**Chapter 2: InetAddress**

**Lab 1**

**Objective: To check whether address is IPv4 or IPv6.**

**Source Code:**

import java.net.\*;

public class CheckIPType {

public static void main(String[] args) {

try {

*// InetAddress object banaune (example: google.com ko lagi)*

InetAddress address = InetAddress.getByName("google.com");

*// Address ko type check garne*

if (address instanceof Inet4Address) {

System.out.println("Yo IPv4 address ho: " + address.getHostAddress());

} else if (address instanceof Inet6Address) {

System.out.println("Yo IPv6 address ho: " + address.getHostAddress());

} else {

System.out.println("Unknown address type: " + address.getHostAddress());

}

} catch (Exception e) {

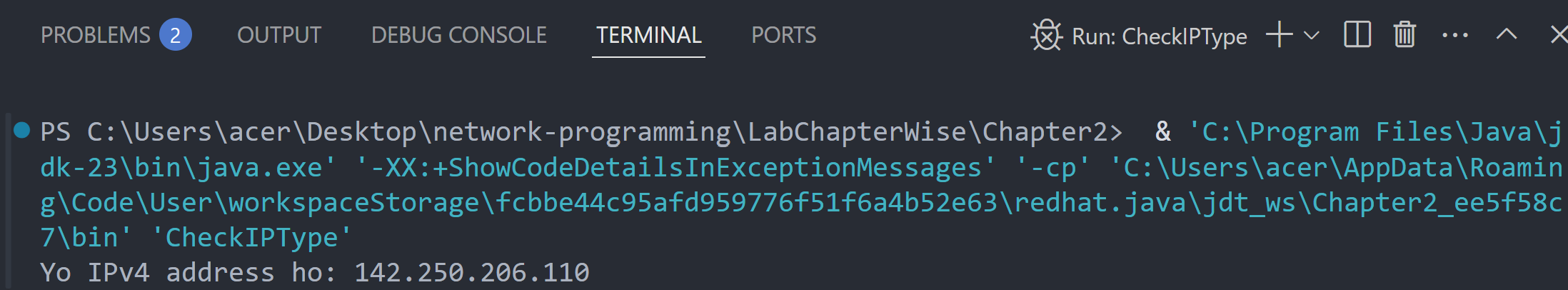
System.out.println("Error aayo: " + e.getMessage());

}

}

}

**Output:**



**Lab 2**

**Objective: To find the address of local machine.**

**Source Code:**

import java.net.\*;

public class LocalAddress {

public static void main(String[] args) {

try {

*// Local machine ko address lina*

InetAddress local = InetAddress.getLocalHost();

System.out.println("Local Address: " + local.getHostAddress());

} catch (Exception e) {

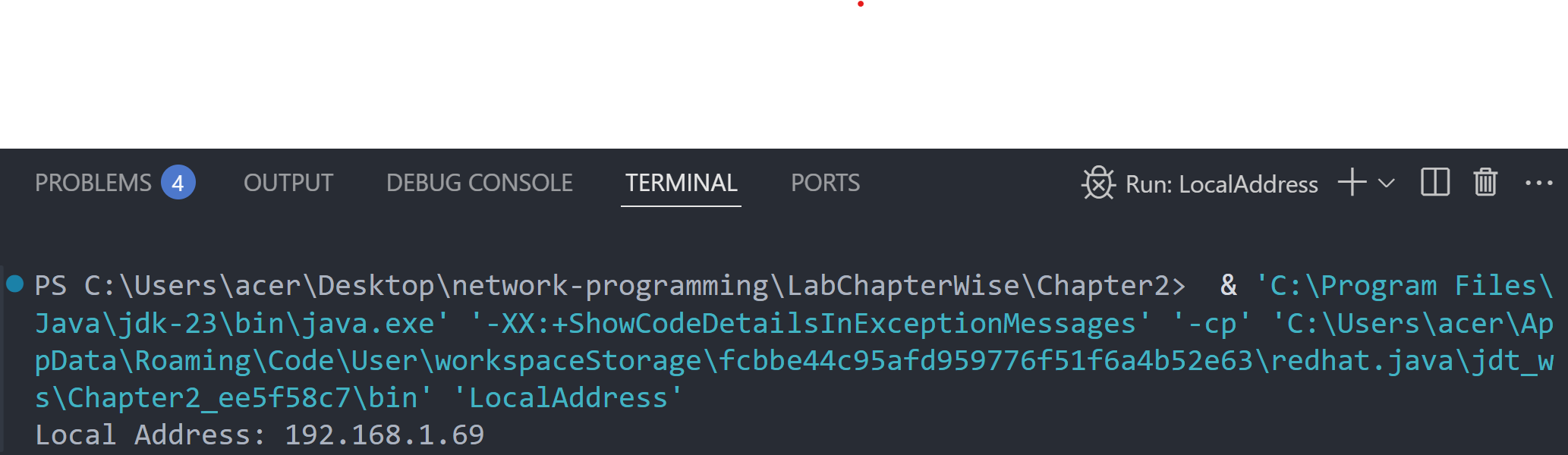
System.out.println("Error aayo: " + e.getMessage());

}

}

}

**Output:**



**Lab 3**

**Objective: To find the IP Address and host name of local machine.**

**Source Code:**

import java.net.\*;

public class LocalIPHost {

public static void main(String[] args) {

try {

*// Local machine ko IP ra Hostname print garne*

InetAddress local = InetAddress.getLocalHost();

System.out.println("IP Address: " + local.getHostAddress());

System.out.println("Host Name: " + local.getHostName());

} catch (Exception e) {

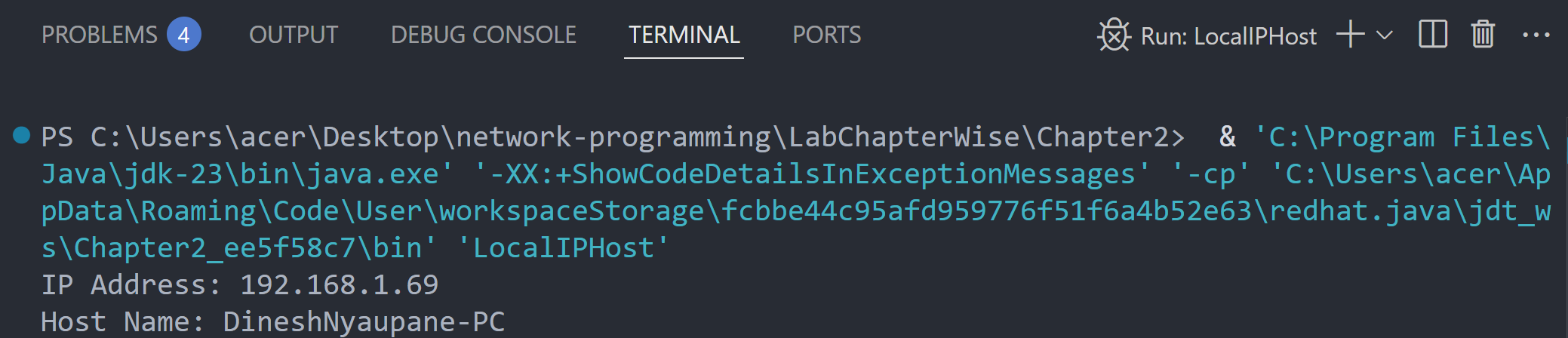
System.out.println("Error aayo: " + e.getMessage());

}

}

}

**Output:**



**Lab 4**

**Objective: To demonstrate the SpamCheck.**

**Source Code:**

import java.net.\*;

public class SpamCheck {

public static void main(String[] args) {

try {

*// Domain name ko address khojne*

InetAddress address = InetAddress.getByName("www.google.com");

*// Hostname spam chaina bhane reply garne*

if (!address.isAnyLocalAddress()) {

System.out.println("Yo spam address huna sakcha!");

} else {

System.out.println("Yo safe address ho.");

}

} catch (Exception e) {

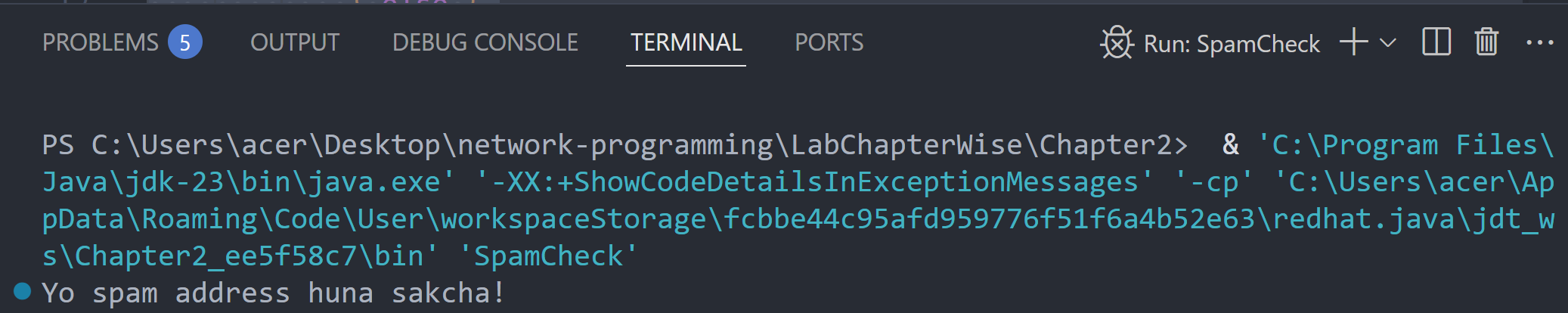
System.out.println("Error aayo: " + e.getMessage());

}

}

}

**Output:**



**Lab 5**

**Objective: To compare the domain names “*www.ibiblio.org*” and “*helios.ibiblio.org*”.**

**Source Code:**

import java.net.\*;

public class CompareDomain {

public static void main(String[] args) {

try {

*// Duita domain ko IP address khojne*

InetAddress addr1 = InetAddress.getByName("www.ibiblio.org");

InetAddress addr2 = InetAddress.getByName("helios.ibiblio.org");

*// Address compare garne*

if (addr1.equals(addr2)) {

System.out.println("Domain haru same address ma point garcha.");

} else {

System.out.println("Domain haru different address ma chha.");

}

} catch (Exception e) {

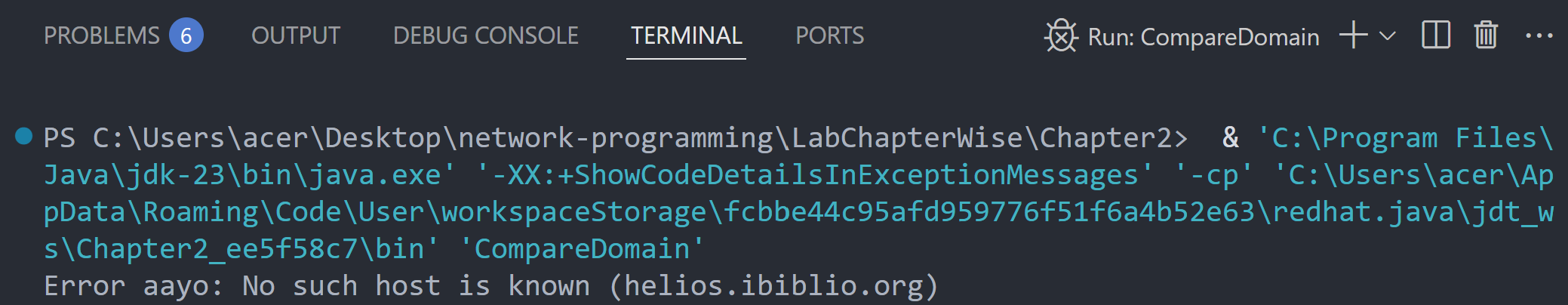
System.out.println("Error aayo: " + e.getMessage());

}

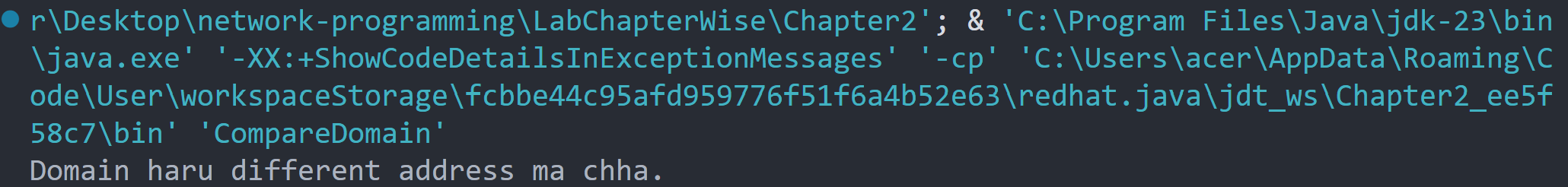
}

}

**Output:**



If we check [www.google.com](http://www.google.com) and [www.youtube.com](http://www.youtube.com) :



**Chapter 3: URLs and URIs**

**Lab 6**

**Objective: To write a program that splits the parts of a URL [Splitting URL into pieces information]**

**Source Code:**

import java.net.URL;

public class SplitURL {

public static void main(String[] args) {

try {

URL url = new URL("https://www.example.com:8080/path/to/resource?query=nepal#section");

System.out.println("Protocol: " + url.getProtocol());

System.out.println("Host: " + url.getHost());

System.out.println("Port: " + url.getPort());

System.out.println("Path: " + url.getPath());

System.out.println("Query: " + url.getQuery());

System.out.println("Reference: " + url.getRef());

} catch (Exception e) {

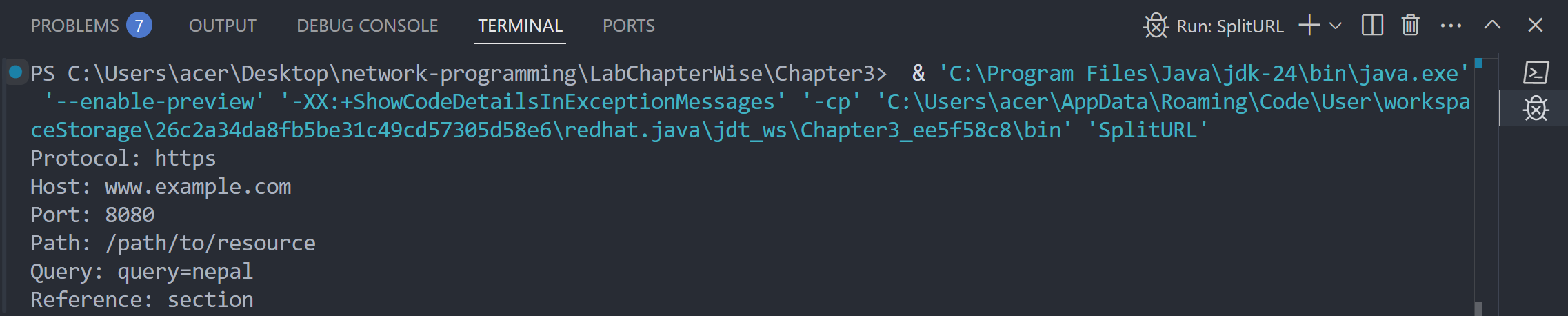
System.out.println("URL ko parts split garna error: " + e);

}

}

}

**Output:**



**Lab 7**

**Objective: To write a program that checks the which protocols does a virtual machine support or not?**

**Source Code:**

import java.net.URLConnection;

import java.util.Arrays;

import java.util.List;

public class CheckProtocols {

public static void main(String[] args) {

List<String> protocols = Arrays.asList("http", "https", "ftp", "file", "mailto");

for (String protocol : protocols) {

try {

URLConnection connection = new java.net.URL(protocol + "://test.com").openConnection();

System.out.println(protocol + " is supported.");

} catch (Exception e) {

System.out.println(protocol + " is NOT supported.");

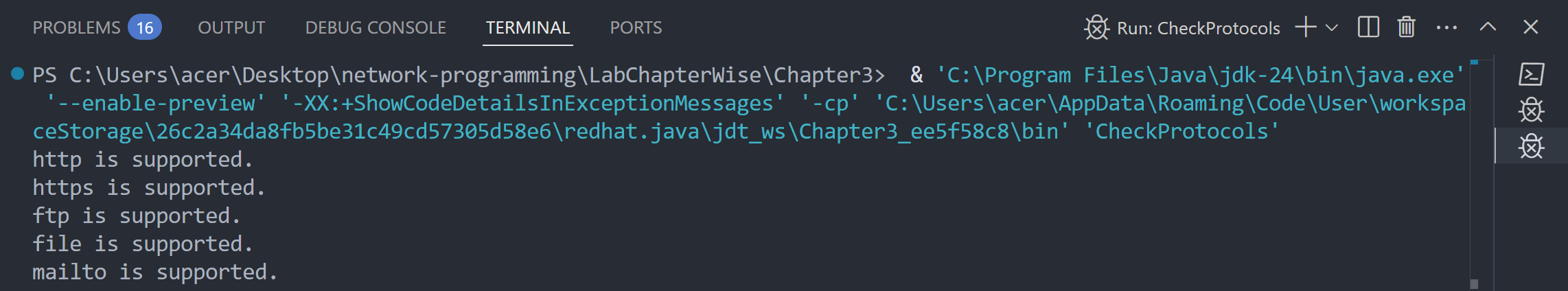
}

}

}

}

**Output:**



**Lab 8**

**Objective: To write a program to download a web page of a given web address.**

**Source Code:**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.URL;

public class DownloadWebPage {

public static void main(String[] args) {

try {

URL url = new URL("https://example.com");

BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

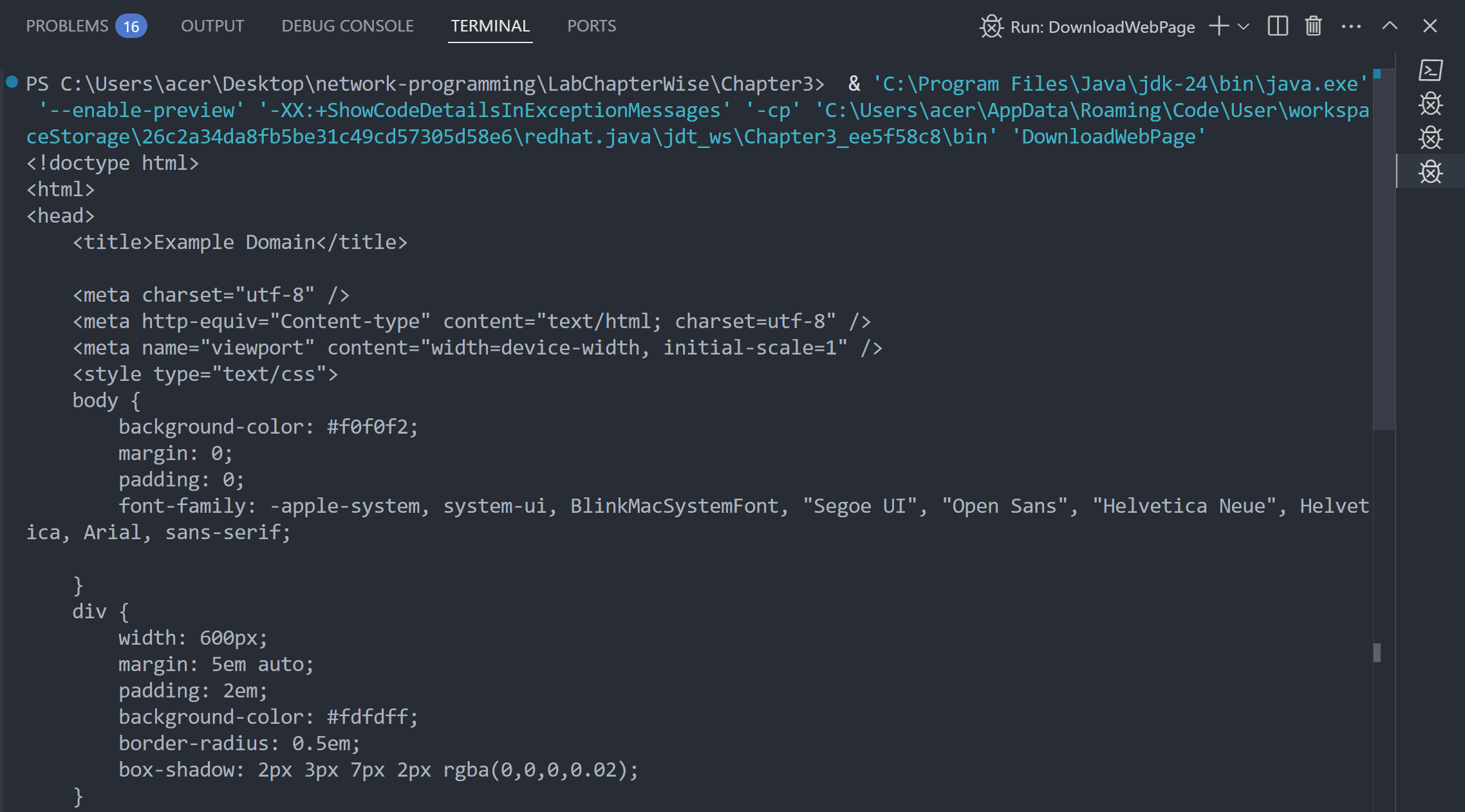
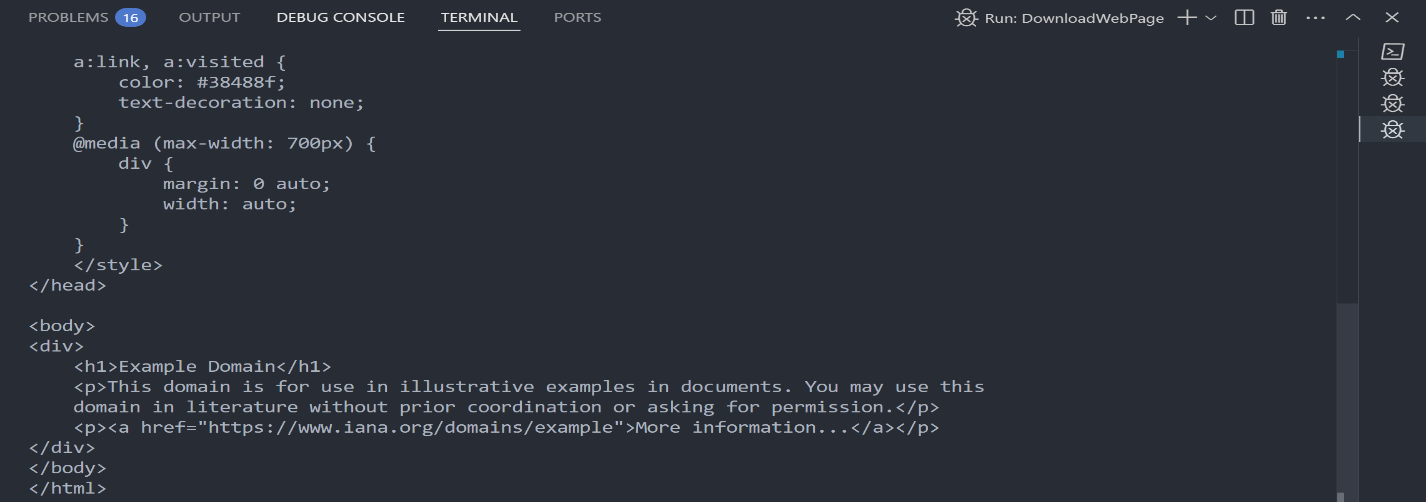
reader.close();

} catch (Exception e) {

System.out.println("Webpage download garna error: " + e);

} } }

**Output:**



**Lab 9**

**Objective: To write a program for resolving relatives URI**

**Source Code:**

import java.net.URI;

public class ResolveRelativeURI {

public static void main(String[] args) {

try {

URI base = new URI("https://www.example.com/folder/");

URI relative = new URI("file.html");

URI resolved = base.resolve(relative);

System.out.println("Resolved URI: " + resolved.toString());

} catch (Exception e) {

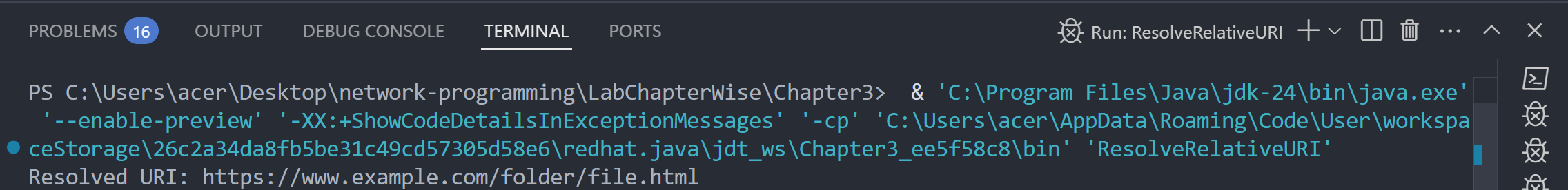
System.out.println("Relative URI resolve garna error: " + e);

}

}

}

**Output:**

****

**Lab 10**

**Objective: To write a program to download an object.**

**Source Code:**

import java.io.FileOutputStream;

import java.io.InputStream;

import java.net.URL;

public class DownloadObject {

public static void main(String[] args) {

try {

URL url = new URL("https://www.example.com");

InputStream input = url.openStream();

FileOutputStream output = new FileOutputStream("downloaded\_sample.pdf");

byte[] buffer = new byte[2048];

int bytesRead;

while ((bytesRead = input.read(buffer)) != -1) {

output.write(buffer, 0, bytesRead);

}

input.close();

output.close();

System.out.println("File download successfully.");

} catch (Exception e) {

System.out.println("File download garna error: " + e);

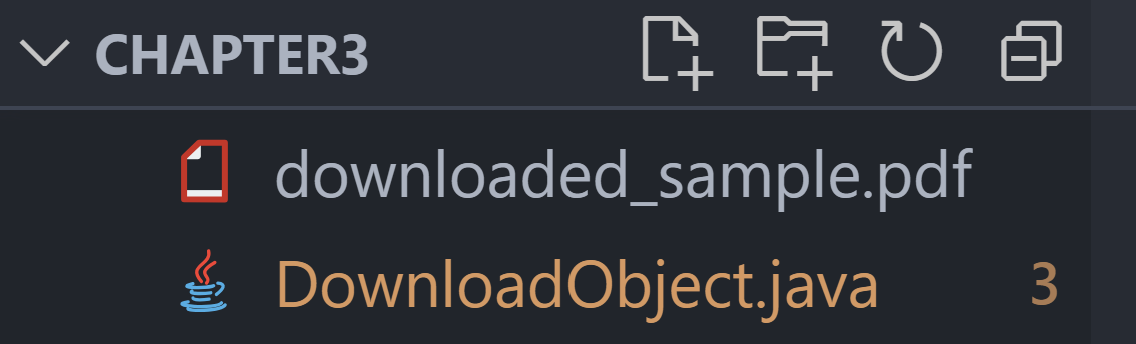
}

}

}

**Output:**





**Lab 11**

**Objective: To write a program to demonstrate the x-www-form-URL encoded strings.**

**Source Code:**

import java.net.URLEncoder;

import java.nio.charset.StandardCharsets;

public class FormURLEncoding {

public static void main(String[] args) {

try {

String name = "Ram Bahadur";

String location = "Nepal Kathmandu";

String encodedName = URLEncoder.encode(name, StandardCharsets.UTF\_8);

String encodedLocation = URLEncoder.encode(location, StandardCharsets.UTF\_8);

System.out.println("Encoded form data:");

System.out.println("name=" + encodedName + "&location=" + encodedLocation);

} catch (Exception e) {

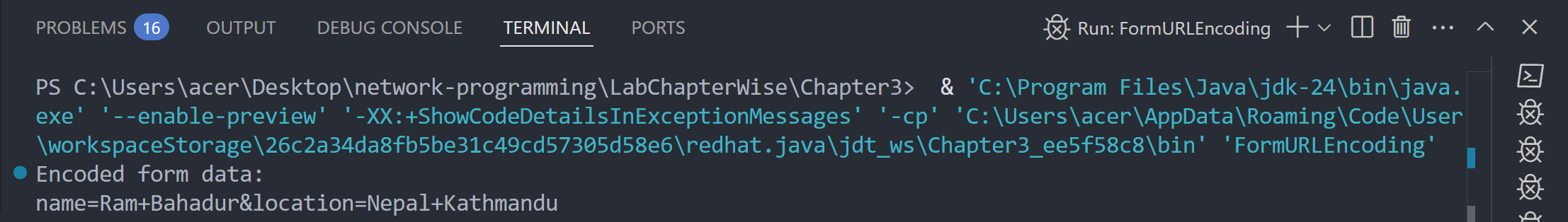
System.out.println("Form URL encoding garna error: " + e);

}

}

}

**Output:**



**Lab 12**

**Objective: To write a program that communicating with Server-Side Programs Through GET.**

**Source Code:**

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;

public class HTTPGetRequest {

public static void main(String[] args) {

try {

URL url = new URL("https://jsonplaceholder.typicode.com/posts/1");

HttpURLConnection conn = (HttpURLConnection) url.openConnection();

conn.setRequestMethod("GET");

int responseCode = conn.getResponseCode();

System.out.println("Response Code: " + responseCode);

BufferedReader reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));

String line;

while ((line = reader.readLine()) != null) {

System.out.println(line);

}

reader.close();

} catch (Exception e) {

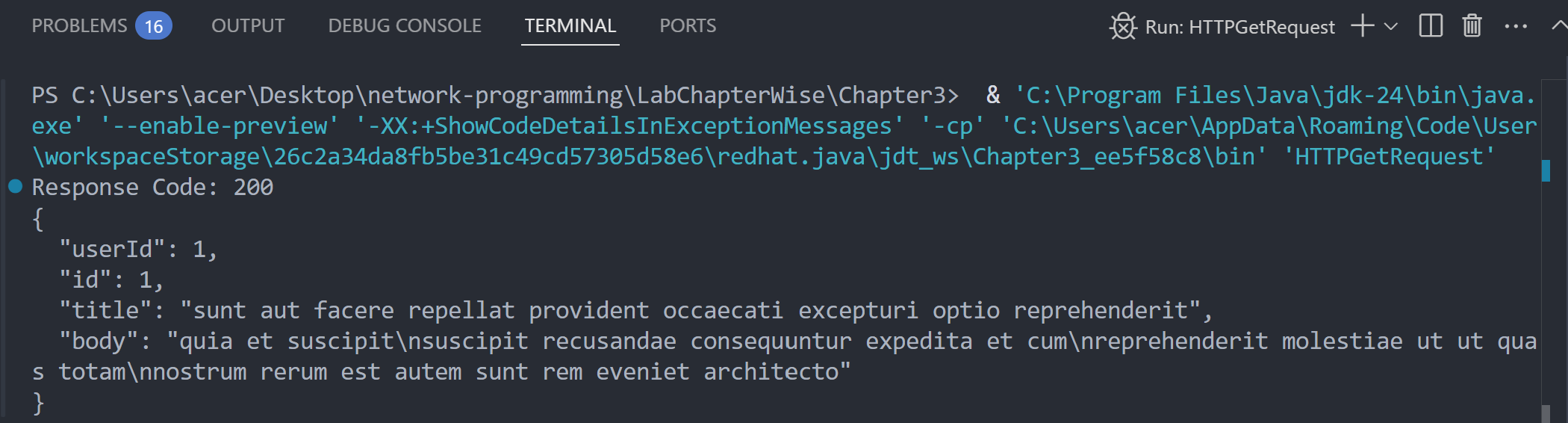
System.out.println("Server sanga GET communication garna error: " + e);

}

}

}

**Output:**



**Chapter 4: HTTP**

**Lab 13**

**Objective: To write a program that shows a simple CookiePolicy that blocks cookies from .gov domains, but allows others.**

**Source Code:**

import java.net.\*;

public class CookiePolicyExample {

public static void main(String[] args) throws Exception {

CookiePolicy policy = new CookiePolicy() {

@Override

public boolean shouldAccept(URI uri, HttpCookie cookie) {

String host = uri.getHost();

return !host.endsWith(".gov.np");

}

};

CookieManager manager = new CookieManager();

manager.setCookiePolicy(policy);

CookieHandler.setDefault(manager);

URI govUri = new URI("http://mofaga.gov.np"); // blocked

URI eduUri = new URI("http://nepathya.edu.np"); // allowed

HttpCookie govCookie = new HttpCookie("session", "gov\_cookie");

HttpCookie eduCookie = new HttpCookie("session", "edu\_cookie");

boolean isGovAccepted = policy.shouldAccept(govUri, govCookie);

boolean isEduAccepted = policy.shouldAccept(eduUri, eduCookie);

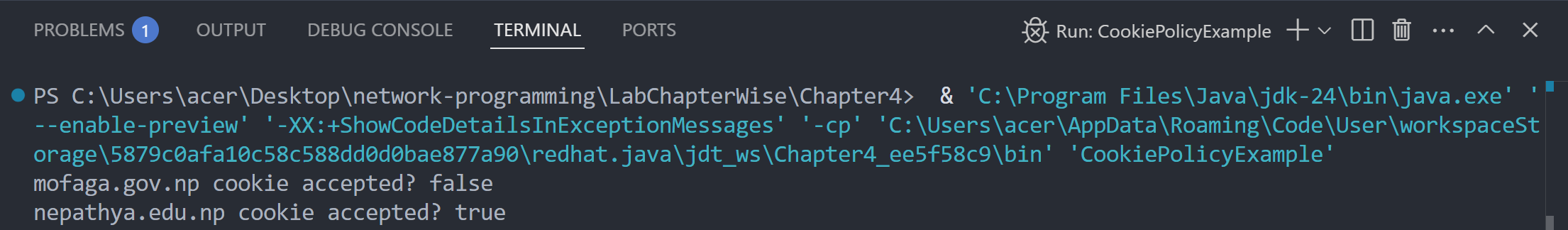
System.out.println("mofaga.gov.np cookie accepted? " + isGovAccepted);

System.out.println("nepathya.edu.np cookie accepted? " + isEduAccepted);

}

}

**Output:**



**Lab 14**

**Objective: To implement the CookieStore Methods (add, read, delete) cookies.**

**Source Code:**

import java.net.\*;

import java.util.\*;

public class CookieStoreExample {

public static void main(String[] args) throws Exception {

CookieManager manager = new CookieManager();

CookieHandler.setDefault(manager);

CookieStore store = manager.getCookieStore();

URI nepUri = new URI("http://nepathya.edu.np");

// Add a cookie

HttpCookie cookie = new HttpCookie("user", "Dinesh-SixthSemester");

store.add(nepUri, cookie);

System.out.println(" Cookie added for nepathya.edu.np: " + cookie);

System.out.println("\n Cookies after addition (" + store.getCookies().size() + "):");

for (HttpCookie c : store.getCookies()) {

System.out.println(" " + c);

}

// Delete cookie

store.remove(nepUri, cookie);

System.out.println("\n Cookie deleted.");

// Verify deletion

List<HttpCookie> cookies = store.getCookies();

System.out.println("\n Cookies after deletion (" + cookies.size() + "):");

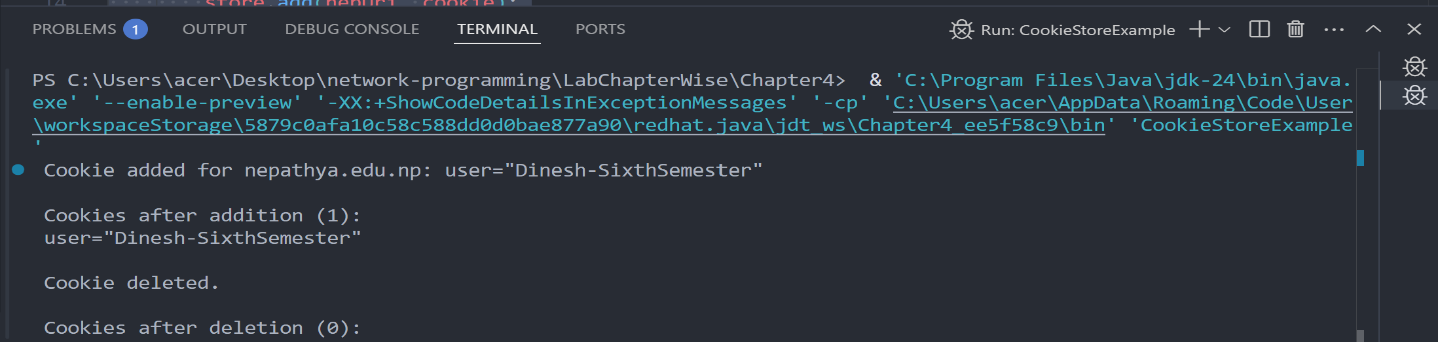
for (HttpCookie c : cookies) {

System.out.println("➡️ " + c);

}

}

}

**Output:**

**Chapter 5: URL Connections**

**Lab 15**

**Objective: To write a program to download a web page using URLConnection.**

**Source Code:**

import java.io.\*;

import java.net.\*;

public class DOwnloadWebPage {

public static void main(String[] args) {

String website = "http://nepathyacollege.edu.np";

try {

URL url = new URL(website);

URLConnection connection = url.openConnection();

BufferedReader reader = new BufferedReader(

new InputStreamReader(connection.getInputStream()));

BufferedWriter writer = new BufferedWriter(

new FileWriter("DownloadedPage.html"));

String line;

while ((line = reader.readLine()) != null) {

writer.write(line);

writer.newLine();

}

reader.close();

writer.close();

System.out.println("Web page downloaded successfully as 'DownloadedPage.html'");

} catch (Exception e) {

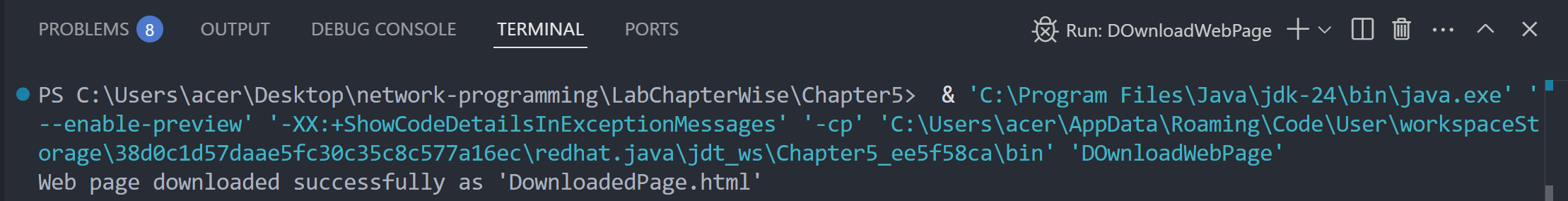
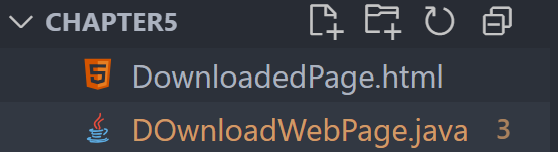
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Lab 16**

**Objective: To write a program to read value of HTTP Header fields.**

**Source Code:**

import java.net.\*;

public class ReadHTTPHeaderFields {

public static void main(String[] args) {

try {

URL url = new URL("http://nepathyacollege.edu.np");

URLConnection connection = url.openConnection();

String contentType = connection.getHeaderField("Content-Type");

String server = connection.getHeaderField("Server");

System.out.println("Content-Type: " + contentType);

System.out.println("Server: " + server);

} catch (Exception e) {

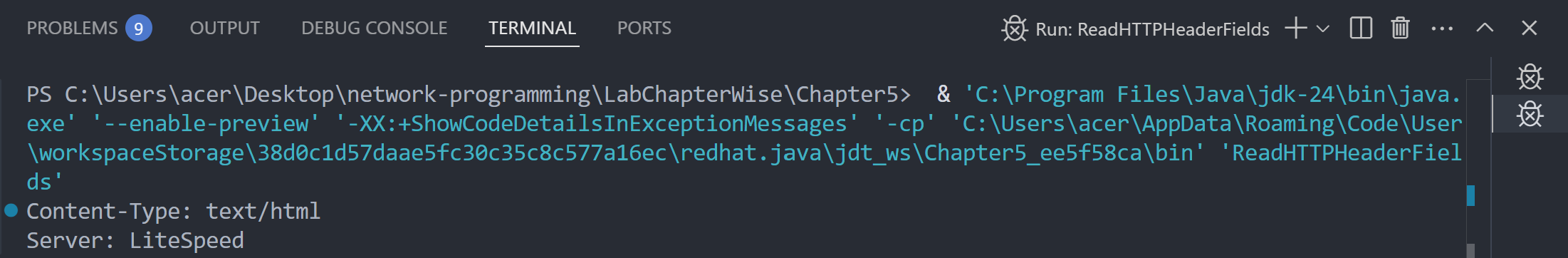
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Lab 17**

**Objective: To write a program to print the entire HTTP header**

**Source Code:**

import java.net.\*;

import java.util.\*;

public class PrintEntireHTTPHeader {

public static void main(String[] args) {

try {

URL url = new URL("http://nepathyacollege.edu.np");

URLConnection connection = url.openConnection();

Map<String, List<String>> headers = connection.getHeaderFields();

for (Map.Entry<String, List<String>> entry : headers.entrySet()) {

String key = entry.getKey();

List<String> values = entry.getValue();

System.out.println(key + ": " + String.join(", ", values));

}

} catch (Exception e) {

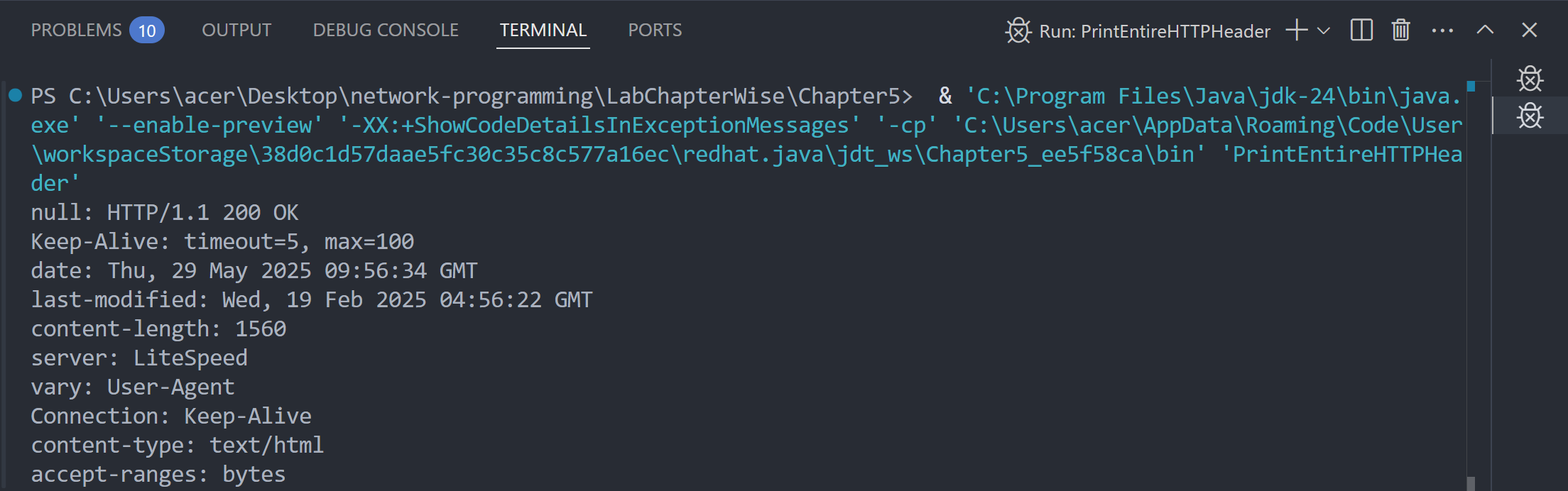
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Lab 18**

**Objective: To write a program for HTTP Request Method.**

**Source Code:**

import java.net.\*;

public class HTTPRequestMethod {

public static void main(String[] args) {

try {

URL url = new URL("http://nepathyacollege.edu.np");

HttpURLConnection connection = (HttpURLConnection) url.openConnection();

connection.setRequestMethod("GET"); // You can try "POST", "HEAD" etc.

System.out.println("HTTP Request Method: " + connection.getRequestMethod());

} catch (Exception e) {

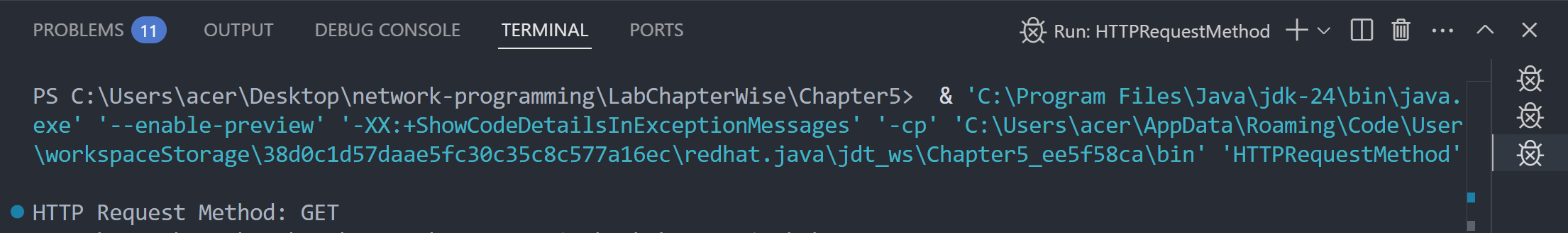
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Lab 19**

**Objective: To write a program to print the URL of a URL Connection to “nepathyacollege.edu.np”**

**Source Code:**

import java.net.\*;

public class PrintURLConnectionURL {

public static void main(String[] args) {

try {

URL url = new URL("http://nepathyacollege.edu.np");

URLConnection connection = url.openConnection();

System.out.println("URL of the connection: " + connection.getURL());

} catch (Exception e) {

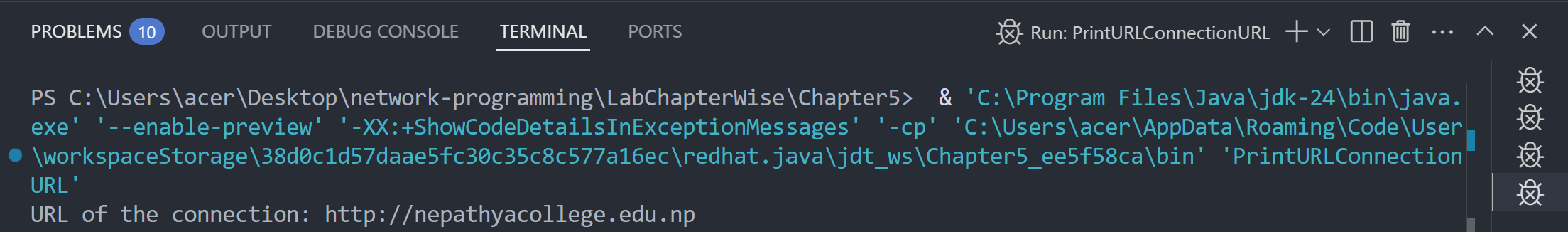
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Lab 20**

**Objective: To write a program to get the time when a URL was last changed.**

**Source Code:**

import java.net.\*;

import java.text.SimpleDateFormat;

import java.util.Date;

public class URLLastModifiedTime {

public static void main(String[] args) {

try {

URL url = new URL("http://nepathyacollege.edu.np");

URLConnection connection = url.openConnection();

long lastModified = connection.getLastModified();

if (lastModified == 0) {

System.out.println("No Last-Modified information.");

} else {

Date date = new Date(lastModified);

SimpleDateFormat sdf = new SimpleDateFormat("dd MMM yyyy HH:mm:ss");

System.out.println("Last Modified: " + sdf.format(date));

}

} catch (Exception e) {

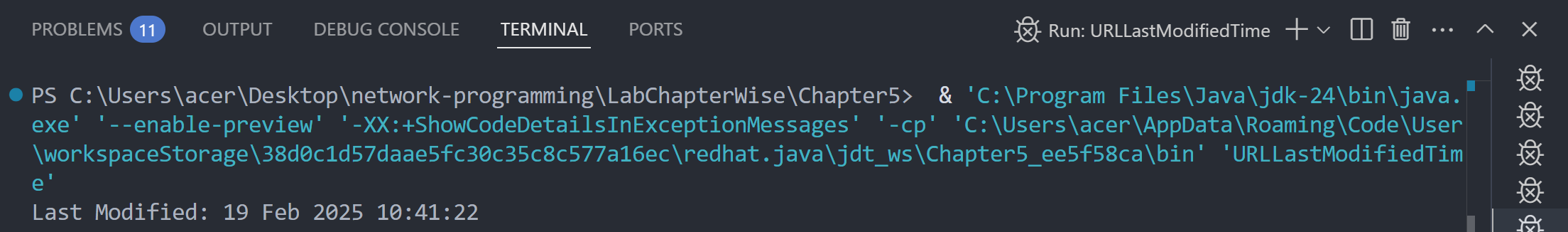
System.out.println("Error: " + e.getMessage());

}

}

}

**Output:**



**Chapter 6: Socket for Clients**

**Lab 21**

**Objective: To write a program socket to client.**

Source Code:

import java.io.\*;

import java.net.\*;

public class SimpleSocketClient {

public static void main(String[] args) {

String serverAddress = "localhost"; *// or IP address of the server*

int port = 5000; *// port number the server is listening on*

try (Socket socket = new Socket(serverAddress, port);

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in))) {

System.out.println("Connected to server at " + serverAddress + ":" + port);

*// Read user input and send to server*

System.out.print("Enter message to send: ");

String message = userInput.readLine();

out.println(message);

*// Read and print server response*

String response = in.readLine();

System.out.println("Server response: " + response);

} catch (IOException e) {

System.err.println("Error: " + e.getMessage());

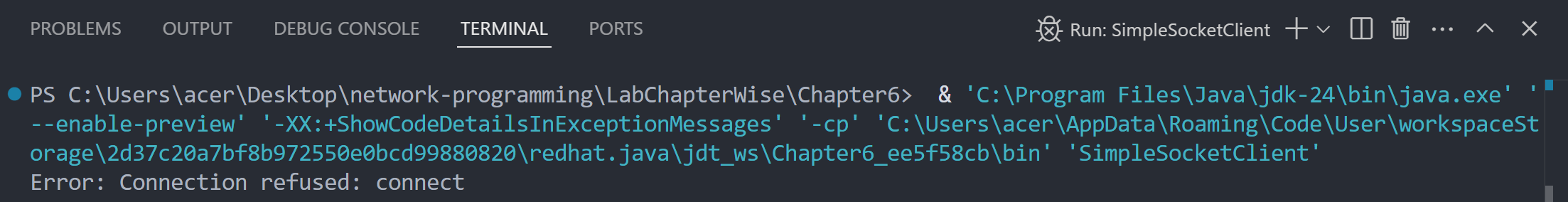
}

}

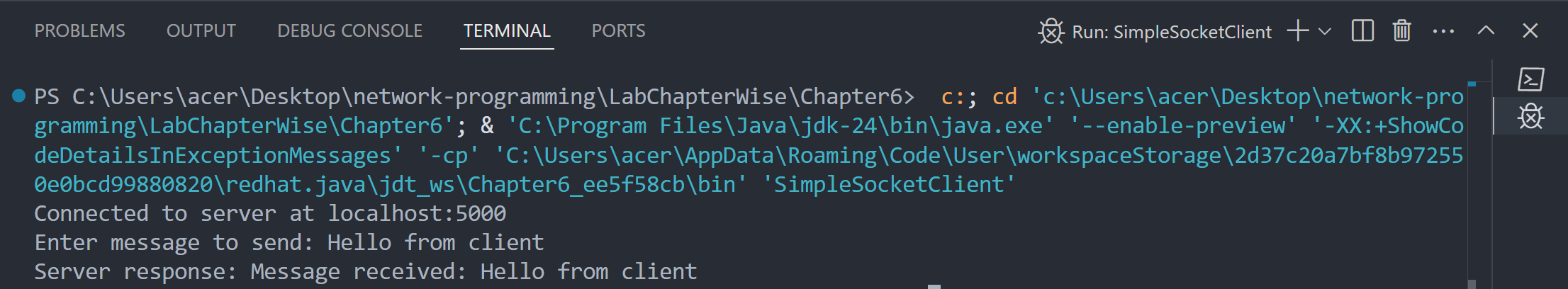
}

Output:

*(before connecting to server)*



*(after connecting to server)*

**

**Chapter 7: Socket for Server**

**Lab 22**

**Objective: To write a program socket to server.**

**Source Code:**

import java.io.\*;

import java.net.\*;

public class SimpleSocketServer {

public static void main(String[] args) {

int port = 5000; *// Server will listen on this port*

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("Server started. Waiting for client connection on port " + port);

Socket clientSocket = serverSocket.accept(); *// Accept client connection*

System.out.println("Client connected: " + clientSocket.getInetAddress());

BufferedReader in = new BufferedReader(

new InputStreamReader(clientSocket.getInputStream()));

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

*// Read message from client*

String message = in.readLine();

System.out.println("Received from client: " + message);

*// Send response to client*

out.println("Message received: " + message);

*// Close client connection*

clientSocket.close();

System.out.println("Client connection closed.");

} catch (IOException e) {

System.err.println("Error: " + e.getMessage());

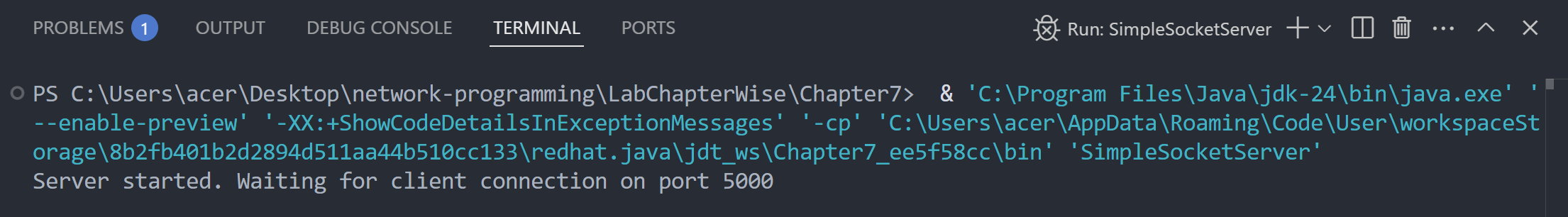
}

}

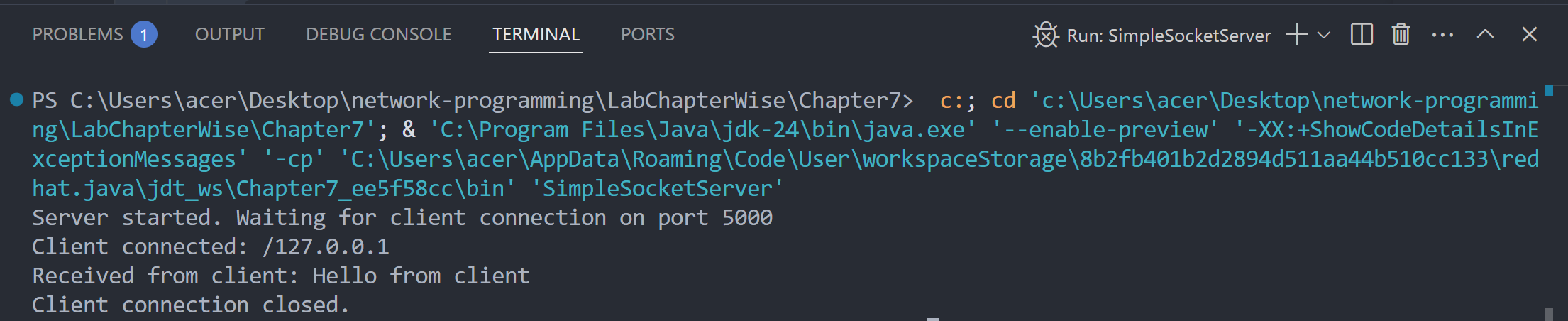
}

**Output:**

*(before any client connects)*



*(after client connects)*

**

**Chapter 8: Secure Sockets**

**Lab 23**

**Objective: To write a program for Creating Secure Sockets with tufohss.edu.np.**

**Source Code:**

import java.io.\*;

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

public class SecureSocketClient {

public static void main(String[] args) {

String host = "tufohss.edu.np";

int port = 443; *// HTTPS default port*

try {

*// Create SSL socket factory and socket*

SSLSocketFactory factory = (SSLSocketFactory) SSLSocketFactory.getDefault();

SSLSocket sslSocket = (SSLSocket) factory.createSocket(host, port);

*// Start handshake (optional, but recommended)*

sslSocket.startHandshake();

*// Output stream to send data to server*

PrintWriter out = new PrintWriter(new BufferedWriter(

new OutputStreamWriter(sslSocket.getOutputStream())));

*// Input stream to read server response*

BufferedReader in = new BufferedReader(

new InputStreamReader(sslSocket.getInputStream()));

*// Send HTTP GET request*

out.println("GET / HTTP/1.1");

out.println("Host: " + host);

out.println("Connection: Close");

out.println(); *// blank line to end request headers*

out.flush();

*// Read and print the response line by line*

String line;

while ((line = in.readLine()) != null) {

System.out.println(line);

}

*// Close streams and socket*

in.close();

out.close();

sslSocket.close();

} catch (Exception e) {

System.err.println("Error: " + e.getMessage());

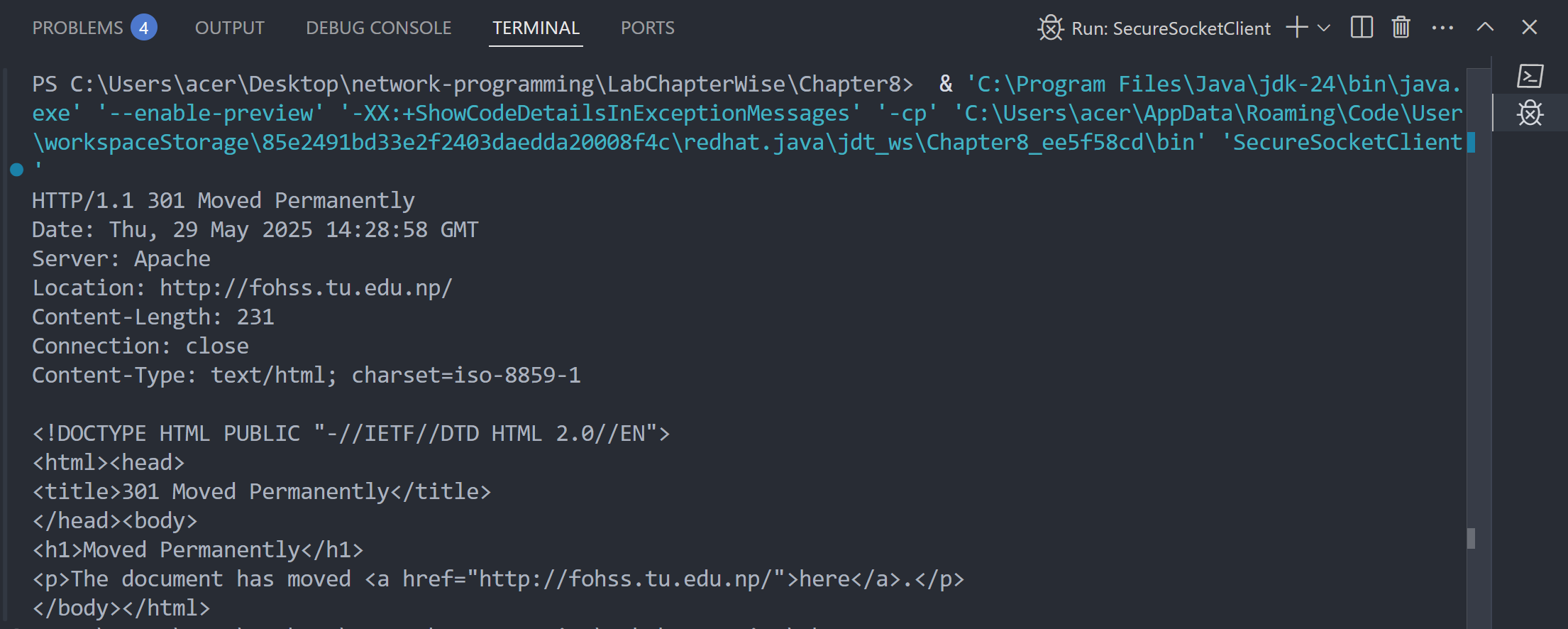
e.printStackTrace();

}

}

}

Output:



**Lab 24**

**Objective: To write a program for Creating Secure Server Sockets and Client Sockets.**

**Source Code:**

**SecureClient.java**

import java.io.\*;

import java.security.KeyStore;

import javax.net.ssl.\*;

public class SecureClient {

public static void main(String[] args) throws Exception {

char[] password = "password".toCharArray();

*// Set system properties to point to the truststore file containing server cert or CA cert*

System.setProperty("javax.net.ssl.trustStore", "clienttruststore.jks");

System.setProperty("javax.net.ssl.trustStorePassword", "password");

*// Load the truststore file (this step is optional if you want to create SSLContext manually)*

KeyStore ts = KeyStore.getInstance("JKS");

try (FileInputStream fis = new FileInputStream("clienttruststore.jks")) {

ts.load(fis, password);

}

*// Initialize TrustManagerFactory with the truststore*

TrustManagerFactory tmf = TrustManagerFactory.getInstance("SunX509");

tmf.init(ts);

*// Set up SSL context using the trust managers from the truststore*

SSLContext sc = SSLContext.getInstance("TLS");

sc.init(null, tmf.getTrustManagers(), null);

*// Create SSLSocketFactory from SSL context*

SSLSocketFactory ssf = sc.getSocketFactory();

*// Create an SSL socket connected to the server*

try (SSLSocket socket = (SSLSocket) ssf.createSocket("localhost", 12345);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {

*// Read response from server*

String response = in.readLine();

System.out.println("Message from Server: " + response);

}

}

}

**SecureServer.java**

import java.io.\*;

import java.security.KeyStore;

import javax.net.ssl.\*;

public class SecureServer {

public static void main(String[] args) throws Exception {

char[] password = "password".toCharArray();

*// Load keystore*

KeyStore ks = KeyStore.getInstance("JKS");

ks.load(new FileInputStream("serverkeystore.jks"), password);

*// Initialize KeyManagerFactory*

KeyManagerFactory kmf = KeyManagerFactory.getInstance("SunX509");

kmf.init(ks, password);

*// Set up SSL context*

SSLContext sc = SSLContext.getInstance("TLS");

sc.init(kmf.getKeyManagers(), null, null);

SSLServerSocketFactory ssf = sc.getServerSocketFactory();

SSLServerSocket serverSocket = (SSLServerSocket) ssf.createServerSocket(12345);

System.out.println("Secure Server started on port 12345...");

while (true) {

SSLSocket socket = (SSLSocket) serverSocket.accept();

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

out.println("Hello from secure server!");

socket.close();

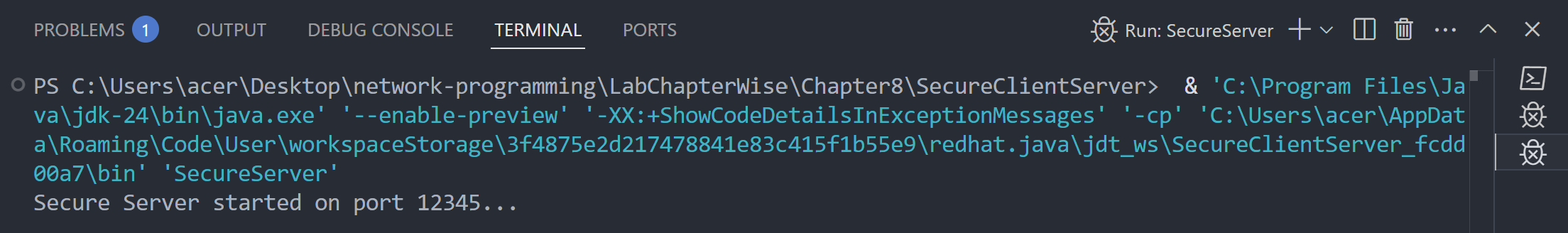
}

}

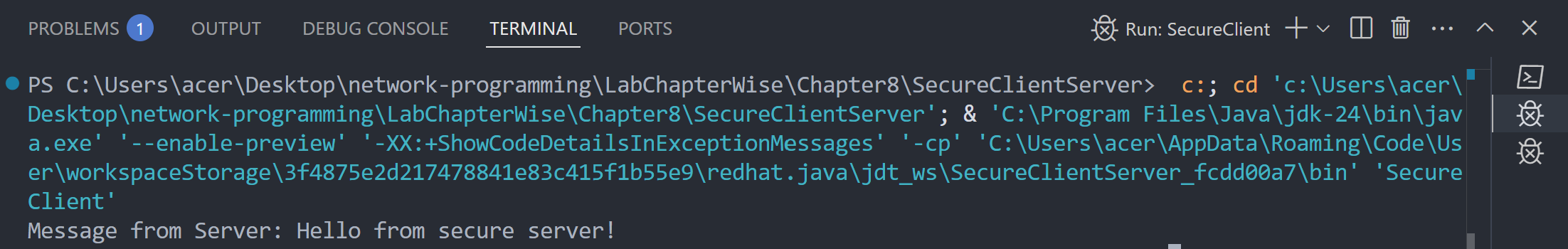
}

**Output:**

*(SecureServer.java)*



*(SecureClient.java)*



**Chapter 9: Non Blocking I/O**

**Lab 25**

**Objective: To write program to list all supported socket options for the different types of network channels**

**Source Code:**

import java.net.\*;

import java.nio.channels.\*;

import java.util.Set;

public class ListSocketOptions {

public static void main(String[] args) throws Exception {

System.out.println("SocketChannel options:");

SocketChannel socketChannel = SocketChannel.open();

Set<SocketOption<?>> socketOptions = socketChannel.supportedOptions();

for (SocketOption<?> option : socketOptions) {

System.out.println(" - " + option.name());

}

System.out.println("\nServerSocketChannel options:");

ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();

Set<SocketOption<?>> serverSocketOptions = serverSocketChannel.supportedOptions();

for (SocketOption<?> option : serverSocketOptions) {

System.out.println(" - " + option.name());

}

System.out.println("\nDatagramChannel options:");

DatagramChannel datagramChannel = DatagramChannel.open();

Set<SocketOption<?>> datagramOptions = datagramChannel.supportedOptions();

for (SocketOption<?> option : datagramOptions) {

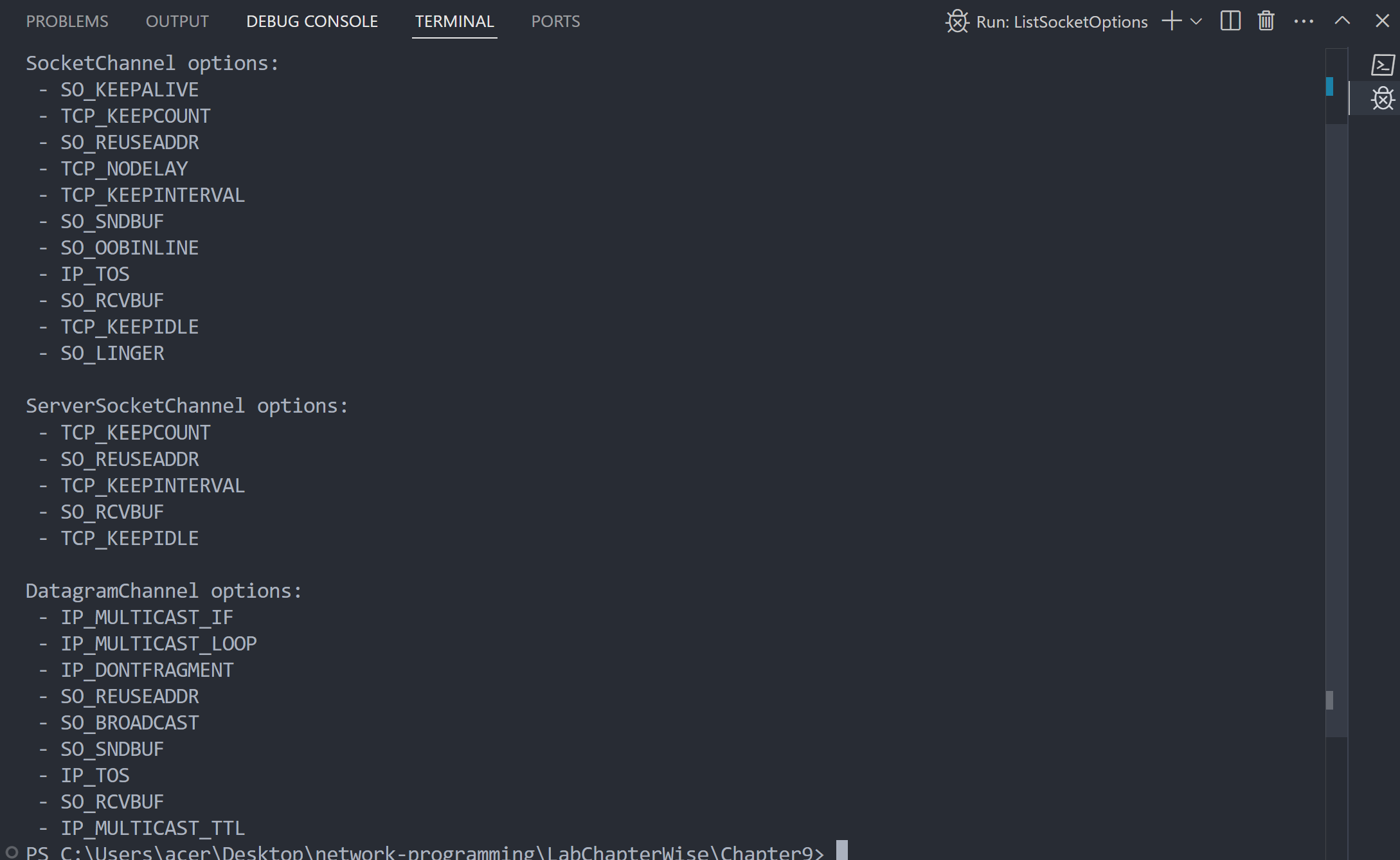
System.out.println(" - " + option.name());

}

}

}

Output:



**Lab 26**

**Objective: To write program to implement the concept on Filling and Draining buffer, duplicating buffer, Slicing buffer, Compact buffer.**

**Source Code:**

import java.nio.ByteBuffer;

public class BufferOperations {

public static void main(String[] args) {

*// Fill buffer*

ByteBuffer buffer = ByteBuffer.allocate(10);

for (byte i = 0; i < 10; i++) {

buffer.put(i);

}

*// Flip (prepare for reading)*

buffer.flip();

*// Drain buffer*

System.out.print("Draining buffer: ");

while (buffer.hasRemaining()) {

System.out.print(buffer.get() + " ");

}

*// Refill*

buffer.clear();

for (byte i = 10; i < 20; i++) {

buffer.put(i);

}

*// Duplicate*

ByteBuffer duplicate = buffer.duplicate();

duplicate.flip();

System.out.print("\nDuplicate buffer: ");

while (duplicate.hasRemaining()) {

System.out.print(duplicate.get() + " ");

}

*// Slice buffer*

buffer.position(2);

buffer.limit(6);

ByteBuffer slice = buffer.slice();

System.out.print("\nSliced buffer: ");

for (int i = 0; i < slice.capacity(); i++) {

System.out.print(slice.get(i) + " ");

}

*//compact*

buffer.clear();

buffer.put((byte) 1);

buffer.put((byte) 2);

buffer.put((byte) 3);

buffer.flip();

buffer.get(); // read one byte

buffer.compact(); // move remaining bytes to beginning

buffer.put((byte) 4); // add more

buffer.flip();

System.out.print("\nCompact buffer: ");

while (buffer.hasRemaining()) {

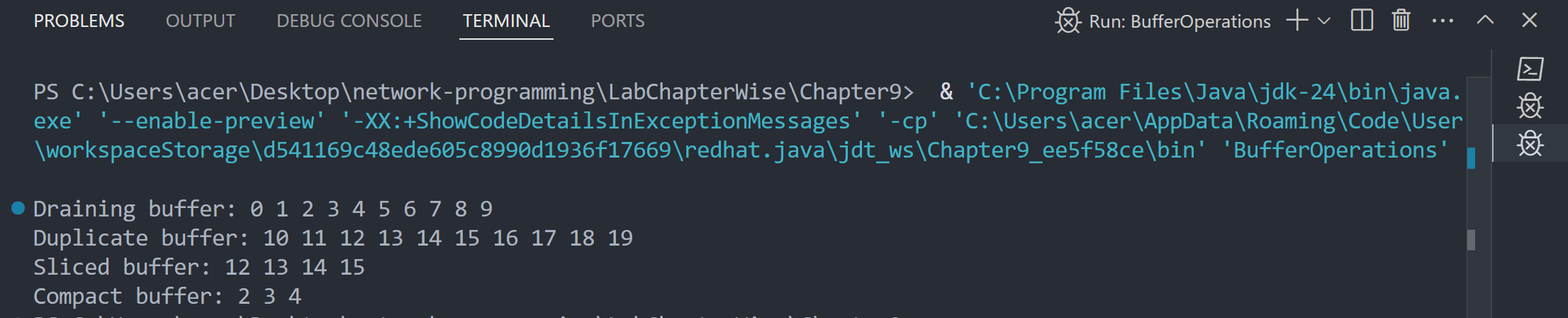
System.out.print(buffer.get() + " ");

}

}

}

**Output:**



**Lab 27**

**Objective: To write a program to implement the concept on Data Conversion**

**Source Code:**

import java.nio.ByteBuffer;

import java.nio.CharBuffer;

import java.nio.charset.Charset;

import java.nio.charset.StandardCharsets;

public class DataConversion {

public static void main(String[] args) {

String input = "Hello, buffer world!";

System.out.println("Original String: " + input);

Charset charset = StandardCharsets.UTF\_8;

ByteBuffer byteBuffer = charset.encode(input);

System.out.print("Encoded Bytes: ");

while (byteBuffer.hasRemaining()) {

System.out.print(byteBuffer.get() + " ");

}

byteBuffer.rewind();

CharBuffer decoded = charset.decode(byteBuffer);

System.out.println("\nDecoded String: " + decoded.toString());

}

}

**Output:**



**Chapter 10: UDP**

**Lab 28**

**Objective: To write a program for UDP Client**

**Source Code:**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class UDPCLient {

public static void main(String[] args) {

try {

DatagramSocket clientSocket = new DatagramSocket();

InetAddress serverAddress = InetAddress.getByName("localhost");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter message to send: ");

String message = scanner.nextLine();

byte[] sendBuffer = message.getBytes();

DatagramPacket sendPacket = new DatagramPacket(sendBuffer, sendBuffer.length, serverAddress, 9876);

clientSocket.send(sendPacket);

*// Receive response*

byte[] receiveBuffer = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);

clientSocket.receive(receivePacket);

String response = new String(receivePacket.getData(), 0, receivePacket.getLength());

System.out.println("Server replied: " + response);

clientSocket.close();

scanner.close();

} catch (Exception e) {

e.printStackTrace();

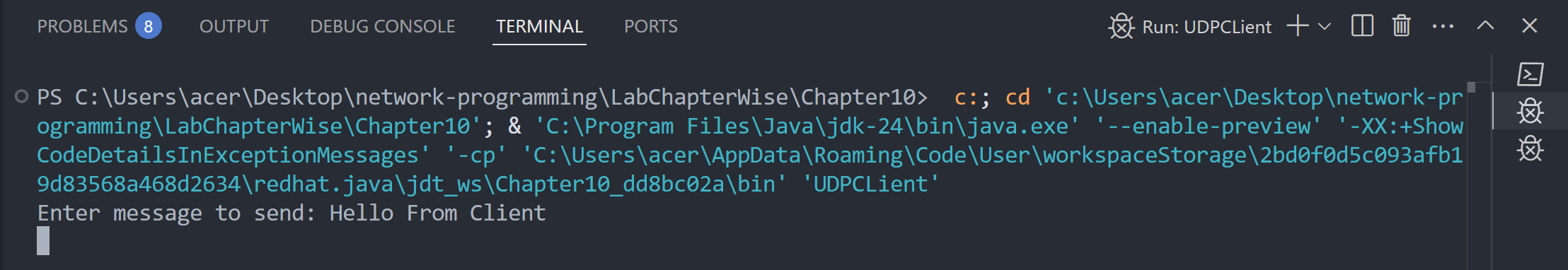
}

}

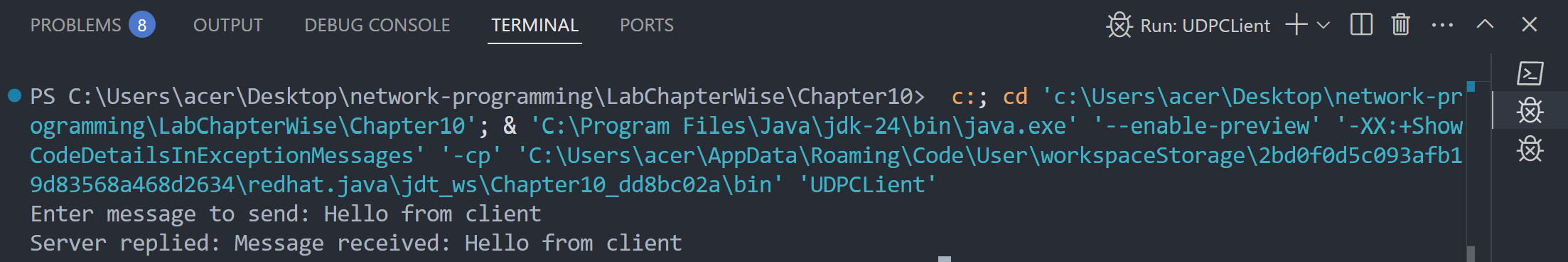
}

**Output:**

*(before connecting to server)*



*(after connecting to server)*



**Lab 29**

**Objective: To write a program for UDP Server.**

**Source Code:**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class UDPServer {

public static void main(String[] args) {

try {

DatagramSocket serverSocket = new DatagramSocket(9876);

byte[] receiveBuffer = new byte[1024];

System.out.println("Server is running and waiting for data...");

*// Receive packet*

DatagramPacket receivePacket = new DatagramPacket(receiveBuffer, receiveBuffer.length);

serverSocket.receive(receivePacket);

String clientMessage = new String(receivePacket.getData(), 0, receivePacket.getLength());

System.out.println("Client says: " + clientMessage);

*// Send response*

String reply = "Message received: " + clientMessage;

byte[] sendBuffer = reply.getBytes();

DatagramPacket sendPacket = new DatagramPacket(

sendBuffer,

sendBuffer.length,

receivePacket.getAddress(),

receivePacket.getPort()

);

serverSocket.send(sendPacket);

serverSocket.close();

} catch (Exception e) {

e.printStackTrace();

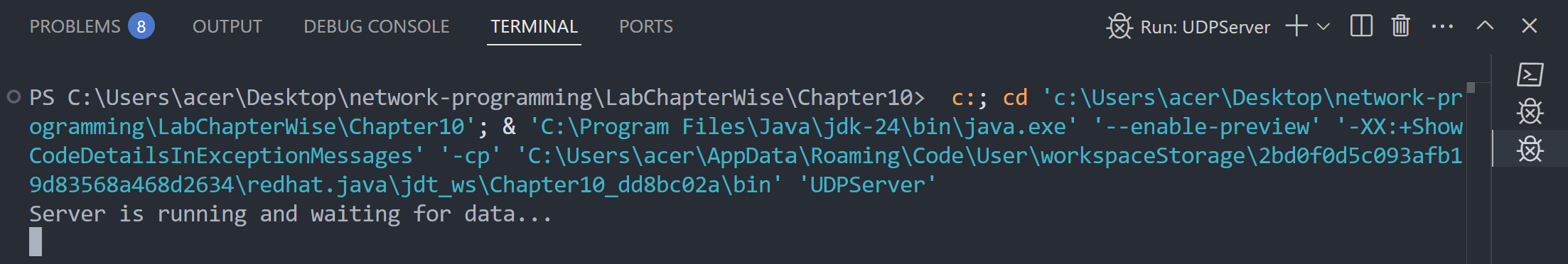
}

}

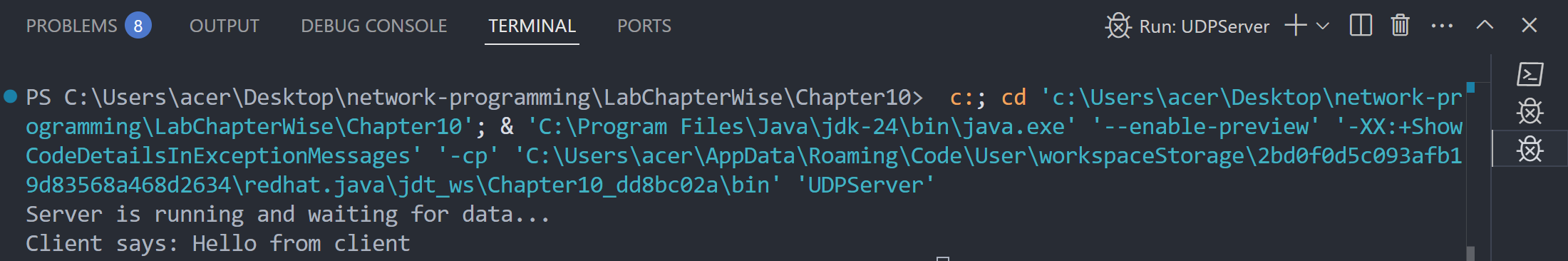
}

**Output:**

*(before any client connects)*



*(after client connects)*



**Chapter 11: IP Multicast**

**Lab 30**

**Objective: To verify that you are receiving multicast data at a particular host.**

**Source Code:**

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.InetAddress;

import java.net.MulticastSocket;

public class MulticastReceiver {

public static void main(String[] args) {

final int PORT = 4446; *// Use the same port as sender*

final String MULTICAST\_GROUP = "230.0.0.0"; // A valid multicast IP

try {

MulticastSocket socket = new MulticastSocket(PORT);

InetAddress group = InetAddress.getByName(MULTICAST\_GROUP);

socket.joinGroup(group);

System.out.println("Joined multicast group. Listening for messages...");

byte[] buffer = new byte[1024];

DatagramPacket packet = new DatagramPacket(buffer, buffer.length);

socket.receive(packet);

String received = new String(packet.getData(), 0, packet.getLength());

System.out.println("Received multicast message: " + received);

socket.leaveGroup(group);

socket.close();

} catch (IOException e) {

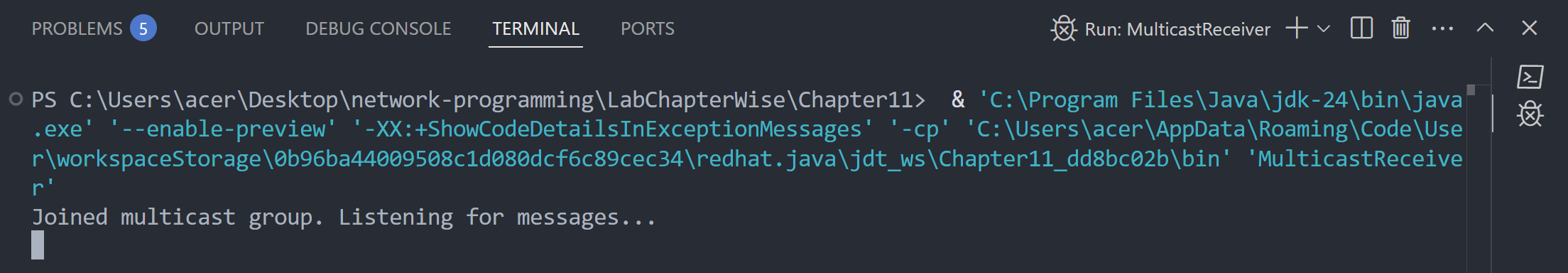
e.printStackTrace();

}

}

}

**Output:**

****

**Chapter 12: RMI**

**Lab 31**

**Objective: To add two numbers using RMI**

**Source Code:**

**Calculator.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calculator extends Remote{

int add(int a, int b) throws RemoteException;

}

**CalculatorRemote.java**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalculatorRemote extends UnicastRemoteObject implements Calculator{

*//constructor*

public CalculatorRemote() throws RemoteException {

super();

}

@Override

public int add(int a, int b){

return a + b;

}

}

**Server.java**

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Server {

public static void main(String[] args) {

try{

CalculatorRemote objCalculatorRemote = new CalculatorRemote();

Registry registry = LocateRegistry.createRegistry(9000);

registry.bind("Multiply", objCalculatorRemote);

System.out.println("Server is ready to accept requests...");

}catch (Exception e) {

System.out.println("Server exception: " + e.getMessage());

e.printStackTrace();

}

}

}

**Client.java**

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class Client {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry("localhost", 9000);

Calculator calculator = (Calculator) registry.lookup("Multiply");

int value = calculator.add(60, 10);

System.out.println("Result is: "+value);

} catch (Exception e) {

System.out.println("Client exception: " + e.getMessage());

e.printStackTrace();

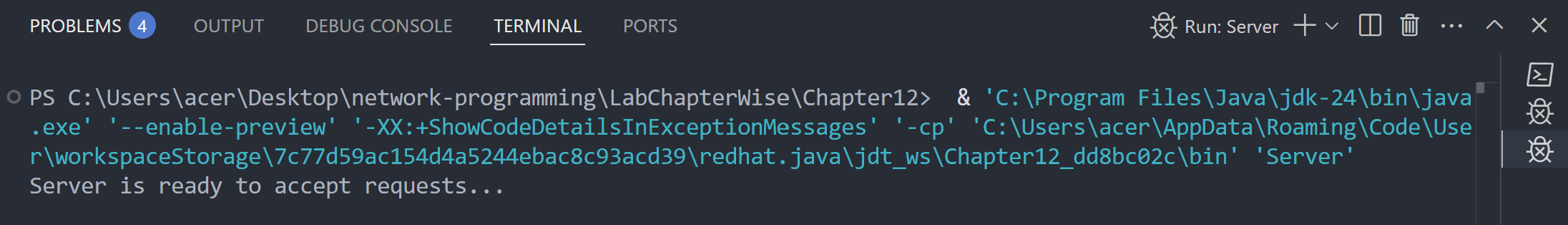
}

}

}

**Output:**

*(server)*



*(client)*

