Lab 14: TCP and UDP Programming with Java

Fakulti Teknologi Maklumat dan Komunikasi

Universiti Teknikal Malaysia Melaka

Table of Contents

Learning Outcomes **1**

Tools **1**

Programming Approach **1**

Part 1: Processing Primitive Data in Java TCP Application **2**

Exercise 1: Executing Reference Code *2*

Exercise 2: Adding New Method to a Controller Class *2*

Exercise 3: Executing TCP Server-Side Application *3*

Exercise 4: Executing TCP Client-Side Application *3*

Exercise 5: Creating New Solution to Process Primitive Data in TCP Application *3*

Part 2: Programming in UDP Application **1**

Exercise 1: Executing Reference of UDP *1*

Exercise 2: Creating New Solutions for UDP *1*

Part 3: Processing Complex Data in TCP Application **2**

Exercise 1: Executing Reference Application *2*

Exercise 2: Creating New Entity Class *2*

Exercise 3: Creating New Controller Class *2*

Exercise 4: Finding a Customer at the Server-Side by Customer Name *4*

Exercise 5: Finding a Customer at the Client-Side by Customer Name *4*

Exercise 6: Finding a Customer at the Server Side by Customer Id *4*

Exercise 7: Finding a Customer at the Client-Side by Customer Id *5*

Exercise 8: Executing Reference Application *5*

Exercise 9: Getting a List of Customers from the Server-Side *5*

Exercise 10: Receiving a List of Customers at the Client-Side *5*

Additional Instruction *5*

Learning Outcomes

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

1. Execute simple TCP applications.
2. Develop simple TCP applications that process primitive and complex data.
3. Execute simple UDP applications.
4. Develop simple UDP applications.

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 the Model-View-Controller design pattern.

Part 1: Processing Primitive Data in Java TCP Application

Exercise 1: Executing Reference Code

1. Download **TCPSummationServerApp.zip** and **TCPSummationClientApp.zip** from the **ulearn**.
2. Import both projects in **Eclipse**.
3. Run both projects separately.
4. Observe the output.

Exercise 2: Adding New Method to a Controller Class

1. Add the method shown in Figure 1 to `NumberCalculator.java`.

```java
/**

 * This method multiplies two numbers

 * @param number1

 * @param number2

 * @return

 */

public int getMultiplication (int number1, int number2) {



        return number1 * number 2;

}
```

Figure 1: Method getMultiplication

2. Fix any errors from in it.
3. Save the class.

Exercise 3: Executing TCP Server-Side Application

1. Open `TCPSummationServerApp.java`.
2. Replace the method that gets the summation of two numbers with the method in Exercise 2.
3. Fix any error in it.
4. Execute `TCPSummationServerApp.java`.

Exercise 4: Executing TCP Client-Side Application

1. Open `TCPSummationClientApp.java`.
2. Execute `TCPSummationClientApp.java`.
3. Observe the output from both sides.
4. Record your observation.

Exercise 5: Creating New Solution to Process Primitive Data in TCP Application

All implementation for these exercises must use `DataInputStream` and `DataOutputStream`.

1. Code implementation for `TCPSummationServerApp.java` to compute the summation and the multiplication of 3 numbers received from the client.
2. Code implementation for `TCPSummationClientApp.java` to send 3 numbers to the server and received two results from the server.
3. Push the solutions on Github with a complete description.
4. Record the solution for this exercise in the Declaration of Solution.

Part 2: Programming in UDP Application

Exercise 1: Executing Reference of UDP

1. Download **UDPGreetingServerApp.zip** and **UDPGreetingClientApp.zip** from the **ulearn**.
2. Import both projects in **Eclipse**.
3. Run both projects separately.
4. Observe the output.

Exercise 2: Creating New Solutions for UDP

1. Write an implementation for a UDP server-side program to count the number of vowels, consonants, and punctuations in sentences. The sentence is received from a client. The server must return the results of these data to the client.
2. Write an implementation for a UDP client-side program to send a sentence to the UDP server. The client will receive a result of the analysis from the server.
3. Push the solutions on Github with a complete description.
4. Record the solution for this exercise in the Declaration of Solution.

Part 3: Processing Complex Data in TCP Application

Exercise 1: Executing Reference Application

1. Download **TCPSalesApp.zip** from the **ulearn**.
2. Execute `TCPProductServer.java` from package `server.app` and `TCPProductServer.java` from package `client.app` separately.
3. Observe the output from both sides.

Exercise 2: Creating New Entity Class

1. Create an entity class to represent a customer in the package model.
2. The class shall contain the customer id and name as the attributes. These attributes are private.
3. Add getters and setters for all attributes.

Exercise 3: Creating New Controller Class

1. Create a controller class to manage the customer data in the package `server.controller`. Name this class appropriately.
2. Define an object that represents a list of customers. This object shall be private.
3. This class shall have four methods. The description of the methods is shown in Table 1.

Table 1: Description of methods for the class that manage the customer data

Method	**Description**
Method 1	This method creates a list of samples of customer data. The list will contain 10 customers. This is a private method.
Method 2	This method searches a customer based on the customer's name from a list of customers. This method will receive a parameter that represents a customer's name. The customer's name will be either full or partial name. The method will return a Customer's object if the name exists. Otherwise, it will return null. This is a public method.
Method 2	This method searches a customer based on the customer's id from a list of customers. This method will receive a parameter that represents a customer's id. This method will return a Customer's object if the name exist. Otherwise, it will return null. This is a public method.
Method 4	This method will return a list of customers. This is a public method.
Constructor 1	This constructor will call Method 1.

4. Implement all the methods. Name the methods appropriately.
5. Additional methods might be needed. Add those methods accordingly.

Exercise 4: Finding a Customer at the Server-Side by Customer Name

1. Create a TCP server-side application to process requests from a TCP client.
2. The server will receive a string of data that represents a customer from a client.
3. The server will find the customer based on the name and return a customer object to the client.
4. The server will log all its operations and interactions with the clients.

Exercise 5: Finding a Customer at the Client-Side by Customer Name

1. Create a TCP client-side application to send several customer names to the server.
2. The application will also receive a customer object from the server.
3. Your solution shall demonstrate sending the customer's full name and several partial names.
4. Your solution shall also demonstrate non-existing customers.
5. Display the details of the object.

Exercise 6: Finding a Customer at the Server Side by Customer Id

1. Create a TCP server-side application to process requests from a TCP client.
2. The server will receive integer data from the client that represents a customer id.
3. The server will search for the client based on the customer id and return a customer object to the client.
4. The server will log all its operations and interactions with the clients.

Exercise 7: Finding a Customer at the Client-Side by Customer Id

1. Create a TCP client-side application to several customer ids to the server.
2. The application will also receive a customer object from the server.
3. Your solution shall also demonstrate non-existing customers.
4. Display the details of the object.

Exercise 8: Executing Reference Application

1. Execute `TCPProductsServer.java` from package `server.app` and `TCPProductsServer.java` from package `client.app` separately.
2. Observe the output from both sides.

Exercise 9: Getting a List of Customers from the Server-Side

1. Create a TCP server-side application that will return a list of customers to the client.
2. The server will log all operations and interactions with the client.

Exercise 10: Receiving a List of Customers at the Client-Side

1. Create a TCP client-side application that will receive a list of customers from the server.
2. Display the details of the client in alphabetical order.

Additional Instruction

1. All source codes must be well documented and written.
2. Push your solutions on Github with a complete description.
3. Record the solution for this exercise in the Declaration of Solution.

End of Document