TRIBHUVAN UNIVERSITY
Faculty of Humanities & Social Sciences

A Lab Report
Of
Network programming

Faculty of Humanities & Social Science
Milton International College

In partial fulfilment of the requirements for the
Bachelors in Computer Application

Submitted To
Faculty of Humanities & Social Science
Prativa Oli

Submitted By
Pema Hojer Lama
Roll no: 7
BCA 6th Semester

Lab Report Questions

S.No.	Experiment Questions.	Remarks
1	WAP to create Client socket and Server socket to establish a connection.	
2	WAP to find which protocols does a virtual machine support?	
3	Program to obtain the information about the a)Host (b) Port (c) Protocol	
4	WAP for SpamCheck.	
5	Creates URL objects for for any two websites and verify if they're the same using the equals() method.	
6	Program to download a web page with a URL Connection.	
7	Read 3 URLs from the command line and uses these six methods to print their content type, content length, content encoding, date of last modification, expiration date, and current date.	
8	WAP to Set ifModifiedSince to 24 hours prior to now.	
9	Program to write to the server using socket and also using the telnet and comparing the response from the server.	
10	WAP to create a multithreaded daytime socket server and connect to at least 2 client at the same time.	
11	In prior to lab program no demonstrates by adding logging to the daytime server.	
12	WAP for a UDP Echo server and UDP client.	
13	RMI Client & Server Side	

1. WAP to create Client socket and Server socket to establish a connection.

Objective

To implement basic socket programming by creating a server socket and a client socket to establish a connection and exchange a simple message.

MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args) {
try {
ServerSocket ss = new ServerSocket(6666); // Create server socket
System.out.println("Server is waiting for client...");
Socket s = ss.accept(); // Accept client connection
System.out.println("Client connected.");

DataInputStream dis = new DataInputStream(s.getInputStream());
String str = dis.readUTF(); // Read UTF string from client
System.out.println("Message = " + str);
ss.close(); // Close server socket
} catch (Exception e) {
System.out.println(e);
}
}
}
```

MyClient.java

```
import java.io.*;
import java.net.*;

public class MyClient {
public static void main(String[] args) {
try {
Socket s = new Socket("localhost", 6666); // Connect to server on port 6666
DataOutputStream dout = new DataOutputStream(s.getOutputStream());

dout.writeUTF("Hello Server"); // Send message
dout.flush(); // Ensure data is sent
dout.close(); // Close output stream
s.close(); // Close socket
} catch (Exception e) {
```

```
System.out.println(e);
}
}
}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains
Server is waiting for client...
Client connected.
Message = Hello Server

Process finished with exit code 0
```

<u>Conclusion</u>:

Successfully established a connection between a client and a server using sockets. The client sent a message, and the server received it, demonstrating basic socket communication.

2. WAP to find which protocols does a virtual machine support?

Objective:

To determine and display the network protocols supported by the Java Virtual Machine using the URL class.

<u>ProtocolSupportChecker.java</u>

```
import java.net.MalformedURLException;
import java.net.URL;
public class ProtocolSupportChecker {
   public static void main(String[] args) {
      try {
// Example 1: HTTP protocol
         URL u1 = new URL("http://www.pemahojer.com.np/");
         System.out.println("Protocol of u1: " + u1.getProtocol());

// Example 2: HTTPS protocol
         URL u2 = new URL("https://www.craftedcreativity.com/");
         System.out.println("Protocol of u2: " + u2.getProtocol());

      } catch (MalformedURLException ex) {
         System.out.println("Malformed URL: " + ex);
      }
   }
}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intell
Protocol of u1: http
Protocol of u2: https

Process finished with exit code 0
```

Conclusion

Identified and printed supported protocols such as HTTP and HTTPS, demonstrating how Java can handle different types of URLs.

3. Program to obtain the information about the

a)Host (b) Port (c) Protocol

Objective

To retrieve and display the protocol, host, and port information from a given URL using the URL class.

HostPortProtocol.java

```
import java.net.MalformedURLException;
import java.net.URL;
public class HostPortProtocol{
   public static void main(String[] args) {
      try {
         URL u1 = new URL("http://www.pemahojer.com.np/");
         System.out.println("URL 1:");
         System.out.println("Protocol: " + u1.getProtocol());
         System.out.println("Host: " + u1.getHost());
         System.out.println("Port: " + u1.getPort()); // returns -1 if no port    is specified
         URL u2 = new URL("http://www.pemahojer.com.np/");
         System.out.println("\nURL 2:");
         System.out.println("Protocol: " + u2.getProtocol());
         System.out.println("Host: " + u2.getHost());
         System.out.println("Port: " + u2.getPort());
      } catch (MalformedURLException ex) {
         System.err.println("Invalid URL: " + ex);
      }
   }
}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\I
URL 1:
Protocol: http
Host: www.pemahojer.com.np
Port: -1

URL 2:
Protocol: http
Host: www.pemahojer.com.np
Port: -1

Process finished with exit code 0
```

Conclusion

Successfully extracted and displayed protocol, host, and port information from given URLs, showing how Java can parse URLs effectively.

4. WAP for SpamCheck.

Objective

To check whether a given host/IP address is listed as a spam source using DNS-based spam blacklists.

SpamCheck.java

```
import java.net.*;

public class SpamCheck {
public static final String BLACKHOLE = "sbl.spamhaus.org";

public static void main(String[] args) {
for (String arg : args) {
if (isSpammer(arg)) {
System.out.println(arg + " is a known spammer.");
} else {
System.out.println(arg + " appears legitimate.");
}
}
}

private static boolean isSpammer(String arg) {
try {
InetAddress address = InetAddress.getByName(arg);
byte[] quad = address.getAddress();
StringBuilder query = new StringBuilder(BLACKHOLE);

for (int i = quad.length - 1; i >= 0; i--) {
int unsignedByte = quad[i] < 0 ? quad[i] + 256 : quad[i];
```

```
query.insert(0, unsignedByte + ".");
}

InetAddress.getByName(query.toString());
return true;
} catch (UnknownHostException e) {
return false;
}
}
}
```

Console

```
PS C:\Users\Acer\Desktop\network\LabReport\src> javac SpamCheck.java
PS C:\Users\Acer\Desktop\network\LabReport\src> java SpamCheck 127.0.0.2
127.0.0.2 appears legitimate.
PS C:\Users\Acer\Desktop\network\LabReport\src> java SpamCheck 157.240.22.23
157.240.22.23 appears legitimate.
PS C:\Users\Acer\Desktop\network\LabReport\src>
```

Conclusion

Program accurately queried DNS blacklists to determine if given IPs were flagged for spam, demonstrating real-world use of **InetAddress** and DNS lookups.

5. Creates URL objects for any two websites and verify if they're the same using the equals() method.

Objective

To create **URL** objects for two websites and compare them using the equals() method to determine if they point to the same resource.

CompareUrls.java

```
import java.net.*;

public class CompareUrls {
   public static void main(String[] args) {
      try {
         // Creating URL objects
         URL url1 = new URL("https://www.apple.com");
         URL url2 = new URL("https://www.apple.com");

         // Comparing the URLs
         if (url1.equals(url2)) {
            System.out.println("The URLs are the same.");
         } else {
```

```
            System.out.println("The URLs are different.");
        }

    } catch (MalformedURLException e) {
        System.out.println("Invalid URL: " + e.getMessage());
    }
  }
}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ I
The URLs are the same.

Process finished with exit code 0
```

<u>Conclusion</u>

Compared two URLs and correctly determined their equality using the equals() method, showcasing how Java compares URL object references.

6. Program to download a web page with a URL Connection.

Objective

To download and analyze a web page using the URLConnection class by retrieving the content type and stream class.

<u>DownloadWebPage.java</u>

```
import java.io.*;
import java.net.*;

public class DownloadWebPage {
    public static void main(String[] args) {
        try {
            String location = "https://web.telegram.com/";
            URL url = new URL(location);

            Class<?>[] types = new Class[3];
            types[0] = String.class;
            types[1] = Reader.class;
            types[2] = InputStream.class;
            Object o = url.getContent(types);

            System.out.println(o.getClass().getName());
        } catch (IOException e){
            System.out.println(e);
```

```
            }
        }

}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program File
sun.net.www.protocol.http.HttpURLConnection$HttpInputStream

Process finished with exit code 0
```

Conclusion

Demonstrated the ability to open a connection to a web page and retrieve its content metadata using Java's networking classes.

7. Read 3 URLs from the command line and uses these six methods to print their content type, content length, content encoding, date of last modification, expiration date, and current date.

Objective

To read 3 URLs from the command line and print content type, content length, encoding, last modified date, expiration date, and current date.

URLInfoPrinter.java

```
import java.net.*;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class URLInfoPrinter {
    public static void main(String[] args) {
        if (args.length < 3) {
            System.out.println("Please provide at least 3 URLs as command line arguments.");
            return;
        }

        SimpleDateFormat sdf = new SimpleDateFormat("EEE, d MMM yyyy HH:mm:ss z");

        for (int i = 0; i < 3; i++) {
            try {
                URL url = new URL(args[i]);
                URLConnection connection = url.openConnection();
```

```
            System.out.println("URL: " + args[i]);

            System.out.println("Content Type: " + connection.getContentType());
            System.out.println("Content Length: " + connection.getContentLength());
            System.out.println("Content Encoding: " + connection.getContentEncoding());

            long lastModified = connection.getLastModified();
            System.out.println("Last Modified: " + (lastModified == 0 ? "Unavailable" :
sdf.format(new Date(lastModified))));

            long expiration = connection.getExpiration();
            System.out.println("Expiration Date: " + (expiration == 0 ? "Unavailable" :
sdf.format(new Date(expiration))));

            long date = connection.getDate();
            System.out.println("Date: " + (date == 0 ? "Unavailable" : sdf.format(new
Date(date))));

        } catch (Exception e) {
            System.out.println("Error with URL " + args[i] + ": " + e.getMessage());
        }
    }
  }
}
```

```
PS C:\Users\Acer\Desktop\network\LabReport\src> java URLInfoPrinter https://www.pemahojer.com.np https://www.karmarong.com https://www.craftedcreativity.com
URL: https://www.pemahojer.com.np
Content Type: text/html
Content Length: -1
Content Encoding: null
Last Modified: Fri, 3 May 2024 20:32:08 NPT
Expiration Date: Unavailable
Date: Thu, 22 May 2025 21:23:57 NPT
URL: https://www.karmarong.com
Content Type: text/html; charset=UTF-8
Content Length: -1
Content Encoding: null
Last Modified: Unavailable
Expiration Date: Unavailable
Date: Thu, 22 May 2025 21:24:03 NPT
URL: https://www.craftedcreativity.com
Content Type: text/html; charset=UTF-8
Content Length: -1
Content Encoding: null
Last Modified: Unavailable
Expiration Date: Unavailable
Date: Thu, 22 May 2025 21:24:07 NPT
PS C:\Users\Acer\Desktop\network\LabReport\src>
```

Conclusion

Successfully extracted and displayed comprehensive metadata for each URL, illustrating effective use of the URLConnection API.

8. WAP to Set ifModifiedSince to 24 hours prior to now.

Objective:
To send an HTTP request with the `If-Modified-Since` header set to 24 hours before the current time and observe the server response.

IfModifiedSince.java

```
import java.io.*;
import java.net.*;

public class IfModifiedSince {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.err.println("Please provide a URL as the first argument.");
            return;
        }

        try {
            URL url = new URL(args[0]);
            HttpURLConnection connection = (HttpURLConnection) url.openConnection();

            // Set If-Modified-Since to 24 hours prior to current time
            long oneDayMillis = 24 * 60 * 60 * 1000L;
            long twentyFourHoursAgo = System.currentTimeMillis() - oneDayMillis;
            connection.setIfModifiedSince(twentyFourHoursAgo);

            connection.connect();

            int responseCode = connection.getResponseCode();

            if (responseCode == HttpURLConnection.HTTP_NOT_MODIFIED) {
                System.out.println("Content not modified since 24 hours ago.");
            } else if (responseCode == HttpURLConnection.HTTP_OK) {
                System.out.println("Content modified. Reading data...");

                try (InputStream in = new BufferedInputStream(connection.getInputStream());
                     Reader reader = new InputStreamReader(in)) {
                    int c;
                    while ((c = reader.read()) != -1) {
                        System.out.print((char) c);
                    }
                }
            } else {
                System.out.println("Response code: " + responseCode);
            }

            connection.disconnect();
```

```
        } catch (MalformedURLException e) {
            System.err.println("Invalid URL: " + e.getMessage());
        } catch (IOException e) {
            System.err.println("IOException: " + e.getMessage());
        }
    }
}
```

Console

```
PS C:\Users\Acer\Desktop\network\LabReport\src> java IfModifiedSince https://www.example.com
Content not modified since 24 hours ago.
PS C:\Users\Acer\Desktop\network\LabReport\src>
```

Conclusion:
Properly used HTTP headers to check if content was modified, showing how Java can optimize web interactions using conditional GET requests.

9. Program to write to the server using socket and also using the telnet and comparing the response from the server.

Objective:
To write a Java program that connects to a time server via socket and compare the response with a Telnet connection.

TimeClient.java

```
import java.io.*;
import java.net.*;

public class TimeClient {
    public static void main(String[] args) {
        try {
            Socket s = new Socket("time.nist.gov", 13);
            InputStream in = s.getInputStream();
            InputStreamReader isr = new InputStreamReader(in, "ASCII");
            BufferedReader br = new BufferedReader(isr);

            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }

            s.close();
        } catch (IOException e) {
            System.out.println("Exception: " + e);
        }
    }
}
```

Console

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ

60817 25-05-22 15:46:01 50 0 0 622.3 UTC(NIST) *

Process finished with exit code 0
```

Conclusion:
Successfully retrieved time data from a remote server using a socket and compared it with Telnet output, showing multiple ways to access server data.

Comparison: Java Socket vs. Telnet

Feature	Java Socket	Telnet Command
Setup	Requires code and compilation	No programming required
Automation	Can loop, parse, or process responses	Manual connection
Response	Exact same data as Telnet	Same—raw server output
Educational Value	Shows socket lifecycle and data stream	Shows protocol-level interaction
Exception Handling	Handled via try-catch	Not available, shows basic errors

10. WAP to create a multithreaded daytime socket server and connect to at least 2 client at the same time.

Objective:
To create a multithreaded server that can handle multiple clients concurrently, simulating a daytime server.

MultithreadedDaytimeServer.java

```
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Date;
public class MultithreadedDaytimeServer {
    public final static int PORT = 13;
    public static void main(String[] args) {
        try (ServerSocket server = new ServerSocket(PORT)) {
            while (true) {
                try {
```

```
                Socket connection = server.accept();
                Thread task = new DaytimeThread(connection);
                task.start();
            } catch (IOException ex) {}
        }
    } catch (IOException ex) {
        System.err.println("Couldn't start server");
    }
  }
  private static class DaytimeThread extends Thread {
    private Socket connection;
    DaytimeThread(Socket connection) {
        this.connection = connection;
    }
    public void run() {
        try {
            Writer out = new OutputStreamWriter(connection.getOutputStream());
            Date now = new Date();
            out.write(now.toString() +"\r\n");
            out.write("From Server" +"\r\n");
            out.flush();
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
  }
}
```

ReadServerSocketBetter.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Socket;

public class ReadServerSocketBetter {
  public static void main(String[] args) throws IOException {
    Socket s=new Socket("localhost",13);
    InputStream in = s.getInputStream();
    InputStreamReader isr = new InputStreamReader(in, "ASCII");
    BufferedReader br = new BufferedReader(isr);
    br.lines().forEach(System.out::println);
  }
}
```

Server

```
Caused by: java.lang.NoClassDefFoundError: MultithreadedDaytimeServer (wrong name: MultithreadedDayTimeServer)
PS C:\Users\Acer\Desktop\network\LabReport\src> javac MultithreadedDayTimeServer.java
PS C:\Users\Acer\Desktop\network\LabReport\src> javac ReadServerSocketBetter.java
PS C:\Users\Acer\Desktop\network\LabReport\src> java MultithreadedDayTimeServer
```

Client1

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Acer\Desktop\network\LabReport> cd src
PS C:\Users\Acer\Desktop\network\LabReport\src> java ReadServerSocketBetter
Thu May 22 21:38:26 NPT 2025
From Server
```

Client2

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Acer\Desktop\network\LabReport> cd src
PS C:\Users\Acer\Desktop\network\LabReport\src> java ReadServerSocketBetter
Thu May 22 21:38:39 NPT 2025
From Server
```

Conclusion:
Achieved parallel client handling with multithreading, demonstrating the scalability and concurrency support of Java sockets.

11. In prior to lab program no demonstrates by adding logging to the daytime server.

Objective:
To enhance the multithreaded daytime server by adding detailed logging for better monitoring and debugging.

DemonstratesLogging.java

```
import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.*;

public class DemonstratesLogging {

    public final static int PORT = 13;
    private static final Logger logger = Logger.getLogger("DaytimeServer");

    public static void main(String[] args) {
```

```
        try (ServerSocket server = new ServerSocket(PORT)) {
            logger.info("Server started on port " + PORT);
            while (true) {
                try {
                    Socket connection = server.accept();
                    logger.info("Client connected: " + connection.getInetAddress());
                    Thread task = new DaytimeThread(connection);
                    task.start();
                } catch (IOException ex) {
                    logger.log(Level.SEVERE, "Error accepting client connection", ex);
                }
            }
        } catch (IOException ex) {
            logger.log(Level.SEVERE, "Couldn't start server", ex);
        }
    }

    private static class DaytimeThread extends Thread {
        private Socket connection;

        DaytimeThread(Socket connection) {
            this.connection = connection;
        }

        public void run() {
            try {
                Writer out = new OutputStreamWriter(connection.getOutputStream());
                Date now = new Date();
                out.write(now.toString() + "\r\n");
                out.write("From Server\r\n");
                out.flush();

                logger.info("Sent time to: " + connection.getInetAddress());
                connection.close();
            } catch (IOException ex) {
                logger.log(Level.WARNING, "Error handling client: " +
connection.getInetAddress(), ex);
            }
        }
    }
}
```

Server Console

```
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Acer\Desktop\network\LabReport> cd src
PS C:\Users\Acer\Desktop\network\LabReport\src> javac DemonstratesLogging.java
PS C:\Users\Acer\Desktop\network\LabReport\src> java DemonstratesLogging
May 22, 2025 9:43:16 PM DemonstratesLogging main
INFO: Server started on port 13
```

Conclusion:
Added meaningful logs to track server and client activities, making the server implementation more robust and easier to maintain.

12.WAP for a UDP Echo server and UDP client.

Objective:
To implement an echo server and client using UDP sockets, where the server responds with the same data received from the client.

Echoserver.java

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;

public class Echoserver {
    public static void main(String[] args) throws Exception {
        DatagramSocket socket = new DatagramSocket(78); // Match port with client
        byte[] receiveData = new byte[1024];
        byte[] sendData;

        System.out.println("UDP Server is running...");

        while (true) {
            // Receive packet
            DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
            socket.receive(receivePacket);

            // Get client data
            String message = new String(receivePacket.getData(), 0, receivePacket.getLength());
            System.out.println("Received from client: " + message);

            // Send back the same data
            sendData = message.getBytes();
            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
                    receivePacket.getAddress(), receivePacket.getPort());
            socket.send(sendPacket);
        }
```

```
    }
}
```

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
UDP Server is running...
Received from client: Hello Pema
```

EchoClient.java

```
import java.net.*;

public class EchoClient {
    public static void main(String[] args) throws Exception {
        DatagramSocket cs = new DatagramSocket();

        InetAddress ia = InetAddress.getByName("localhost");
        int port = 78; // Match port with server

        String message = "Hello Milton Collage";
        byte[] sendData = message.getBytes();
        byte[] receiveData = new byte[1024];

        // Create packet and send to server
        DatagramPacket sendPkt = new DatagramPacket(sendData, sendData.length, ia, port);
        cs.send(sendPkt);

        // Prepare to receive server's response
        DatagramPacket receivePkt = new DatagramPacket(receiveData, receiveData.length);
        cs.receive(receivePkt);

        String response = new String(receivePkt.getData(), 0, receivePkt.getLength());
        System.out.println("Received from server: " + response);

        cs.close(); // Close the socket
    }
}
```

```
C:\Users\Acer\.jdks\openjdk-23.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\I
Received from server: Hello Pema

Process finished with exit code 0
```

Conclusion:
Demonstrated successful message exchange between server and client over UDP, verifying non-connection-based communication.

13.RMI Client & Server Side

Objective:
To implement Remote Method Invocation (RMI) for performing remote addition and subtraction operations between a client and server.

Server.java
```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Server {
   public static void main(String[] args) {
      try {
         Rem_impt obj = new Rem_impt(); // create remote object
         Registry registry = LocateRegistry.createRegistry(1888); // create RMI registry
         registry.rebind("Rem", obj); // bind remote object
         System.out.println("Server ready.");
      } catch (Exception e) {
         System.out.println("Server error: " + e);
      }
   }
}
```

Client.java
```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.Scanner;

public class Client {
   public static void main(String[] args) {
      try {
         // Get reference to registry running on localhost and port 1888
         Registry registry = LocateRegistry.getRegistry("localhost", 1888);

         // Look up the remote object
         Rem stub = (Rem) registry.lookup("Rem");

         Scanner sc = new Scanner(System.in);
         System.out.println("Enter first number:");
         int a = sc.nextInt();
         System.out.println("Enter second number:");
         int b = sc.nextInt();

         // Remote method calls
         System.out.println("Sum = " + stub.addNum(a, b));
         System.out.println("Difference = " + stub.subNum(a, b));
      } catch (Exception e) {
         System.out.println("Client error: " + e);
      }
   }
}
```

Rem_impt.java

```
import java.rmi.*;
import java.rmi.server.*;

public class Rem_impt extends UnicastRemoteObject implements Rem {

    public Rem_impt() throws RemoteException {
        super(); // required to export the object
    }

    public int addNum(int a, int b) {
        return a + b;
    }

    public int subNum(int a, int b) {
        return a - b;
    }
}
```

Rem.java

```
import java.rmi.*;
public interface Rem extends Remote{

    public int addNum(int a,int b)throws RemoteException;
    public int subNum(int a,int b)throws RemoteException;

}
```

Server

```
PS C:\Users\Acer\Desktop\network\LabReport\src> javac Server.java
PS C:\Users\Acer\Desktop\network\LabReport\src> javac Client.java
PS C:\Users\Acer\Desktop\network\LabReport\src> javac Rem_impt.java
PS C:\Users\Acer\Desktop\network\LabReport\src> javac Rem.java
PS C:\Users\Acer\Desktop\network\LabReport\src> java Server
Server ready.
```

Client

```
Terminal    Local ×    Local (2) ×    +    ⌄
Error: Could not find or load main class client
Caused by: java.lang.NoClassDefFoundError: Client (wrong name: client)
PS C:\Users\Acer\Desktop\network\LabReport\src> java Client
Enter first number:
33
Enter second number:
66
Sum = 99
Difference = -33
PS C:\Users\Acer\Desktop\network\LabReport\src>
```

Conclusion:
Successfully executed remote method calls using Java RMI, proving Java's capability to perform distributed computing.