LABWORK
COURSE: DISTRIBUTED SYSTEMS
CHAPTER 1: INTRODUCTION

1. Web server apache2

1.1. Content

In the class, we have studied the chapter1: Introduction of Distributed Systems. We have learned the definition and some features of DS. We knew that all the network services based on the theory of DS. In this labwork, let's try to construct a WWW system. Concretely, we will install a web server.

1.2. Requirements

1.2.1. Theory

- Handling Unix OS
- Base of Computer Network

1.2.2. Devices

- PC or a VM

1.2.3. Software

1.3. Practical Steps

1.3.1. Install web server apache2

First, we need to install a webserver software. Today, the most commonly used webserver is *apache*. Run the following command:

```
sudo apt install apache2
```

Try to access this webserver in typing the IP of this PC in another PC (they must be in the same LAN network). If it appears the default page of apache (something like "Apache2 Ubuntu default page"), that means you installed successfully the apache webserver.

Question 1: What is the path of the html file that contains the content of the default website apache? /var/www/html

Question 2: What is the default port on which webserver is listening? port 80

1.3.2. Install virtual hosts for apache2

Web server apache2 is able to host several virtual machines with only one IP address. Now we try to make running 2 domains: example.com and test.com
First, create two folders containing content for these two domaims:

```
sudomkdir -p /var/www/example.com/public_html
sudomkdir -p /var/www/test.com/public_html
```

Change permission:

```
sudochmod -R 755 /var/www
```

Question 3: Explain what permission 755 means. user, group permission: read, write, execute
other user permission: read, execute

Write the content for these 2 website (edit the index.html file in 2 folders public_html you have just created)

```
<html>
<head>
<title>Welcome to Example.com!</title>
</head>
<body>
<h1>Success!  The  example.com  virtual  host  is
working!</h1>
</body>
</html>
```

(change to `test.com` with the correspondent file).

The default configuration file of virtual host of apache is:

```
/etc/apache2/sites-available/000-default.conf
```

Now, create the two following new files:
`/etc/apache2/sites-available/example.com.conf`
`/etc/apache2/sites-available/test.com.conf`

Here is the content of file example.com.conf

```
<VirtualHost *:80>
ServerAdmin admin@example.com
ServerName example.com
ServerAlias www.example.com
DocumentRoot /var/www/example.com/public_html
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Now, for file test.com.conf :

```
<VirtualHost *:80>
ServerAdmin admin@test.com
ServerName test.com
ServerAlias www.test.com
DocumentRoot /var/www/test.com/public_html
```

```
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Run these two following commands to activate these files above:

```
sudo a2ensite example.com.conf
sudo a2ensite test.com.conf
```

Restart the apache service:

```
sudo service apache2 restart
```

Open the file `/etc/hosts` and add these lines:

```
127.0.0.1 example.com
127.0.0.1 test.com
```

Now, open the web browser and test the 2 addresses: *example.com* and *test.com*

Because we bind two website aliases to the same localhost ip, thus when we call each it will navigate to the document in the root folder we set for each alias

Question 4: What do you see after typing these 2 addresses? Explain it.

See the content of index.html for each site

Question 5: Try to make other machines in the same LAN access to these 2 addresses.

Add local ip address of the machine running apache2 virtual host into the hosts file of the computer that wants to access

2. Interface in Java

2.1. Content

In the class, we have studied the characteristic Openness of Distributes Systems. In order to guarantee the feature Openness, we have to construct *interface* between components of the system. In this section, we'll construct a simple client-server model, where the client sends a series of numbers to the server. The latter will sort these received numbers in using the method *sort* declared in an interface. There will be different way of implementing the method *sort* of this interface.

2.2. Requirements

2.2.1. Theory

- Java programming

2.2.2. Devices

- PC

2.2.3. Software

- Eclipse IDE

- Installed JDK/JRE

2.3. Practical Steps

2.3.1. Install requirements

- Download and install IDE Eclipse:
https://www.eclipse.org/downloads/packages/
- Download JDK/JRE:
https://www.oracle.com/technetwork/java/javase/downloads/index.html

2.3.2. Construct the program

First, open Eclipse and create a new java project in choosing *File → New → Java project*. You name it whatever you want.
After creating the project, you'll see a folder named *src*. This is the folder that contains the source code.
Create two packages in that folder in right-click in *src* and choose *New → Package*.
Name these two packages as follows:
*com.hust.soict.**your_name**.client_server*
*com.hust.soict.**your_name**.helper*
(Attention: replace ***your_name*** by your name.)

In the *client_server* package, create a two classes and name it: Client and Server.
Now, it's time to write code.

2.3.2.1. Client

Now you open and edit the Client.java file.
First, you have to import some classes of Java libraries:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
```

Then, in the *main* method, you initialize a socket instant in using *Socket* class:

```
Socket socket = new Socket("127.0.0.1", 9898);
```

You can replace the IP address and the port number as you like, but make attention that this is the IP of the server and the port the server program is listening on.

Now, we have to initialize two instances of two classes *BufferedReader* and *PrintWriter* for sending and receiving data.

```
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
```

Initialize an instance of Scanner class

```
System.out.println(in.readLine());
Scanner scanner = new Scanner(System.in);
```

Now, you have to write yourself a *while* loop to get numbers from user and send it to server, until user types empty.
Suggestion: you can use this line to get message string:

```
String message = scanner.nextLine();
```

Question 6: What is the code of the while loop?

```
String clientMsg = new String("");
while (true) {
    clientMsg = scanner.nextLine();
    if (clientMsg.isEmpty()) break;
    System.out.println("Client input: " + clientMsg);
    out.println(clientMsg);
    System.out.println(in.readLine());
}
```

In the end, don't forget to close the *socket* and th

```
socket.close();
scanner.close();
```

2.3.2.2. Server

Now you open and edit the Server.java file. The goal is to construct a multi-threaded server that receives numbers from client and sort them and send back the result to the client.
First, you have to import some classes of Java libraries:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import com.hust.soict.haianh.helper.*;
import java.util.Arrays;
```

In the *main* method, write the code as follows:

```
System.out.println("The Sorter Server is running!");
int clientNumber = 0;
try (ServerSocket listener = new ServerSocket(9898)) {
     while (true) {
        new Sorter(listener.accept(), clientNumber++).start();
     }
}
```

Make attention that the port number must be the same as the one in the client code.
Outside of the main method, create a class Sorter for a thread. In fact, this class extends the class Thread:

```
    private static class Sorter extends Thread {
        private Socket socket;
        private int clientNumber;

        public Sorter(Socket socket, int clientNumber) {
            this.socket = socket;
            this.clientNumber = clientNumber;
```

```
            System.out.println("New client #" + clientNumber + " connected
at " + socket);
        }

        public void run() {
            try {
                BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);

                // Send a welcome message to the client.
                out.println("Hello, you are client #" + clientNumber);

                // Get messages from the client, line by line; Each line has
several numbers separated by a space character
                while (true) {
                    String input = in.readLine();
                    if (input == null || input.isEmpty()) {
                        break;
                    }
                    //Put it in a string array
                    String[] nums = input.split(" ");

                    //Convert this string array to an int array
                    int[] intarr = new int[ nums.length ];

                    int i = 0;

                    for ( String textValue : nums ) {
                           intarr[i] = Integer.parseInt( textValue );
                      i++;
                    }

                    //Sort the numbers in this int array
                    new SelectionSort().sort(intarr);
                    //Convert the int array to String
                    String strArray[] = Arrays.stream(intarr)
                                              .mapToObj(String::valueOf)
                                              .toArray(String[]::new);
                    //Send the result to Client
                    out.println(Arrays.toString(strArray));
                }
            } catch (IOException e) {
                System.out.println("Error handling client #" +
clientNumber);
            } finally {
                try { socket.close(); } catch (IOException e) {}
                System.out.println("Connection with client # " +
clientNumber + " closed");
            }
        }
    }
```

The role of method run is to wait for the number input from client side and then execute the sort function then print out the sorted array to client side.
The run method is run when the start method is called after class Sorter is initialized after a client is connected.

Question 7: What is the role of the method *run*? When is it called?

In the code above, you can see the call of the method *sort* of the class *SelectionSort*. Now, we can construct an interface and declare the method *sort* inside. The class *SelectionSort* is one of the classes that implement this interface.

2.3.2.3. Interface and different implementation

Now, right-click on the package *com.hust.soict.your_name.helper*, choose *New → Interface*
Create a new interface and name it *NumberSorter*.
In this file, write the code below:

```
public interface NumberSorter {
	void sort(int arr[]);
}
```

In the same package, create a new class and name it *SelectionSort*.
In fact, the class *SelectionSort* will implement the interface *NumberSorter* and define concretely the method *sort* based on the algorithm *selection sort*.
Open the file SelectionSort.java and write the code below:

```
public class SelectionSort implements NumberSorter{
	public void sort(int arr[]) {
		int n = arr.length;

	  // One by one move boundary of unsorted subarray
	  for (int i = 0; i < n-1; i++)
	  {
		// Find the minimum element in unsorted array
		int min_idx = i;
		for (int j = i+1; j < n; j++)
		    if (arr[j] < arr[min_idx])
			min_idx = j;

		// Swap the found minimum element with the first
		// element
		int temp = arr[min_idx];
		arr[min_idx] = arr[i];
		arr[i] = temp;
	  }
	}
}
```

2.3.2.4. Run the program

Now, try to run the whole program. Don't forget to run the server before the client.

2.3.2.5. Implement other sort algorithms

It's time to implement yourself other sort algorithms. You do the same thing as above (create new class in the package helper and implement the *NumberSorter* interface). Try to implement the 3 algorithms below:

- Bubble sort
- Insertion sort
- Shell sort