University of Westminster
Faculty of Science and Technology

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

Tutorial title	**Getting acquainted with concepts and tools**
Tutorial type	**Guided and indepenent and non-marked**
Week 01	**21/01/2021**

Contents

Learning Goals

This tutorial focuses on three main learning goals:

- to get acquainted with the (some of the) tools you will be using during the module, and,
- to review basic java programming skills,
- to lay the foundation for self-assessement of the status of their basic java skills

It is divided into two separate sections, the student will perform the first task (1-10) 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 10)

1) Start Netbeans (Version 8.2) 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) Create a new java project in Netbeans (Figure 1 to 3)

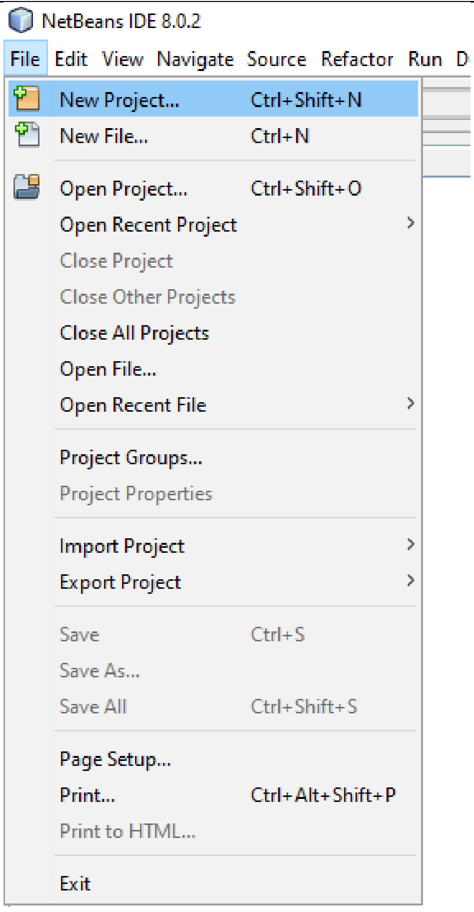
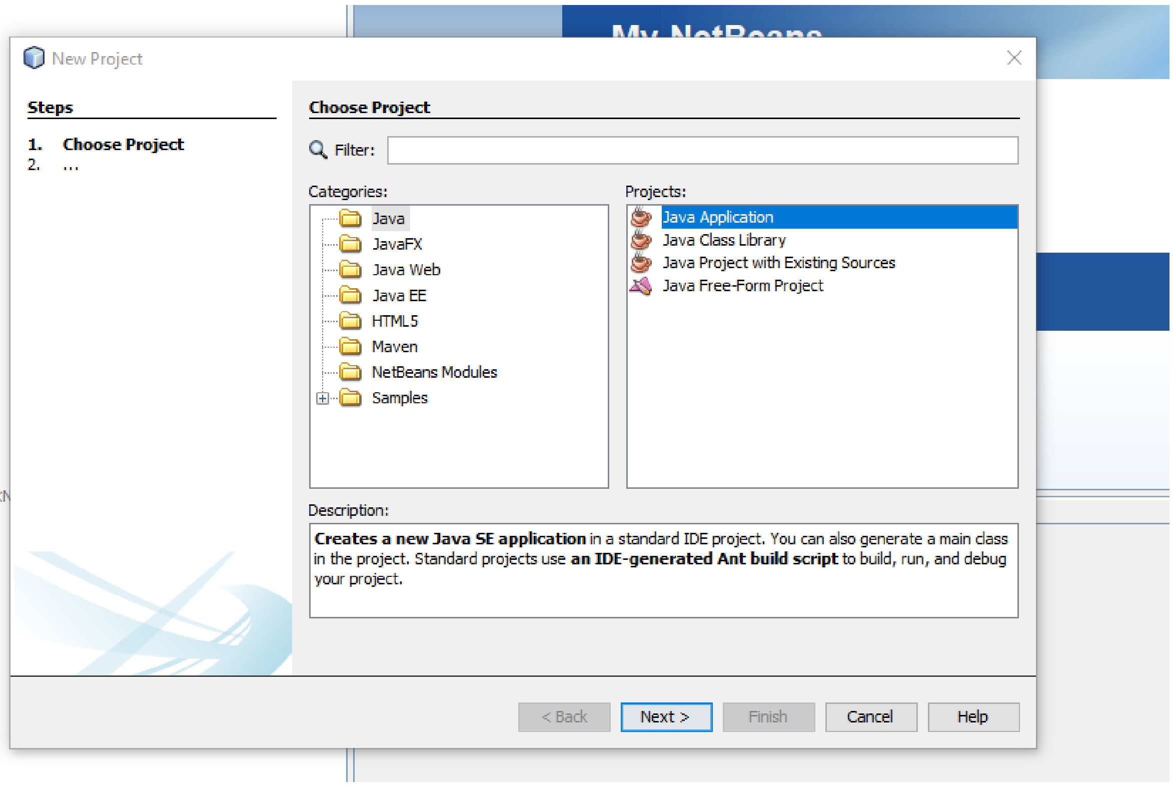
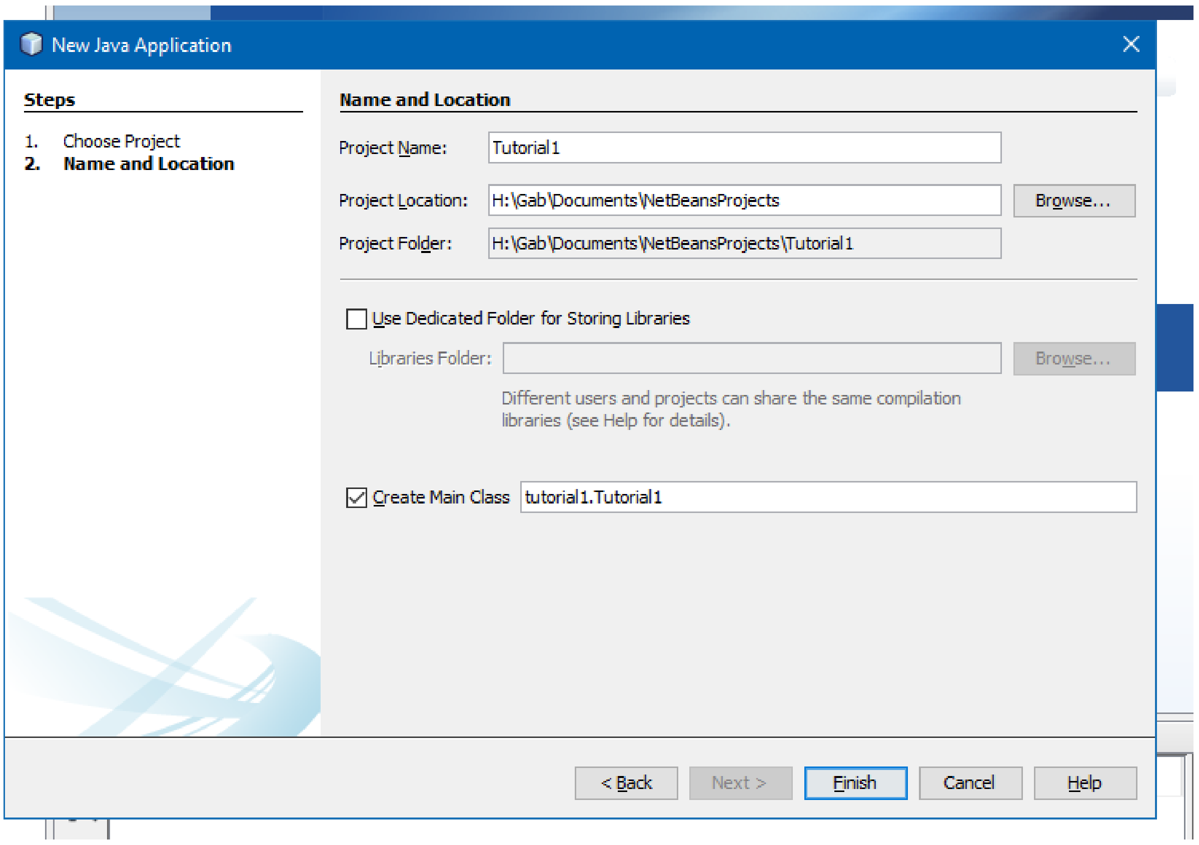


Figure 1, Create a New Project in NetBeans (1)



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 2, Create a New Project in NetBeans (2)



Project Name: Tutorial1
Project Location: H:\Gab\Documents\NetBeansProjects
Project Folder: H:\Gab\Documents\NetBeansProjects\Tutorial1
☐ Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries (see Help for details).
☑ Create Main Class tutorial1.Tutorial1

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

3) The default Java Project created by Netbeans already has an empty Tutorial1 with and empty constructor (Figure 4). We will use this Tutorial1 empty class as the skeleton for our dummy client.

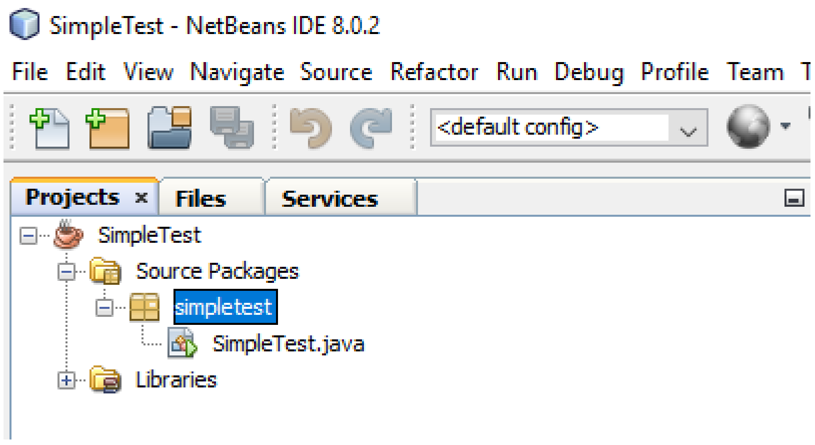


Figure 4, Standard Java Project in NetBeans

4) Create a new java class in the project (Figure 1 to 3) which will represent our Dummy Server.

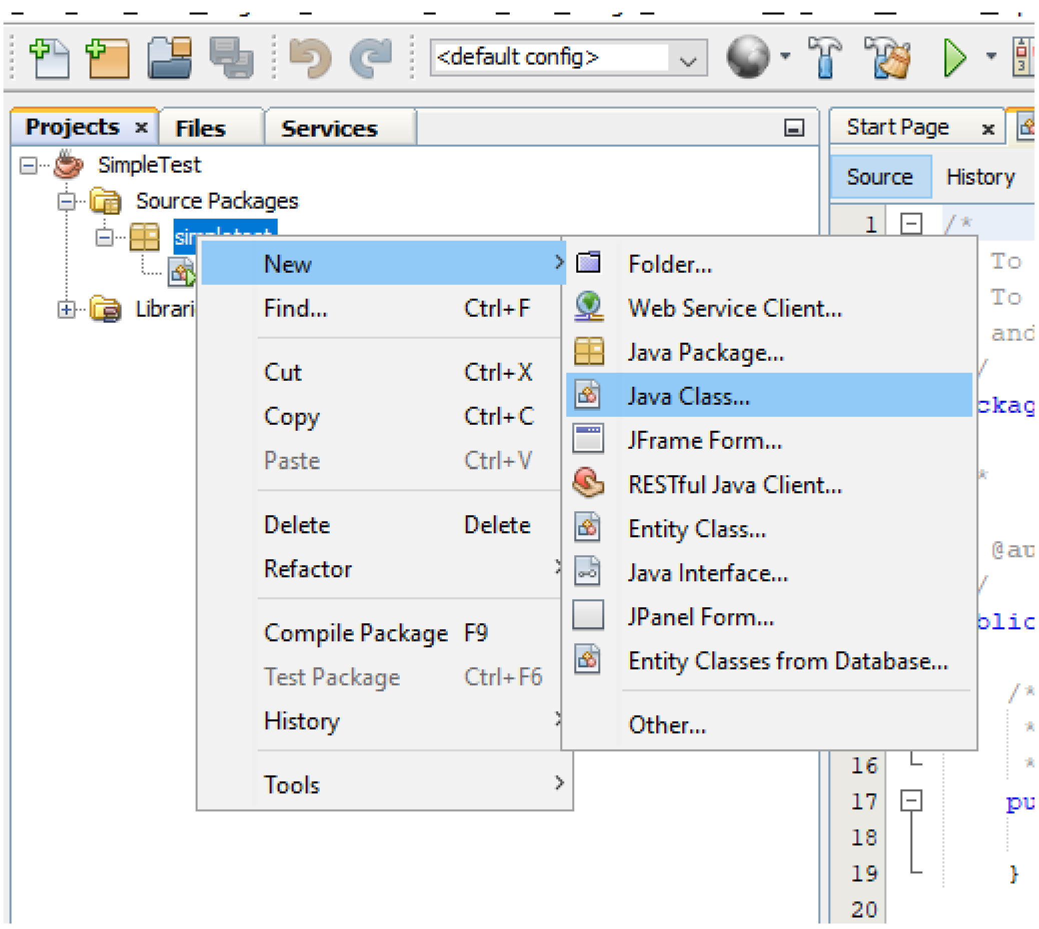
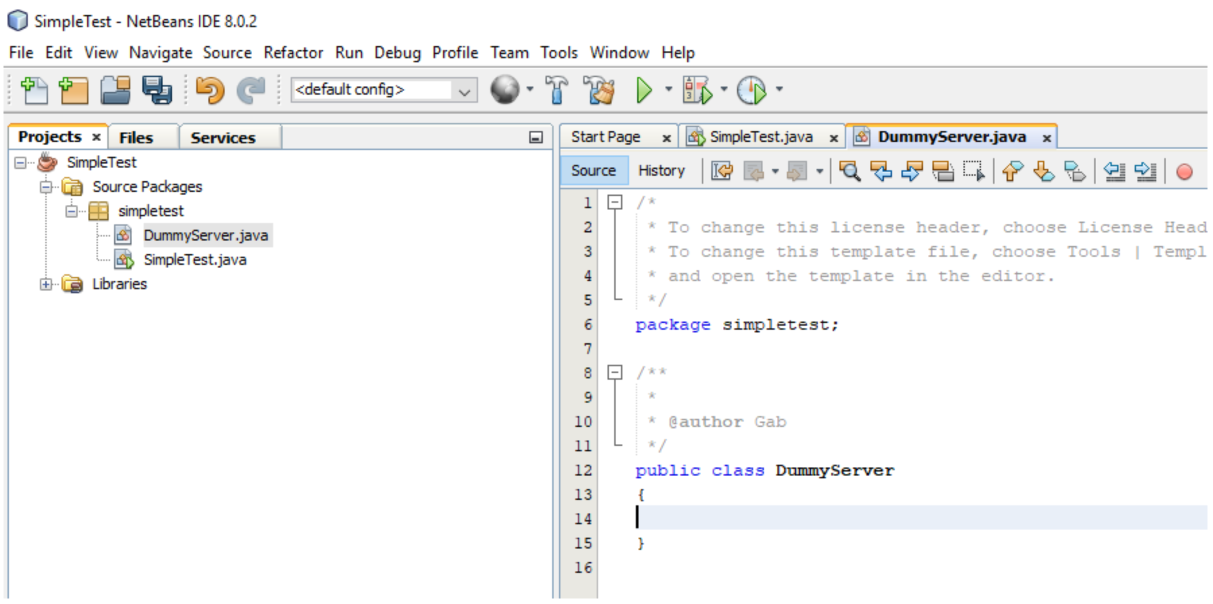


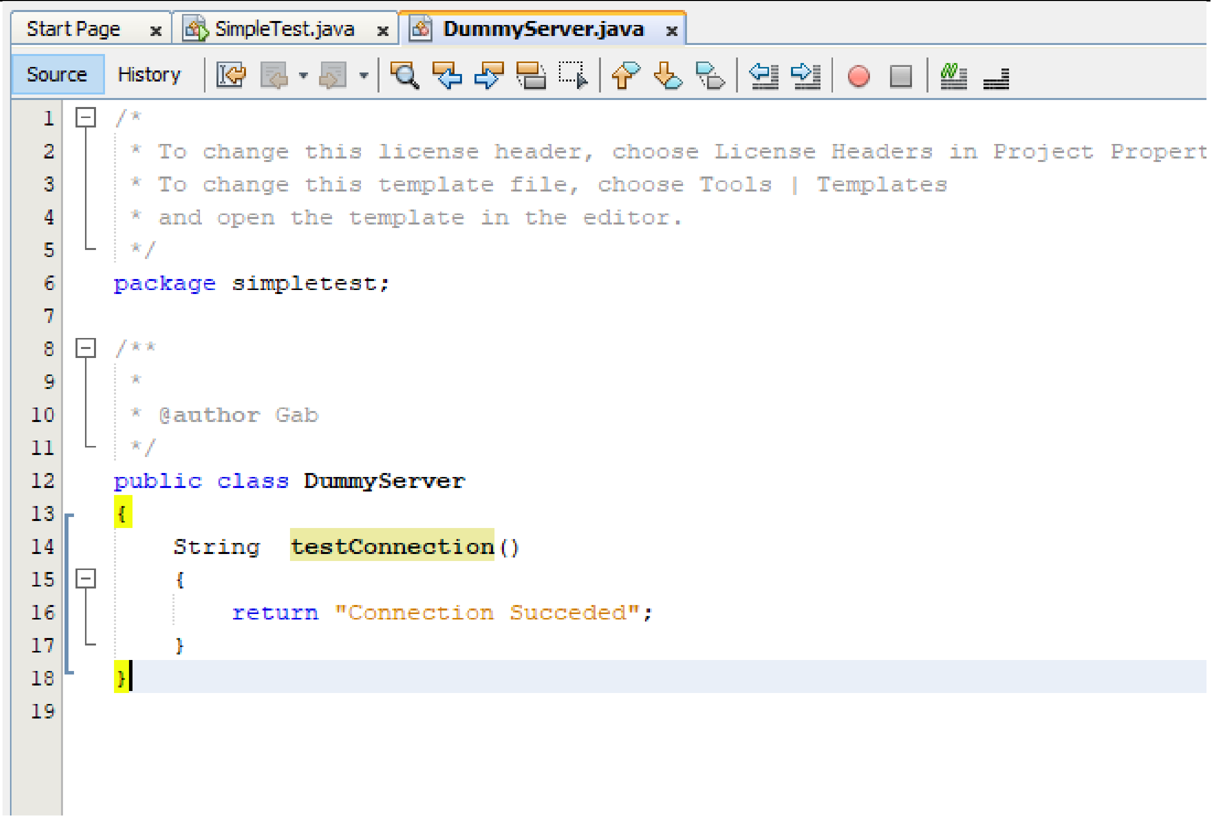
Figure 5, Create a new Java Class in NetBeans (1)



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

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

}
```

Figure 6, Create a new Java Class in NetBeans (2)

5) Add a method (testConnection) to the DummyServer class(Figure 6) that returns a simple string which we can test to see if the server is running.



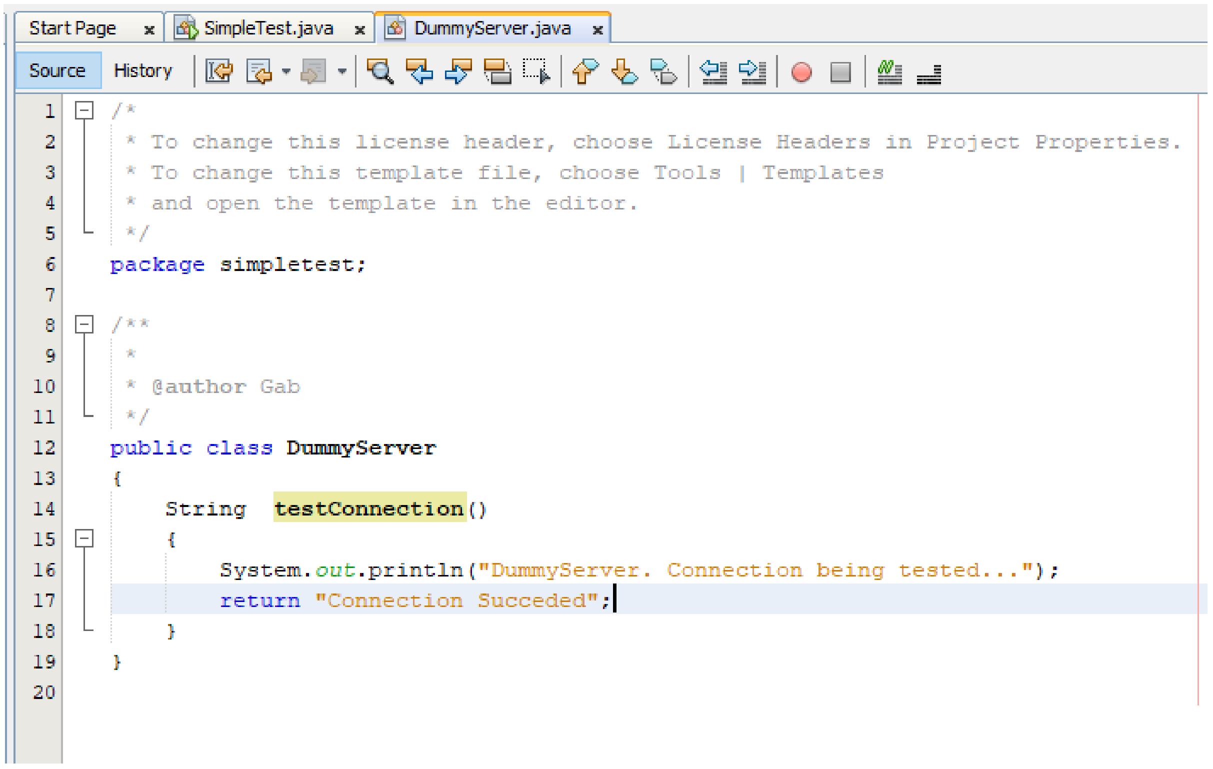
Start Page | SimpleTest.java | DummyServer.java

Source | History

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

/**
 *
 * @author Gab
 */
public class DummyServer
{
    String  testConnection()
    {
        return "Connection Succeded";
    }
}
```

Figure 7, Add simple method to the DummyServer Class

6) Add a log method to the DummyServer class that prints a simple string which we can test to see the debug the server while it is running.



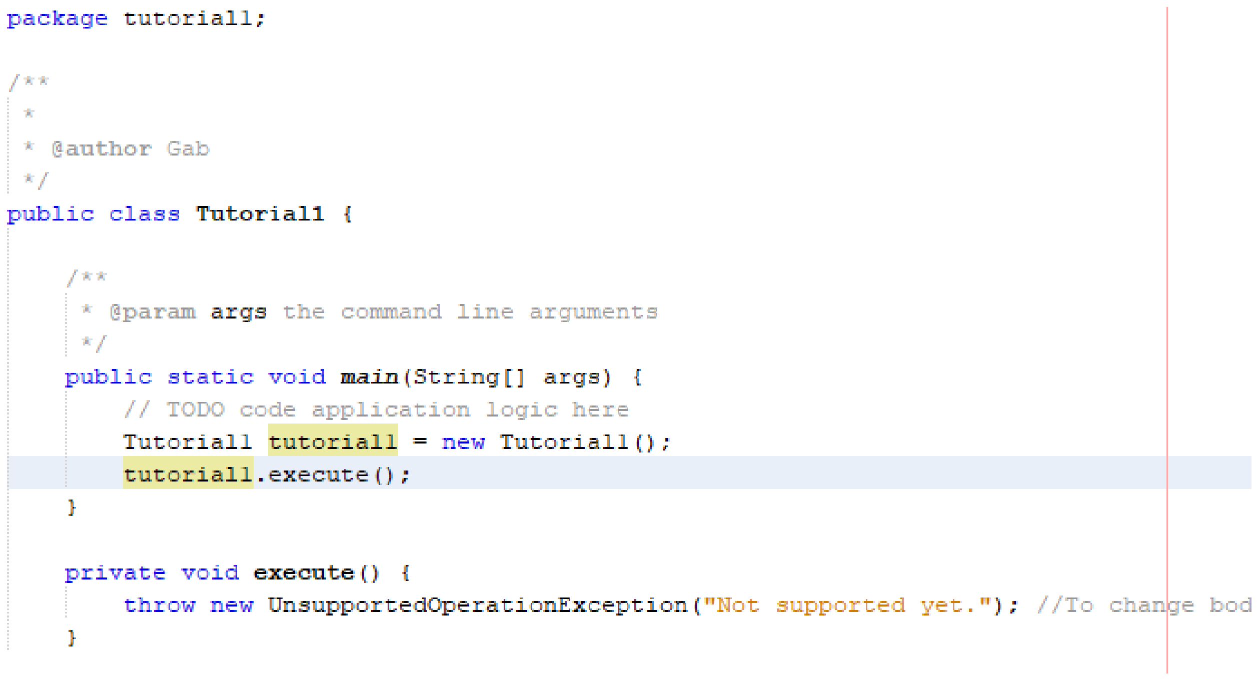
Start Page | SimpleTest.java | DummyServer.java

Source | History

```java
/*
 * 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 simpletest;

/**
 *
 * @author Gab
 */
public class DummyServer
{
    String  testConnection()
    {
        System.out.println("DummyServer. Connection being tested...");
        return "Connection Succeded";
    }
}
```

Figure 8, Add simple logging to the method

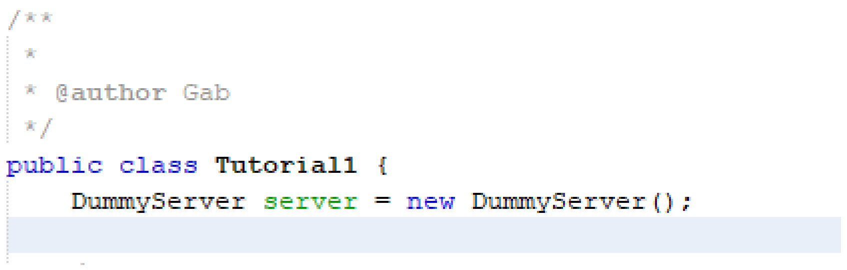
7) Create an instance of the Tutorial1 class and an empty method execute (Figure 9)



```
package tutorial1;

/**
 *
 * @author Gab
 */
public class Tutorial1 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Tutorial1 tutorial1 = new Tutorial1();
        tutorial1.execute();
    }

    private void execute() {
        throw new UnsupportedOperationException("Not supported yet."); //To change bod
    }
```

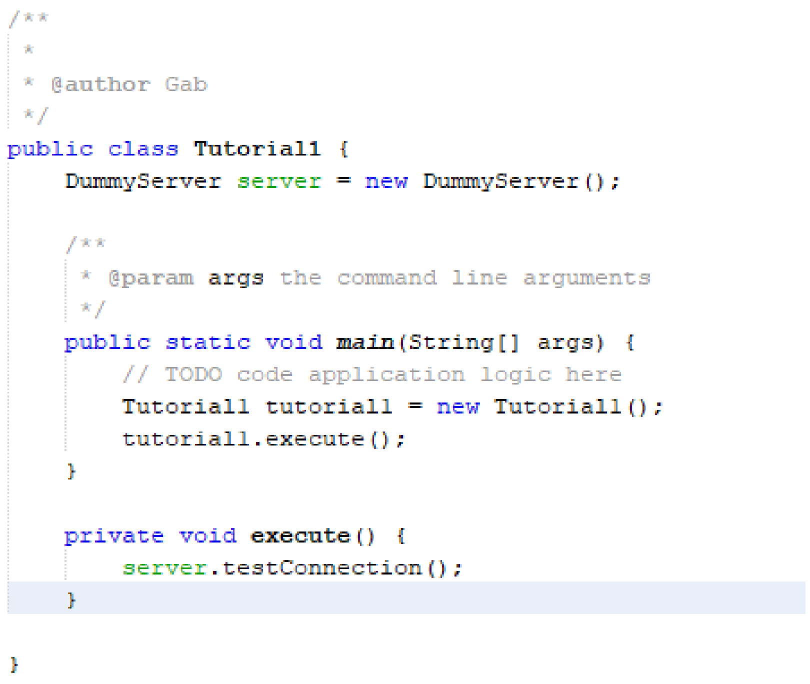
Figure 9, Create instance and execute method.

8) Declare the type of DummyServer and declare one instance into your Tutorial1 class



```
/**
 *
 * @author Gab
 */
public class Tutorial1 {
    DummyServer server = new DummyServer();
```

9) Fill in the execute method by invoking the testConnection method of the DummyServer class.



```
/**
 *
 * @author Gab
 */
public class Tutorial1 {
    DummyServer server = new DummyServer();

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Tutorial1 tutorial1 = new Tutorial1();
        tutorial1.execute();
    }

    private void execute() {
        server.testConnection();
    }

}
```

10) Run the client (Figure 10-11) by right clicking on the class with the main method you want to execute (Tutorial1) and selecting "Run File"

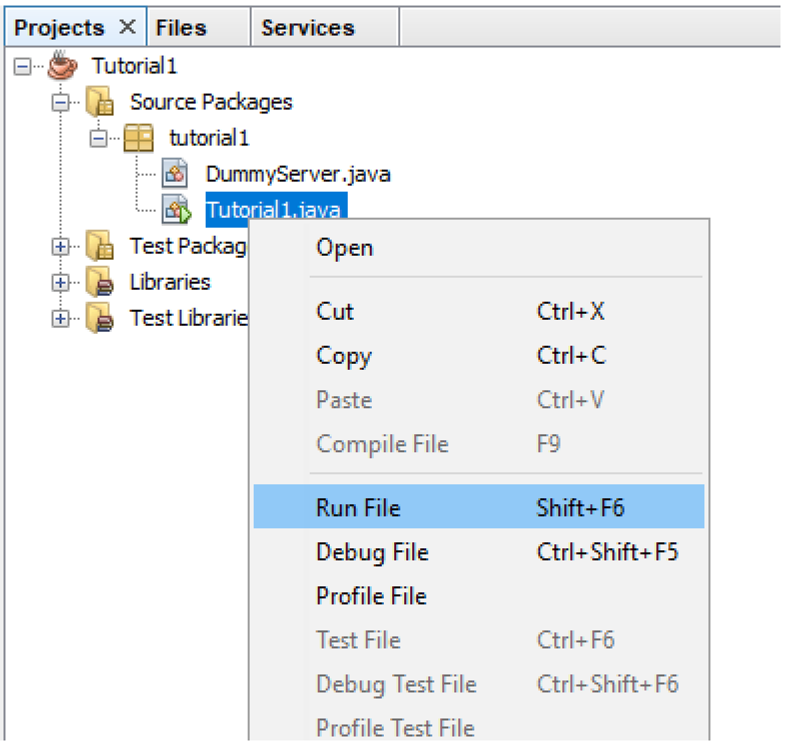
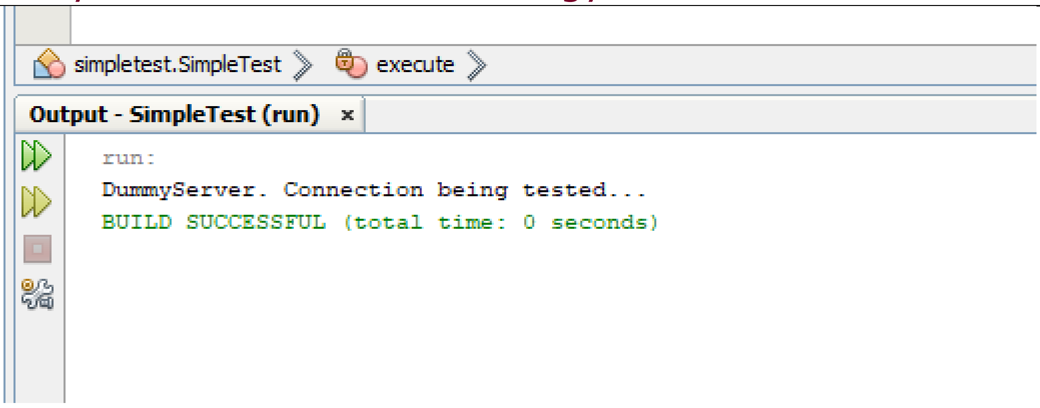


Figure 10, Run tutorial1 (1)



```
run:
DummyServer. Connection being tested...
BUILD SUCCESSFUL (total time: 0 seconds)
```

Figure 11, Run tutorial1 (2)

11) Export the project as zip file on Netbeans (Figure 12-13)

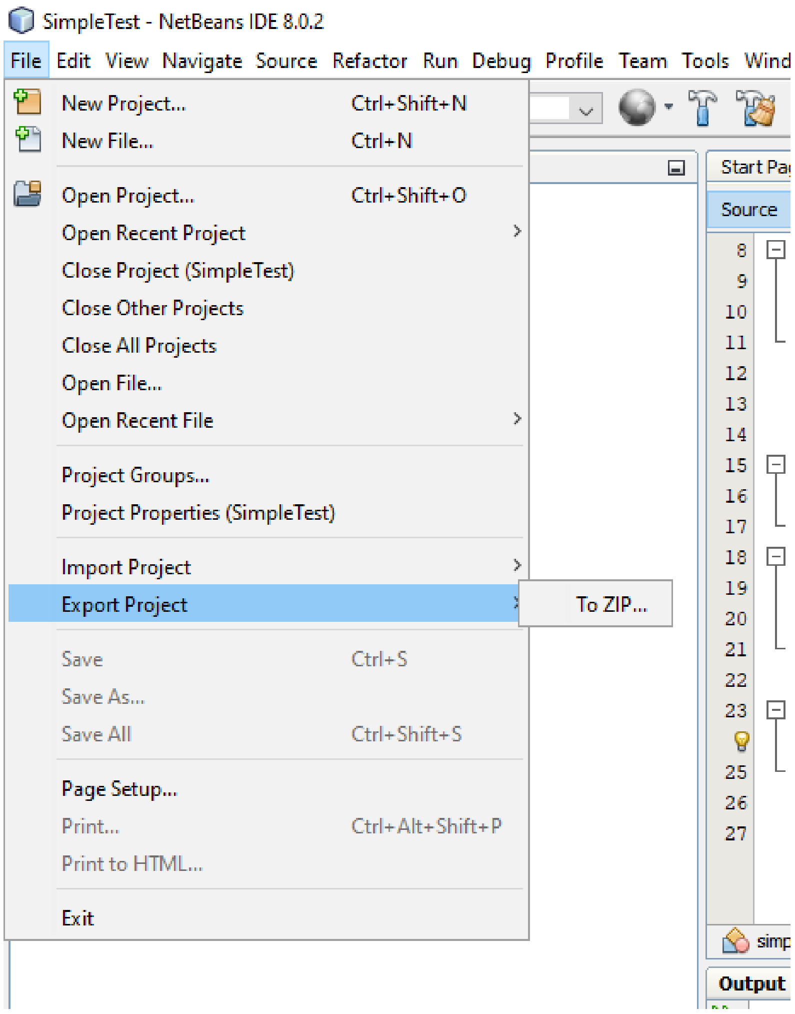


Figure 12, Export the project as zip file (1)

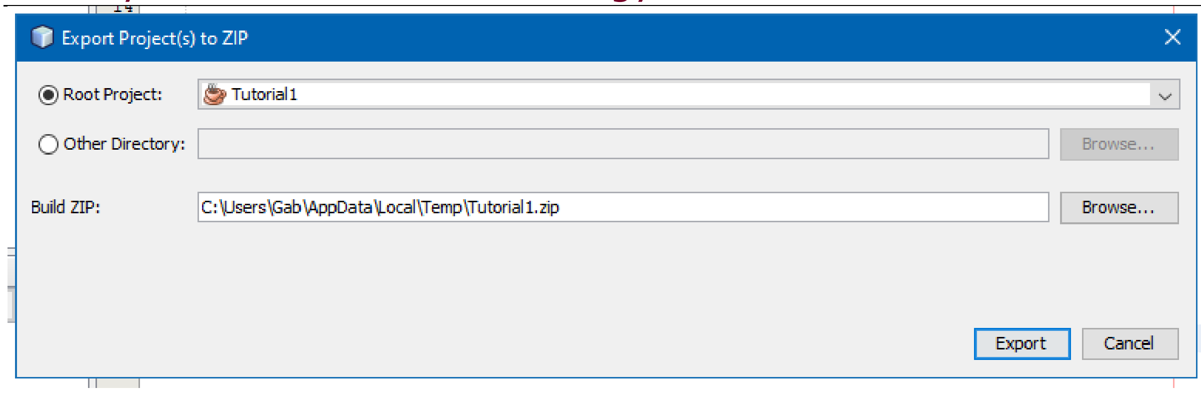


Figure 13,Export the project as zip file (2)

Figure 14, Formative Assessment

TASKS to BE PERFORMED Independently be the student (from Task 11 to Task 15) (Formative Assessment)

12) Modify the Client so that it prints on standard output what is returned by the server.
13) 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)
14) Modify the DummyServer Class so that the server has a name (as a string) so that the returned string from the method testConnection is (e.g. Server….. : Connection from client ….. succeded)
15) Modify the Client and DummyServer Class so that they can add a time stamp to methods execute and testConnection (e.g. [Date and Time] - Server….. : Connection from client ….. succeded)