BITP 3123 DISTRIBUTED APPLICATION DEVELOPMENT

EMALIANA BINTI KASMURI

LAB 4: DEVELOPING TCP APPLICATION

NAME	MATRIC NO
THAQIF ASHRAF BIN ZULKIFLEE	B032010481

Exercise 1

Create New Eclipse's Workspace

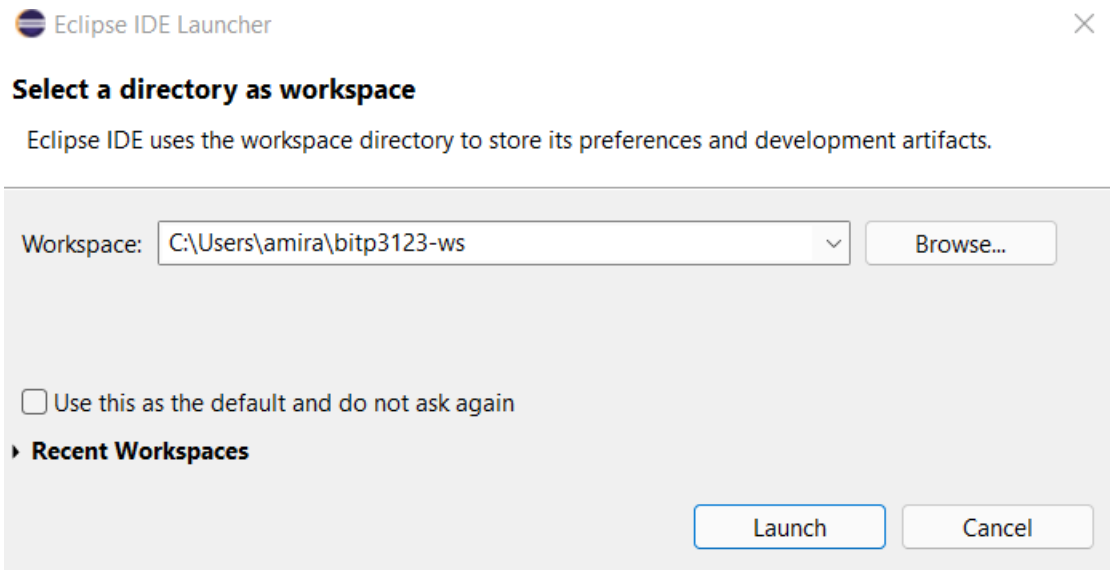

Figure 1: Window to change workspace

Exercise 2

Execute a Simple TCP Application

Execute Server-Side Application

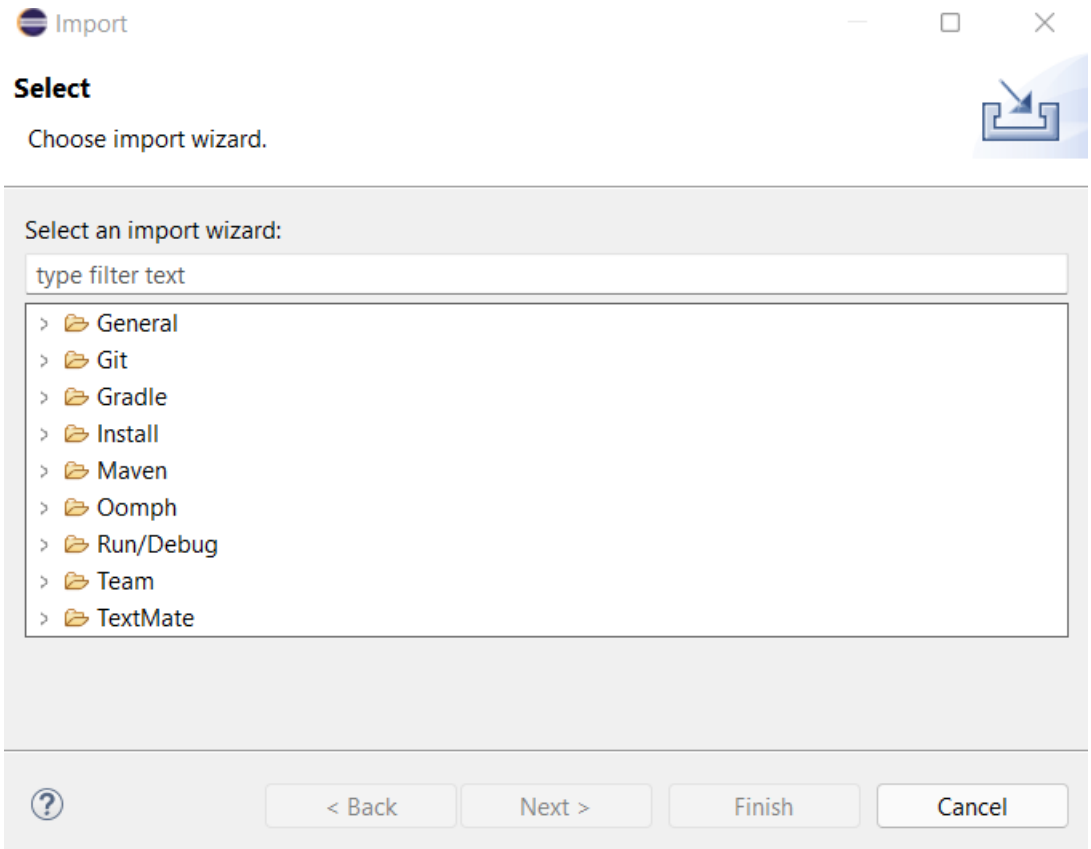

Figure 2: Window to import files into the project

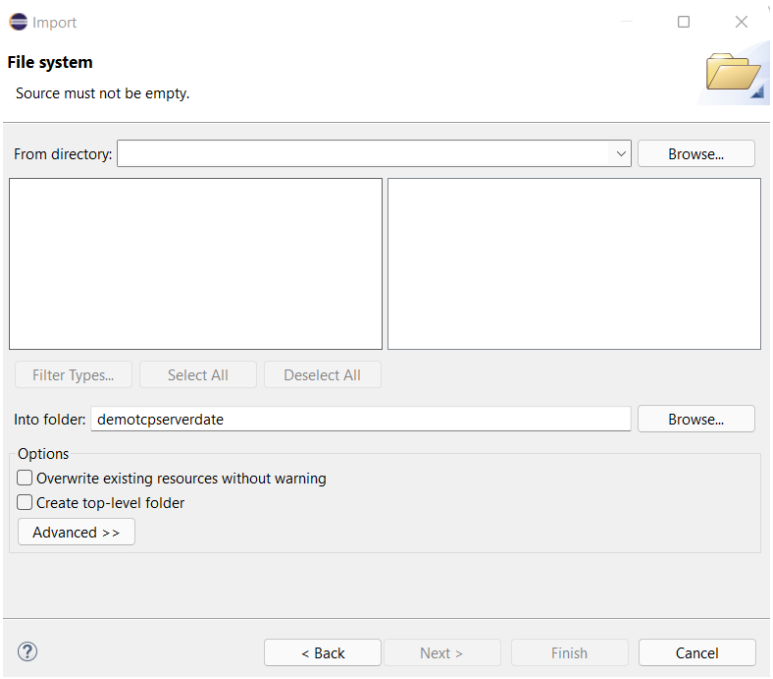

Figure 3: Window to specify file to be import into the project

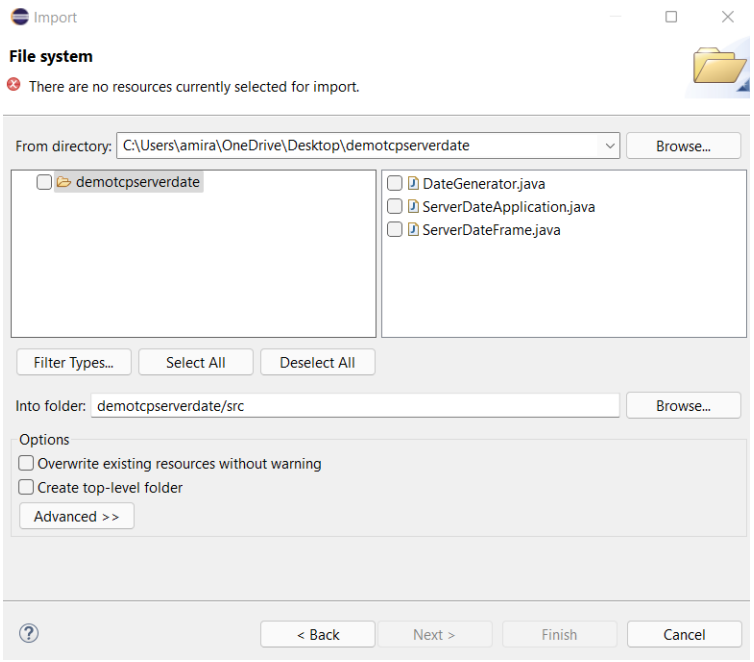

Figure 4: Example of the selected directory

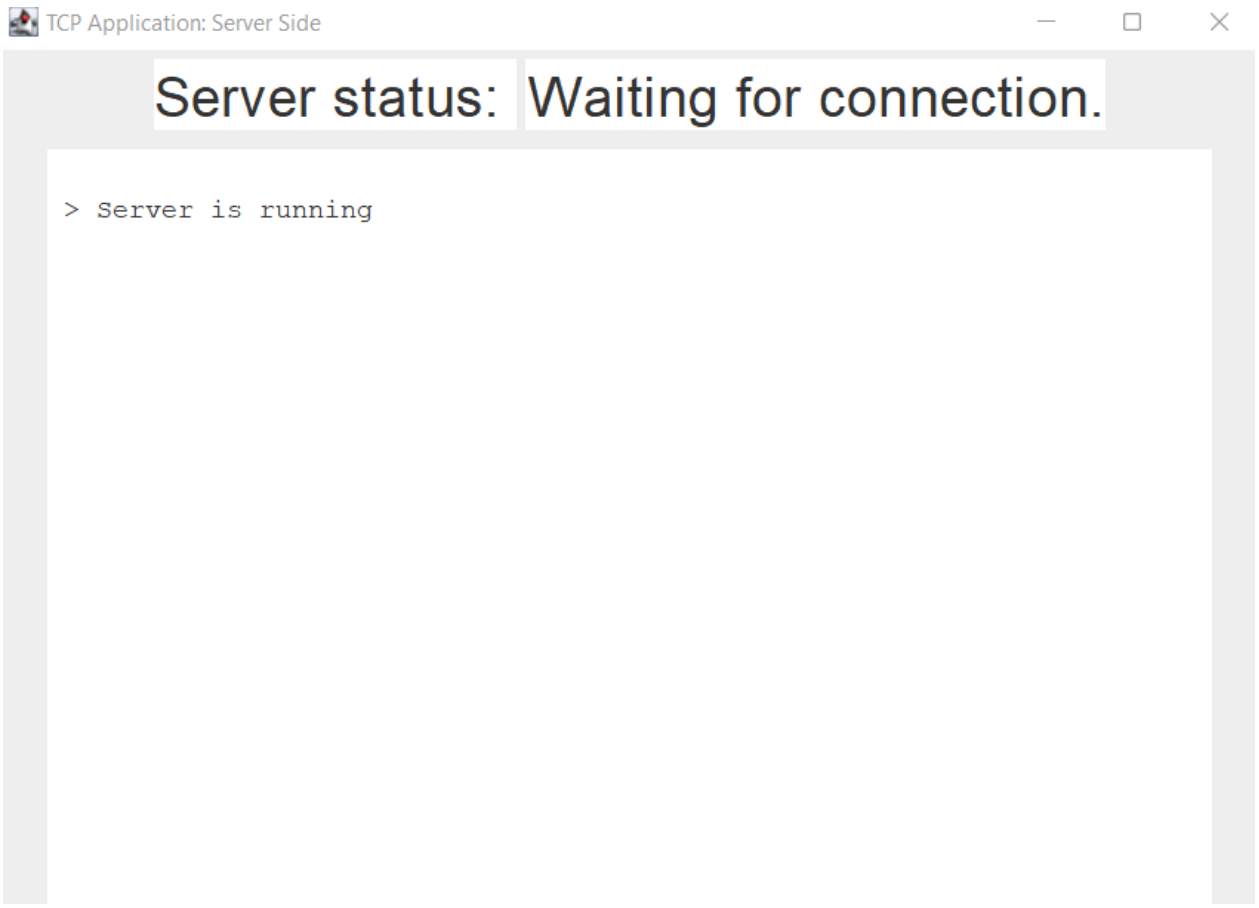

Figure 5: Window for server-side application

Execute Client-Side Application

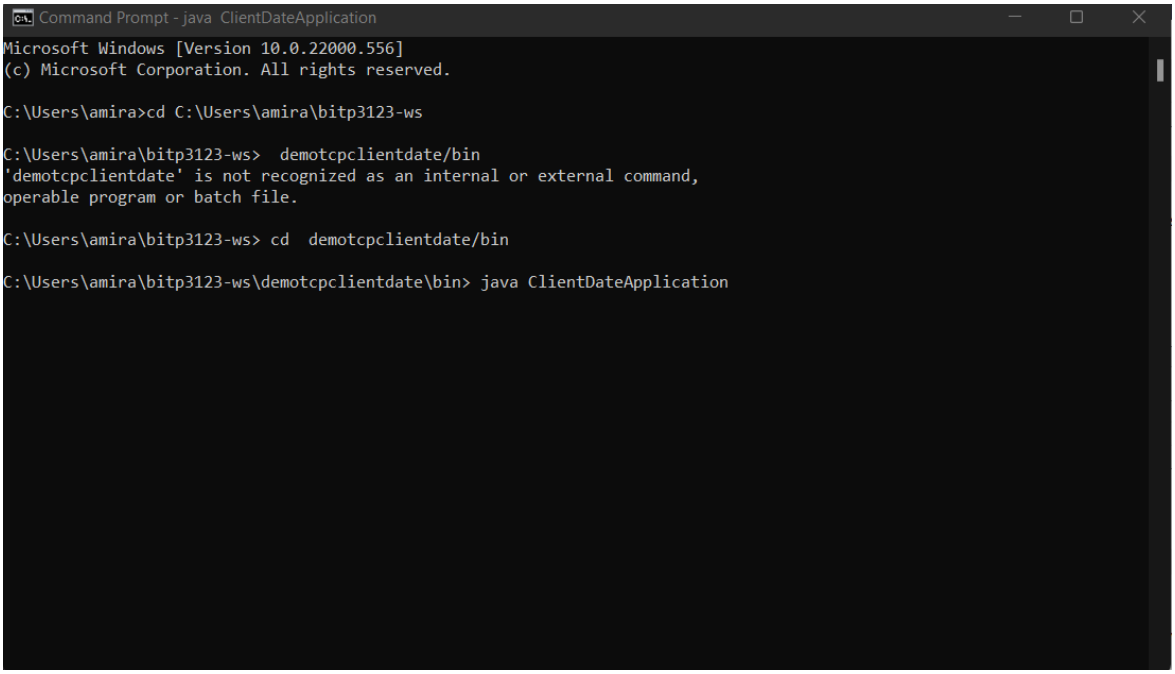

Figure 6: Window for execute ClientDateApplication in cmd

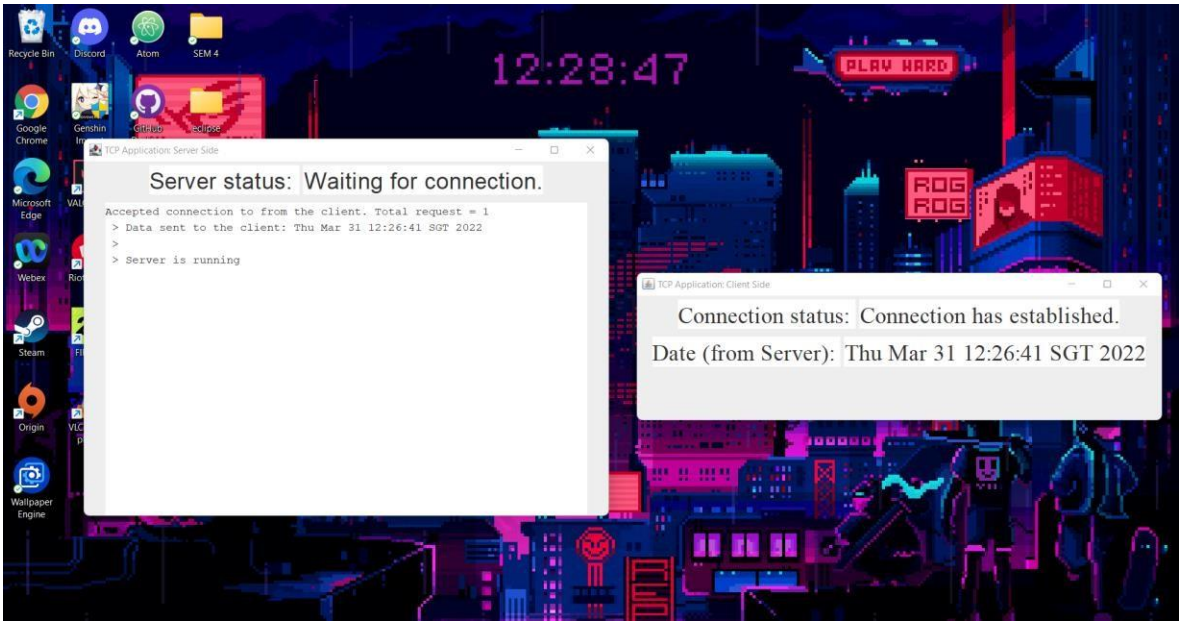

Figure 7: Output for both server-side application and client-side application

Exercise 3

Create a TCP-Based client-server application to process a length of a text

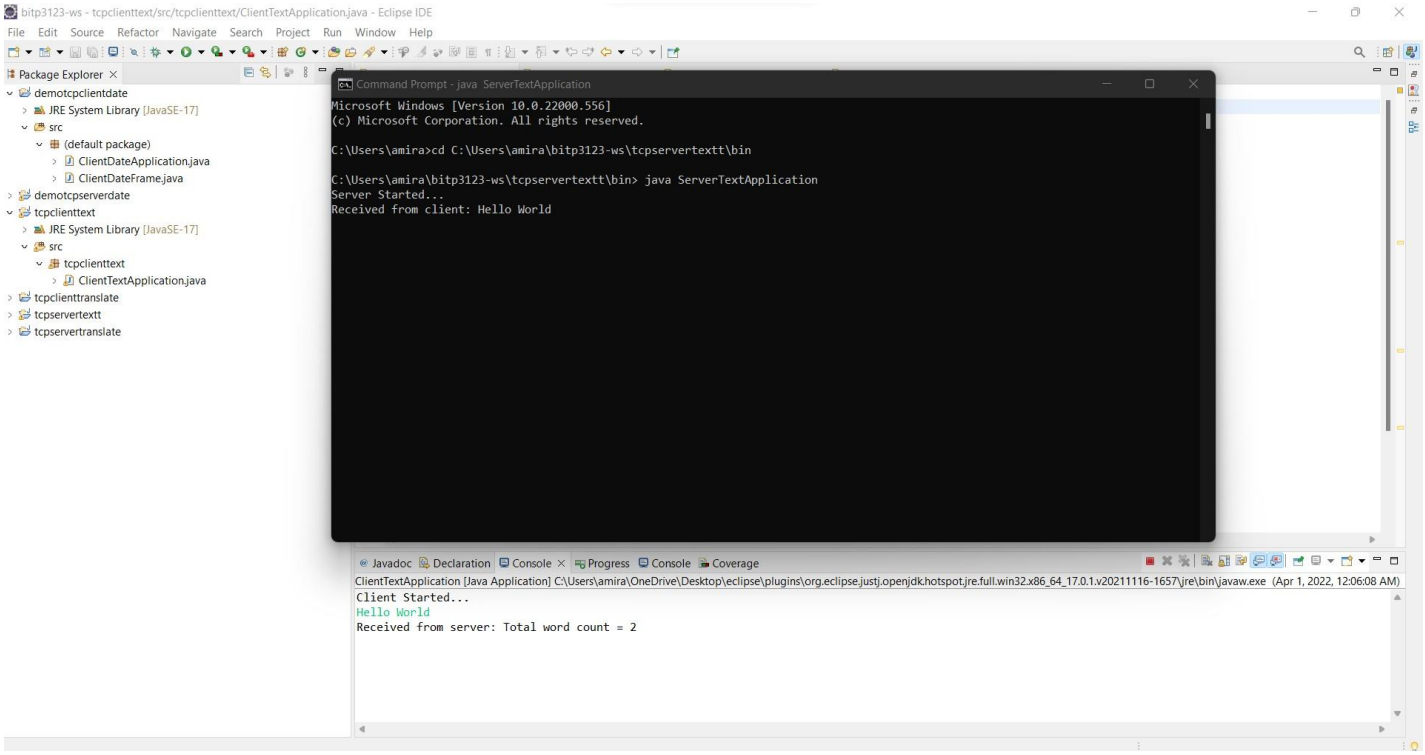

Figure 8: Output of TCP-Based client-server application to process a length of a text

Exercise 4

Write the following codes and run the application

```java
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 9: ServerTranslationApplication.java

Figure 10: Output ServerTranslationApplication.java

```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 11: ClientTranslation.java

Figure 12: Output ClientTranslation.java