

JAVA CLIENT-SERVER APP WITH XML (SOAP) WEB SERVICES

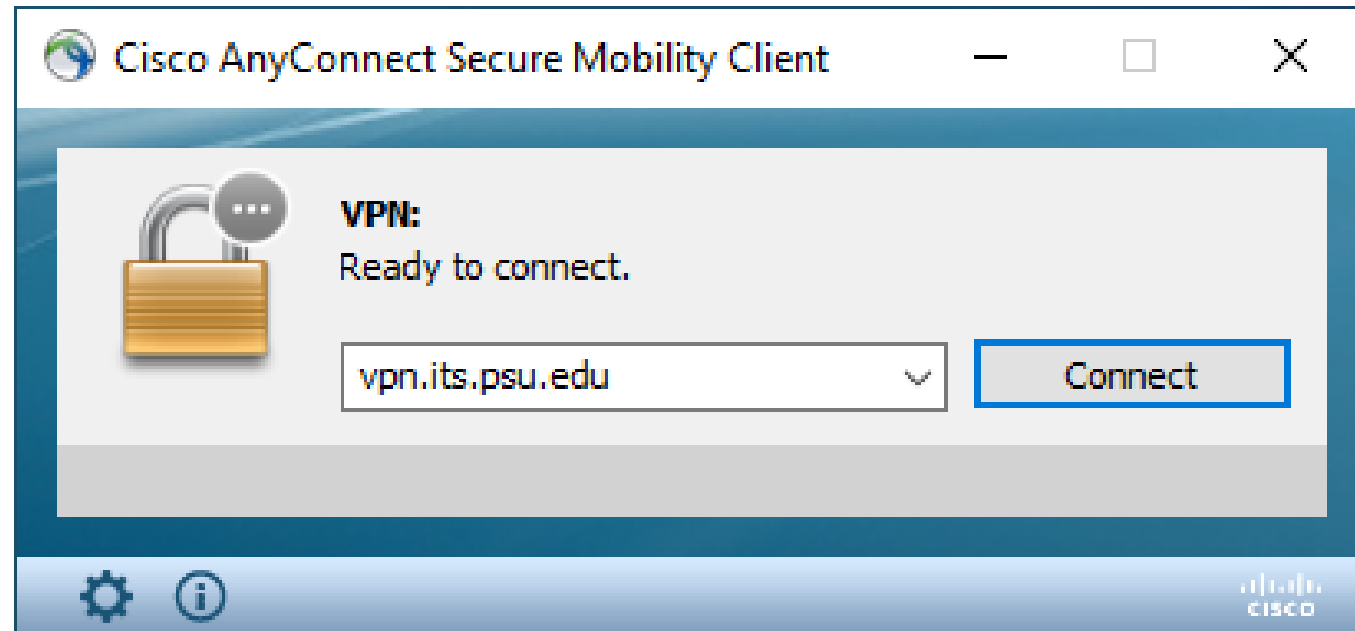
This exercise was adapted from the following tutorials, which I encourage you to review for more information:

- [CodeJava: Java Client Server XML Web Services \(JAX-WS\) Tutorial](#) by [Nam Ha Minh](#)
- LinkedIn Learning: Java EE 7: Web Services (by Kesha Williams) *Available through the IST411/MIS466 LinkedIn Learning Collection

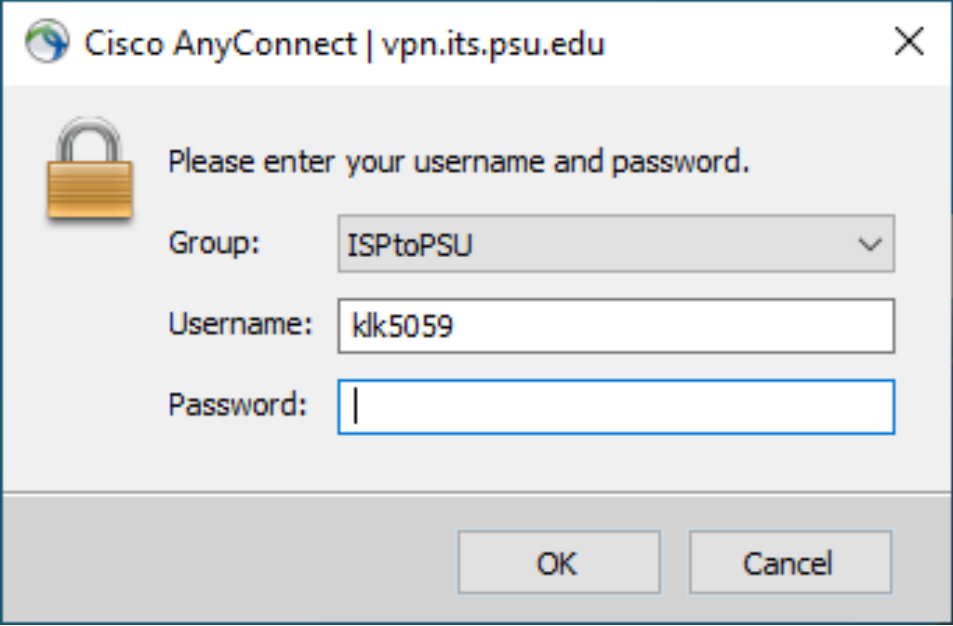


PART 1: CONNECT TO VPN AND SIGN INTO VIRTUAL MACHINE

STEP 1: Open the Cisco AnyConnect Secure Mobility Client application. Select *vpn.its.psu.edu* and click the **Connect** button.



STEP 2: In the *Cisco AnyConnect | vpn.its.psu.edu* dialog, enter your Penn State username and password and click the **OK** button.

A screenshot of a Cisco AnyConnect login dialog box. The title bar reads "Cisco AnyConnect | vpn.its.psu.edu" with a close button (X) on the right. The main area has a light gray background. On the left is a yellow padlock icon. To its right is the text "Please enter your username and password." Below this are three input fields: a "Group:" dropdown menu showing "ISPtoPSU" with a downward arrow, a "Username:" text box containing "klk5059", and a "Password:" text box with a single vertical line cursor. At the bottom are two buttons: "OK" and "Cancel".

Cisco AnyConnect | vpn.its.psu.edu

Please enter your username and password.

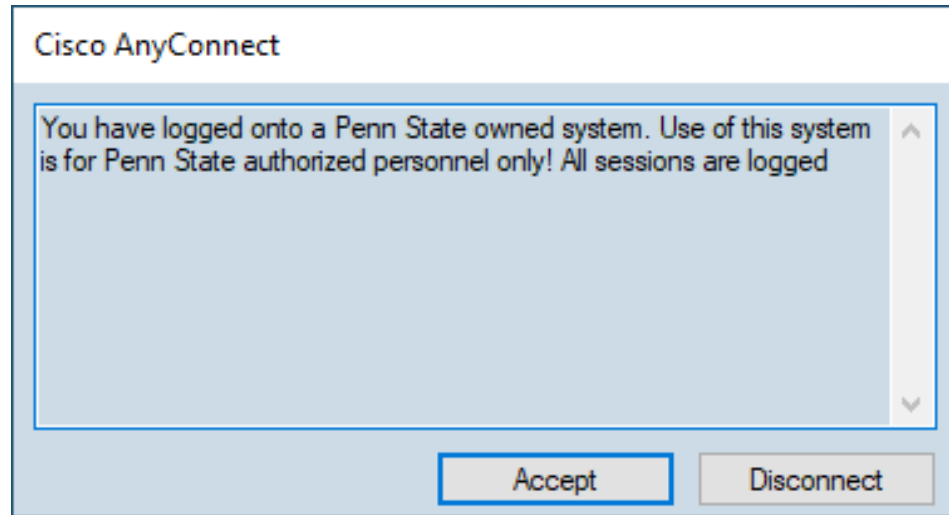
Group: ISPtoPSU

Username: klk5059

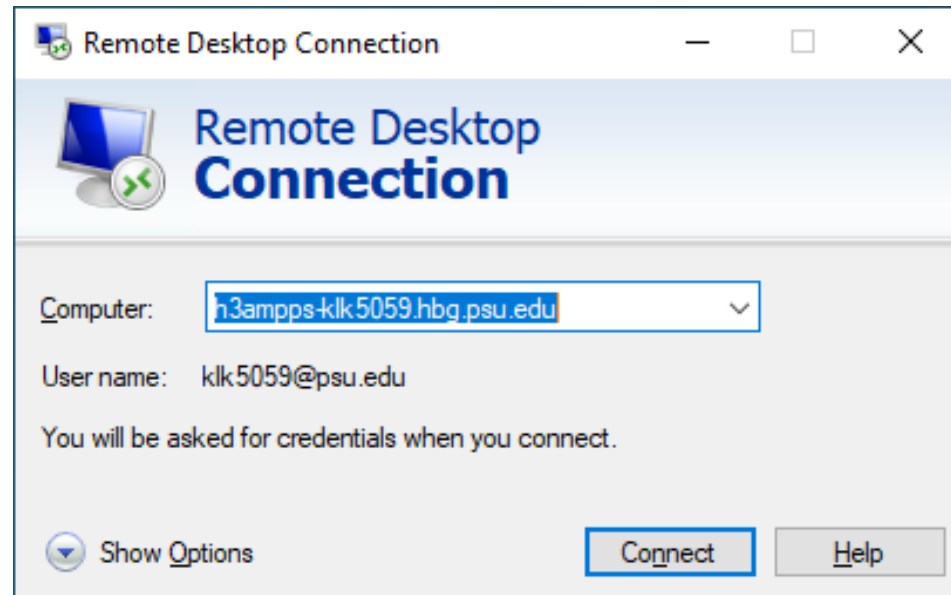
Password: |

OK Cancel

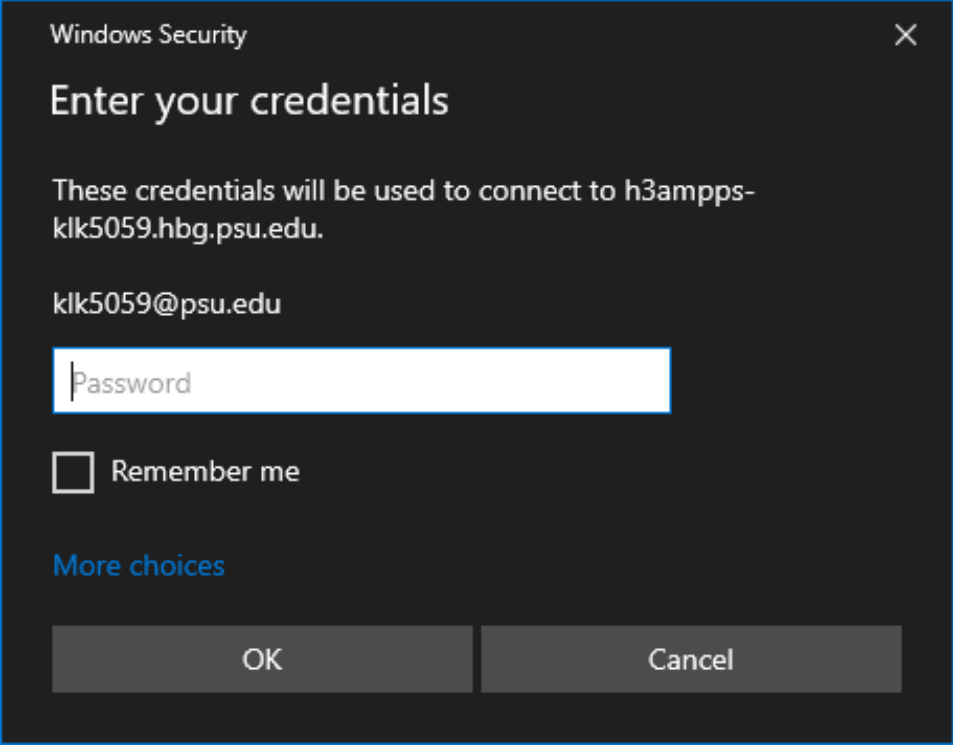
STEP 3: In the *Cisco AnyConnect* dialog, click the *Accept* button.



STEP 4: Open the Remote Desktop Connection application and enter *h3ampps-yourpsuusername.hbg.psu.edu* (where *yourpsuusername* is your Penn State username, for example: h3ampps-klk5059.hbg.psu.edu). Click the **Connect** button.



STEP 5: In the *Windows Security* dialog, select *More choices*.



A screenshot of a Windows Security dialog box titled "Enter your credentials". The dialog has a dark background and a light blue border. At the top, it says "Windows Security" with a close button (X) on the right. Below the title, it says "Enter your credentials". A message states: "These credentials will be used to connect to h3ampps-klk5059.hbg.psu.edu." Below this, the email address "klk5059@psu.edu" is displayed. There is a text input field with the placeholder text "Password". Below the input field is a checkbox labeled "Remember me". At the bottom, there is a link "More choices" in blue text. At the very bottom, there are two buttons: "OK" and "Cancel".

Windows Security

Enter your credentials

These credentials will be used to connect to h3ampps-klk5059.hbg.psu.edu.

klk5059@psu.edu

Password

☐ Remember me

[More choices](#)

OK Cancel

STEP 6: Select *Use a different account*. Enter your Penn State *email address* and password. Click the *OK* button.

Windows Security

Enter your credentials


These credentials will be used to connect to h3ampps-
klk5059.hbg.psu.edu.


User name

Password

☐ Remember me

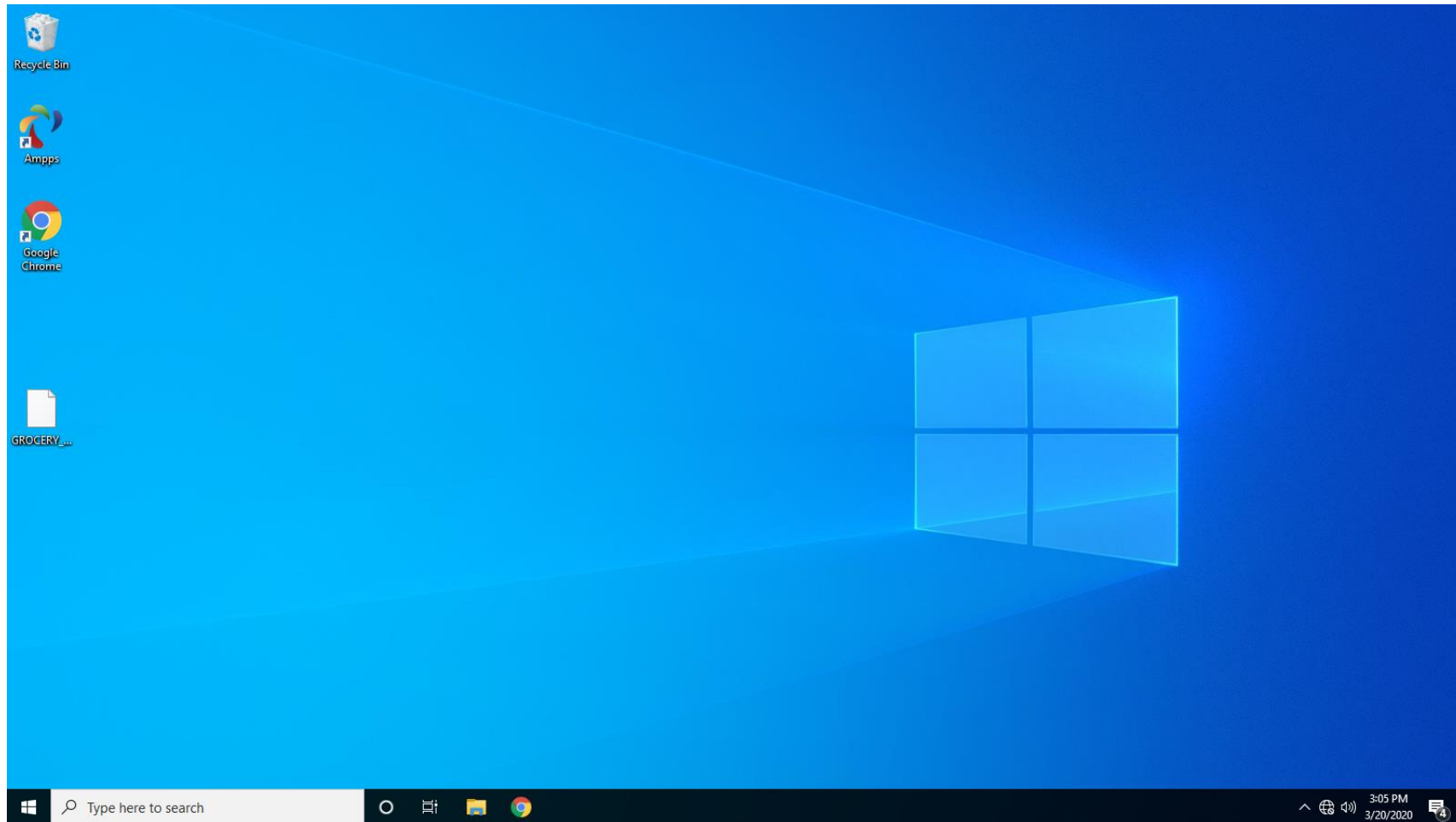
[More choices](#)

 klk5059@psu.edu

 Use a different account

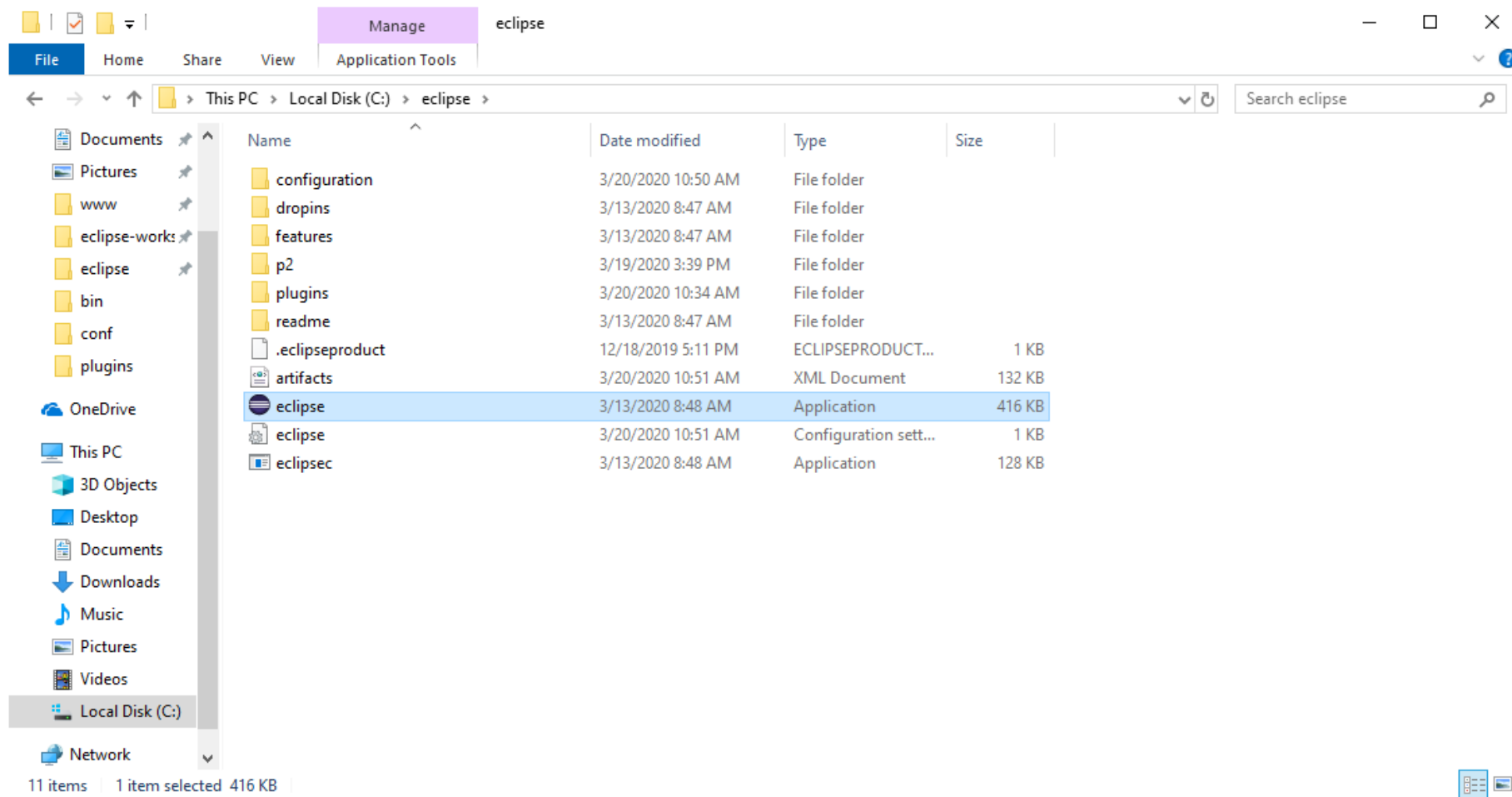
OK Cancel

STEP 7: You should now see the desktop of your virtual machine.

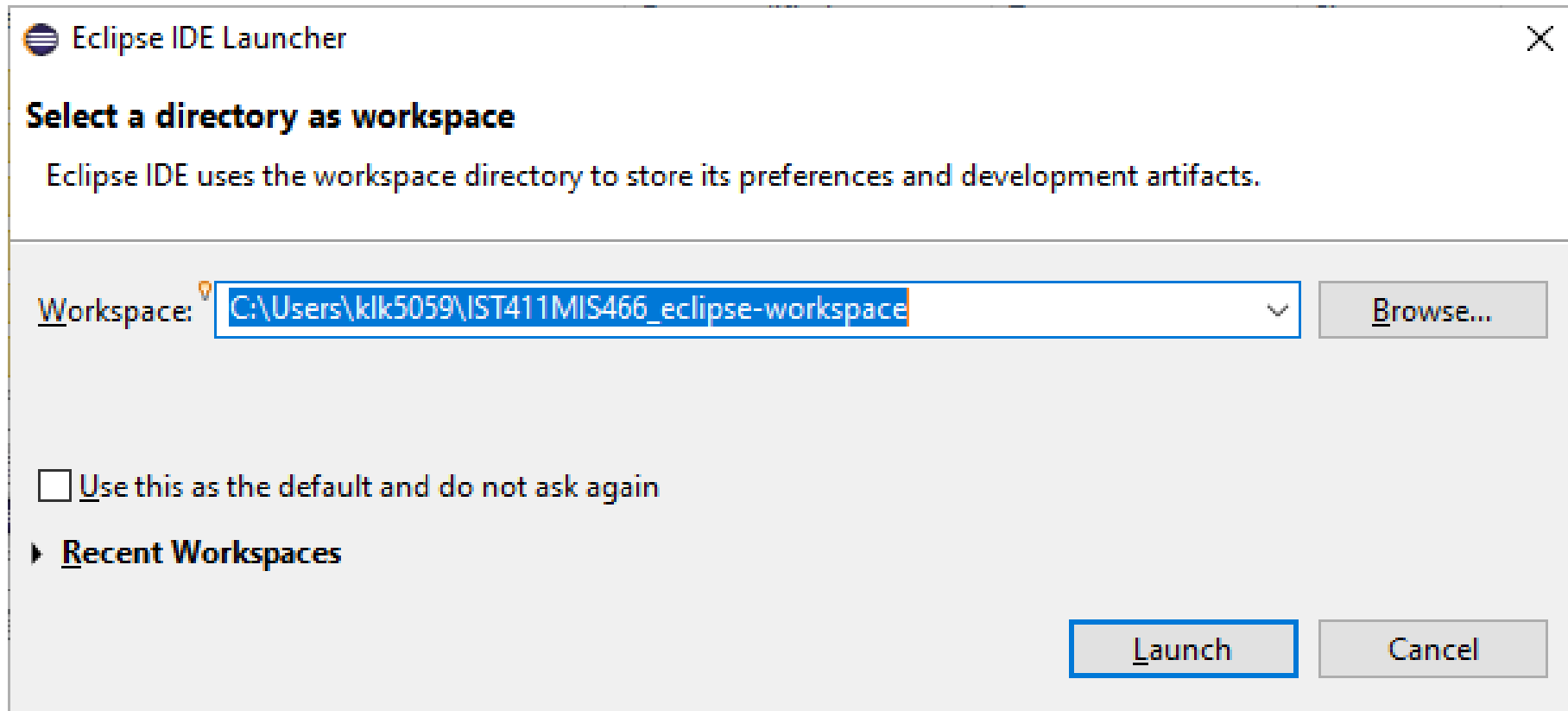


PART 2: OPEN ECLIPSE

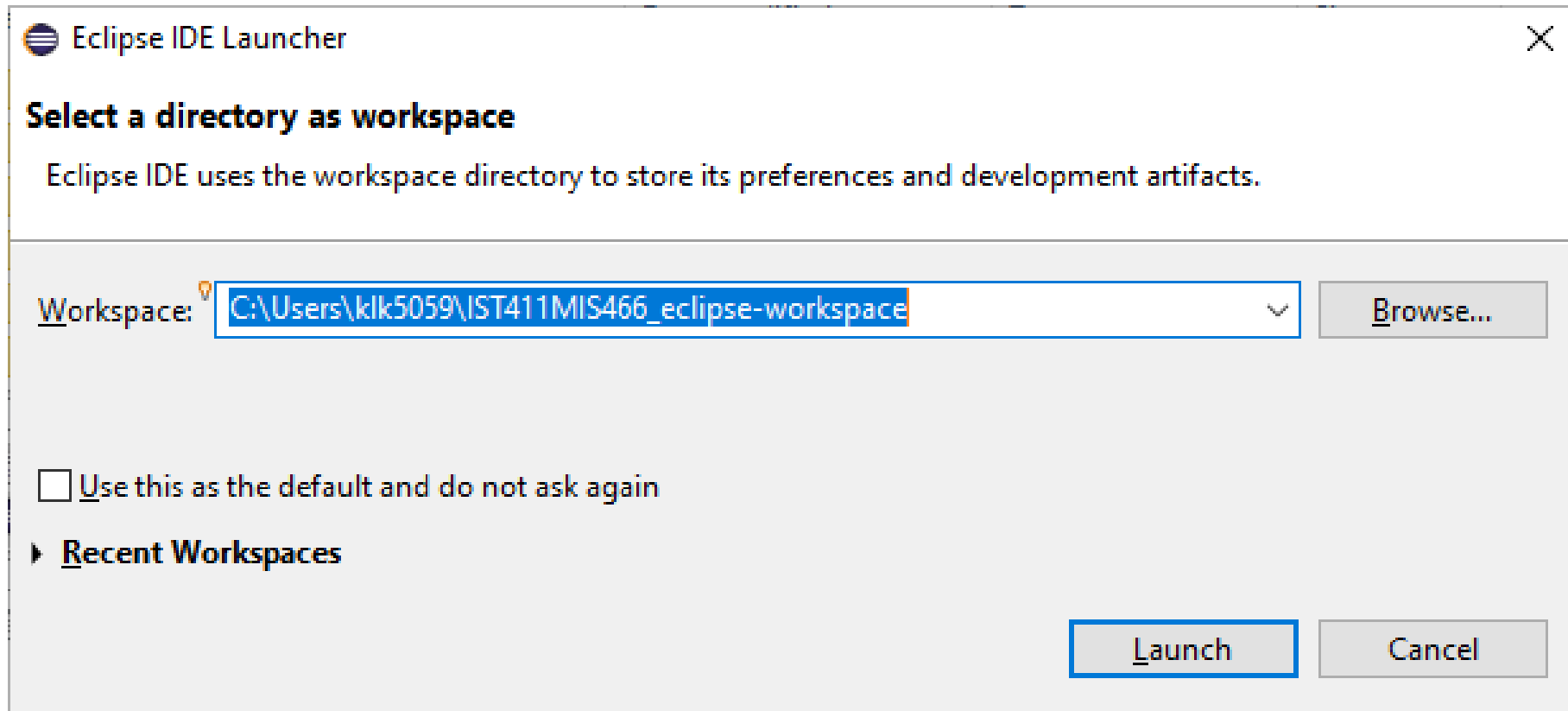
STEP 1: In a File Explorer window, navigate to *C:\eclipse* and launch the Eclipse application (highlighted in the screenshot below).



STEP 2: In the *Eclipse IDE Launcher* dialog, select (or enter) a directory that you want to use as your workspace. I created a folder called *IST411MIS466_eclipse-workspace*, in the directory *C:\Users\klk5059* to use for my workspace. Click the *Launch* button.

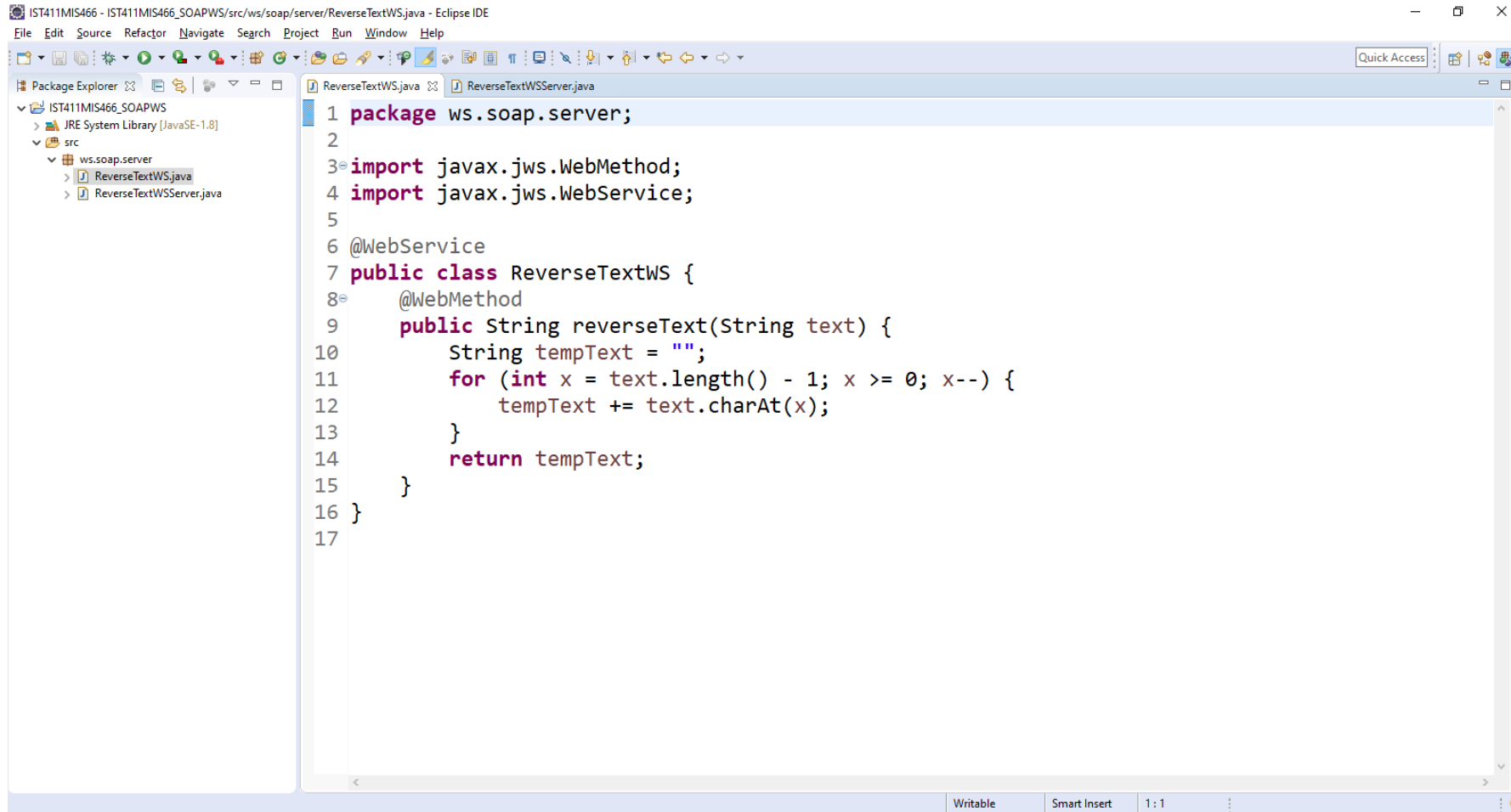


STEP 3: In the *Eclipse IDE Launcher* dialog, select (or enter) a directory that you want to use as your workspace. I created a folder called *IST411MIS466_eclipse-workspace*, in the directory *C:\Users\klk5059* to use for my workspace. Click the *Launch* button.



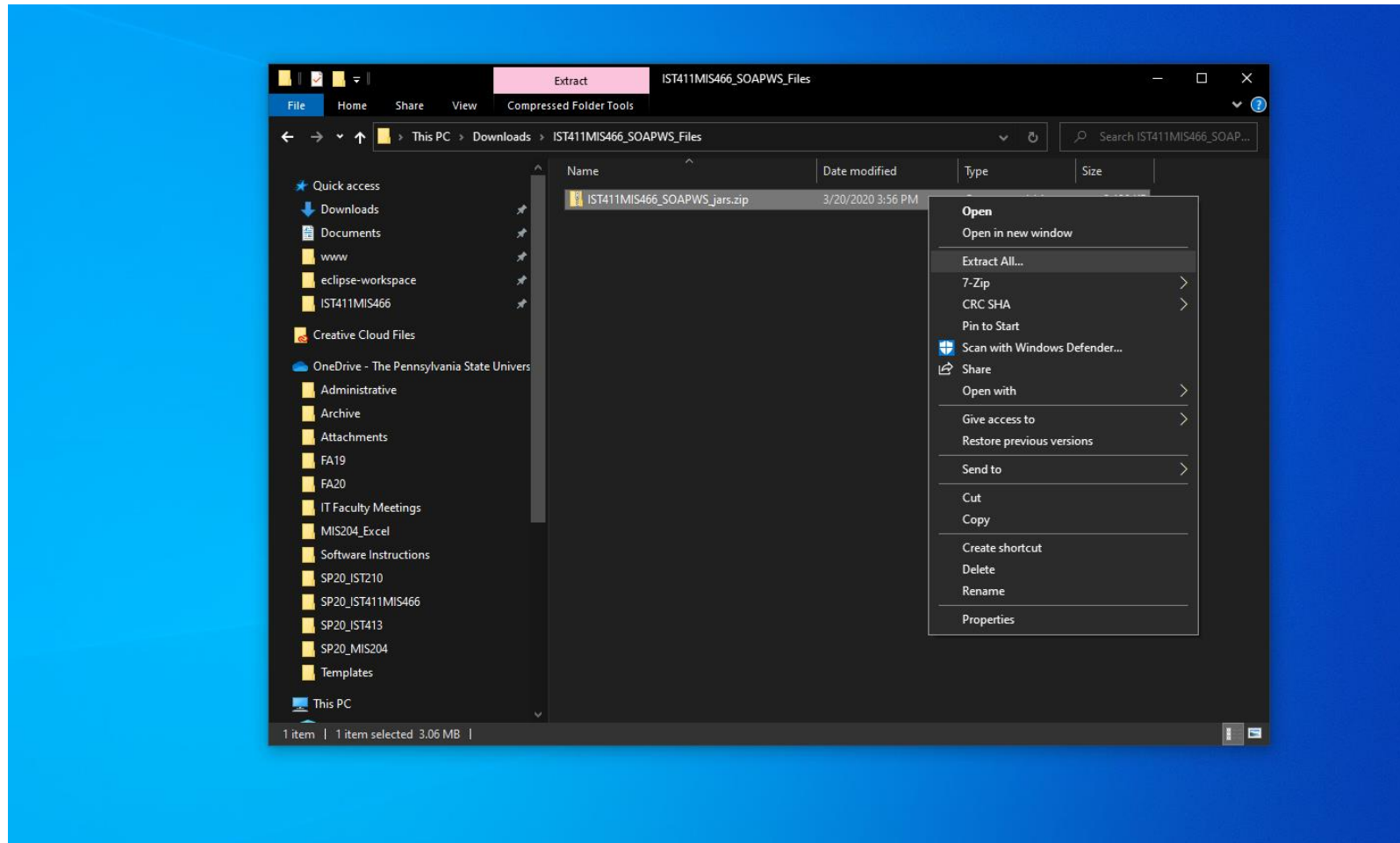
PART 3: CREATE THE WEB SERVICE CLASS

STEP 1: In a new Java project, create a new package called **ws.soap.server**. In that package, create a new class called **ReverseTextWS**, which will be the web service. Add the following code to the class. This class uses the annotations (indicated by the @ symbol). According to the Java documentation: “Annotations, a form of metadata, provide data about a program that is not part of the program itself. Annotations have no direct effect on the operation of the code they annotate.” The **@WebService** annotation indicates that the class is a web service. The **@WebMethod** annotation indicates that the **reverseText()** method is exposed to web service clients. You will see errors on lines 3, 4, 6, and 8. We’ll resolve those next.

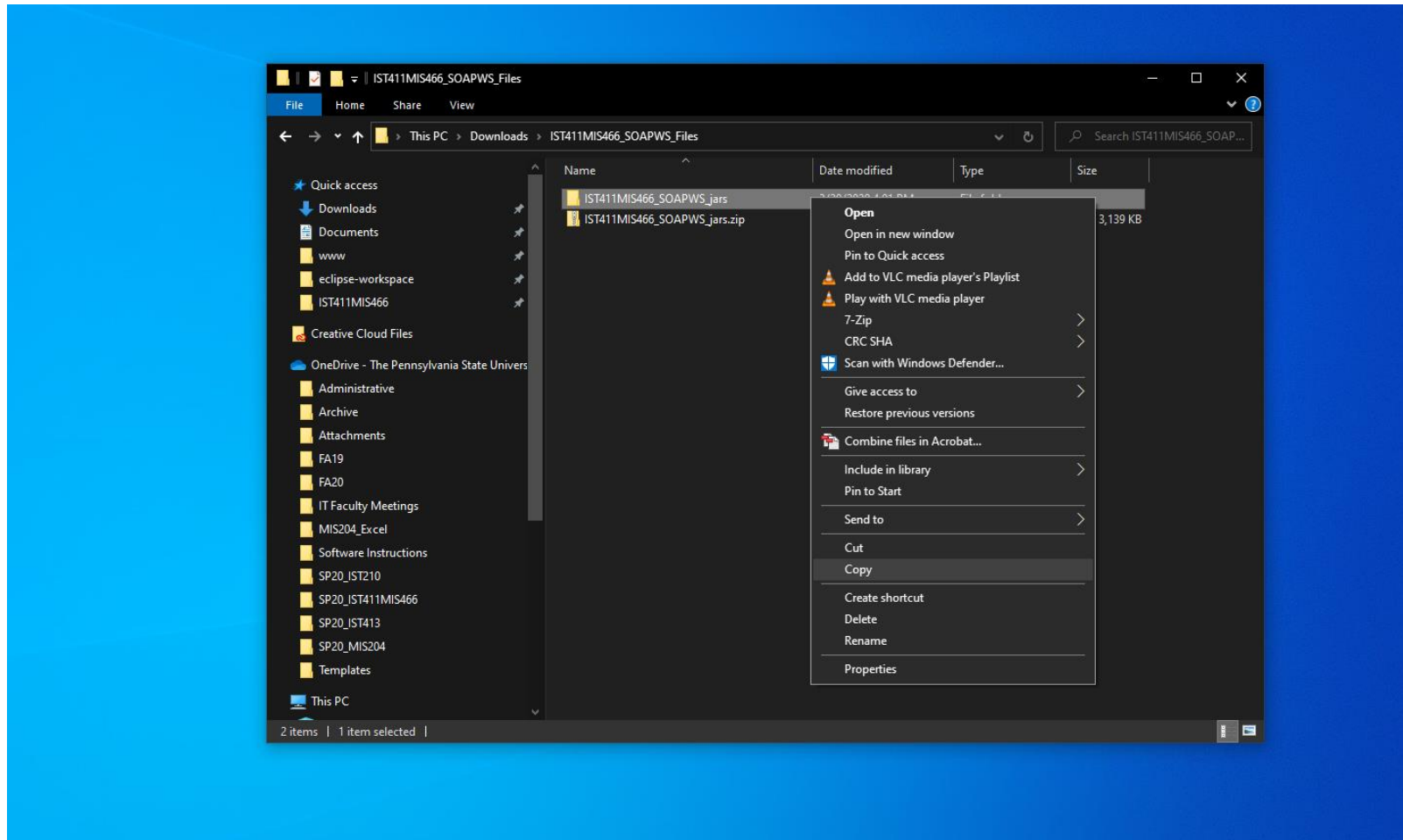


```
1 package ws.soap.server;
2
3 import javax.jws.WebMethod;
4 import javax.jws.WebService;
5
6 @WebService
7 public class ReverseTextWS {
8     @WebMethod
9     public String reverseText(String text) {
10         String tempText = "";
11         for (int x = text.length() - 1; x >= 0; x--) {
12             tempText += text.charAt(x);
13         }
14         return tempText;
15     }
16 }
17
```

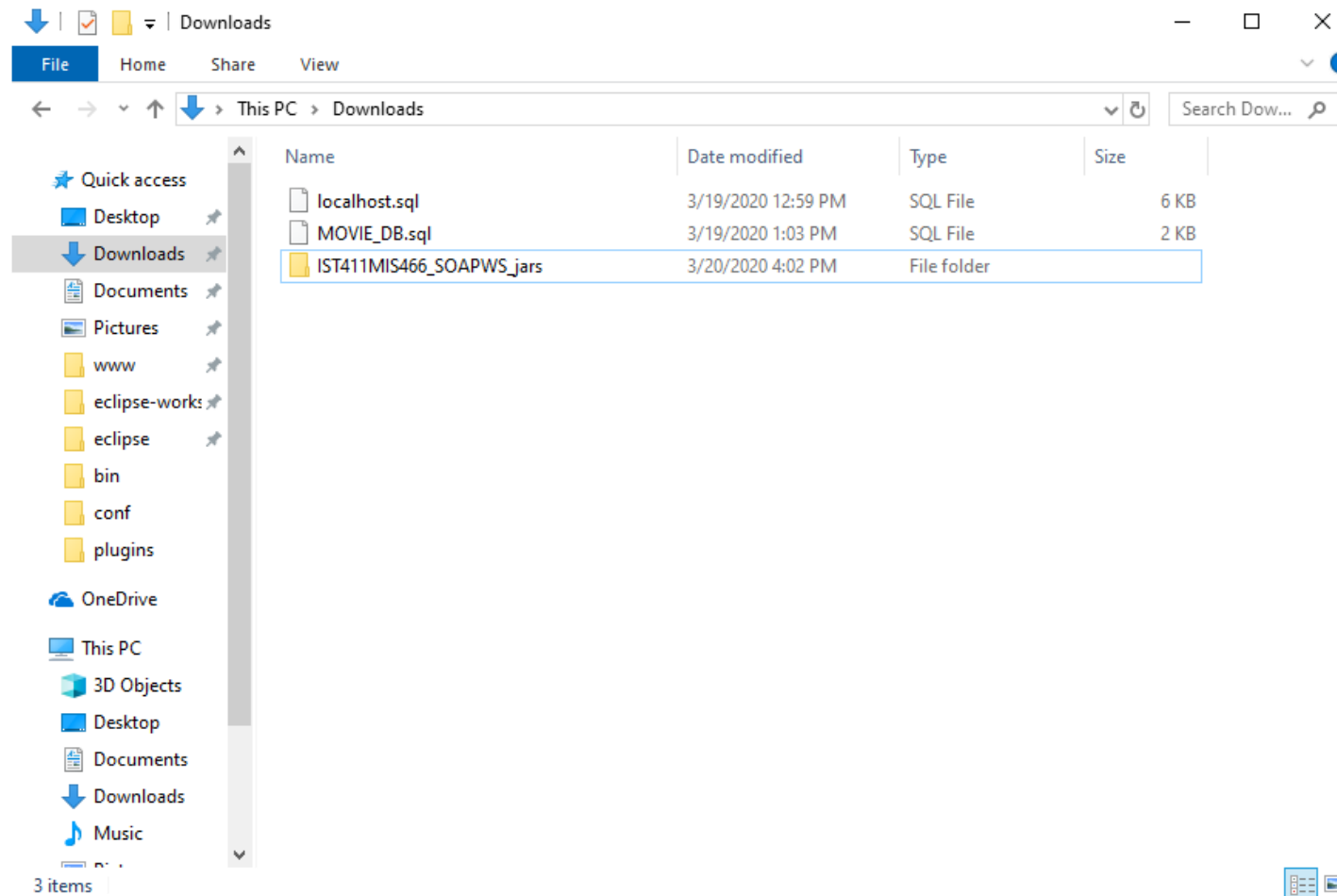
STEP 2: Download the IST411MIS466_SOAPWS_jars.zip folder from Canvas and extract its contents onto your computer.



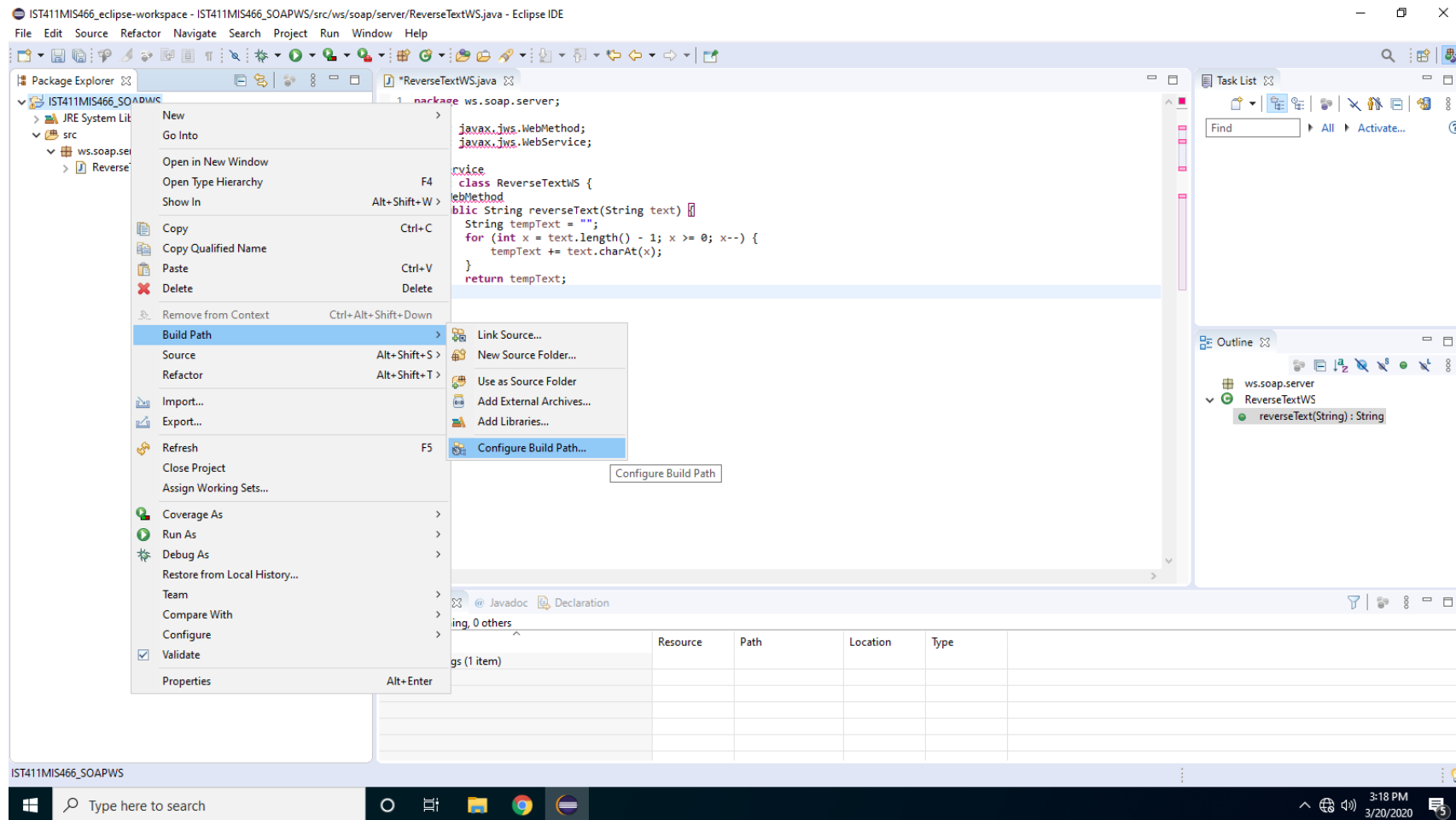
STEP 3: Copy the extracted folder.



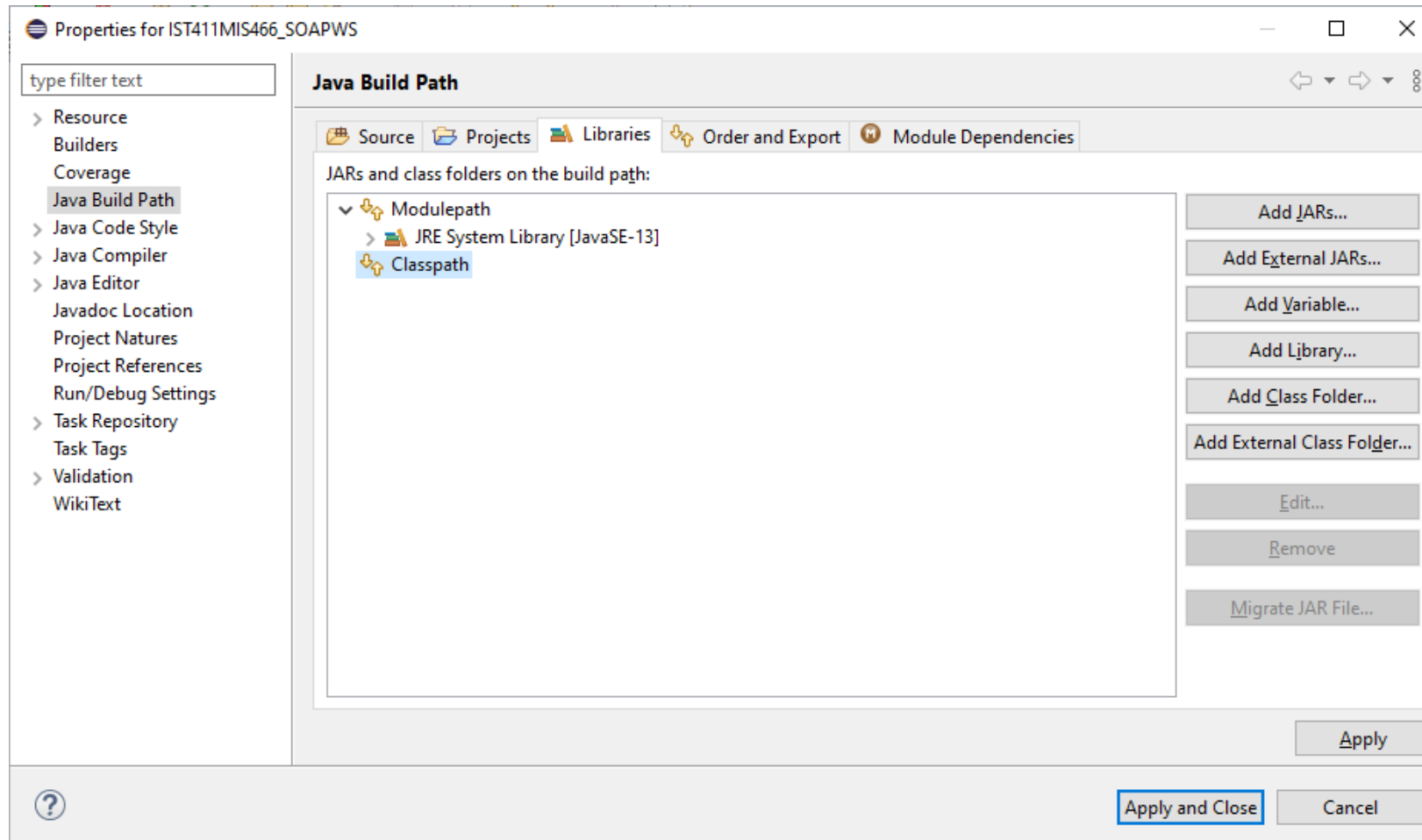
STEP 4: Back in your virtual machine, in a File Explorer window, open the **Downloads** directory. Paste the folder that you copied in the previous step.



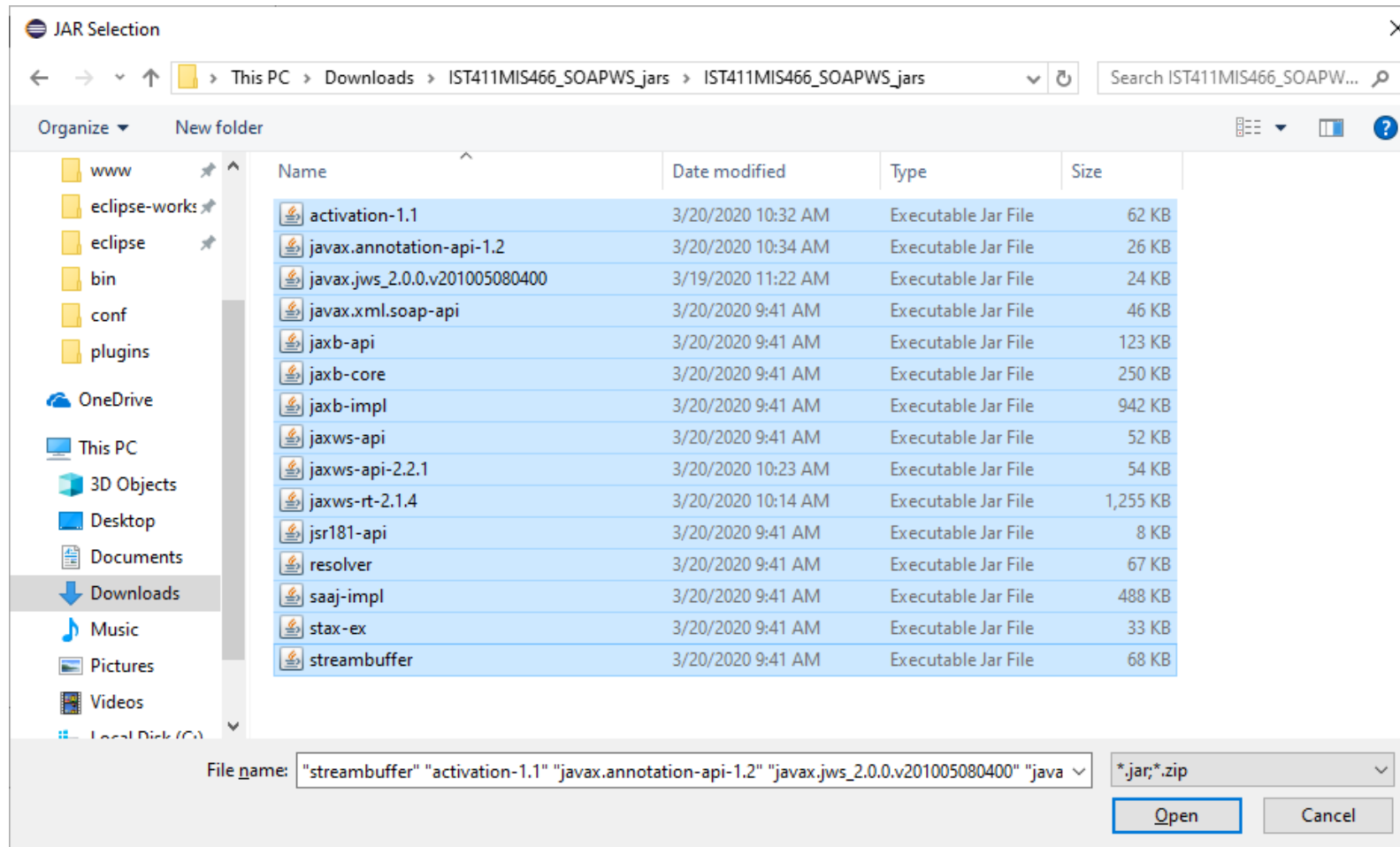
STEP 5: Back in Eclipse, right-click on the project and select **Build Path**, then **Configure Build Path....**



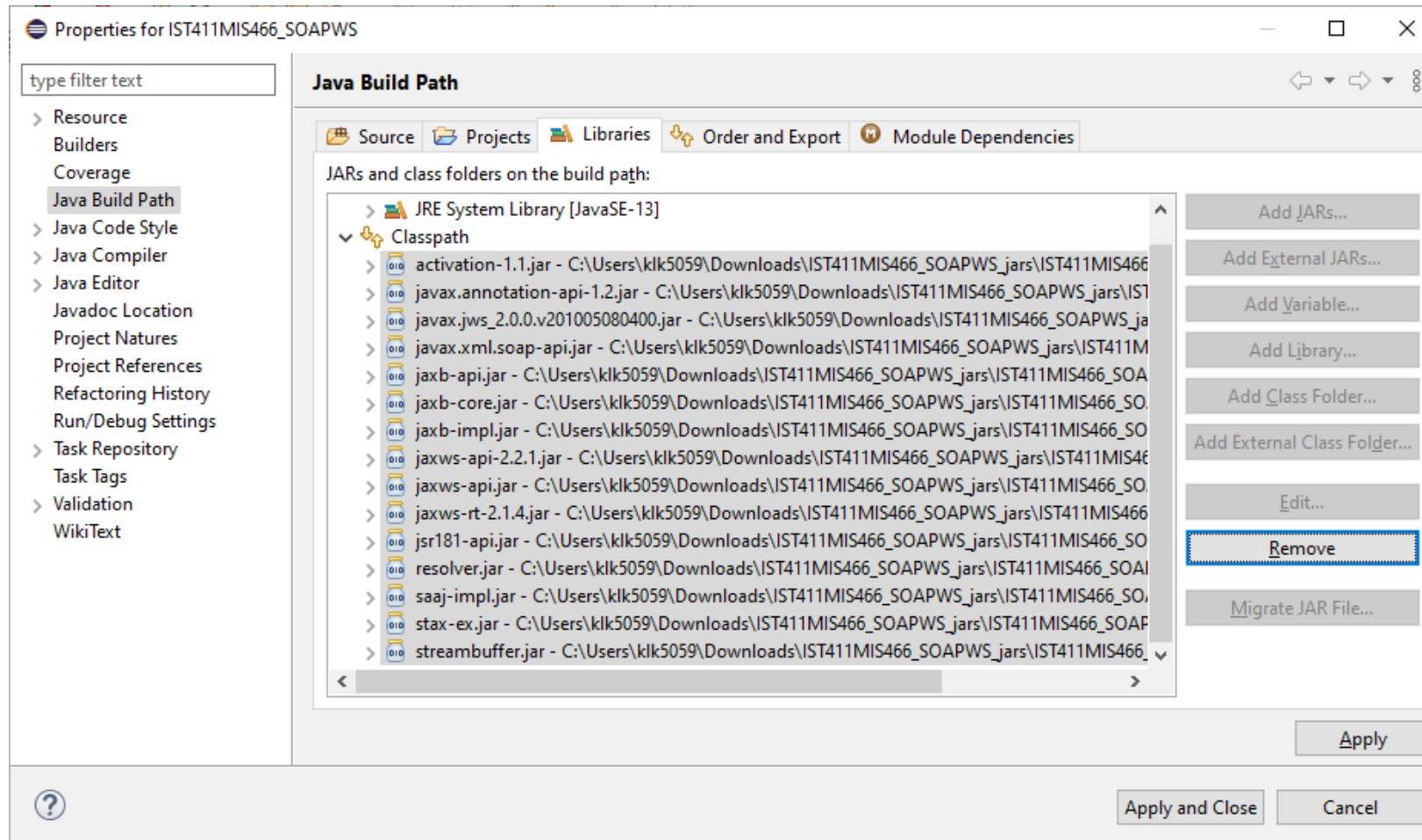
STEP 6: In the *Properties for...* dialog, select the *Libraries* tab, then click on *Classpath*. Click the *Add External JARs...* button.



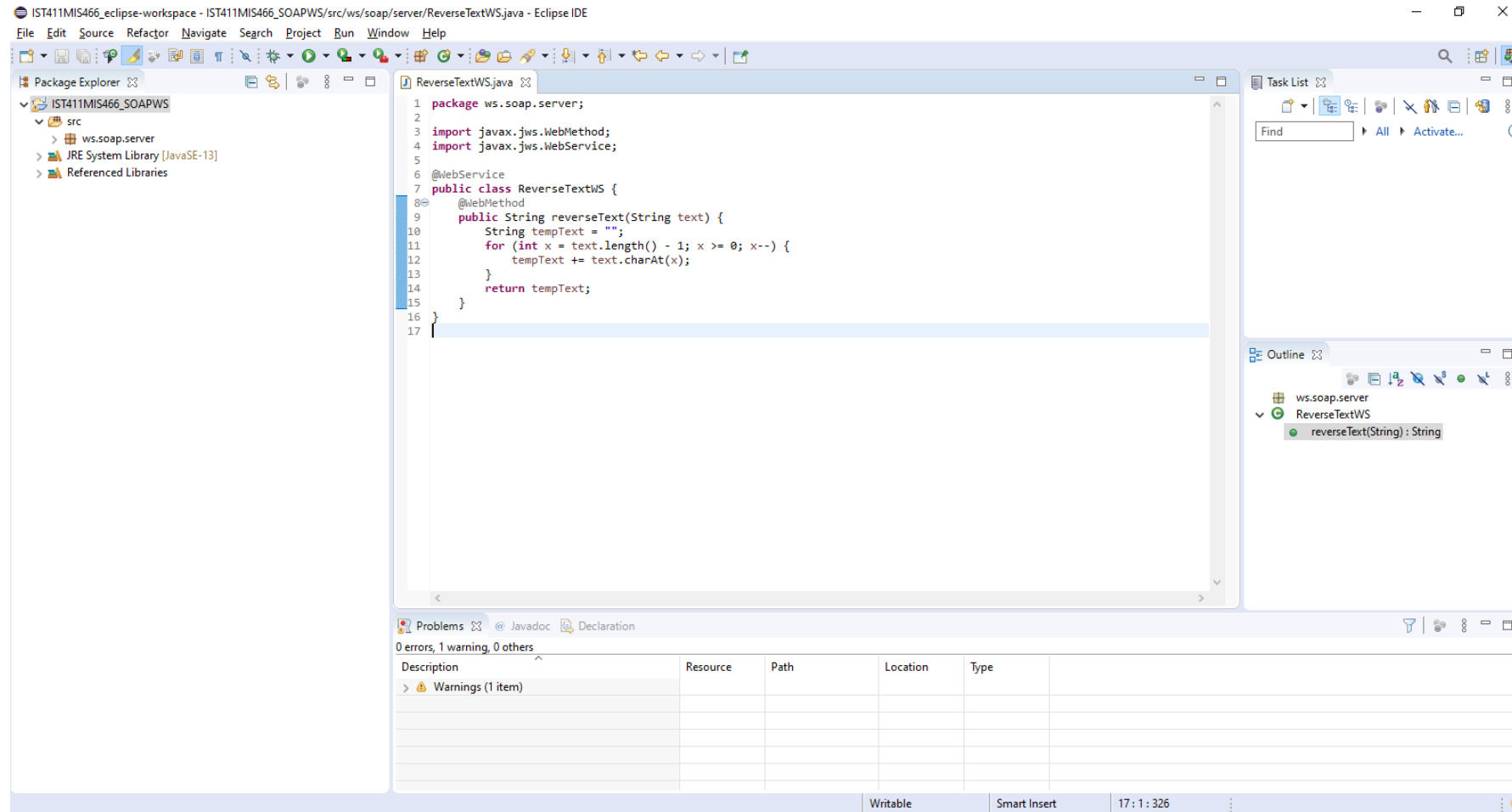
STEP 7: In the **JAR Selection** dialog, navigate to the **Downloads\IST411MIS466_SOAPWS_jars** directory and select the .jar files from the folder that you pasted there in STEP 4 (use the **Shift** key or the **Ctrl** key while selecting the .jar files to select all of them). Click the **Open** button.



STEP 8: Back in the *Properties for...* dialog, you should see all the .jar files listed under *Classpath*. Click the *Apply and Close* button.

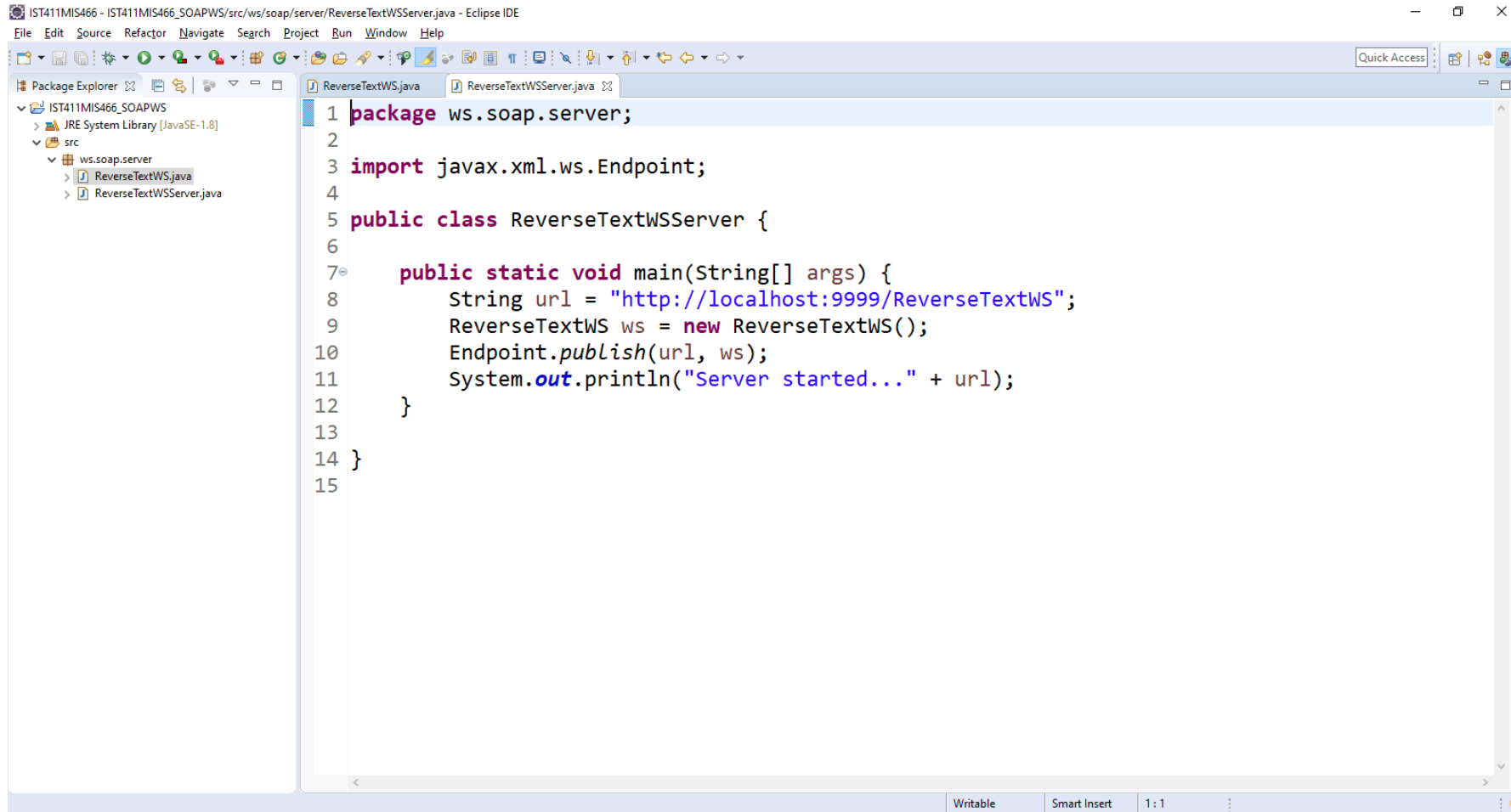


STEP 9: The errors in the *ReverseTextWS.java* file should now be resolved.



PART 2: CREATE THE WEB SERVICE SERVER

STEP 1: In the same package (*ws.soap.server*), create another class called *ReverseTextWSServer*, which will be the web service server and will host the *ReverseTextWS* web service. Add the following code to the class. The *url* variable is where the web service will be published. We will publish it to the *localhost* on port *9999* as a resource called *ReverseTextWS*. Next, we create the web service that will be published as an instance of *ReverseTextWS*. Finally, we call the *Endpoint.publish()* method to create the endpoint, or the address from which the web service can be accessed by the clients.

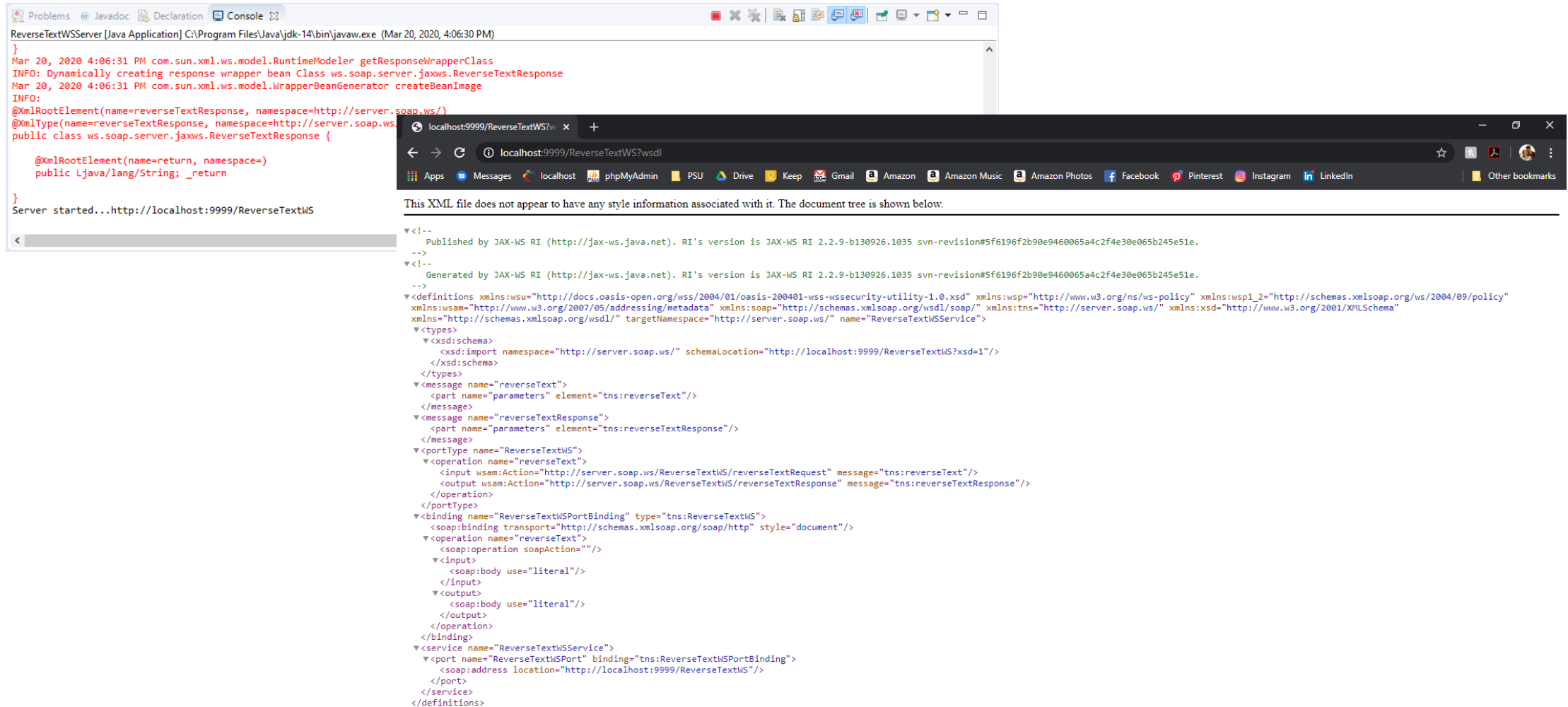


The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: IST411MIS466_SOAPWS > src > ws.soap.server. The main editor window shows the code for ReverseTextWSServer.java. The code defines a package, imports javax.xml.ws.Endpoint, and creates a public class ReverseTextWSServer with a main method that sets up the web service endpoint.

```
1 package ws.soap.server;
2
3 import javax.xml.ws.Endpoint;
4
5 public class ReverseTextWSServer {
6
7     public static void main(String[] args) {
8         String url = "http://localhost:9999/ReverseTextWS";
9         ReverseTextWS ws = new ReverseTextWS();
10        Endpoint.publish(url, ws);
11        System.out.println("Server started..." + url);
12    }
13
14 }
15
```

PART 3: PUBLISH THE WEB SERVICE TO THE SERVER

STEP 1: Run the *ReverseTextWSServer* class. In the console you may see warning/info messages about deprecated libraries/classes. The last line of the console should display *Server started...http://localhost:9999/ReverseTextWS*. Then in a browser, go to: <http://localhost:9999/ReverseTextWS?wsdl>. You should see the WSDL for the *ReverseTextWS* web service.



The screenshot shows two windows. The left window is an IDE console for the *ReverseTextWSServer* application. It displays several log messages from the *com.sun.xml.ws.model* package, including *getResponseWrapperClass*, *Dynamically creating response wrapper bean Class ws.soap.server.jaxws.ReverseTextResponse*, and *createBeanImage*. The final message is *Server started...http://localhost:9999/ReverseTextWS*. The right window is a web browser showing the WSDL file at *localhost:9999/ReverseTextWS?wsdl*. The browser displays the XML content of the WSDL, which defines the *ReverseTextWS* service and its *reverseText* operation. The XML includes namespaces for *wsu*, *wsam*, *wsdl*, *tns*, and *xsd*, and defines the *ReverseTextWS* service with a *reverseText* operation that takes a *reverseTextRequest* and returns a *reverseTextResponse*.

```
ReverseTextWSServer [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 20, 2020, 4:06:30 PM)
}
Mar 20, 2020 4:06:31 PM com.sun.xml.ws.model.RuntimeModeler getResponseWrapperClass
INFO: Dynamically creating response wrapper bean Class ws.soap.server.jaxws.ReverseTextResponse
Mar 20, 2020 4:06:31 PM com.sun.xml.ws.model.WrapperBeanGenerator createBeanImage
INFO:
@XmlRootElement(name=reverseTextResponse, namespace=http://server.soap.ws/)
@XmlType(name=reverseTextResponse, namespace=http://server.soap.ws/)
public class ws.soap.server.jaxws.ReverseTextResponse {

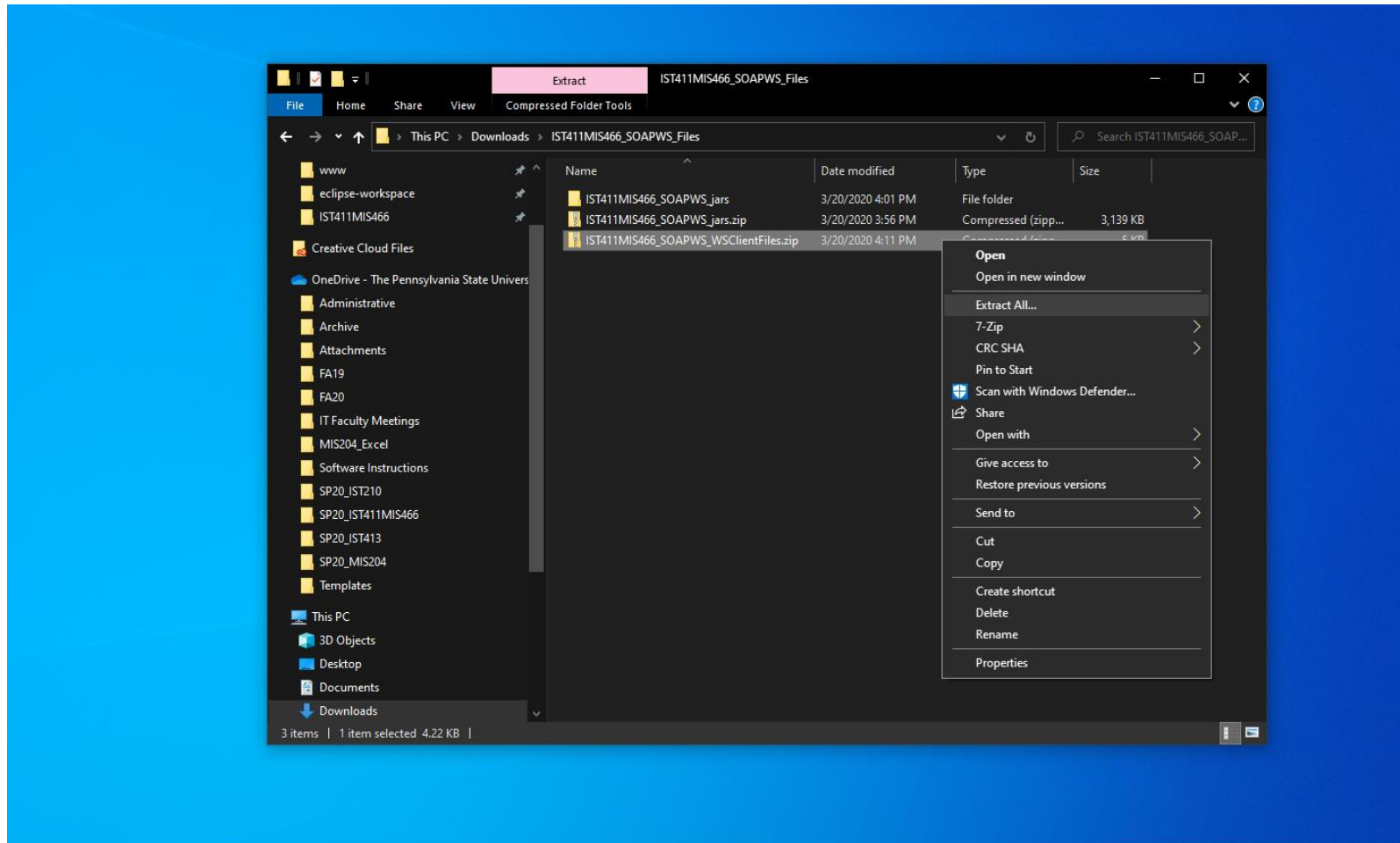
    @XmlElement(name=return, namespace=)
    public Ljava/lang/String; _return

}
Server started...http://localhost:9999/ReverseTextWS

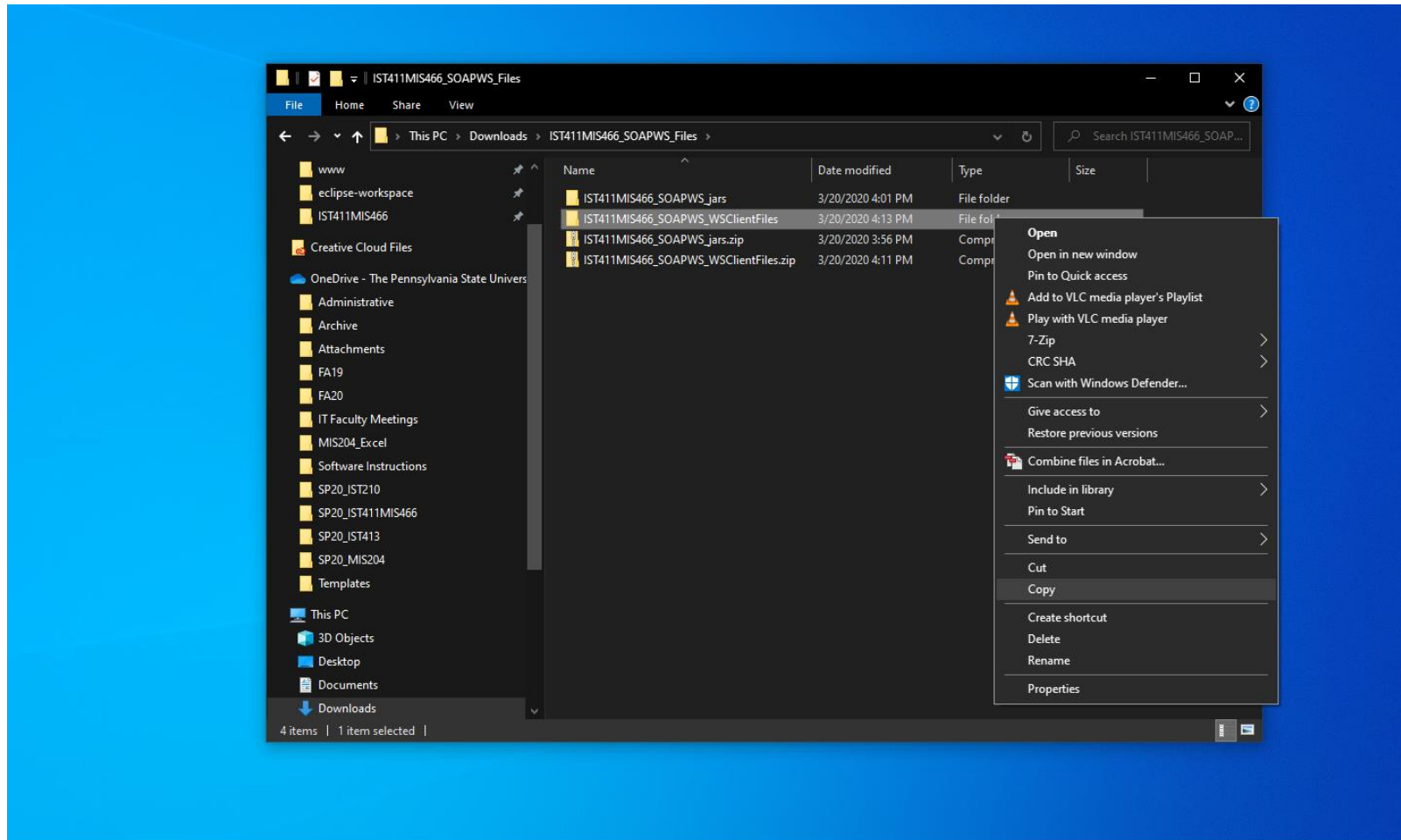
localhost:9999/ReverseTextWS?wsdl
This XML file does not appear to have any style information associated with it. The document tree is shown below.
<?xml version='1.0' encoding='UTF-8'>
  <!--
    Published by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.
  -->
  <!--
    Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is JAX-WS RI 2.2.9-b130926.1035 svn-revision#5f6196f2b90e9460065a4c2f4e30e065b245e51e.
  -->
  <definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/soap/" xmlns:tns="http://server.soap.ws/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://server.soap.ws/" name="ReverseTextWSService">
    <types>
      <xsd:schema>
        <xsd:import namespace="http://server.soap.ws/" schemaLocation="http://localhost:9999/ReverseTextWS?xsd=1"/>
      </xsd:schema>
    </types>
    <message name="reverseText">
      <part name="parameters" element="tns:reverseText"/>
    </message>
    <message name="reverseTextResponse">
      <part name="parameters" element="tns:reverseTextResponse"/>
    </message>
    <portType name="ReverseTextWS">
      <operation name="reverseText">
        <input wsam:Action="http://server.soap.ws/ReverseTextWS/reverseTextRequest" message="tns:reverseText"/>
        <output wsam:Action="http://server.soap.ws/ReverseTextWS/reverseTextResponse" message="tns:reverseTextResponse"/>
      </operation>
    </portType>
    <binding name="ReverseTextWSPortBinding" type="tns:ReverseTextWS">
      <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
      <operation name="reverseText">
        <soap:operation soapAction="">
          <input>
            <soap:body use="literal"/>
          </input>
          <output>
            <soap:body use="literal"/>
          </output>
        </operation>
      </binding>
    <service name="ReverseTextWSService">
      <port name="ReverseTextWSPort" binding="tns:ReverseTextWSPortBinding">
        <soap:address location="http://localhost:9999/ReverseTextWS"/>
      </port>
    </service>
  </definitions>
```

PART 4: CREATE THE WEB SERVICE CLIENT FILES AND THE WEB SERVICE CLIENT

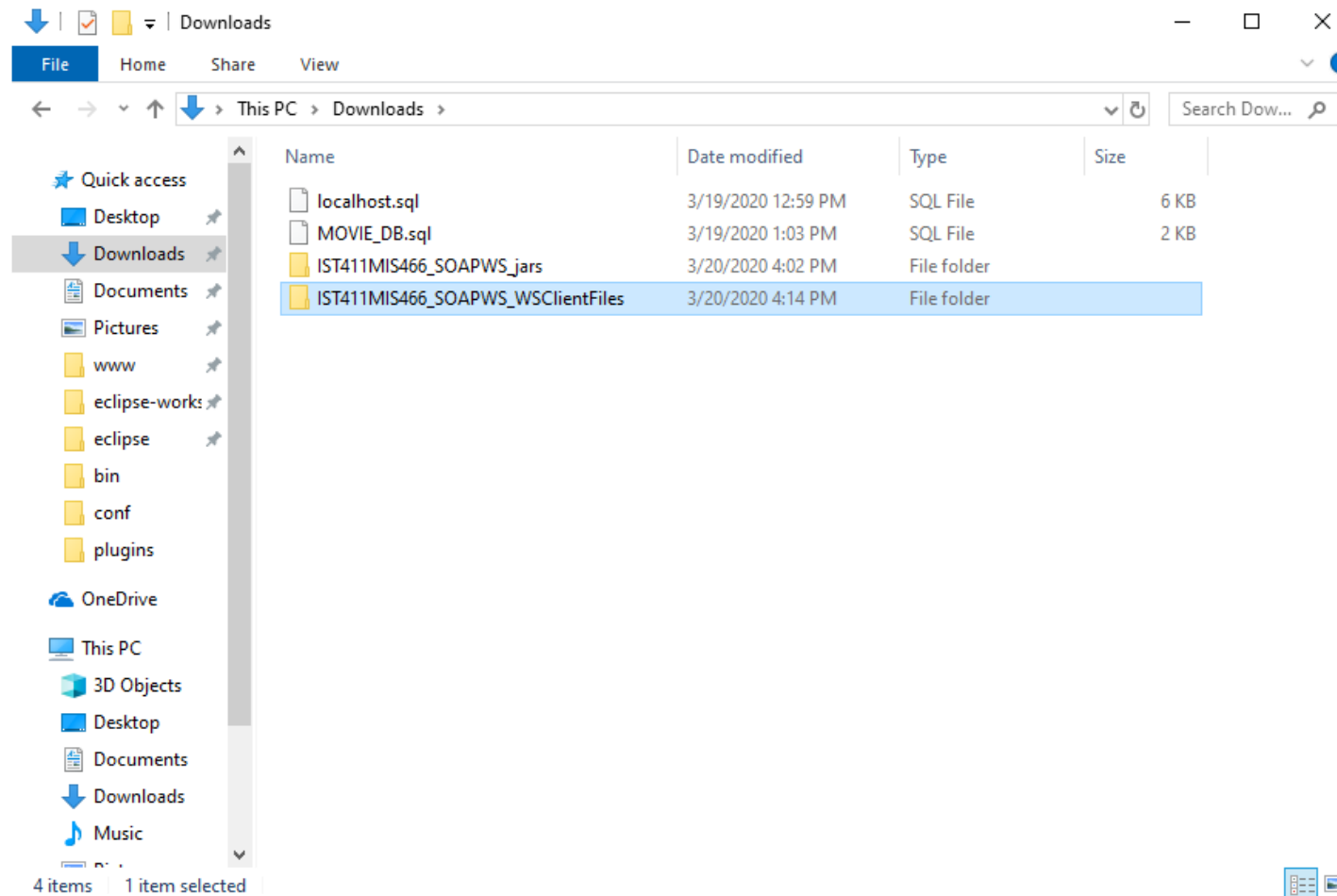
STEP 1: Download the IST411MIS466_SOAPWS_WSClientsFiles.zip folder from Canvas and extract its contents onto your computer.



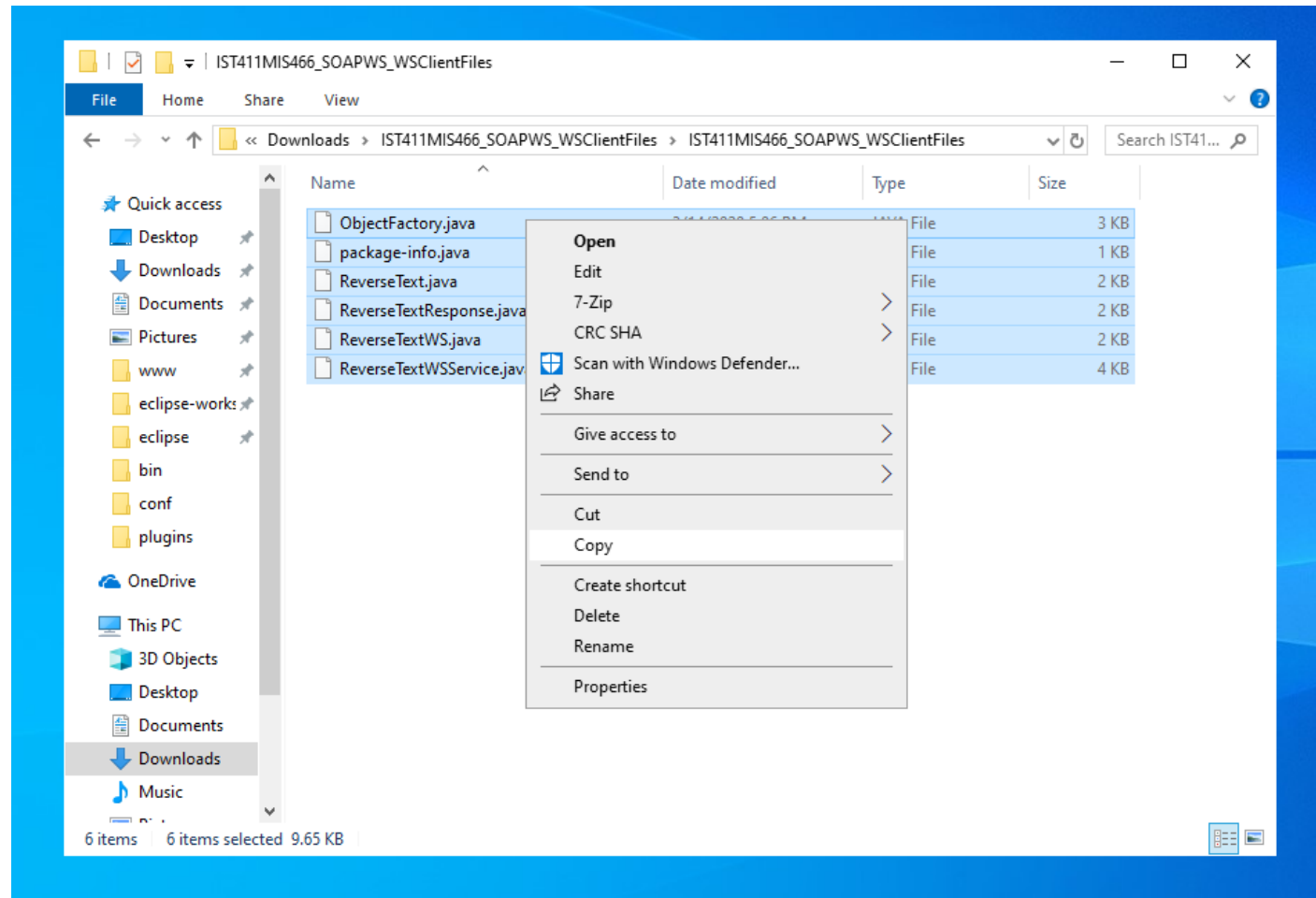
STEP 2: Copy the extracted folder.



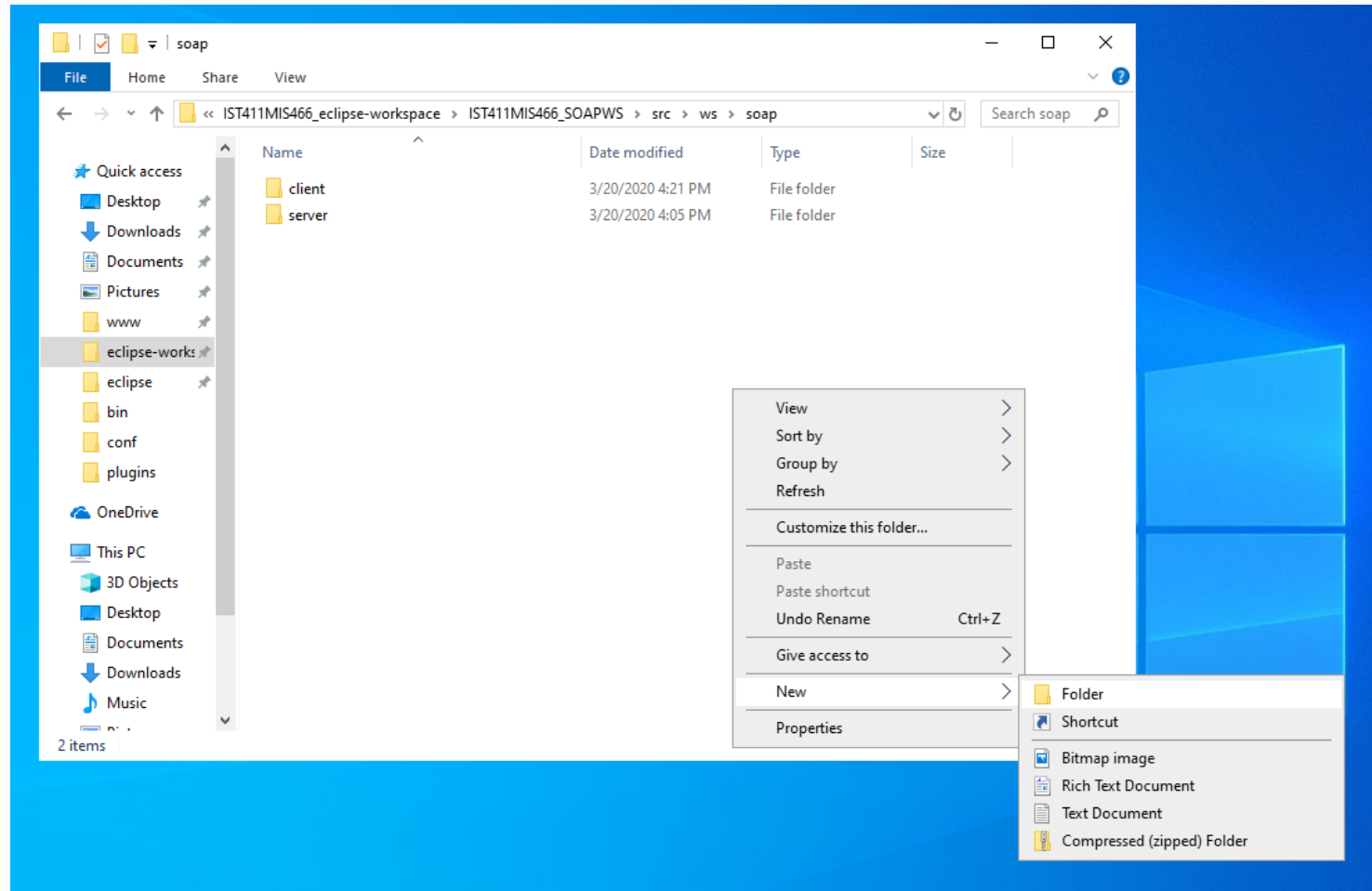
STEP 3: Back in your virtual machine, in a File Explorer window, open the **Downloads** directory. Paste the folder that you copied in the previous step.



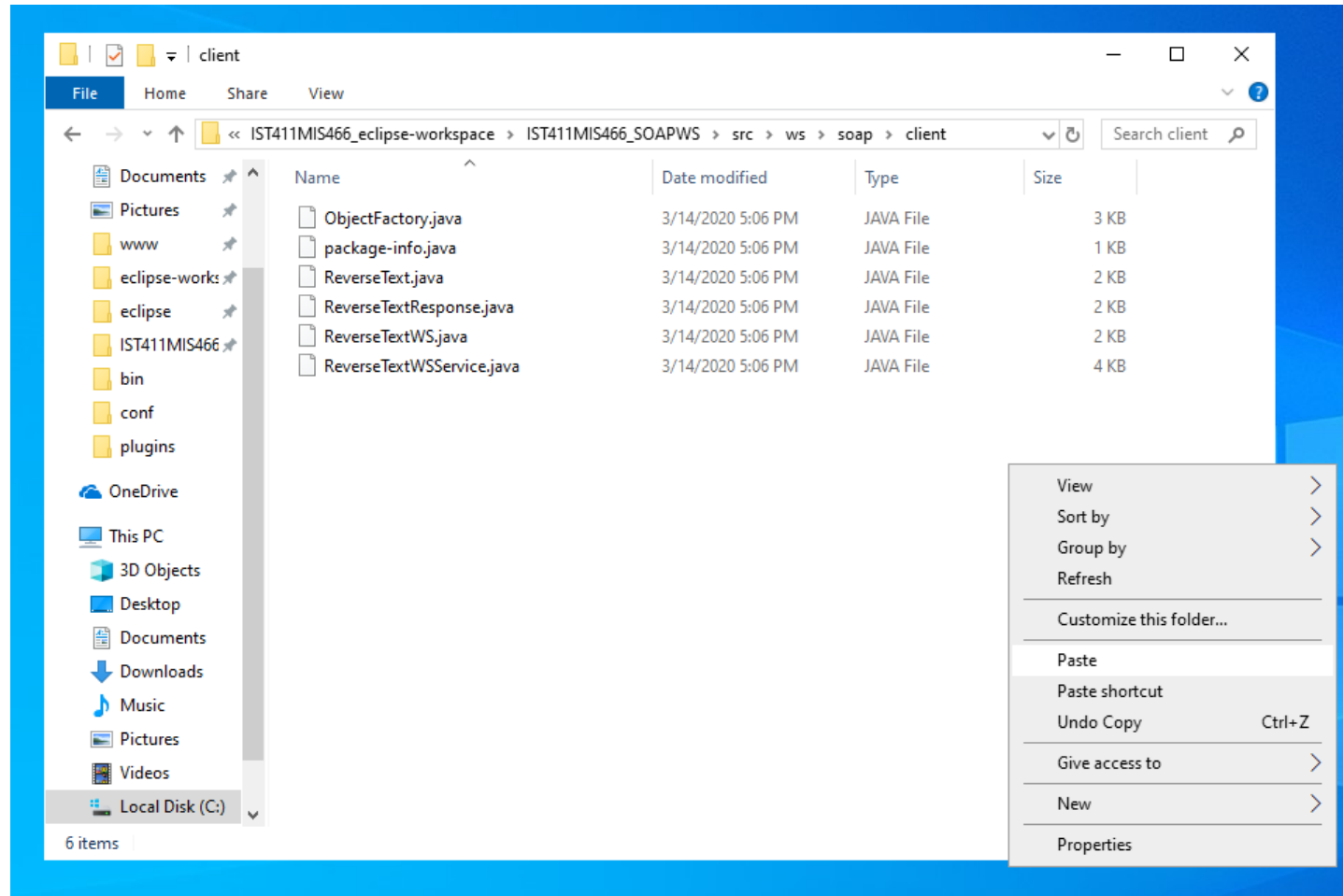
STEP 4: Open the ***Downloads\IST411MIS466_SOAPWS_WSClientFiles*** directory and copy the Java files. You should have a total of 6 files to copy (use the ***Shift*** key or the ***Ctrl*** key while selecting the .java files to select all of them).



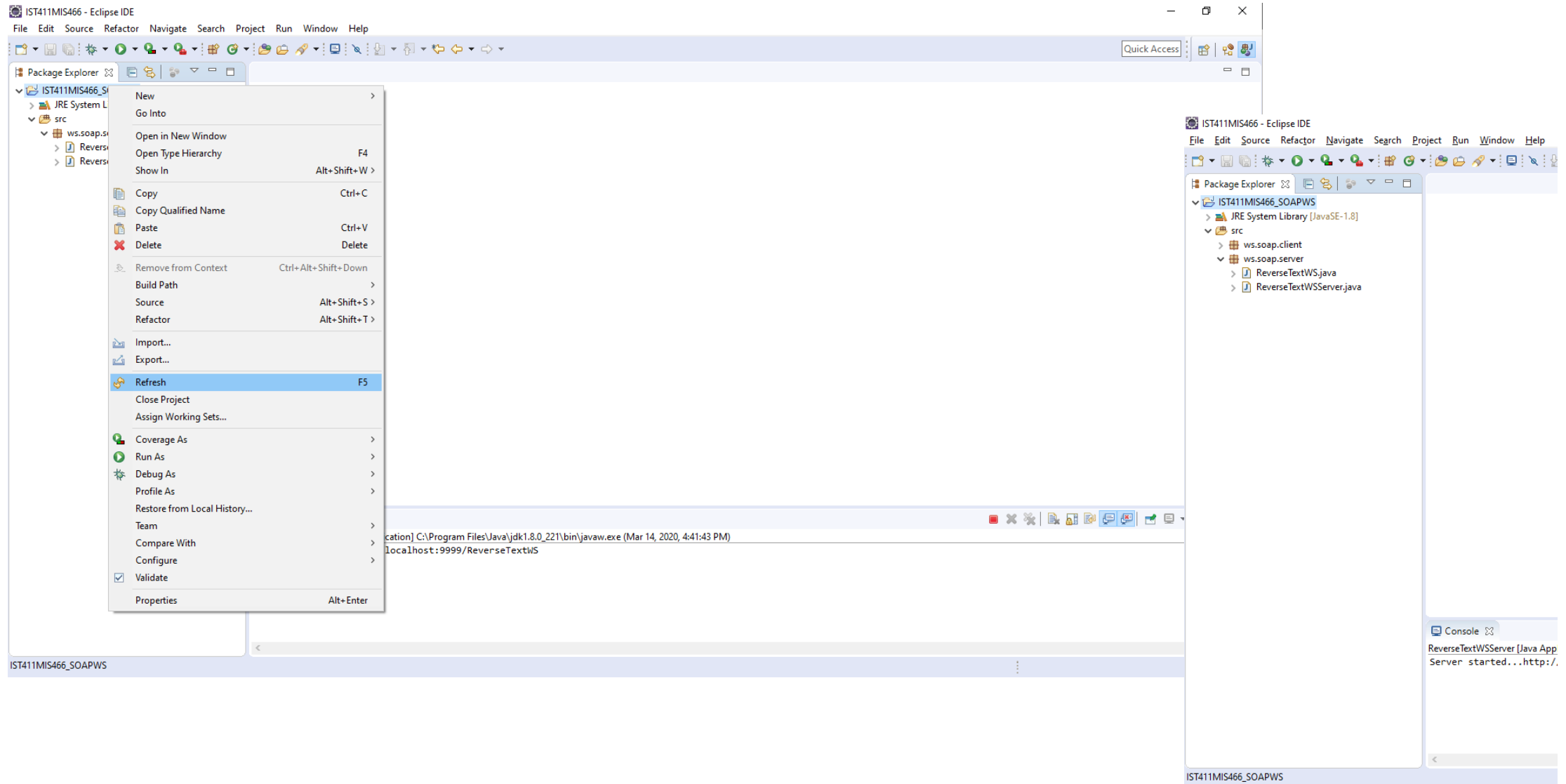
STEP 5: In a File Explorer window, navigate to your project (this should be your Eclipse workspace location, followed by your project name). Navigate into the *src* folder, then the *ws* folder, then the *soap* folder. In that directory, create a new folder named *client*.



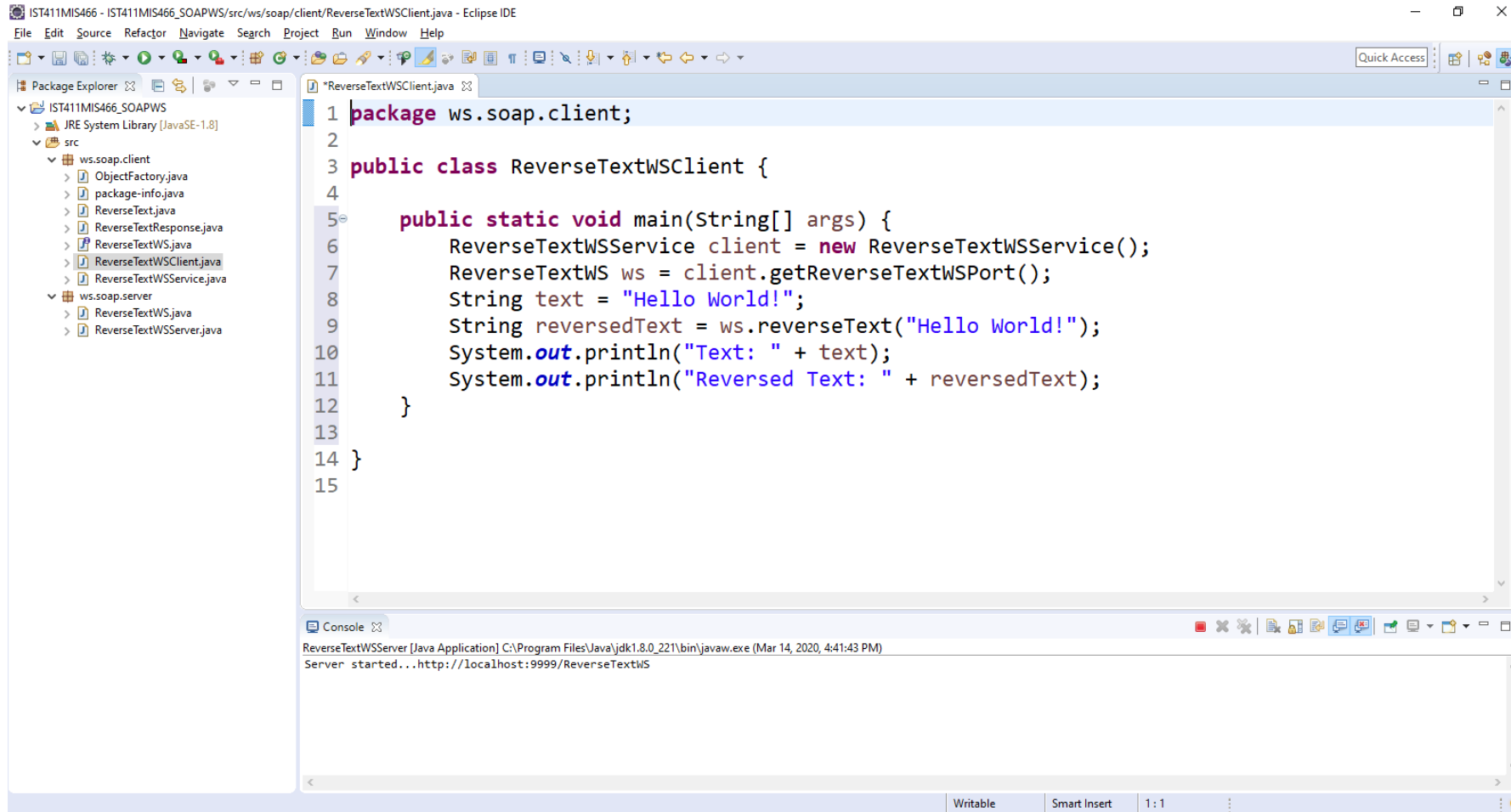
STEP 6: Navigate into the new client folder, and paste the Java files that you copied in STEP 4.



STEP 7: In Eclipse, right-click on the project and select **Refresh Project**. You should now see a **ws.soap.client** package with all of the files that were generated from the **wsimport** Java command in STEP 4B.



STEP 8: In the **ws.soap.client** package, create a new class called **ReverseTextWSClient**, which will be the web service client. Add the following code to the class. We create a client view of the web service as an instance of **ReverseTextWSService**, and use that to create the client web service interface as an instance of **ReverseTextWS** (the interface class in the client package that was generated by the **wsimport** Java command). This instance of the interface tells the client which methods the **ReverseTextWS** web service provides, which is why we use it to call the **reverseText()** method.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: IST411MIS466_SOAPWS, JRE System Library [JavaSE-1.8], src, ws.soap.client, and ws.soap.server. The ws.soap.client package contains several files, including ReverseTextWSClient.java, which is the active file. The main method in ReverseTextWSClient.java creates a ReverseTextWSService client, obtains a ReverseTextWS interface instance, and calls the reverseText() method with the string "Hello World!". The console at the bottom shows the output of the application, indicating that the server started successfully at http://localhost:9999/ReverseTextWS.

```
1 package ws.soap.client;
2
3 public class ReverseTextWSClient {
4
5     public static void main(String[] args) {
6         ReverseTextWSService client = new ReverseTextWSService();
7         ReverseTextWS ws = client.getReverseTextWSPort();
8         String text = "Hello World!";
9         String reversedText = ws.reverseText("Hello World!");
10        System.out.println("Text: " + text);
11        System.out.println("Reversed Text: " + reversedText);
12    }
13 }
14 }
15
```

ReverseTextWSServer [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (Mar 14, 2020, 4:41:43 PM)
Server started...http://localhost:9999/ReverseTextWS

PART 5: RUN THE WEB SERVICE CLIENT

STEP 1: Run the *ReverseTextWSClient* class. In the console you may see warning/info messages about deprecated libraries/classes. The last lines of the console should display *Text: Hello World!* and *Reversed Text: !dlroW olleH*. These are the original text value *Hello World!* and the reversed text value *!dlroW olleH* which was returned from the *ReverseTextWS* web service that is being hosted on the *ReverseTextWSServer* at *localhost:9999/ReverseTextWS*.



The screenshot shows an IDE console window with the following content:

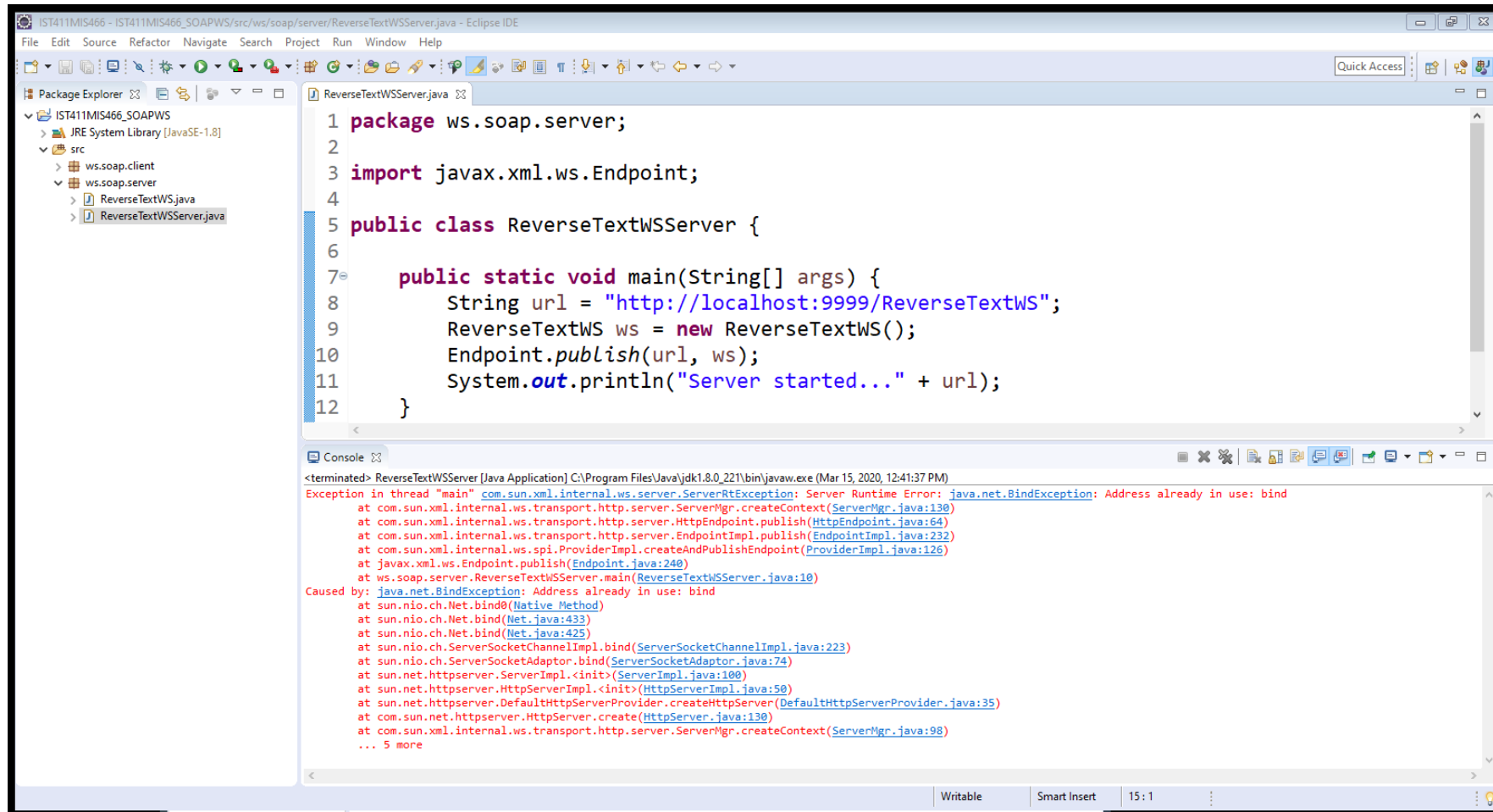
```
<terminated> ReverseTextWSClient [Java Application] C:\Program Files\Java\jdk-14\bin\javaw.exe (Mar 20, 2020, 4:26:12 PM – 4:26:13 PM)
WARNING: Illegal reflective access by com.sun.xml.bind.v2.runtime.reflect.opt.Injector (file:/C:/Users/klk5059/Downloads/IST411MIS466_SOAPWS_jars/IST411MIS466_SOAPWS_ji
WARNING: Please consider reporting this to the maintainers of com.sun.xml.bind.v2.runtime.reflect.opt.Injector
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Mar 20, 2020 4:26:13 PM javax.xml.soap.FactoryFinder find
WARNING: Using deprecated META-INF/services mechanism with non-standard property: javax.xml.soap.MetaFactory. Property javax.xml.soap.SAAJMetaFactory should be used in:
Mar 20, 2020 4:26:13 PM javax.xml.soap.FactoryFinder find
WARNING: Using deprecated META-INF/services mechanism with non-standard property: javax.xml.soap.MetaFactory. Property javax.xml.soap.SAAJMetaFactory should be used in:
Mar 20, 2020 4:26:13 PM javax.xml.soap.FactoryFinder find
WARNING: Using deprecated META-INF/services mechanism with non-standard property: javax.xml.soap.MetaFactory. Property javax.xml.soap.SAAJMetaFactory should be used in:
Mar 20, 2020 4:26:13 PM javax.xml.soap.FactoryFinder find
WARNING: Using deprecated META-INF/services mechanism with non-standard property: javax.xml.soap.MetaFactory. Property javax.xml.soap.SAAJMetaFactory should be used in:
Original Text: Hello World!
Reversed Text: !dlroW olleH
```

TROUBLESHOOTING



TROUBLESHOOTING THE SERVER:

When running the server component of this exercise, if you get a runtime error like the one shown below ***Address already in use: bind***... you likely didn't stop a previous instance of the server from running. You can check the console in Eclipse to see if you can find the running instance and use the ***stop*** button to terminate the program. However, if you cannot find the programming in Eclipse you can follow these step to terminate the process and free the port.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure: IST411MIS466_SOAPWS, JRE System Library [JavaSE-1.8], src, ws.soap.client, ws.soap.server, ReverseTextWS.java, and ReverseTextWSServer.java. The main editor shows the code for ReverseTextWSServer.java, which defines a package, imports javax.xml.ws.Endpoint, and contains a main method that publishes an endpoint at http://localhost:9999/ReverseTextWS. The Console window at the bottom shows a terminated exception: java.net.BindException: Address already in use: bind. The stack trace indicates the error occurred during the creation and publishing of the endpoint.

```
1 package ws.soap.server;
2
3 import javax.xml.ws.Endpoint;
4
5 public class ReverseTextWSServer {
6
7     public static void main(String[] args) {
8         String url = "http://localhost:9999/ReverseTextWS";
9         ReverseTextWS ws = new ReverseTextWS();
10        Endpoint.publish(url, ws);
11        System.out.println("Server started..." + url);
12    }
13 }
```

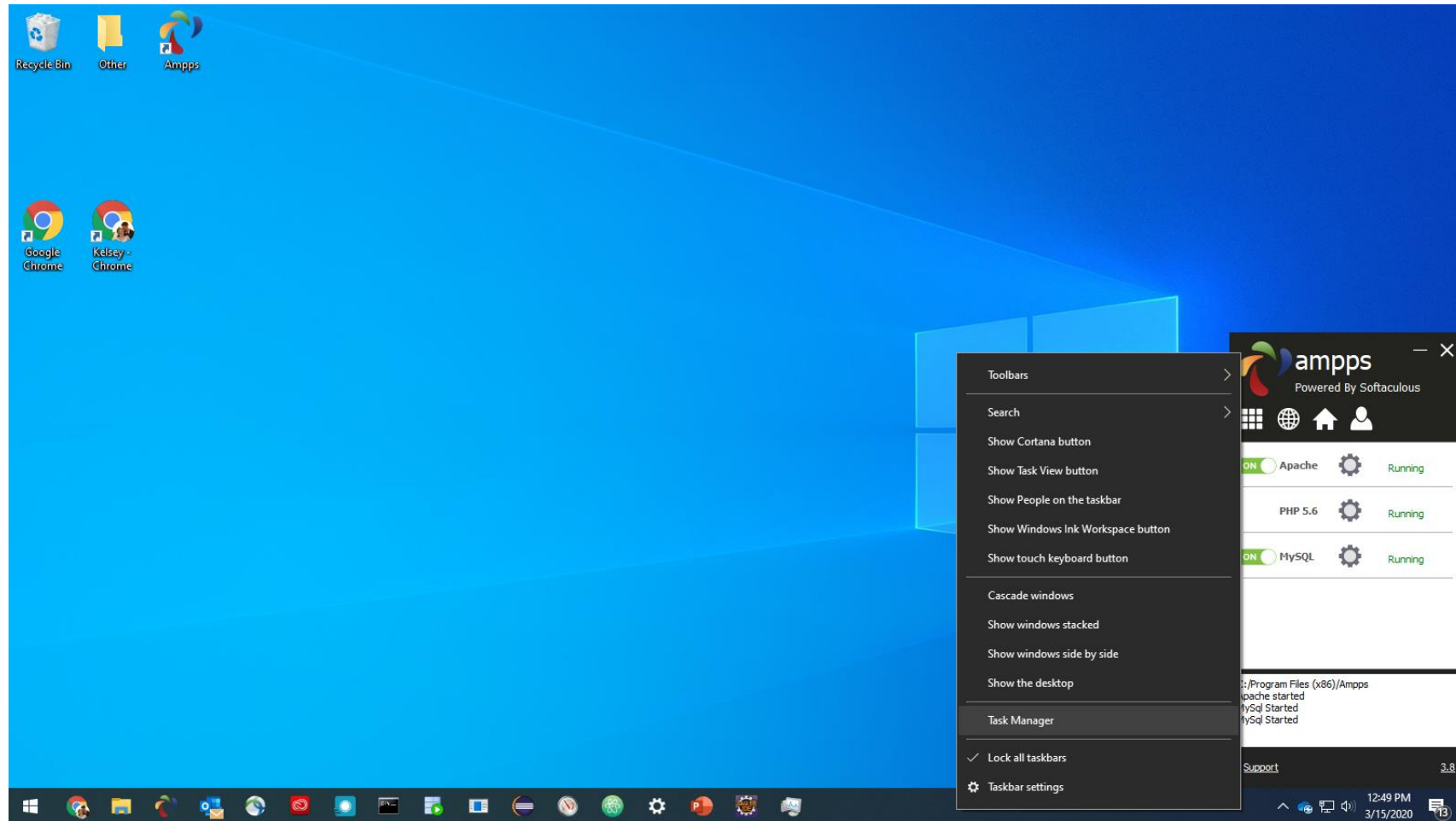
Console Output:

```
<terminated> ReverseTextWSServer [Java Application] C:\Program Files\Java\jdk1.8.0_221\bin\javaw.exe (Mar 15, 2020, 12:41:37 PM)
Exception in thread "main" com.sun.xml.internal.ws.server.ServerRtException: Server Runtime Error: java.net.BindException: Address already in use: bind
    at com.sun.xml.internal.ws.transport.http.server.ServerMgr.createContext(ServerMgr.java:130)
    at com.sun.xml.internal.ws.transport.http.server.HttpEndpoint.publish(HttpEndpoint.java:64)
    at com.sun.xml.internal.ws.transport.http.server.EndpointImpl.publish(EndpointImpl.java:232)
    at com.sun.xml.internal.ws.spi.ProviderImpl.createAndPublishEndpoint(ProviderImpl.java:126)
    at javax.xml.ws.Endpoint.publish(Endpoint.java:240)
    at ws.soap.server.ReverseTextWSServer.main(ReverseTextWSServer.java:10)
Caused by: java.net.BindException: Address already in use: bind
    at sun.nio.ch.Net.bind0(Native Method)
    at sun.nio.ch.Net.bind(Net.java:433)
    at sun.nio.ch.Net.bind(Net.java:425)
    at sun.nio.ch.ServerSocketChannelImpl.bind(ServerSocketChannelImpl.java:223)
    at sun.nio.ch.ServerSocketAdaptor.bind(ServerSocketAdaptor.java:74)
    at sun.net.httpserver.ServerImpl.<init>(ServerImpl.java:100)
    at sun.net.httpserver.HttpServerImpl.<init>(HttpServerImpl.java:50)
    at sun.net.httpserver.DefaultHttpServerProvider.createHttpServer(DefaultHttpServerProvider.java:35)
    at com.sun.net.httpserver.HttpServer.create(HttpServer.java:130)
    at com.sun.xml.internal.ws.transport.http.server.ServerMgr.createContext(ServerMgr.java:98)
    ... 5 more
```



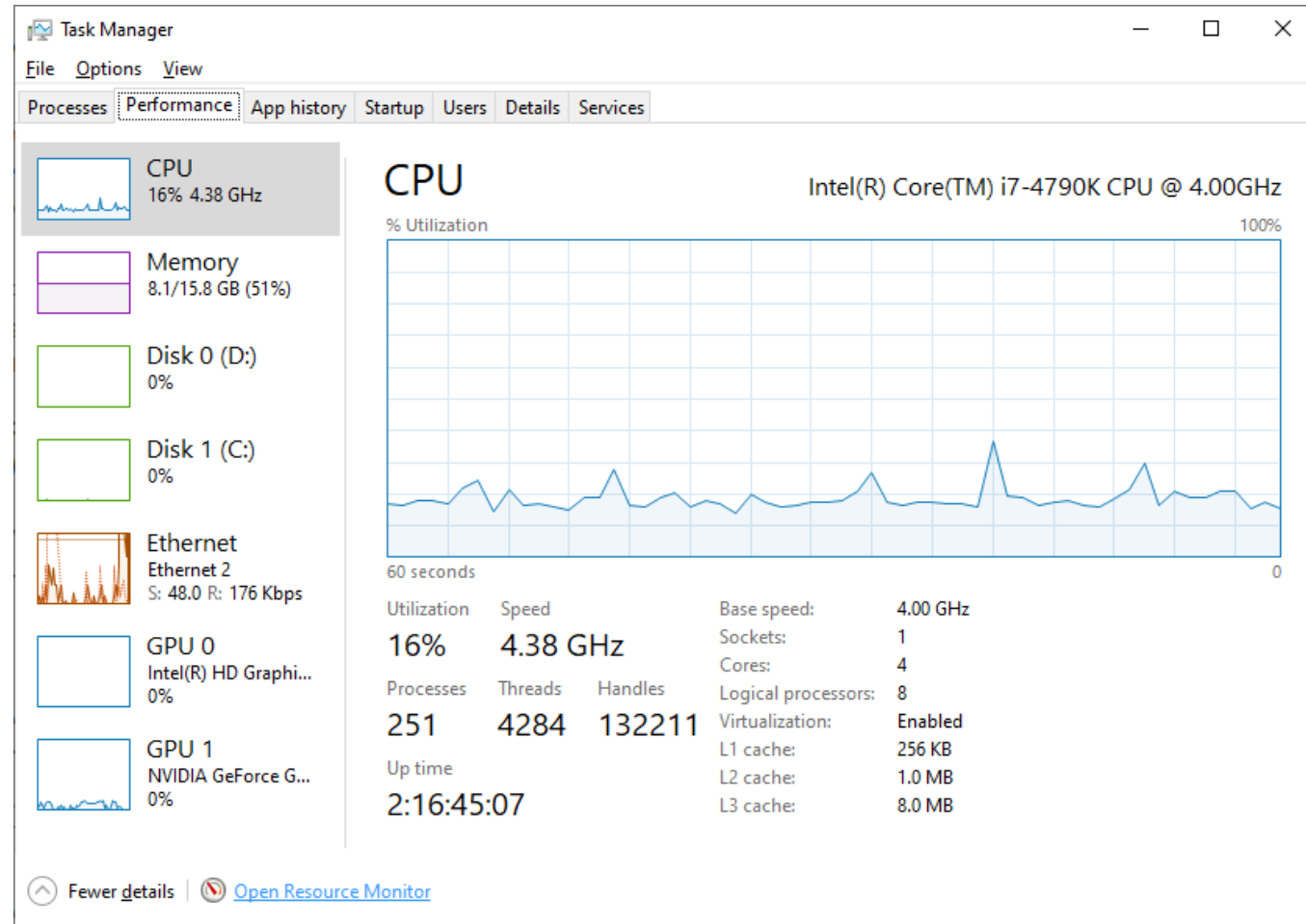
TROUBLESHOOTING THE SERVER

STEP 1: To terminate the server, you will need to use the Task Manager and the Resource Monitor. First, open your task manager. One way to do this is to right-click in the task bar and select *Task Manager*.



TROUBLESHOOTING THE SERVER

STEP 2: In the Task Manager, select the *Performance* tab. At the bottom of the window, click on *Open Resource Monitor*.



TROUBLESHOOTING THE SERVER

STEP 3: In the Resource Monitor, in the **Listening Ports** section, find the port for the server (in this example it's 9999). You can sort the ports in numeric order by click the **Port** column heading. When you find the port, make a note of the **PID**. In this case, it's 5836.

The screenshot shows the Windows Resource Monitor window with the 'Network' tab selected. The 'Processes with Network Activity' section lists several processes, including 'javaw.exe' with PID 5836. The 'Network Activity' section shows 23 Kbps Network I/O and 0% Network Utilization. The 'TCP Connections' section is empty. The 'Listening Ports' section shows a list of ports, with 'javaw.exe' listening on port 9999 (TCP) with Firewall Status 'Allowed, not r...'. The 'Network' section on the right shows a network diagram with Ethernet 3 and Ethernet 2.

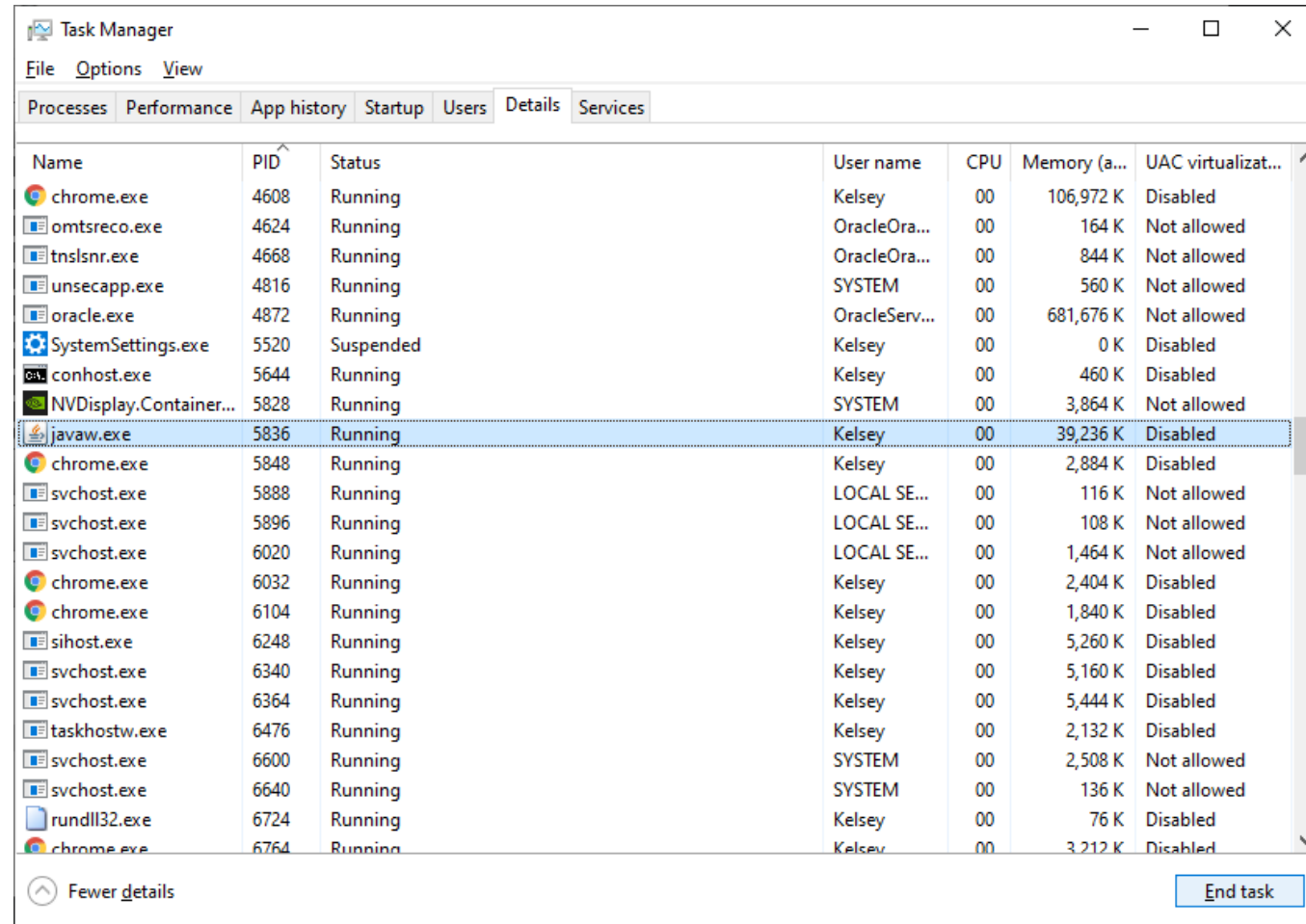
Image	PID	Send (B/sec)	Receive (B/sec)	Total (B/sec)
Image				
POWERPNT.EXE	11040	2,417	1,881	4,298
OneDrive.exe	4428	1,131	549	1,680
svchost.exe (LocalServiceAn...)	3856	0	1,178	1,178
chrome.exe	15752	90	75	165
tnslsnr.exe	4668	42	70	112
oracle.exe	4872	70	42	112
svchost.exe (NetworkService...)	1684	48	55	103
svchost.exe (netsvc -p)	3912	15	25	39
System	4	34	0	34
OUTLOOK.EXE	11710	0	11	11

Image	PID	Address	Port	Protocol	Firewall Status
CricutTaskbarApplication.exe	11520	IPv4 unspecified	7041	TCP	Allowed, not r...
svchost.exe (NetworkService -p)	16976	IPv6 unspecified	7680	TCP	Not allowed, ...
svchost.exe (NetworkService -p)	16976	IPv4 unspecified	7680	TCP	Not allowed, ...
javaw.exe	5836	IPv4 loopback	9999	TCP	Allowed, not r...
Adobe Desktop Service.exe	12964	IPv4 loopback	15292	TCP	Allowed, not r...
Adobe Desktop Service.exe	12964	IPv4 loopback	15393	TCP	Allowed, not r...
node.exe	692	IPv4 loopback	45623	TCP	Allowed, not r...
lsass.exe	752	IPv6 unspecified	49664	TCP	Not allowed, ...
lsass.exe	752	IPv4 unspecified	49664	TCP	Not allowed, ...
minitask.exe	673	IPv6 unspecified	49665	TCP	Not allowed, ...



TROUBLESHOOTING THE SERVER

STEP 4: Return to the Task Manager, and select the **Details** tab. Locate the process by finding the **PID** that you made a note of in the previous step. Again, you can sort the processes numerically by clicking on the **PID** column header. When you locate the process, click on it (the row should be highlighted) and then click the **End task** button.



The screenshot shows the Windows Task Manager window with the 'Details' tab selected. The table below lists the running processes, with 'javaw.exe' (PID 5836) highlighted. The 'End task' button is located at the bottom right of the window.

Name	PID	Status	User name	CPU	Memory (a...)	UAC virtualizat...
chrome.exe	4608	Running	Kelsey	00	106,972 K	Disabled
omtsreco.exe	4624	Running	OracleOra...	00	164 K	Not allowed
tnslsnr.exe	4668	Running	OracleOra...	00	844 K	Not allowed
unsecapp.exe	4816	Running	SYSTEM	00	560 K	Not allowed
oracle.exe	4872	Running	OracleServ...	00	681,676 K	Not allowed
SystemSettings.exe	5520	Suspended	Kelsey	00	0 K	Disabled
conhost.exe	5644	Running	Kelsey	00	460 K	Disabled
NVDisplay.Container...	5828	Running	SYSTEM	00	3,864 K	Not allowed
javaw.exe	5836	Running	Kelsey	00	39,236 K	Disabled
chrome.exe	5848	Running	Kelsey	00	2,884 K	Disabled
svchost.exe	5888	Running	LOCAL SE...	00	116 K	Not allowed
svchost.exe	5896	Running	LOCAL SE...	00	108 K	Not allowed
svchost.exe	6020	Running	LOCAL SE...	00	1,464 K	Not allowed
chrome.exe	6032	Running	Kelsey	00	2,404 K	Disabled
chrome.exe	6104	Running	Kelsey	00	1,840 K	Disabled
sihost.exe	6248	Running	Kelsey	00	5,260 K	Disabled
svchost.exe	6340	Running	Kelsey	00	5,160 K	Disabled
svchost.exe	6364	Running	Kelsey	00	5,444 K	Disabled
taskhostw.exe	6476	Running	Kelsey	00	2,132 K	Disabled
svchost.exe	6600	Running	SYSTEM	00	2,508 K	Not allowed
svchost.exe	6640	Running	SYSTEM	00	136 K	Not allowed
rundll32.exe	6724	Running	Kelsey	00	76 K	Disabled
chrome.exe	6764	Running	Kelsey	00	3,212 K	Disabled



TROUBLESHOOTING THE SERVER

STEP 5: You can now try to run the server again in Eclipse. You should see the *Server started...* message in the console.

