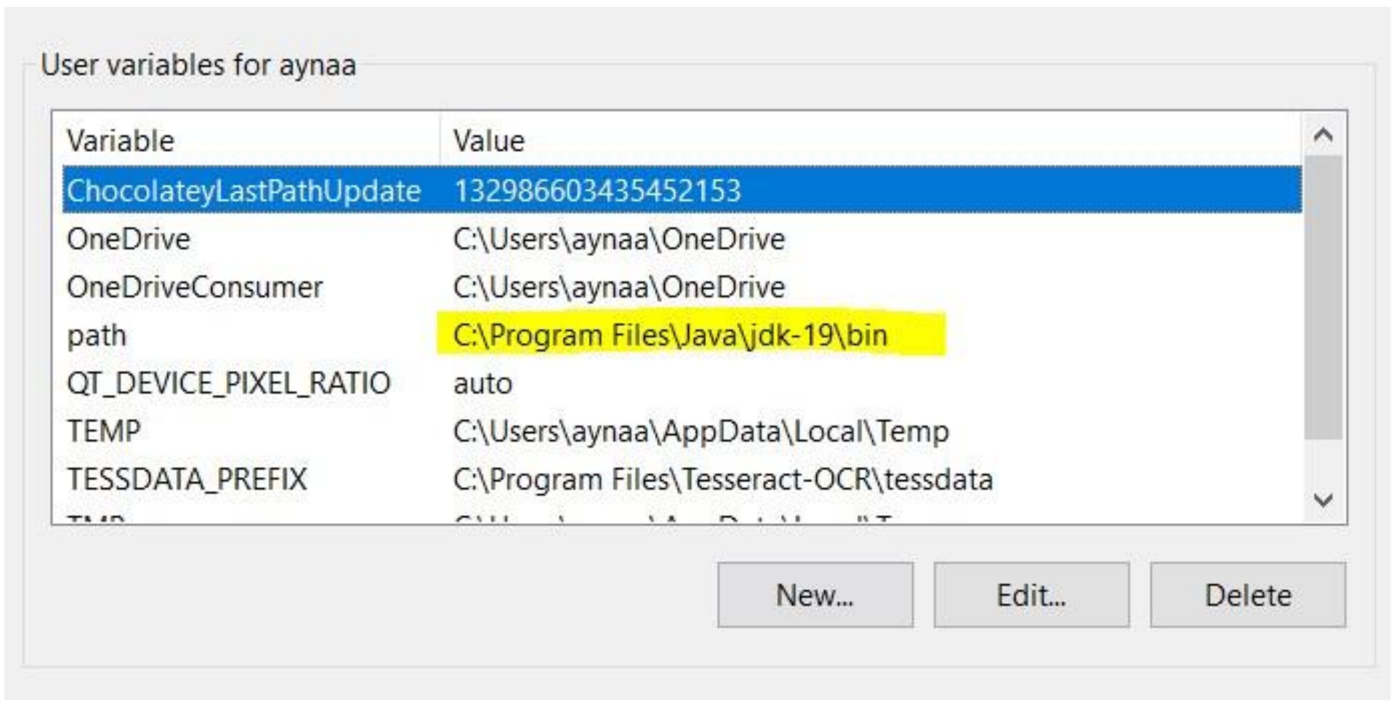
SA Practical

Things to Download

- Java jdk :https://www.oracle.com/in/java/technologies/downloads/#jdk19-windows(download x64 MSI Installer version) Set path environment variable for java
  Also install Eclipse IDE for Enterprise Java and Web Developers in Eclipse Ide options.



User variables for aynaa

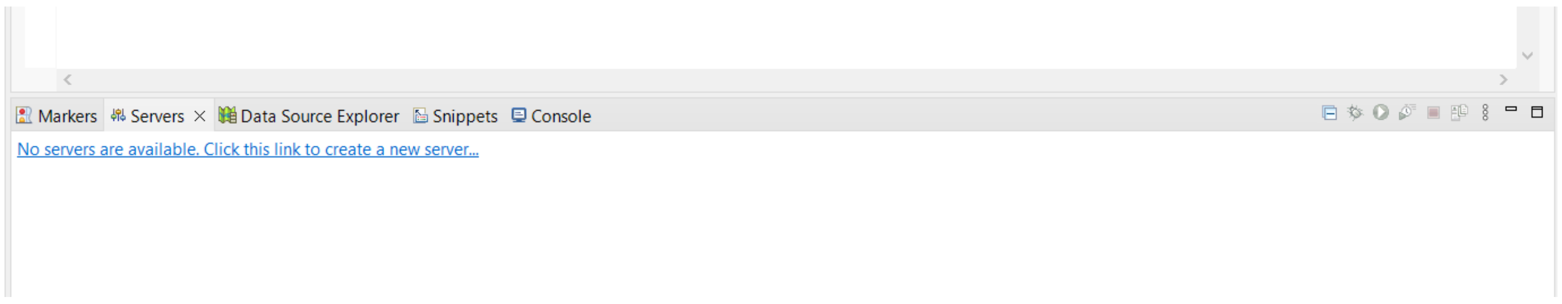
Variable	Value
ChocolateyLastPathUpdate	132986603435452153
OneDrive	C:\Users\aynaa\OneDrive
OneDriveConsumer	C:\Users\aynaa\OneDrive
path	C:\Program Files\Java\jdk-19\bin
QT_DEVICE_PIXEL_RATIO	auto
TEMP	C:\Users\aynaa\AppData\Local\Temp
TESSDATA_PREFIX	C:\Program Files\Tesseract-OCR\tessdata

- Java Eclipse IDE :https://www.eclipse.org/downloads/
- Tomcat:https://tomcat.apache.org/download-80.cgi(Go to core and under it download zip version 8.5)

ArchStudio URL: http://www.isr.uci.edu/projects/archstudio-4/updatesite-4.2/

[redacted] Exp 3 code

1) Create a dynamic project (Don't Forget to change perspective to Java EE)
2) Name: (Anyname)
   Dynamic web module version: 2.5
   Check Add Project to an EAR
3) Finish
4) Go to servers panel
5) click on link "No servers are available..."



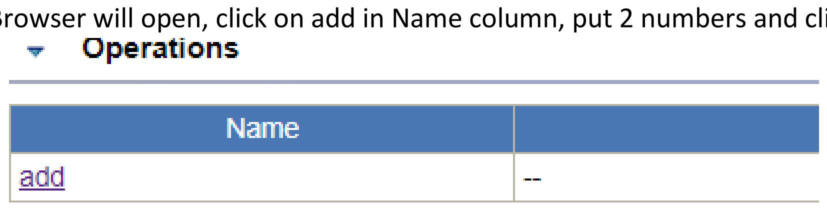
No servers are available. Click this link to create a new server...

6) Now, select Apache-->Tomcat v8.5
   Next, Tomcat installation directory --> Browse : Give the path of downloaded apache tomcat 8.5
   Next, select your project and click on add, Finish
7) Right click on project --> new class
   Give Name (e.g addClass), Finish
8) Create function:

```
public class addClass {
        public int add(int a,int b) {
                return a+b;
        }
}
```

9) Right click on Tomcat in servers panel, and click on start

10) Right click on file (addClass), New-->Other-->Web Services-->Web Service.
    Next, in 2nd image bring pointer at top (Test Client)
    Next, Next, Error(OK), Next, Warning (Yes to all), Click on Launch
11) Browser will open, click on add in Name column, put 2 numbers and click on go.



Name	
add	--

RMI Exp 4 code

IHello.java

```
import java.rmi.*;
public interface IHello extends Remote{
public String message() throws RemoteException;
}
```

HelloImpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class HelloImpl extends UnicastRemoteObject
implements IHello{
	public HelloImpl() throws RemoteException {
	//There is no action need in this moment.
	}
	public String message() throws RemoteException {
		return ("Hello");
	}
}
```

HelloServer.java

```
import java.rmi.*;
public class HelloServer {
private static final String host = "localhost";
public static void main(String[] args) throws Exception {
	//** Step 1
	//** Declare a reference for the object that will be implemented
	HelloImpl temp = new HelloImpl();
	//** Step 2
	//** Declare a string variable for holding the URL of the object&#39;s name
	String rmiObjectName = "rmi://" + host + "/Hello";
	//Step 3
	//Binding the object reference to the object name.
	Naming.rebind(rmiObjectName, temp);
	//Step 4
	//Tell to the user that the process is completed.
	System.out.println("Binding complete...\n");
	}
}
```

HelloClient.java

```
import java.rmi.ConnectException;
import java.rmi.Naming;
public class HelloClient
{
private static final String host = "localhost";
public static void main(String[] args)
{
	try
	{
		//We obtain a reference to the object from the registry and next,
		//it will be typecasted into the most appropiate type.
		IHello greeting_message = (IHello) Naming.lookup("rmi://"+ host + "/Hello");
		//Next, we will use the above reference to invoke the remote
		//object method.
		System.out.println("Message received:"+greeting_message.message());
	}
	catch (ConnectException conEx)
```

```
		{
			System.out.println("Unable to connect to server!");
			System.exit(1);
		}
		catch (Exception ex)
		{
			ex.printStackTrace();

			System.exit(1);
		}
}
}
```

Compile Each file, Then run “start rmiregistry”, then run “java HelloClient.java”

RMI Exp 5 code

Server.java

```
public class Server implements interfaceCalculator{
	public int add(int a,int b){
		return a+b;
	}
	public int sub(int a,int b){
		return a-b;
	}
}
```

InterfaceCalculator.java

```
public interface interfaceCalculator{
	public int add(int a,int b);
	public int sub(int a,int b);
}
```

Client.java

```
public class Client {
	public static void main(String [] args)
	{
		interfaceCalculator i=new Server();
		System.out.println(i.add(12,13));
		System.out.println(i.sub(12,12));
	}
}
```

Just Run Client.java

Wrapper Exp 6

DSender.java

```
import java.net.*;
import java.util.*;
public class Dsender{
  public static void main(String[] args) throws Exception {
   Scanner scn= new Scanner(System.in);
   System.out.println("Enter your message : ");
   String str= scn.nextLine();
   DatagramSocket ds = new DatagramSocket();
   InetAddress ip = InetAddress.getByName("127.0.0.1");
   DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(), ip, 3000);
   ds.send(dp);
   ds.close();
   System.out.println("Message has been sent to Receiver Class Please Check: "+ str);
   }
}
```

DReceiver.java

```
import java.net.*;
public class DReceiver{
  public static void main(String[] args) throws Exception {
   System.out.println("Waiting for Sender to send the Message");
   DatagramSocket ds = new DatagramSocket(3000);
   byte[] buf = new byte[1024];
   DatagramPacket dp = new DatagramPacket(buf, 1024);
   ds.receive(dp);
   String str = new String(dp.getData(), 0, dp.getLength());
   System.out.println(str);
   ds.close();
   System.out.println("Message received successfully");
  }
}
```