

Module code and title: 5COSC010C-Client Service Architecture
Tutorial Manual

Tutorial title	Methods, Types and Exceptions
Tutorial type	Guided and independent and non-marked
Week 02	28/01/2021

Contents

Learning Goals	1
TASKS to be Performed under the instruction of the Tutor (from Task 1 to Task 16)	1
TASKS to BE PERFORMED Independently by the student (from Task 22 to Task 27) (Formative Assessment).....	13

Learning Goals

This tutorial focuses on two main learning goals:

- to advance your basic java programming skills (specifically types and exceptions)
- to self-assess your advancement in your basic java skills

It is divided into two separate sections, the student will perform the first tasks (1-16) 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 16)

- 1) Start Netbeans in your system.
- 2) Create a new java project in Netbeans (Figure 1 to 3). We will call this project SimpleTestTwo and it will consist in a simple calculator based on a client/server dummy architecture. **Modify the name of the standard main class to SimpleClient** as in Figure 3

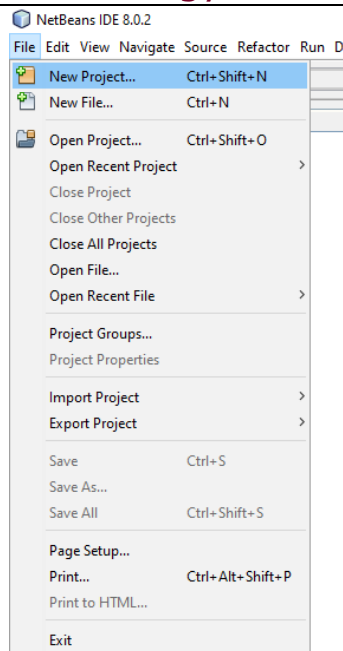


Figure 1, Create a New Project in NetBeans (1)

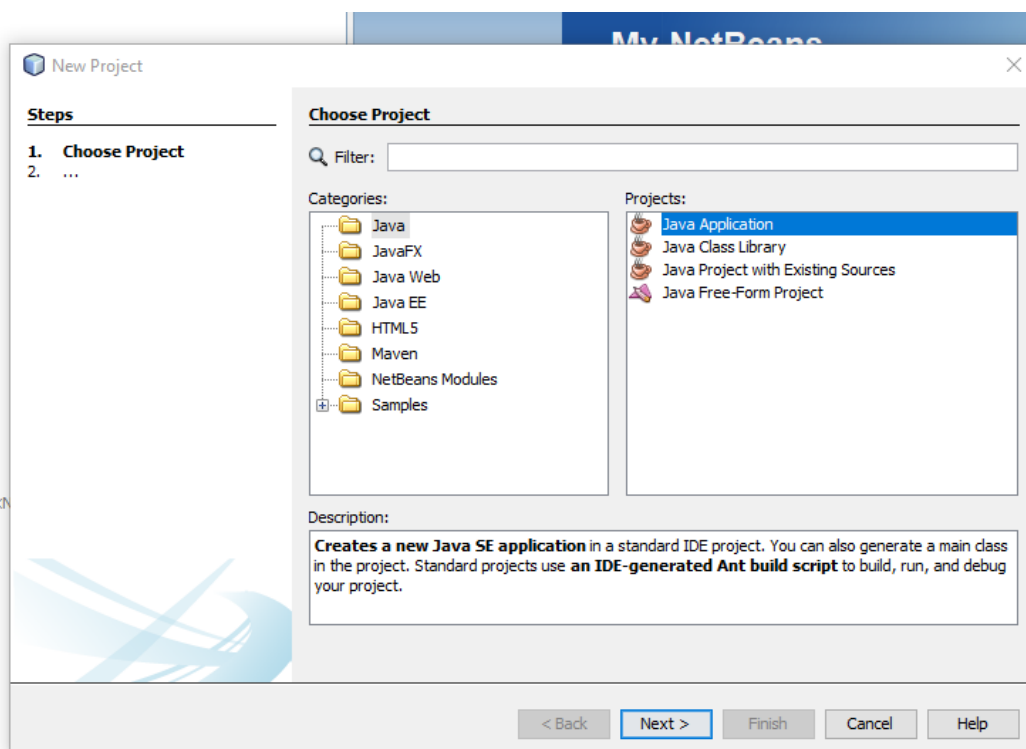


Figure 2, Create a New Project in NetBeans (2)

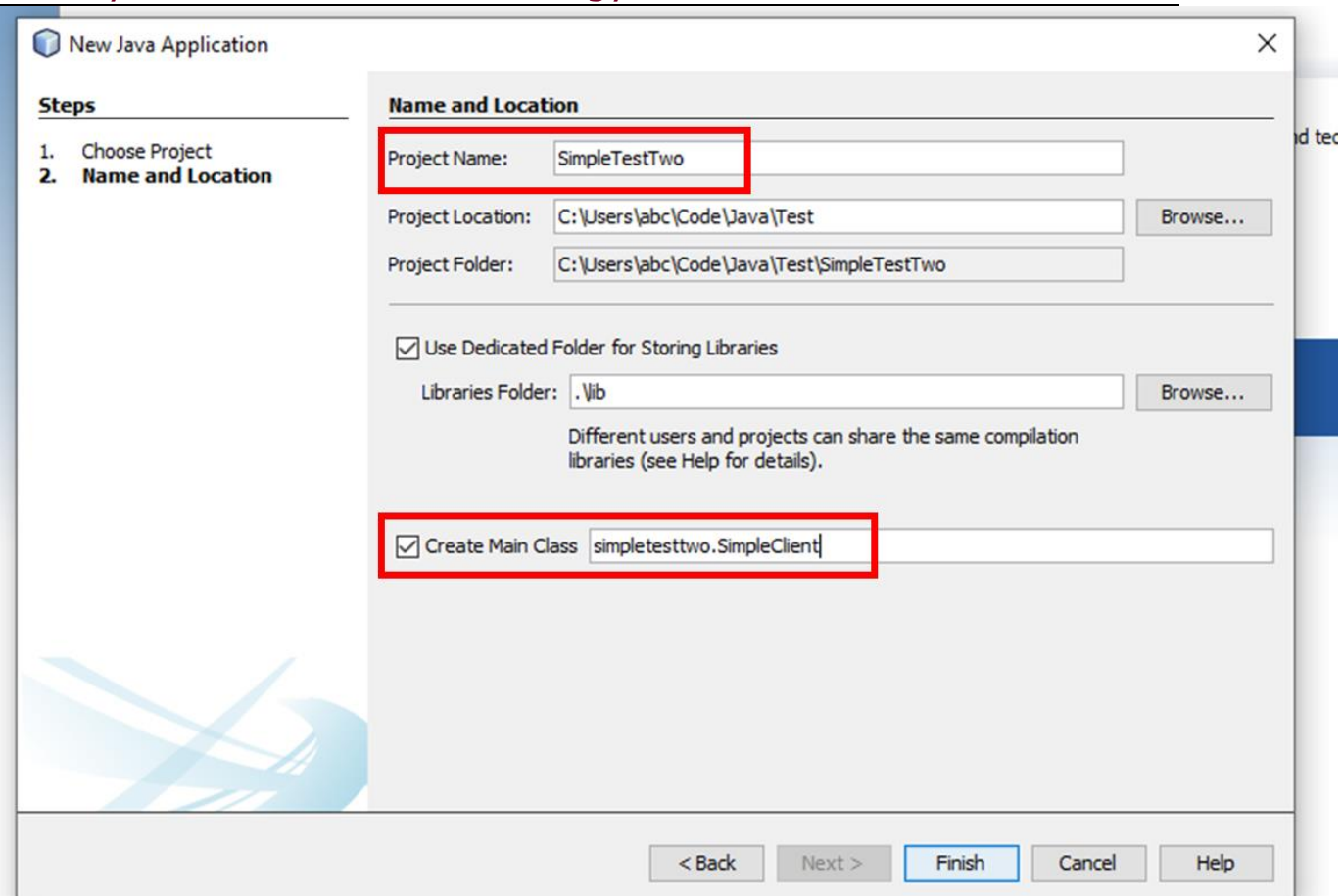


Figure 3, Create a New Project in NetBeans (3)

- 3) Create a new java class in the project which will represent our SimpleCalculatorServer (Figure 4 and Figure 5, Create a new Java Class in NetBeans(2))

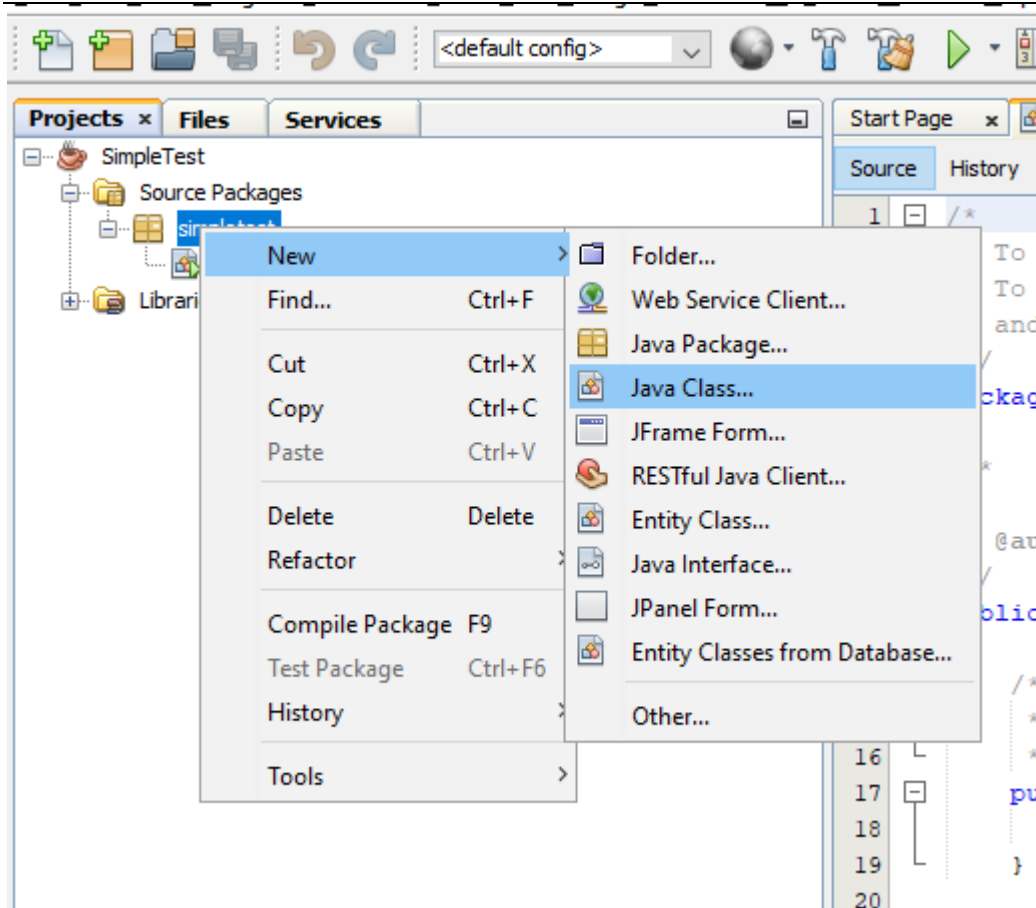


Figure 4, Create a new Java Class in NetBeans (1)

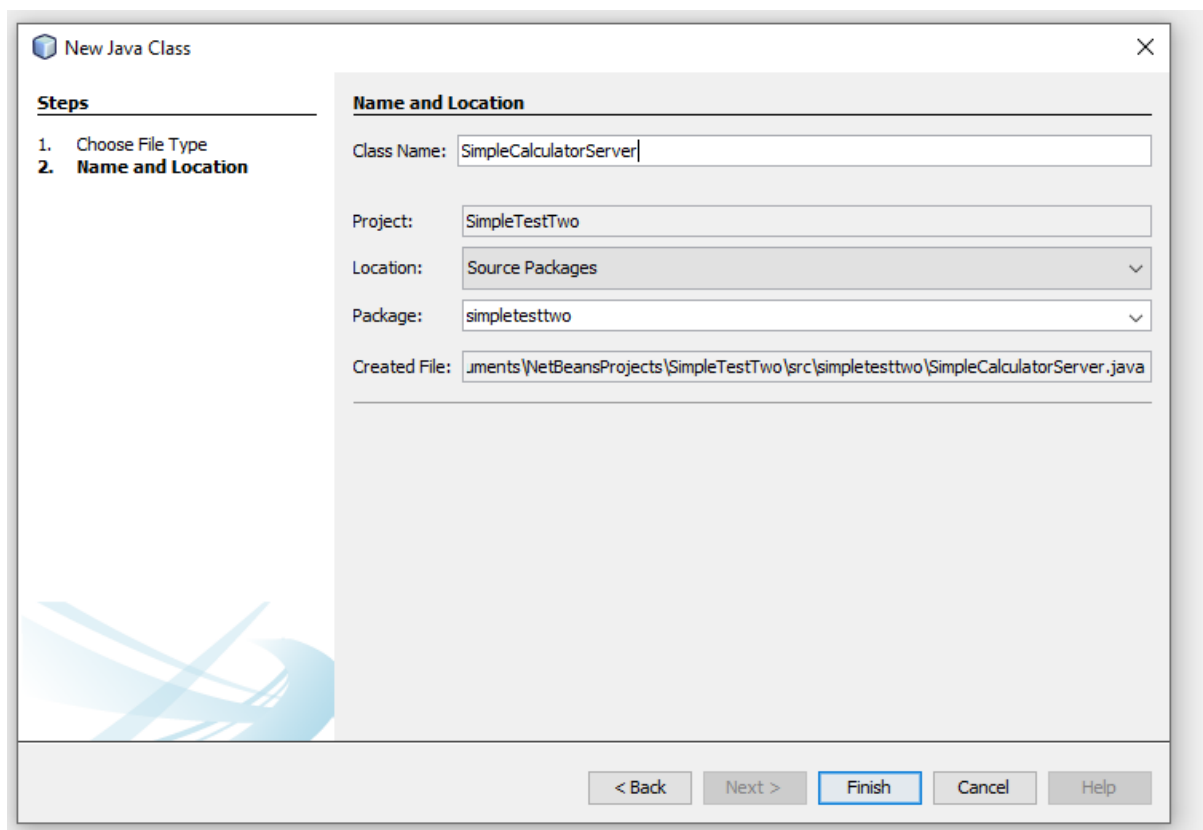


Figure 5, Create a new Java Class in NetBeans(2)

- 4) Add a method to the SimpleCalculatorServer that returns a simple true which we can test to see if the server is running (Figure 6)

```
8  /**
9  *
10 * @author Gab
11 */
12 public class SimpleCalculatorServer
13 {
14     public boolean isServerConnected()
15     {
16         System.out.println("[SERVER] - Connection is being tested...");
17         return true;
18     }
19
20
21 }
22
```

Figure 6, Add simple method to the SimpleCalculatorServer Class

- 5) Create an instance of the SimpleClient class and an empty method execute and add the invocation of the server in the execute method, add the invocation of the connection test (Figure 7). Also declare an instance of the SimpleCalculatorServer at Class Level as in Figure 6

```
/**
 *
 * @author Gab
 */
public class SimpleClient
{
    SimpleCalculatorServer server = new SimpleCalculatorServer();
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        SimpleClient simpleClient = new SimpleClient();
        simpleClient.executeTest();
    }

    private void executeTest()
    {
        System.out.println("[CLIENT] - Testing if the server is connected...");
        if(server.isServerConnected())
        {
            System.out.println("[CLIENT] - The Client is connected, the test can proceed...");
        }
        else
        {
            System.out.println("[CLIENT] - The Client is NOT connected, the test has failed...");
        }
    }
}
}
```

Figure 7, Create instance and execute method.

6) Run the client (Figure 8) to obtain the output described in Figure 9

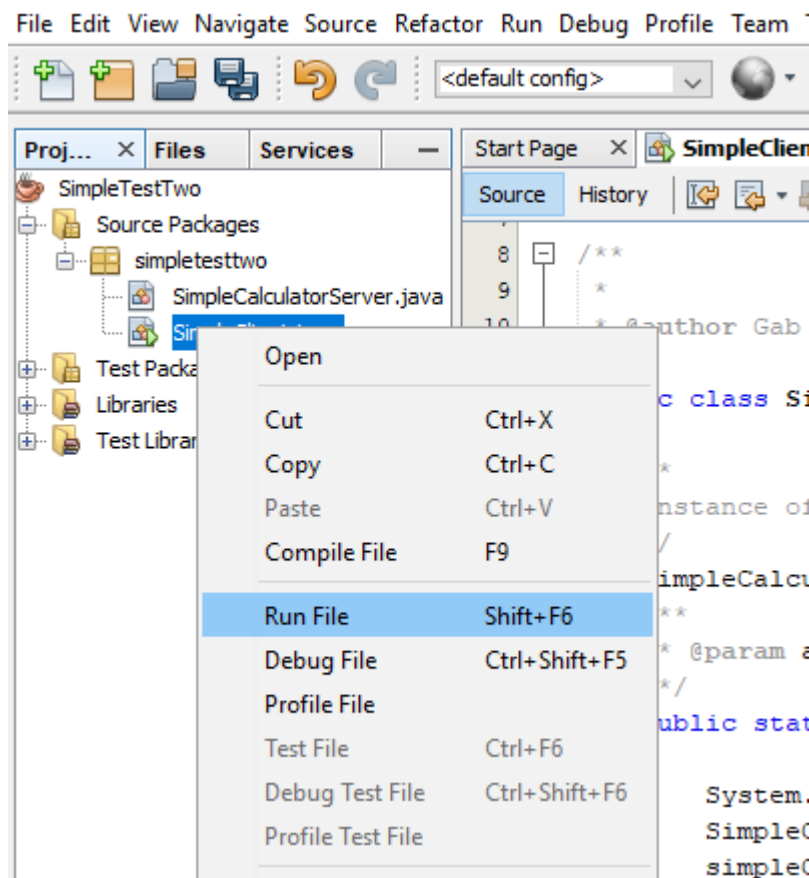


Figure 8, Run the Client (1)

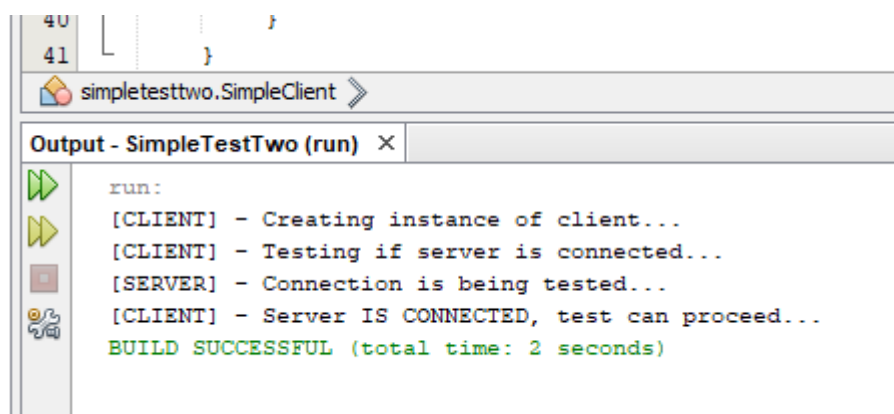


Figure 9, Client Execution Output

7) Add a method to the server that adds two integer numbers (Figure 10) with some simple logging information.

```
6 package simpletesttwo;
7
8 /**
9  *
10  * @author Gab
11  */
12 public class SimpleCalculatorServer
13 {
14     public boolean isServerConnected()
15     {
16         System.out.println("[SERVER] - Connection is being tested...");
17         return true;
18     }
19
20     public int addIntegerNumbers(int a, int b)
21     {
22         System.out.println("[SERVER] - Adding " + a + " and " + b);
23         return a+b;
24     }
25 }
```

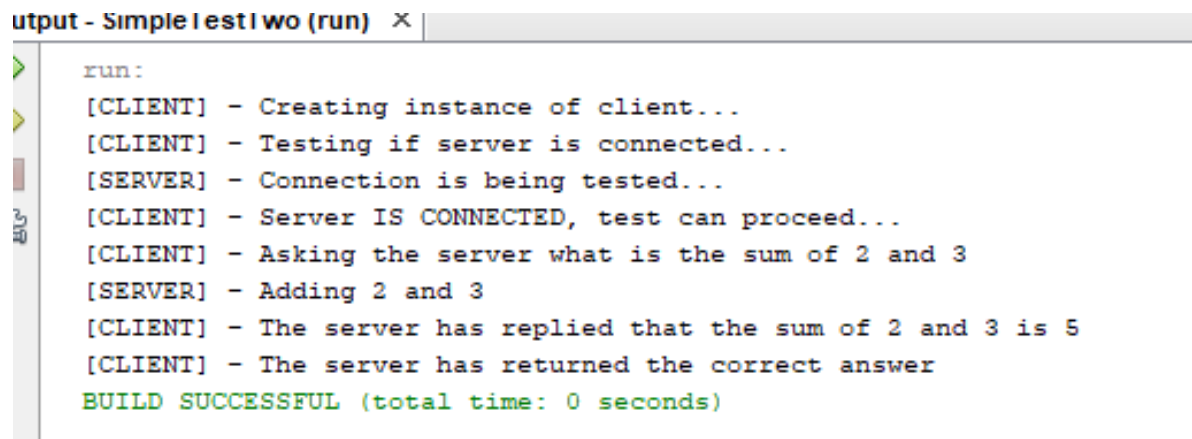
Figure 10, Add method that adds two numbers to the Server

- 8) Add the code on the client to invoke the addIntegerNumbers on the server and check that the result is correct (Figure 11)

```
private void execute()
{
    System.out.println("[CLIENT] - Testing if server is connected...");
    if(server.isServerConnected())
    {
        System.out.println("[CLIENT] - Server IS CONNECTED, test can proceed...");
        int x = 2;
        int y = 3;
        System.out.println("[CLIENT] - Asking the server what is the sum of " + x + " and " + y);
        int z = server.addIntegerNumbers(x, y);
        System.out.println("[CLIENT] - The server has replied that the sum of " + x + " and " + y + " is " + z);
        if(z == x+y)
            System.out.println("[CLIENT] - The server has returned the correct answer");
        else
            System.out.println("[CLIENT] - The server has returned the WRONG answer");
    }
    else
    {
        System.out.println("[CLIENT] - Server IS NOT CONNECTED, test CANNOT proceed...");
    }
}
```

Figure 11, Invoke the method that adds two numbers in the client

9) Execute the client and observe the output ()



```
Output - SimpleTestTwo (run) ×
run:
[CLIENT] - Creating instance of client...
[CLIENT] - Testing if server is connected...
[SERVER] - Connection is being tested...
[CLIENT] - Server IS CONNECTED, test can proceed...
[CLIENT] - Asking the server what is the sum of 2 and 3
[SERVER] - Adding 2 and 3
[CLIENT] - The server has replied that the sum of 2 and 3 is 5
[CLIENT] - The server has returned the correct answer
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 12, Output of the Client Execution

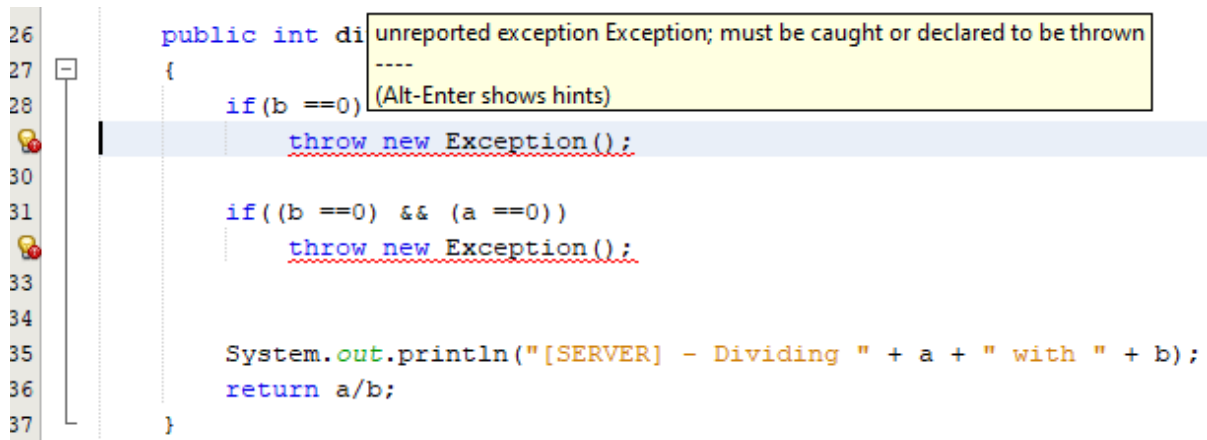
10) Now, add a method to the server that divides two integer numbers (Figure 13)



```
7
8 /**
9  *
10 * @author Gab
11 */
12 public class SimpleCalculatorServer
13 {
14     public boolean isServerConnected()
15     {
16         System.out.println("[SERVER] - Connection is being tested...");
17         return true;
18     }
19
20     public int addIntegerNumbers(int a, int b)
21     {
22         System.out.println("[SERVER] - Adding " + a + " and " + b + "...");
23         return a+b;
24     }
25
26     public int divideIntegerNumbers(int a, int b)
27     {
28         System.out.println("[SERVER] - Dividing " + a + " and " + b + "...");
29         return a/b;
30     }
31 }
32
33
```

Figure 13, Add divide method in the server

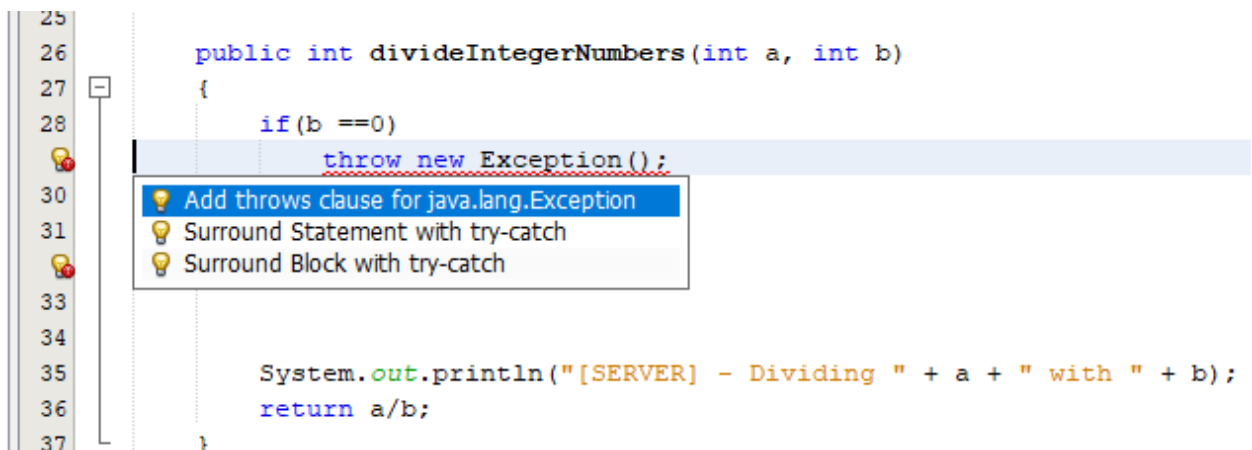
- 11) We must add an exception if b is zero (you cannot divide a number with zero) and if both numbers are zero (zero divided by zero is indetermined). Let's do so (Figure 14). Netbeans tells us that we must either catch or throw the exception (red underlighting on Figure 14).



```
26 public int divideIntegerNumbers(int a, int b)
27 {
28     if(b == 0)
29         throw new Exception();
30
31     if((b == 0) && (a == 0))
32         throw new Exception();
33
34     System.out.println("[SERVER] - Dividing " + a + " with " + b);
35     return a/b;
36 }
37
```

Figure 14, Add exceptions to divide methods.

- 12) By clicking on the lightbulb on the left, Netbeans also tells us what we could do to deal with the Exception we raise in the method, we decide to throw it to back to the calling method (in the client) and let it deal with it (Figure 15).



```
25
26 public int divideIntegerNumbers(int a, int b)
27 {
28     if(b == 0)
29         throw new Exception();
30
31
32
33
34
35     System.out.println("[SERVER] - Dividing " + a + " with " + b);
36     return a/b;
37 }
```

Figure 15, Throwing the exception back to the client

- 13) Now Netbeans also tells us that we must deal with the Exception in the client, as we decided that the server just throws it back to the client. (Figure 16)

```
public int divideIntegerNumbers(int a, int b) throws Exception
{
    if(b ==0)
        throw new Exception();

    if((b ==0) && (a ==0))
        throw new Exception();

    System.out.println("[SERVER] - Dividing " + a + " with " + b);
    return a/b;
}
```

Figure 16, Throw the exception back to the calling method.

- 14) Now, we have to deal with the exception problem on the client as we are still not managing the exception. You can over with the pointer of the error (underlined in red) to see what the problems is (Unreported Exception)

```
x = 4;
y = 2;
System.out.println("[CLIENT] - Server says that " + x + " / " + y + " is " + server.divideIntegerNumbers(x, y));
```

Figure 17, Netbeans tells us that there is an un-managed exception in the client.

- 15) We can ask help to the lightbulb as in step 12, see Figure 18

```
10 // Adding divide method
11 */
12
13 x = 4;
14 y = 2;
15 System.out.println("[CLIENT] - Asking the server what is the ratio of " + x + " and " + y);
16 z = server.divideIntegerNumbers(x, y);
17
18 Add throws clause for java.lang.Exception
19 Surround Statement with try-catch
20 Surround Block with try-catch
21 [CLIENT] - The server has replied that the ratio of " + x + " and " + y + " is:
22 [CLIENT] - The server has returned the correct answer";
23
24 else
25 System.out.println("[CLIENT] - The server has returned the WRONG answer");
26
27
28
29
30
31
32
```

Figure 18, Suggestions to surround with try-catch

- 16) We decide to surround the server invocation that can throw an exception with a try-catch block (Figure 19)

```
if(z == x/y)
    System.out.println("[CLIENT] - The server has returned the correct answer");
else
    System.out.println("[CLIENT] - The server has returned the WRONG answer");

/*
Testing divide method
*/
x = 4;
y = 2;
System.out.println("[CLIENT] - Asking the server what is the ratio of " + x + " and " + y);
try {
    z = server.divideIntegerNumbers(x, y);
} catch (Exception ex) {
    Logger.getLogger(SimpleClient.class.getName()).log(Level.SEVERE, null, ex);
}
System.out.println("[CLIENT] - The server has replied that the ratio of " + x + " and " + y);
if(z == x/y)
    System.out.println("[CLIENT] - The server has returned the correct answer");
else
    System.out.println("[CLIENT] - The server has returned the WRONG answer");
```

Figure 19, Catching exception on the client

- 17) Ensure that all code related to the division inside the try scope (within the try curly brackets) as in Figure 20, that ensures that no code is executed if the exception is thrown. To test this we added the line "This line will not be seen. Only the added printout "Test Completed" will be displayed if there is an exception.

```
x = 4;
y = 0;
try {
    System.out.println("[CLIENT] - Server says that " + x + " / " + y + " is " + server.divideIntegerNumbers(x, y));
    System.out.println("[CLIENT] - This line will not be seen !!!!");
} catch (Exception ex) {
    System.out.println("[CLIENT] - Exception Caught !!!!");
    Logger.getLogger(SimpleClient.class.getName()).log(Level.SEVERE, "Cannot divide by zero !!!!", ex);
}
```

Figure 20, Testing the exception

- 18) Run the client to check if both methods run correctly in normal circumstances (Figure 21)

```
run:
[CLIENT] - Creating instance of client...
[CLIENT] - Testing if server is connected...
[SERVER] - Connection is being tested...
[CLIENT] - Server IS CONNECTED, test can proceed...
[CLIENT] - Asking the server what is the sum of 2 and 3
[SERVER] - Adding 2 and 3
[CLIENT] - The server has replied that the sum of 2 and 3 is 5
[CLIENT] - The server has returned the correct answer
[CLIENT] - Asking the server what is the ratio of 4 and 2
[SERVER] - Adding 4 and 2
[CLIENT] - The server has replied that the ratio of 4 and 2 is 2
[CLIENT] - The server has returned the correct answer
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 21, Running the client

19) Let's see if the exception mechanism works well. Set the variable b to zero as in Figure 20

20) And let's see if it works by running the client (Figure 22). It worked correctly, no divide related code was executed when $y = 0$ and the last print method (added now to test the exception) is printed correctly. You can click on the blue pointers to go to the code that triggered the exception.

```
run:
[CLIENT] - Creating instance of client...
[CLIENT] - Testing if server is connected...
[SERVER] - Connection is being tested...
[CLIENT] - Server IS CONNECTED, test can proceed...
[CLIENT] - Asking the server what is the sum of 2 and 3
[SERVER] - Adding 2 and 3
[CLIENT] - The server has replied that the sum of 2 and 3 is 5
[CLIENT] - The server has returned the correct answer
[CLIENT] - Asking the server what is the ratio of 4 and 2
[SERVER] - Adding 4 and 2
[CLIENT] - The server has replied that the ratio of 4 and 2 is 2
[CLIENT] - The server has returned the correct answer
[CLIENT] - Asking the server what is the ratio of 4 and 0
[CLIENT] - Test Completed
Jan 27, 2020 8:58:33 PM simpletesttwo.SimpleClient execute
SEVERE: null
java.lang.Exception
    at simpletesttwo.SimpleCalculatorServer.divideIntegerNumbers(SimpleCalculatorServer.java:29)
    at simpletesttwo.SimpleClient.execute(SimpleClient.java:74)
    at simpletesttwo.SimpleClient.main(SimpleClient.java:28)

BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 22, Running the Client to test the exception

21) Export the project as zip file on Netbeans (Figure 19)

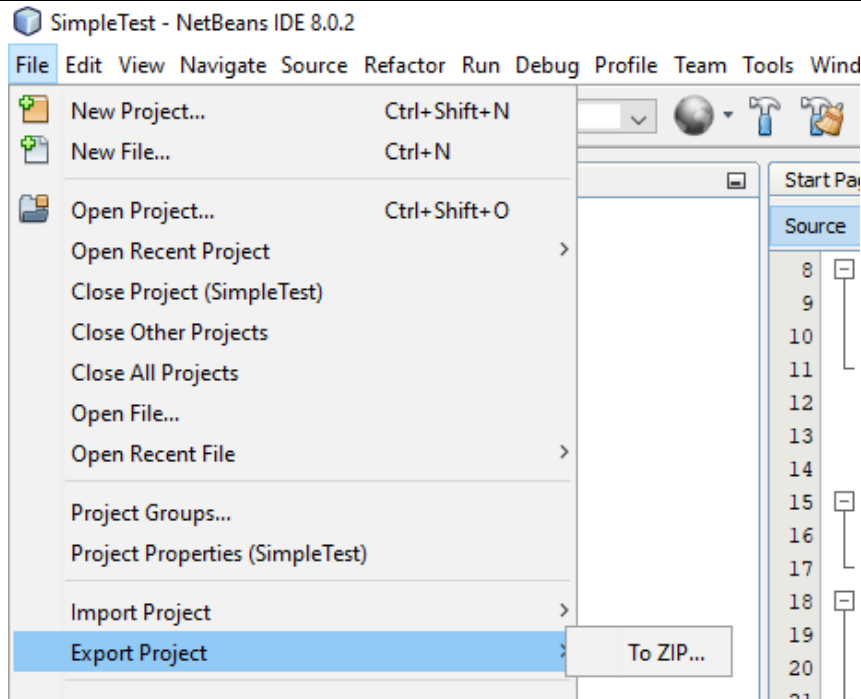


Figure 23, Export the project as zip file (1)

TASKS to BE PERFORMED Independently by the student (from Task 22 to Task 27) (Formative Assessment)

- 22) Modify the server by adding methods that subtract and multiply numbers.
- 23) Test these new methods from the client.
- 24) What happens if you divide two numbers like 4 and 3? The result is 1, is it true that $4/3$ is equal to one? Where is the problem? How can you solve it? **Type your answer using comment block in the client class at the bottom.**
- 25) Modify the server by adding methods that perform the four operations on arrays and not just simple numbers. As an example $[1,2,3] + [3,4,5] = [4,6,8]$.
- 26) Think on the exceptions that these methods need to raise, what if you try to perform an operation of arrays of different sizes? As an example $[1,2,3] + [3,4]$, what would you do? Raise an exception? If you decide so, add the code that throws the exception in the server and catches it in the client.
- 27) Think on the exceptions that these methods need to raise, what if you try to perform a division of arrays and only some numbers are not valid? As an example $[1,2,3,0] / [3,0,4,0]$, what would you do? Raise an exception and return no results (even for the valid numbers)? Add the code that throws the exception in the server and catches it in the client.