server by invoking the simple testing interface as in Figure 9. **(If you do not know how to test a Web Service, re-do Tutorial5.)**



Tutorial8WebService 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.Double server.Tutorial8WebService.add(java.lang.Double,java.lang.Double)

public abstract java.lang.Double server.Tutorial8WebService.divide(java.lang.Double,java.lang.Double)

public abstract java.lang.Double server.Tutorial8WebService.subtract(java.lang.Double,java.lang.Double)

public abstract java.lang.Double server.Tutorial8WebService.multiply(java.lang.Double,java.lang.Double)

public abstract java.lang.Boolean server.Tutorial8WebService.isConnected()

public abstract java.lang.String server.Tutorial8WebService.hello(java.lang.String)

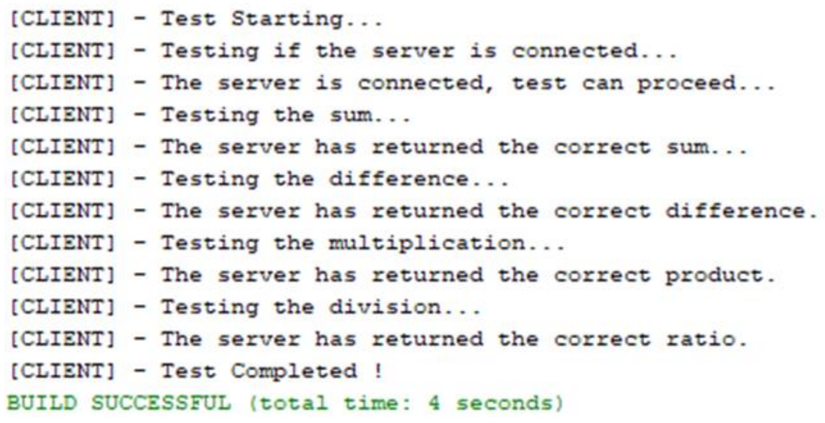
Figure 9, Simple Testing Interface.

11) Create the client project and call it Tutorial8Client, name the main class Tutorial8Client.

12) Create the client stubs for the Tutorial8Server. **<u>(If you do not know how to create a Client Stub, re-do Tutorial5)</u>**

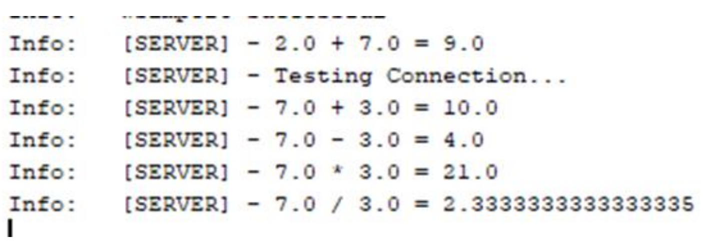
13) Implement the logic on the client, you can re-use most of the DummyClient taking into account that you must substitute the DummyServer instance with the Client Stubs.

14) Execute the Client, you should see the separate outputs (for the Client and the Server) of



```
[CLIENT] - Test Starting...
[CLIENT] - Testing if the server is connected...
[CLIENT] - The server is connected, test can proceed...
[CLIENT] - Testing the sum...
[CLIENT] - The server has returned the correct sum...
[CLIENT] - Testing the difference...
[CLIENT] - The server has returned the correct difference.
[CLIENT] - Testing the multiplication...
[CLIENT] - The server has returned the correct product.
[CLIENT] - Testing the division...
[CLIENT] - The server has returned the correct ratio.
[CLIENT] - Test Completed !
BUILD SUCCESSFUL (total time: 4 seconds)
```

Figure 10, Client Execution Output



```
Info:   [SERVER] - 2.0 + 7.0 = 9.0
Info:   [SERVER] - Testing Connection...
Info:   [SERVER] - 7.0 + 3.0 = 10.0
Info:   [SERVER] - 7.0 - 3.0 = 4.0
Info:   [SERVER] - 7.0 * 3.0 = 21.0
Info:   [SERVER] - 7.0 / 3.0 = 2.3333333333333335
```

Figure 11, Server Execution Output

TASKS to BE PERFORMED Independently be the student (from Task 15 to Task 16) (Formative Assessment)

15) Investigate how to implement exceptions on the real Client/Server environment.

16) Investigate how to implement methods that perform the four arithmetic operations on ArrayList of Double and not single numbers.

Appendix A – Dummy Server Documentation

```
public class DummyServer
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
DummyServer()

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
java.lang.Double	add(java.lang.Double a, java.lang.Double b)
java.lang.Double	divide(java.lang.Double a, java.lang.Double b)
java.lang.Boolean	isConnected()
java.lang.Double	multiply(java.lang.Double a, java.lang.Double b)
java.lang.Double	subtract(java.lang.Double a, java.lang.Double b)

Constructor Detail

DummyServer

```
public DummyServer()
```

Method Detail

isConnected

```
public java.lang.Boolean isConnected()
```

Returns:

true if the server is connected, false otherwise

add

```
public java.lang.Double add(java.lang.Double a,
                            java.lang.Double b)
```

Parameters:

a - the first addendum

b - the second addendum

Returns:

the sum of a and b

subtract

```
public java.lang.Double subtract(java.lang.Double a,
                                 java.lang.Double b)
```

Parameters:

a - the minuend

b - the subtrahend

Returns:

the difference of a and b

multiply

```
public java.lang.Double multiply(java.lang.Double a,
                                 java.lang.Double b)
```

Parameters:

a - the multiplicand

b - the multiplier

Returns:

the product of the a and b

divide

```
public java.lang.Double divide(java.lang.Double a,
                               java.lang.Double b)
```

Parameters:

a - the dividend

b - the divisor

Returns:

a divided by b