Lab 13: Developing TCP Application

BITP 3123 DISTRIBUTED APPLICATION DEVELOPMENT

EMALIANA BINTI KASMURI
FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

Table of Contents

Lab 13: Developing TCP Application

Learning Outcomes

When the student finished all the exercises, the student should be able to,

1. Create a new Eclipse workspace.
2. Execute simple TCP applications.
3. Develop simple TCP applications.

Reference Materials

1. Lecture Slides from Lecture 13.
2. Supplementary files named demotcpclientdate.zip and demotcpserverdate.zip. The files are available in ulearn. Download and extract these files to your computer.

Tools

1. Eclipse for Java EE

Programming Approach

The solution of all exercises must be implemented using the object-oriented approach that is complied with Java Coding Standard and Model-View-Controller design pattern.

Exercise 1: Create New Eclipse's Workspace

It is advisable to create a new Eclipse workspace for this BITP 3123. The following are steps to create a new workspace in Eclipse.

Estimated Time to Completion: 10 minutes

1. Create a new directory on your computer. Named the directory as **bitp3123-ws**.
2. Launch Eclipse from your computer.
3. Select **File > Switch Workspace > Others**. A window entitled **Eclipse IDE Launcher** as shown in Figure 1 will be displayed on the screen.

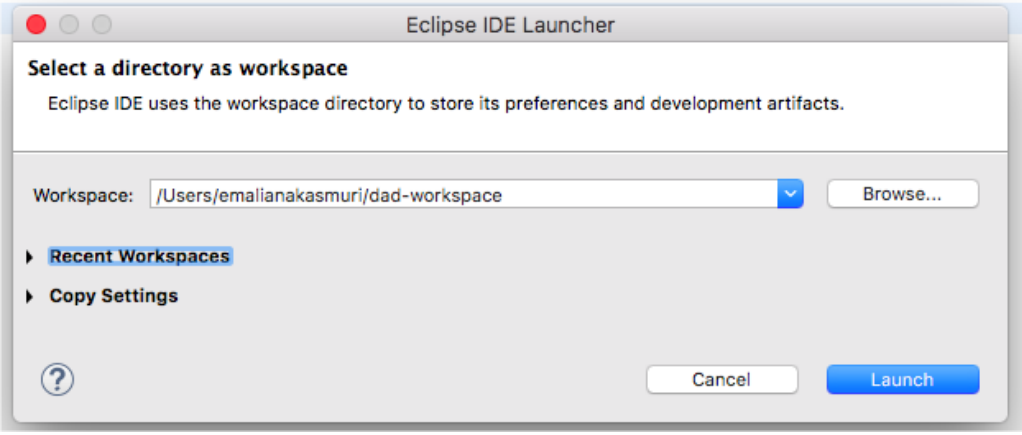

Figure 1: A window to change workspace

4. Click Browse to locate the directory created in Step 1.
5. Then, click Launch. The current workspace will be changed to the new workspace.

Exercise 2: Execute a Simple TCP application

Execute Server-Side Application

Estimated Time to Completion: 10 minutes

The following are the steps to execute the server-side application.

1. Create a new Java project. Name the project as **demotcpserverdate**.
2. Double-click the project to expand its content.
3. Right-click on the **src** folder from the Project Explorer in Eclipse.
4. Select **Import**. A new window entitled **Select,** as shown in Figure 2 will be displayed.

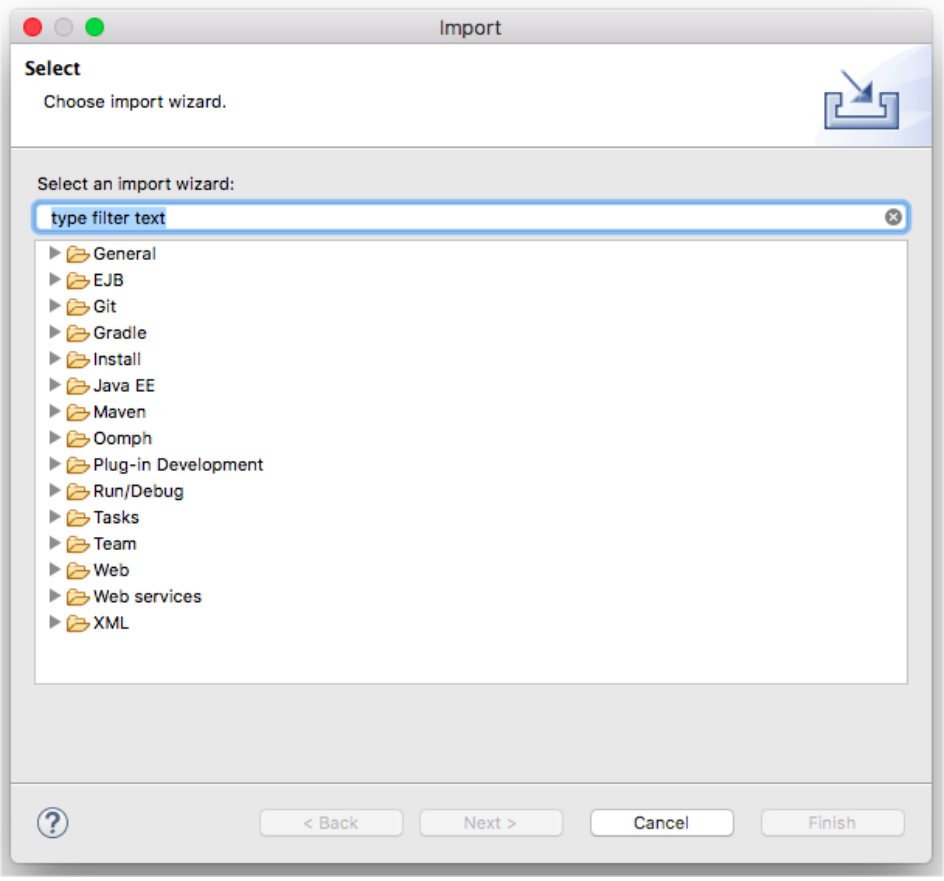

Figure 2: Window to import files into the project

5. Click **General** from the window shown in Figure 2 to expand its content.
6. Then select **File System**.

7. After that, click **Next**. A window entitled **File system,** as shown in Figure 3 will be displayed on the screen.

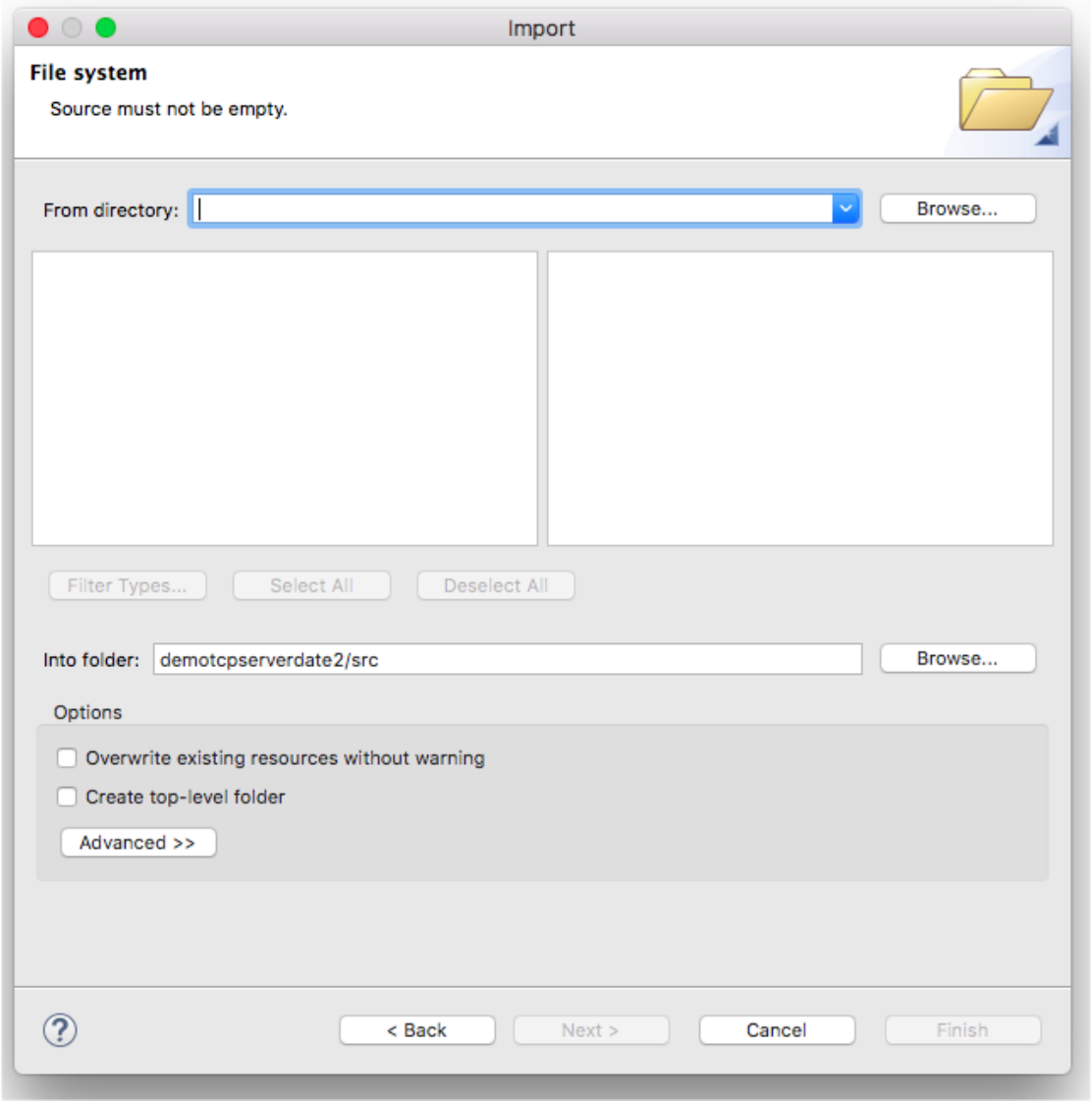

Figure 3: Window to specify file to be import into the project

8. Click **Browse** to locate the directory extracted from **demotcpserverdate.zip**. Select the directory. Figure 4 shows an example of the selected directory.

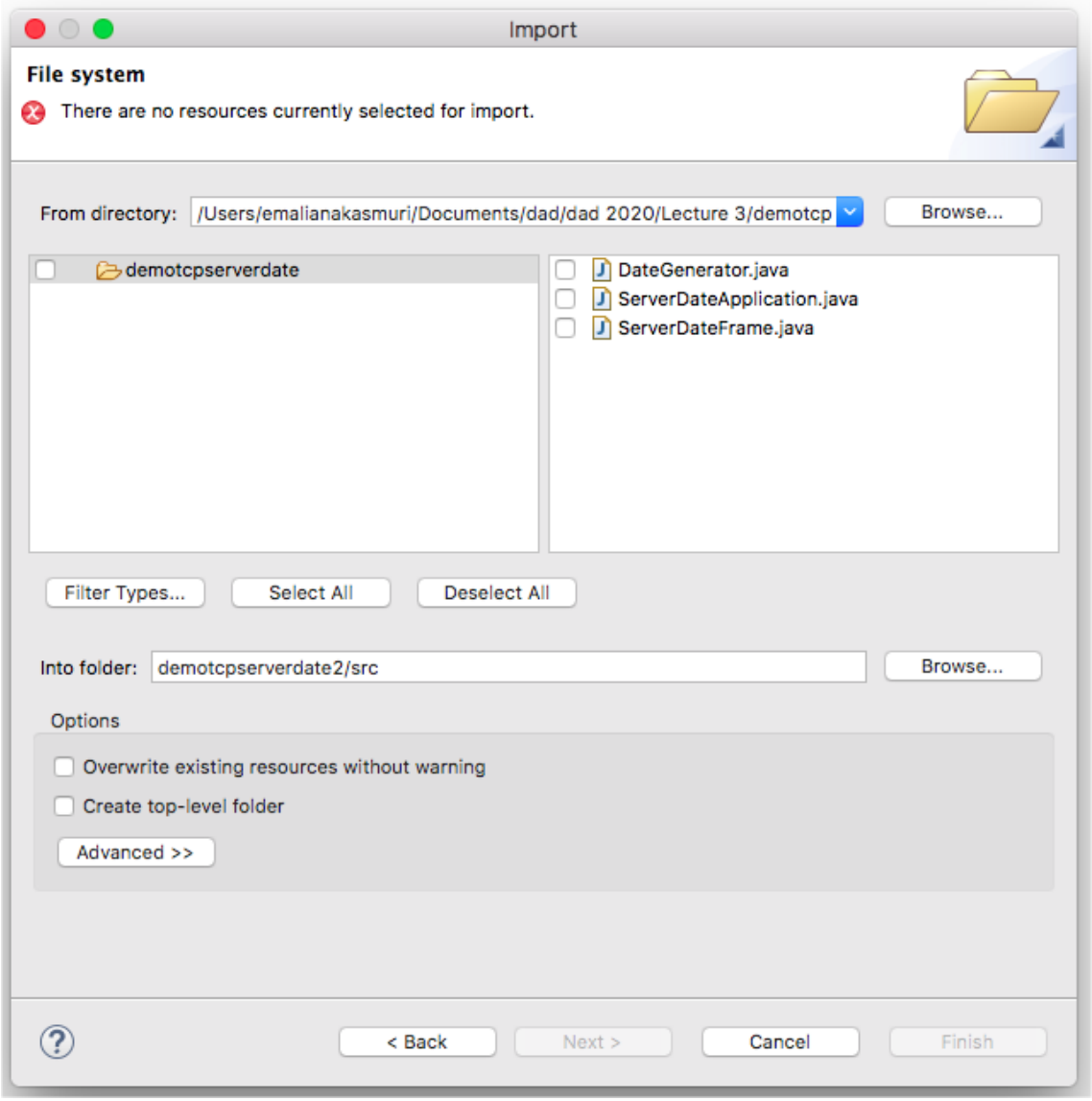

Figure 4: Example of the selected directory

9. Tick all the checkboxes to import all files.
10. Then click **Finish**. The selected files will be imported into the project in the **src** folder.

11. Open the class with the `main()` method.

12. Execute that class. The output should be similar to Figure 5.

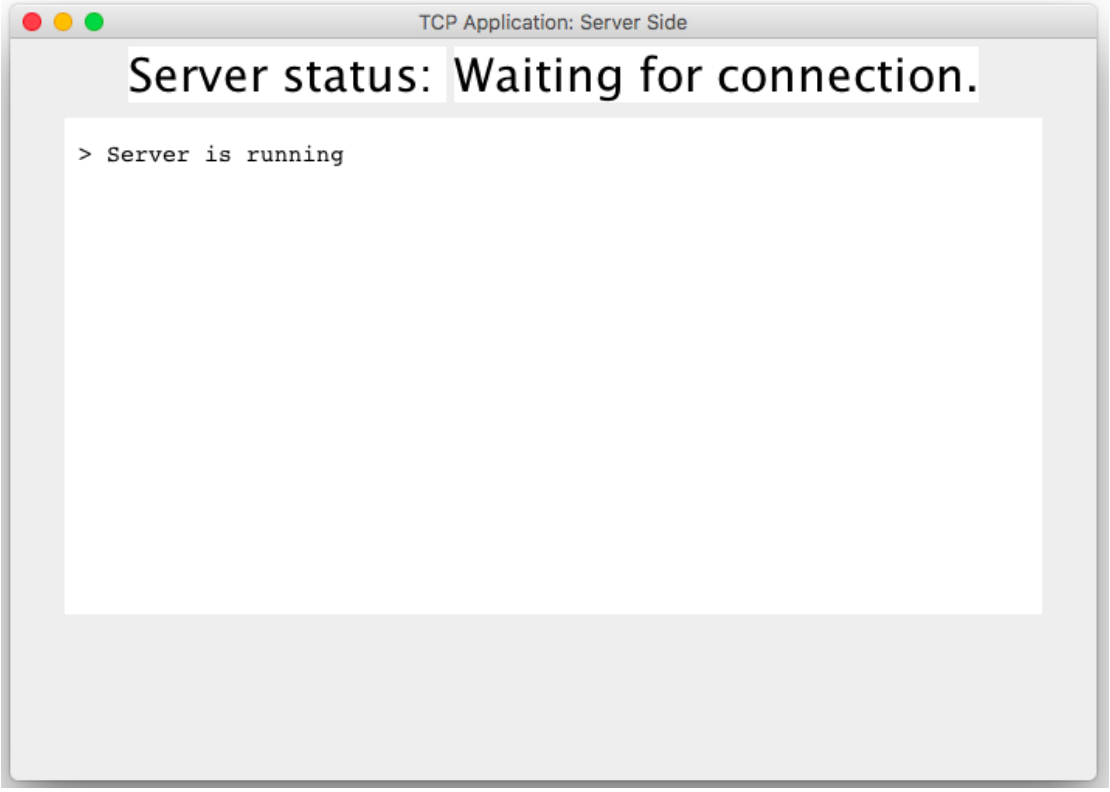

Figure 5: Window for server-side application

Execute Client-Side Application

Estimated Time to Completion: 10 minutes

The following are the steps to execute the client-side application.

1. Create a new Java project. Name the project as **demotcpclientdate**.
2. Double-click the project to expand its content.
3. Right-click on the **src** folder from the Project Explorer.
4. Repeat step 4 until step 9 from section Execute Server-Side Application to import files for the client-side application.
5. Open a **terminal/MS Dos console**.
6. Change the directory to **demotcpclientdate/bin**.
7. Execute **ClientDateApplication**. A new window entitled **TCP Application: Client-Side** will be displayed on the screen.
8. Observe the output at the server-side application.
9. Repeat steps 5 to 8 for another client.

Exercise 3 Text Processing Client-Server Application using TCP

Estimated Time to Completion: 20 minutes

1. Design a TCP-Based client-server application to process a length of a text using a class diagram.
2. The server will count the number of words in the text and returns it to the client.
3. Be creative with your solutions.
4. Implement the design created in step 1 using Java.
5. Push your solutions on Github with complete description.
6. Record the solution for this exercise in Declaration of Solution.

Exercise 4 Creating Simple TCP Application

Estimated Time to Completion: 20 minutes

1. Write the code shown in Figure 6 for a server-side application.

```
public class ServerTranslationApplication {

    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = null;

        try {
            // Bind Serversocket to a port
            int portNo = 4228;
            serverSocket = new ServerSocket(portNo);

            String text1 = "Good afternoon";
            System.out.println("Waiting for request");

            while (true) {

                // Accept client request for connection
                Socket clientSocket = serverSocket.accept();

                // Create stream to write data on the network
                DataOutputStream outputStream = new DataOutputStream(clientSocket.getOutputStream());

                // Send current date back to the client
                outputStream.writeUTF(text1);

                // Close the socket
                clientSocket.close();

            }

            // Closing is not necessary because the code is unreachable

        } catch (IOException ioe) {

            if (serverSocket != null)
                serverSocket.close();

            ioe.printStackTrace();
        }
    }
}
```

Figure 6: ServerTranslationApplication.java

2. Import the necessary classes.
3. Save and run the application.
4. Observe the output from the run.
5. Write the following code for a client-side application.

```java
public class ClientTranslationApplication {

    public static void main(String[] args) {


        try {
            // Connect to the server at localhost, port 4228
            Socket socket = new Socket(InetAddress.getLocalHost(), 4228);

            // Create input stream
            BufferedReader bufferedReader = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()));

            // Read from the network and display the current date
            String text = bufferedReader.readLine();
            System.out.println(text);

            // Close everything
            bufferedReader.close();
            socket.close();

        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }



    }

}
```

Figure 7: ClientTranslation.java

6. Import the necessary classes.
7. Save and run the application.
8. Observe the output from the run.

Exercise 5 Analysis of Observation

The following questions are related to the observation from Exercise 4. Record your answer in ulearn.

1. Based on your observation from the Java program created in Figure 6 and Figure 7, which Java class from `java.net` marks the distinction between a server-side and a client-side application?
2. What type of stream is used in Figure 6 to respond to the client's request?
3. What technique is used to write the data on the network in Figure 6?
4. State the Java instruction that connects the program in Figure 6 to the network.
5. What is the object's name that allows the data to be written onto the network in Figure 6?
6. State the Java instruction that writes the data on the network in Figure 6.
7. State the Java instruction that requests the server's connection in Figure 7.
8. What is the type of stream used in Figure 7 to read the response from the server?
9. State the name of the object that allows the program in Figure 7 to read the data from the network.
10. State the Java instruction that reads the data from the network in Figure 7.

Exercise 6 TCP Console-Based Text Translation Application

Estimated Time to Completion: 30 minutes

1. Design a TCP client-server application using class diagrams to process the multilingual text as shown in Table 1.
2. The design of the application shall include the following requirements:-
    a. The client-side application should send an English text and the target translated language.
    b. The server-side application should translate the text received from the client into the targeted language and display it.
    c. The applications shall be executed in thin-client architecture.
    d. Both sides of application are console application.
3. Be creative with your solution.
4. Implement the design using Java. The client-side application shall send several requests of translation to the server-side application.
5. Push your solutions on Github with complete description.
6. Record the solution for this exercise in Declaration of Solution.

Table 1: Multilingual text

English	Bahasa Malaysia	Arabic	Korean
Good morning	Selamat pagi	صباح الخير	좋은 아침
Good night	Selamat malam	طاب مساؤك	안녕히 주무세요
How are you?	Apa khabar?	كيف حالك؟	어떻게 지내세요?
Thank you	Terima kasih	شكرا لك	감사합니다
Goodbye	Selamat tinggal	مع السلامة	안녕
What's up?	Ada apa?	ما أخبارك؟	뭐야?

Exercise 7 TCP Console-Based Text Translation Application

Estimated Time to Completion: 40 minutes

1. Review the design in Exercise 6.
2. Make the necessary changes to accommodate the following requirements.
    a. The client-side application should send an English text and the target translated language.
    b. The server-side application should translate the text received from the client into the targeted language and display it.
    c. The server-side application should keep track the number of request made to the server.
    d. The server side application should keep track the details of each request and the respond return for each request.
    e. Requirement c and d must be visible.
    f. The applications shall be executed in thin-client architecture.
    g. Both sides of application are GUI-based application.
3. Be creative with your solution.
4. Implement the design using Java. The client-side application shall send several requests of translation to the server-side application.
5. Push your solutions on Github with complete description.
6. Record the solution for this exercise in Declaration of Solution.

End of Document