Practical No. :- 1

Aim:- **To develop a program for multi-client chat server using Socket.**

Theory:-

A **multi-client chat server** using sockets allows multiple clients to connect to a server and communicate in real-time. It leverages the **TCP/IP protocol** for reliable message exchange. The server listens on a specific port, accepting client connections using **Java's Socket** and **ServerSocket** classes. When a client connects, the server spawns a new thread to handle each client independently, enabling concurrent communication. The **client program** connects to the server using the server's IP address and port number. A **broadcast mechanism** is implemented to send messages from one client to all other connected clients. This approach is scalable and can be used for real-time chat applications. In cloud computing, such a program can be deployed on cloud platforms like AWS or Google Cloud, utilizing their scalable infrastructure for a high number of concurrent connections.

Algorithm/Concept:-

□ **Start Server:**

- Create a server socket on a specific port.
- Initialize a list to keep track of connected clients.

□ **Listen for Clients:**

- Accept new client connections in a loop.
- Create a new thread for each connected client.

□ **Client Thread Execution:**

- Read messages from the client.
- Broadcast the message to all other connected clients.

□ **Broadcast Message:**

- Iterate through the list of clients and send the message.

□ **Client Disconnect:**

- Remove the client from the list upon disconnection.

□ **Stop Server:**

- Close all sockets and terminate the server.

Code/ Program:-

Server.java:- import

java.io.*; import

java.net.*; import

java.util.*; public

class Server {

```
    private static Set<ClientHandler> clientHandlers = new HashSet<>();
    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(1234);
            System.out.println("Server started. Waiting for clients...");
            while (true) {
                Socket socket = serverSocket.accept();
                System.out.println("New client connected: " + socket);
                ClientHandler clientHandler = new
                ClientHandler(socket);
                clientHandlers.add(clientHandler);
                Thread thread = new
                Thread(clientHandler); thread.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                if (serverSocket != null) {
```

```
                serverSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} public static void broadcastMessage(String message, ClientHandler
    sender) { for (ClientHandler clientHandler : clientHandlers) {
        if (clientHandler != sender) {
            clientHandler.sendMessage(message);
        }
    }
} public static void removeClient(ClientHandler
    clientHandler) { clientHandlers.remove(clientHandler);
}
}
    public ClientHandler(Socket socket) {
        this.socket = socket;
            try {
            in = new BufferedReader(new
            InputStreamReader(socket.getInputStream())); out = new
            PrintWriter(socket.getOutputStream(), true);
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
}@Override
public void run() {
   String   message;
   try {
      while ((message = in.readLine()) != null) {
         System.out.println("Received: " + message);
         Server.broadcastMessage(message, this);
         }
   } catch (IOException e) {
      e.printStackTrace();
   } finally {
      Server.removeClient(this);
      try {
         socket.close();
         } catch (IOException e) { e.printStackTrace();
      }
   }
   public void sendMessage(String message) {
      out.println(message);
   }
}
```

Client.java:- import

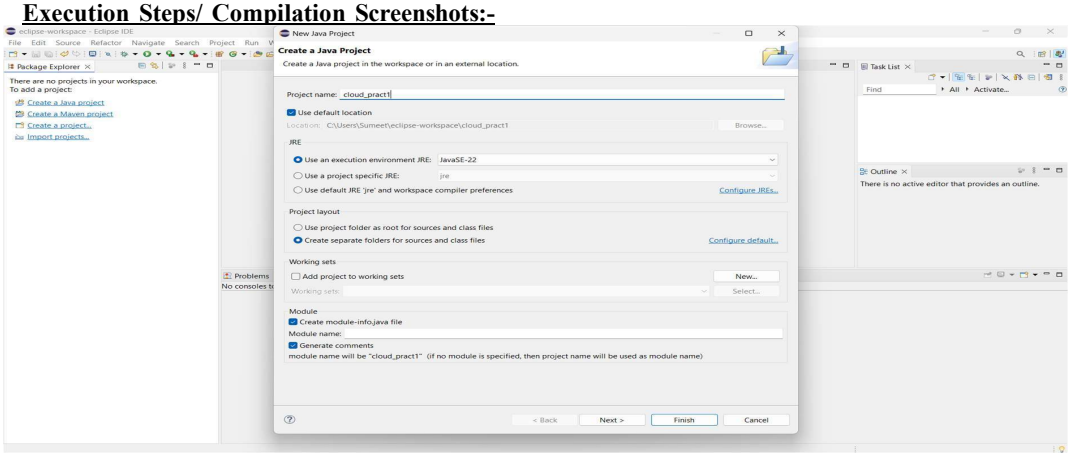
java.io.*; import

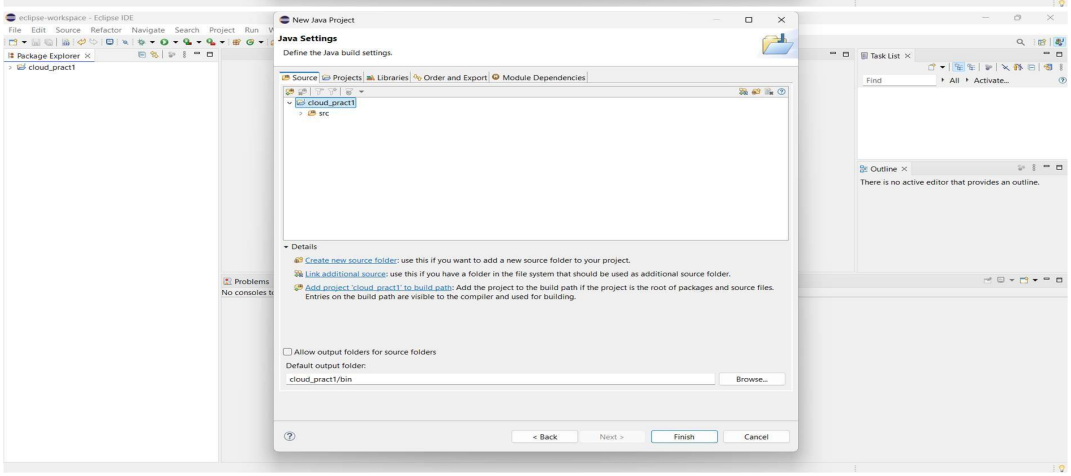
java.net.*; import

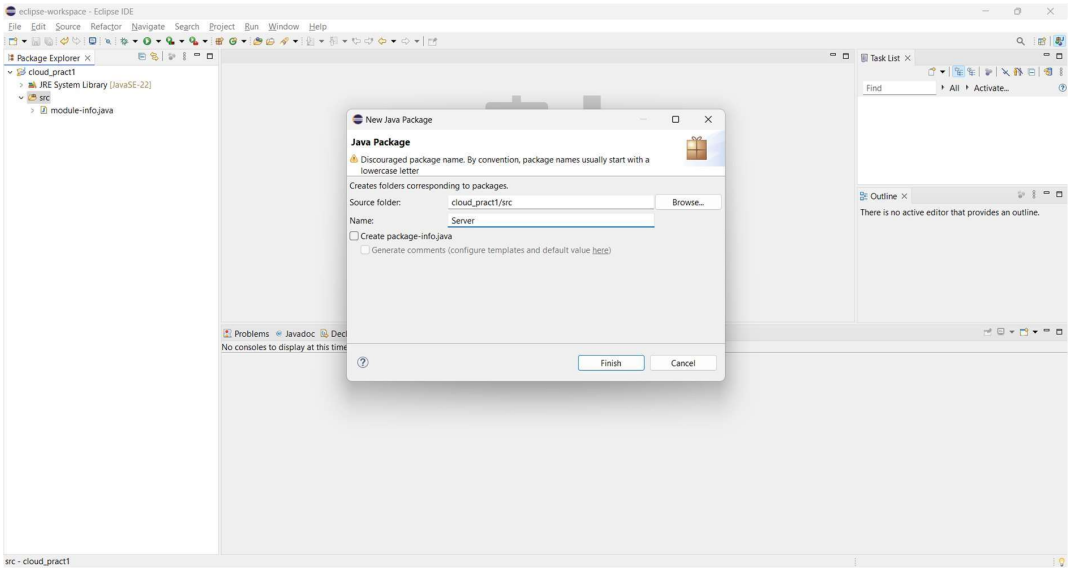
```
java.util.*;

public class Client {

    public static void main(String[] args) {

        Socket socket = null; BufferedReader

        input = null; PrintWriter output =

        null;

        try {

            socket = new Socket("localhost", 1234);

            System.out.println("Connected to the chat server");

            input = new BufferedReader(new InputStreamReader(System.in));

            output = new PrintWriter(socket.getOutputStream(), true); BufferedReader serverInput =

            new BufferedReader(new
InputStreamReader(socket.getInputStream())); Thread

            readThread = new Thread(() -> {

                 try {

                String message;

                while ((message = serverInput.readLine()) != null) { System.out.println(message);

              } } catch (IOException e) {

                e.printStackTrace();

               }

            }); readThread.start();

            String userMessage;

            while (true) {

                userMessage = input.readLine();

                if (userMessage.equalsIgnoreCase("exit")) {

                    break;

                } output.println(userMessage);
```

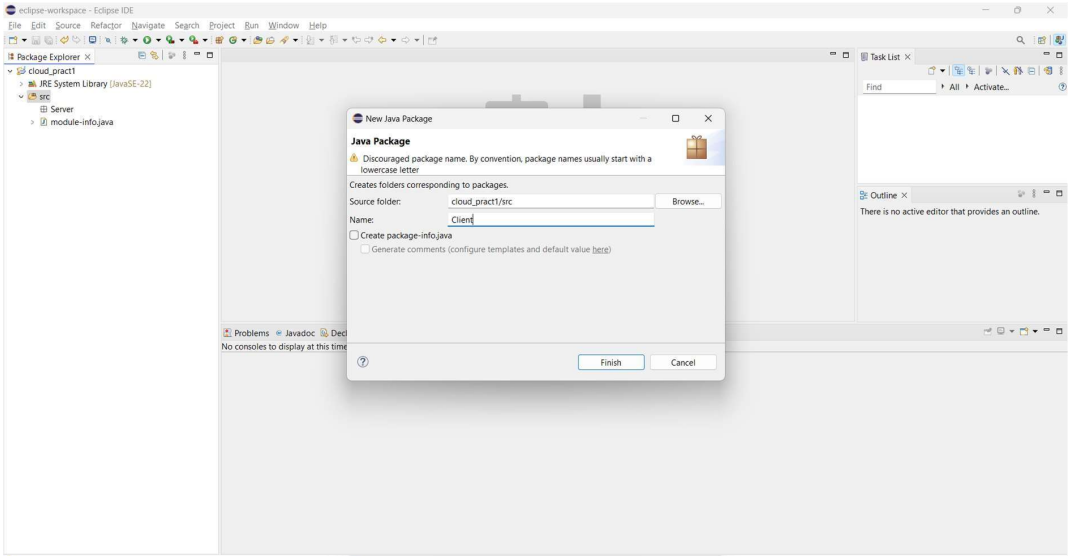
```
        }
      } catch (IOException e) {
        e.printStackTrace();
        } finally { try {
            if (socket != null) {
                socket.close();
            } System.exit(0);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
} }
```

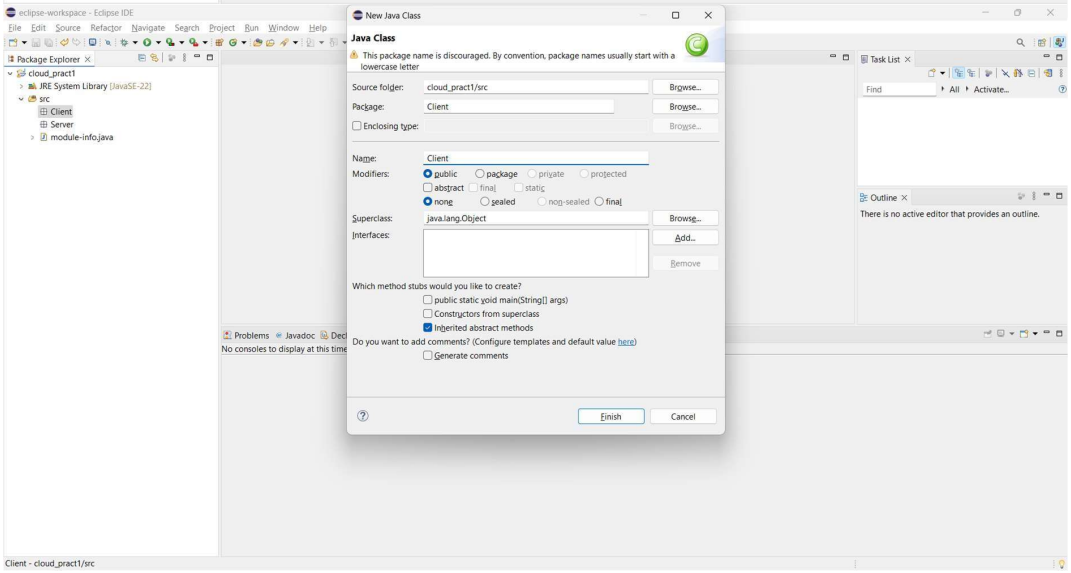
Execution Steps/ Compilation Screenshots:-



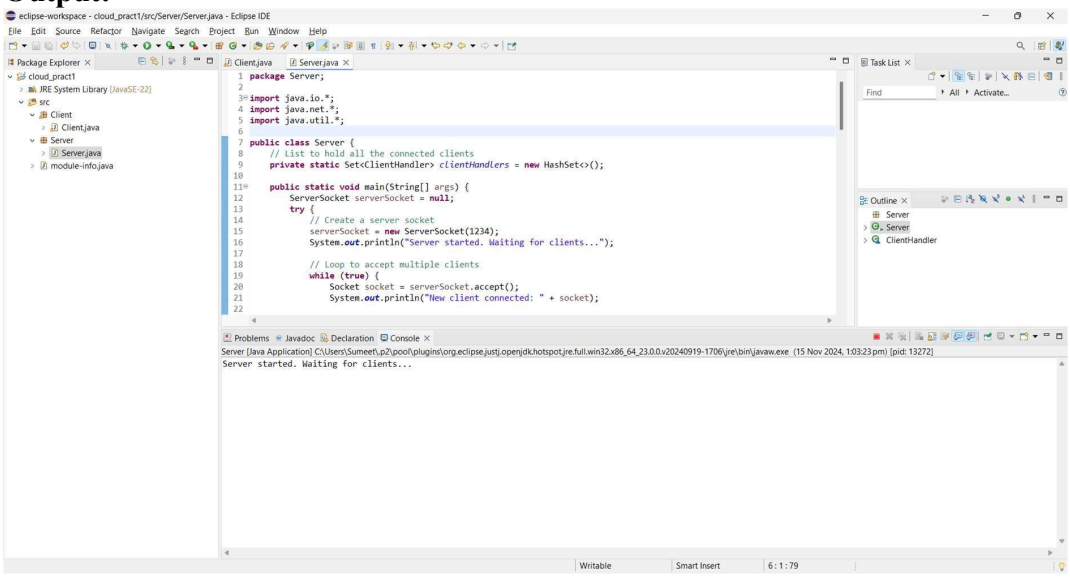


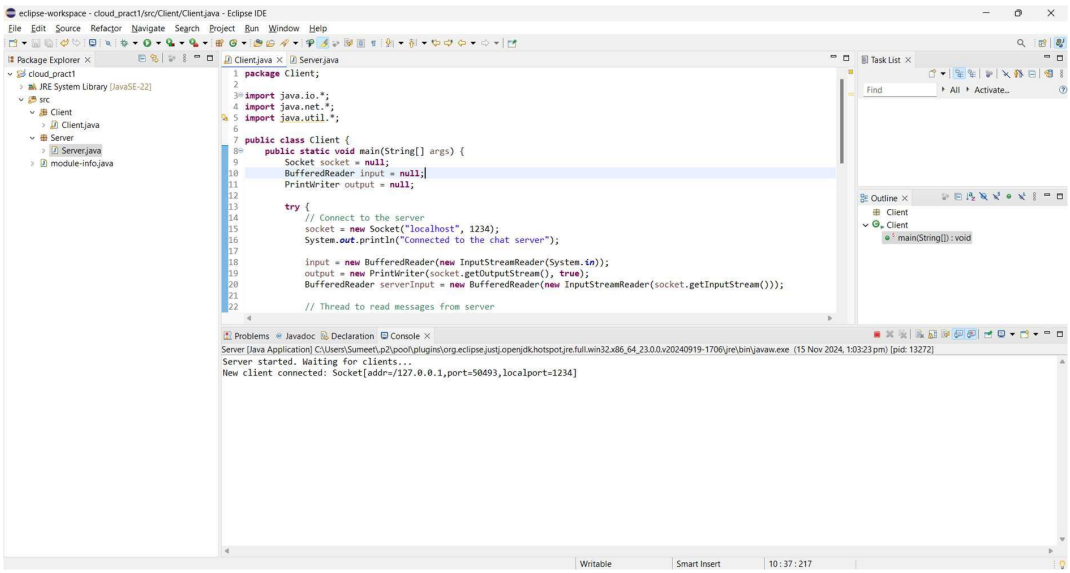

eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
JRE System Library [JavaSE-22]
src
module-info.java
New Java Package
Java Package
Discouraged package name. By convention, package names usually start with a lowercase letter
Creates folders corresponding to packages.
Source folder: cloud_pract1/src
Browse...
Name: Server
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel
Task List
Find
All
Activate...
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time
src - cloud_pract1


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
JRE System Library [JavaSE-22]
src
Server
module-info.java
New Java Package
Java Package
Discouraged package name. By convention, package names usually start with a lowercase letter
Creates folders corresponding to packages.
Source folder: cloud_pract1/src
Browse...
Name: Client
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel
Task List
Find
All
Activate...
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time


eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
JRE System Library [JavaSE-22]
src
Client
Server
module-info.java
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract1/src
Browse...
Package: Client
Browse...
Enclosing type:
Browse...
Name: Client
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Browse...
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Task List
Find
All
Activate...
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time
Client - cloud_pract1/src

Output:





Reference:-

https://www.geeksforgeeks.org/multi-threaded-chat-application-set-1/

https://stackoverflow.com/questions/56454187/multi-client-chat-application-in-java

Practical No. :- 2

Aim:- To implement a Server calculator using RPC concept. (Make use of datagram).

Theory:-

A **Server Calculator using RPC (Remote Procedure Call)** and Datagram is a client-server model where the client can request mathematical operations from the server, which executes the operations and sends the results back. RPC allows functions to be executed on a remote server as if they were local, abstracting the complexities of network communication. By using **Datagram** (UDP protocol), the application handles communication without establishing a persistent connection, making it lightweight and faster. The server listens for requests on a specific port. When a client sends a calculation request (e.g., addition, subtraction), the server performs the operation and returns the result as a response. In a **cloud computing** environment, this setup can be deployed on cloud instances, benefiting from cloud scalability and low latency for rapid calculations.

Algorithm/Concept:-

□ **Start Server:**

- Create a datagram socket and bind it to a specific port.

□ **Listen for Requests:**

- Wait for client requests using a datagram packet.

□ **Process Request:**

- Extract the operation type and operands from the received packet.
- Perform the requested arithmetic operation (e.g., addition, subtraction).

□ **Send Response:**

- Package the result into a new datagram packet.
- Send the result back to the client.

□ **Repeat or Stop:**

- Continue listening for new requests or terminate the server based on a condition.

Code/ Program:-

CalculatorServer:-

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

```
import java.net.DatagramSocket;
import java.net.InetAddress;
public class CalculatorServer {

public static void main(String[] args) {
DatagramSocket serverSocket = null;
try {
        serverSocket = new DatagramSocket(9876);
        System.out.println("Calculator Server is running...");
while (true) {

        byte[] receiveData = new byte[1024];

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

        serverSocket.receive(receivePacket);

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

        System.out.println("Received request: " + request);
        InetAddress clientAddress = receivePacket.getAddress();
        int clientPort = receivePacket.getPort();
        String[] tokens = request.split(" ");
        if (tokens.length != 3) {
        sendResponse(serverSocket, clientAddress, clientPort, "Invalid input          format");
        continue;}
        String operation = tokens[0].toLowerCase();
        double num1 = Double.parseDouble(tokens[1]);
        double num2 = Double.parseDouble(tokens[2]);
        double result = 0;
        String response;
        switch (operation) {
        case "add":
        result = num1 + num2; response = "Result: " + result; break;
        case "sub":
        result = num1 - num2; response = "Result: " + result; break;
        case "mul":
        result = num1 * num2; response = "Result: " + result; break;
        case "div":
        if (num2 == 0) {
                response = "Error: Division by zero";
        } else { result = num1 / num2;
                 response = "Result: " + result;}
        break;
        default:
        response = "Error: Invalid operation"; }
        sendResponse(serverSocket, clientAddress, clientPort, response);}
        } catch (Exception e) { e.printStackTrace();
```

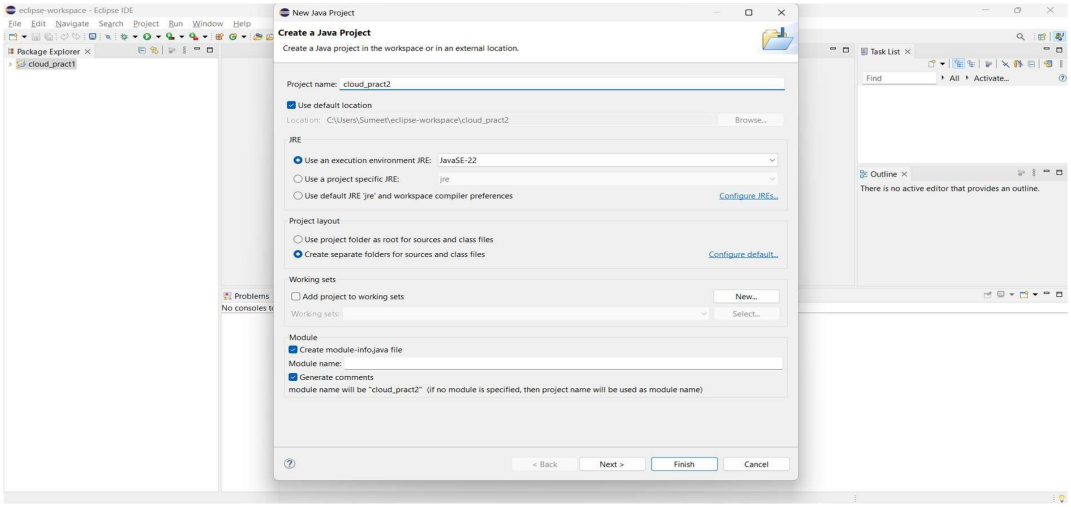
```
        } finally {
        if (serverSocket != null && !serverSocket.isClosed())
        { serverSocket.close();}
        }

}
private static void sendResponse(DatagramSocket socket, InetAddress address, int port, String
response) {

try {

byte[] responseData = response.getBytes();

DatagramPacket responsePacket = new DatagramPacket(responseData, responseData.length,
address, port);

socket.send(responsePacket);

} catch (Exception e) { e.printStackTrace();}}}
```

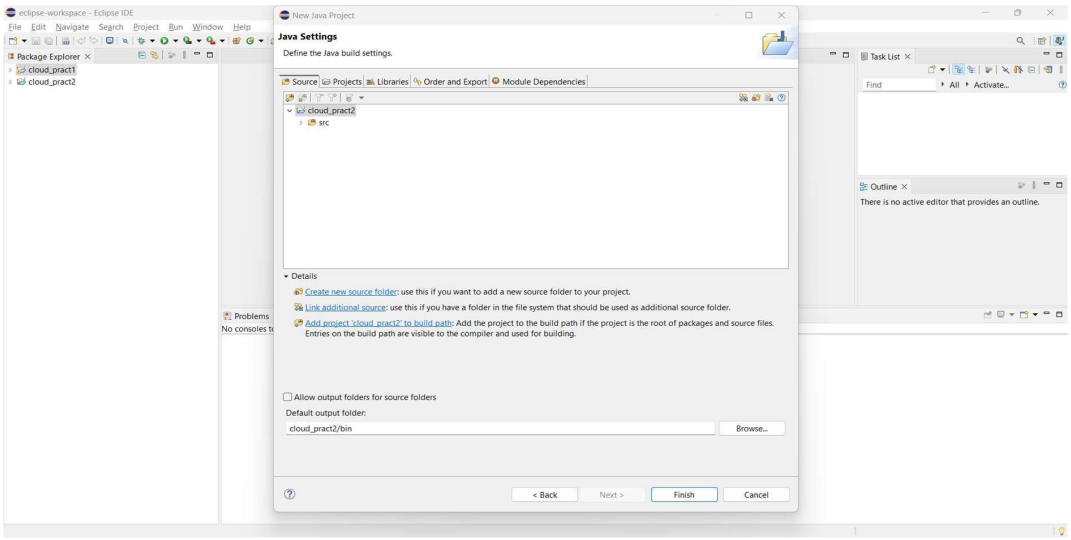
CalculatorClient:-

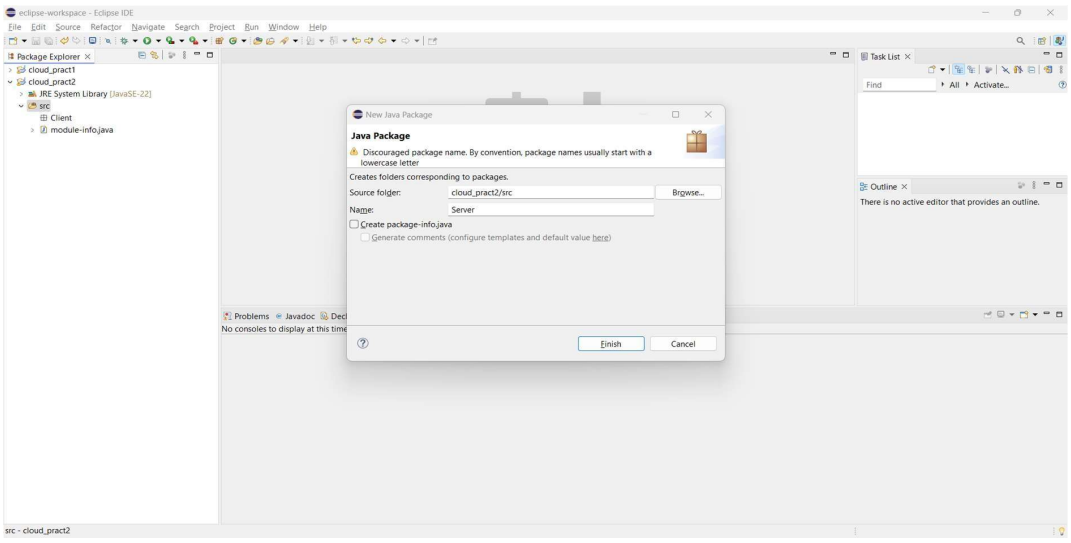
```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class CalculatorClient {

public static void main(String[] args)
        { DatagramSocket clientSocket = null;
                try {clientSocket = new DatagramSocket();
                        InetAddress serverAddress =
        InetAddress.getByName("localhost");
        Scanner scanner = new Scanner(System.in);
while (true) { System.out.println("Enter operation (e.g., add 5 3) or 'exit' to quit:");
String request = scanner.nextLine();
if (request.equalsIgnoreCase("exit")) { System.out.println("Exiting..."); break;
}
byte[] sendData = request.getBytes();

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

clientSocket.send(sendPacket); byte[] receiveData = new byte[1024];
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Server Response: " + response);
}
```

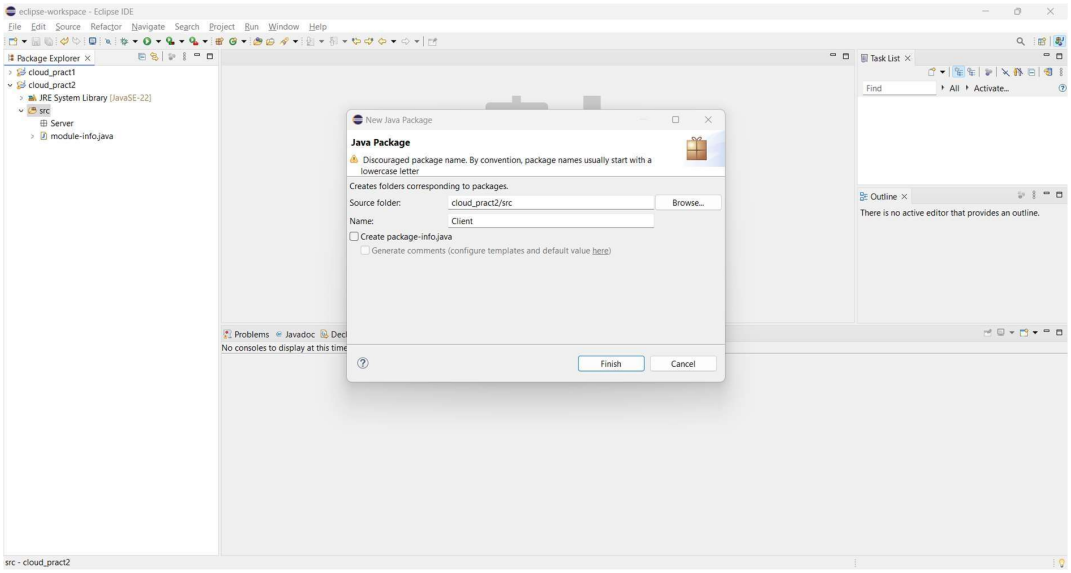
```
} catch (Exception e) { e.printStackTrace();
} finally {
if (clientSocket != null && !clientSocket.isClosed()) { clientSocket.close();
}}}}
```

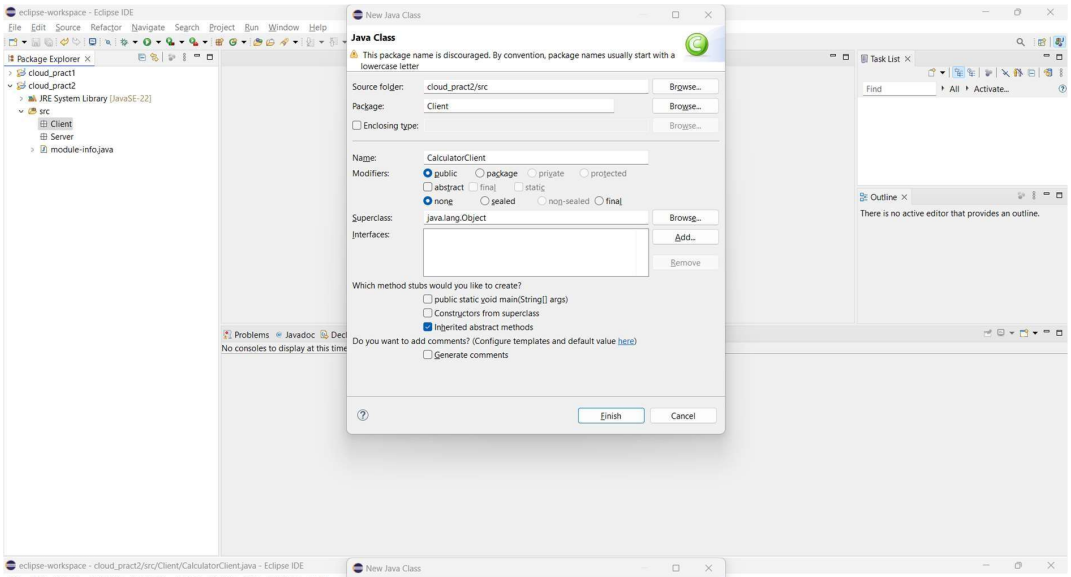
Execution/ Compilation Screenshot:-

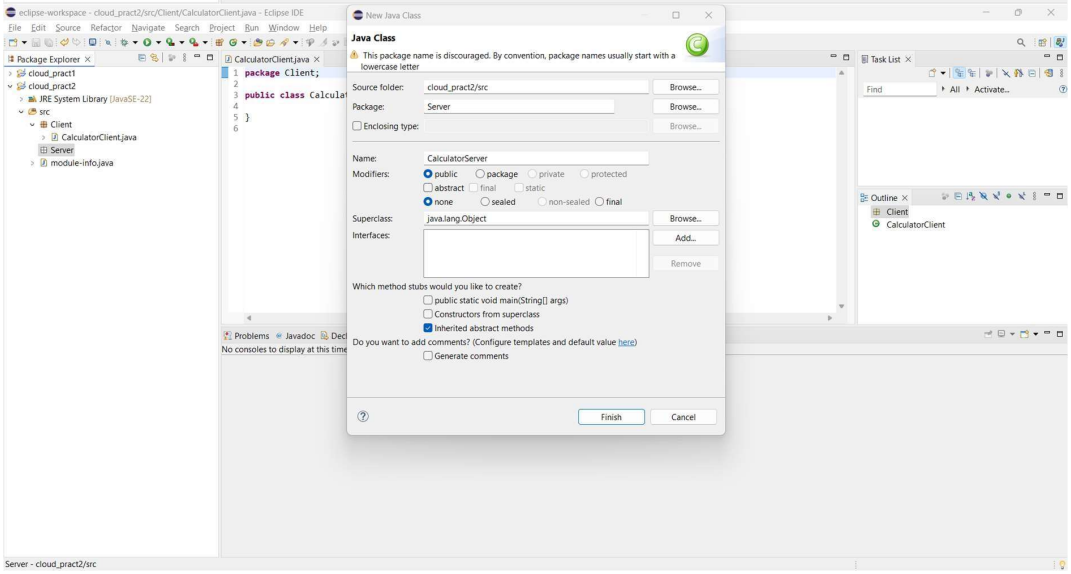




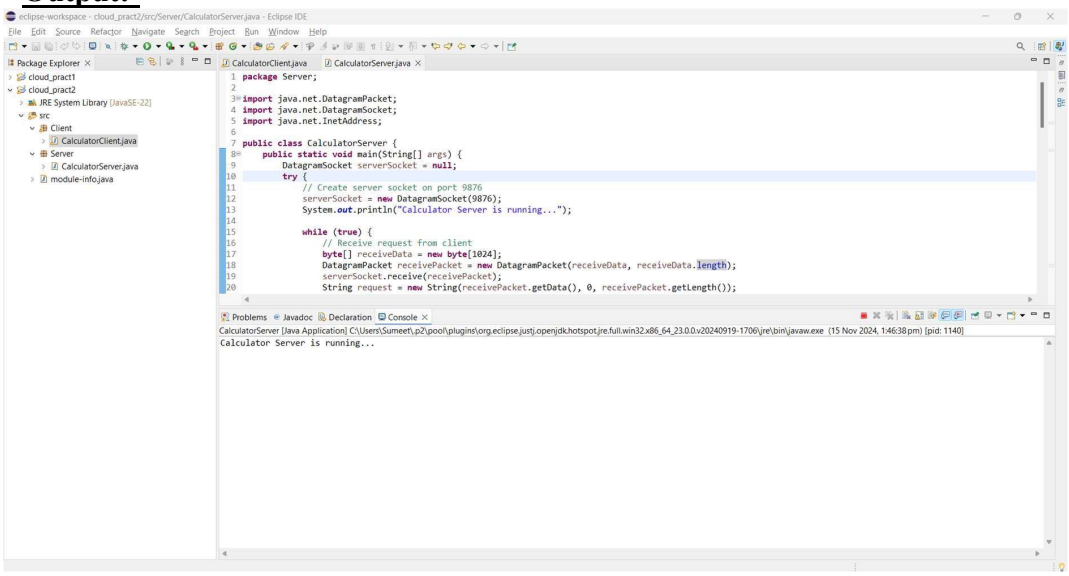


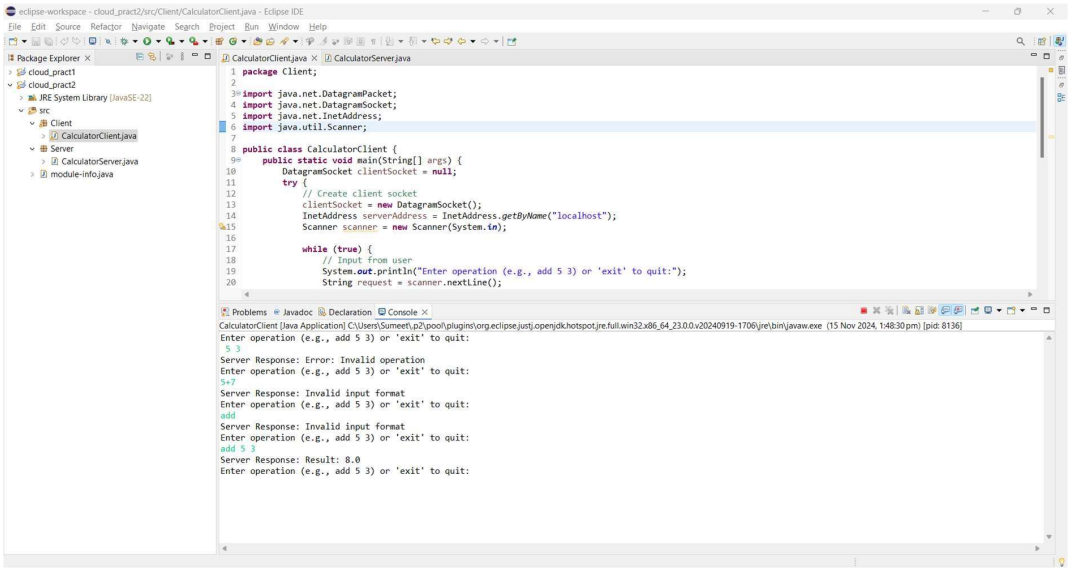

eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
cloud_pract2
JRE System Library [JavaSE-22]
src
Server
module-info.java
New Java Package
Java Package
Discouraged package name. By convention, package names usually start with a lowercase letter
Creates folders corresponding to packages.
Source folder: cloud_pract2/src
Browse...
Name: Client
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel
Task List
Find
All
Activate...
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time
src - cloud_pract2


eclipse-workspace - Eclipse IDE
Package Explorer
cloud_pract1
cloud_pract2
JRE System Library [JavaSE-22]
src
Client
Server
module-info.java
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract2/src
Package: Client
Enclosing type:
Name: CalculatorClient
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Outline
There is no active editor that provides an outline.
Problems
Javadoc
No consoles to display at this time


eclipse-workspace - cloud_pract2/src/Client/CalculatorClient.java - Eclipse IDE
Package Explorer
cloud_pract1
cloud_pract2
JRE System Library [JavaSE-22]
src
Client
CalculatorClient.java
Server
module-info.java
CalculatorClient.java
1 package Client;
3 public class Calcula
5 }
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract2/src
Package: Server
Enclosing type:
Name: CalculatorServer
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Outline
Client
CalculatorClient
Problems
Javadoc
No consoles to display at this time
Server - cloud_pract2/src

Output:-





Reference:-

https://www.geeksforgeeks.org/simple-calculator-via-udp-in-java/

https://www.geeksforgeeks.org/simple-calculator-via-udp-in-java/

Practical No. :- 3

Aim:- To implement a Date Time Server using RPC concept. (Make use of datagram).

Theory:-

A **Date Time Server** using the **RPC (Remote Procedure Call)** concept and **Datagram (UDP protocol)** allows clients to remotely request the current date and time from the server. RPC abstracts the network communication, letting clients invoke server-side functions as if they were local. Using **Datagram sockets**, which rely on the UDP protocol, ensures a connectionless and lightweight communication mechanism. The server listens for incoming requests on a specified port. When a client sends a request for the current date and time, the server retrieves the system's date and time, packages it into a response, and sends it back. This approach is efficient for simple queries like date and time, avoiding the overhead of establishing persistent connections. Deploying this on cloud infrastructure can leverage the cloud's ability to handle multiple simultaneous client requests with minimal latency.

Algorithm/ Concept:-

□ **Initialize Server:**

- Create a datagram socket and bind it to a designated port.

□ **Wait for Client Request:**

- Listen for incoming datagram packets from clients.

□ **Process Request:**

- Extract the request from the packet (e.g., request for current date and time).

□ **Fetch Date and Time:**

- Retrieve the current system date and time.

□ **Send Response:**

- Package the date and time information into a datagram packet.
- Send the packet back to the client.

□ **Repeat or Terminate:**

- Continue processing further requests or close the server based on conditions.

Code/ Program:-

DateTimeServer:-

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

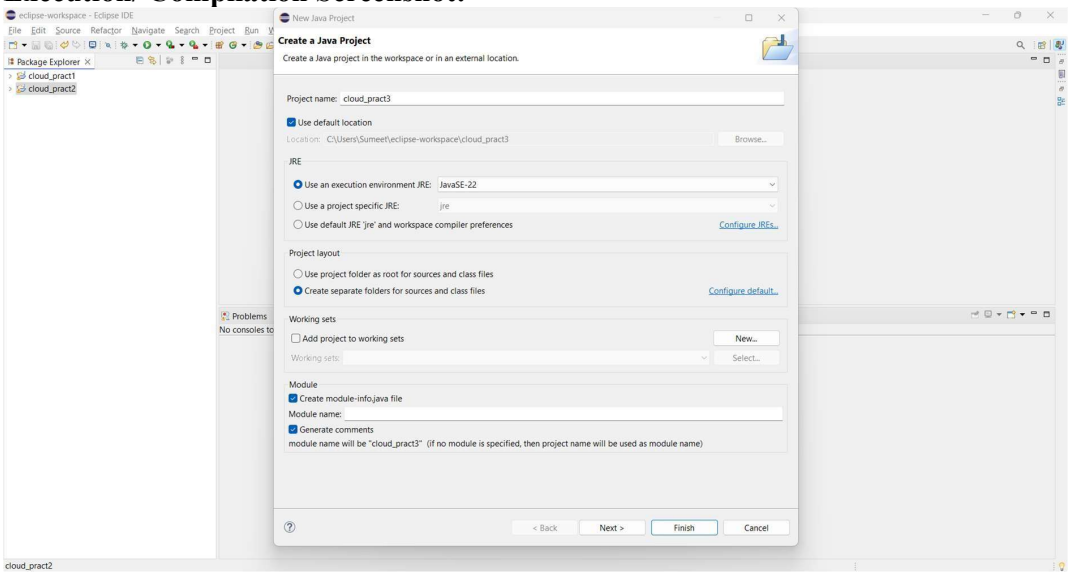
```
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateTimeServer {
public static void main(String[] args)
        { DatagramSocket serverSocket = null;
        try {serverSocket = new DatagramSocket(9876);
        System.out.println("Date Time Server is running...");
        while (true) {
        byte[] receiveData = new byte[1024];
        DatagramPacket receivePacket = new DatagramPacket(receiveData,
        receiveData.length); serverSocket.receive(receivePacket);
        String request = new String(receivePacket.getData(), 0,
        receivePacket.getLength());
        System.out.println("Received request: " + request); InetAddress      clientAddress =
receivePacket.getAddress(); int clientPort =  receivePacket.getPort();
        String response;
        if (request.equalsIgnoreCase("datetime")) {
        SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date date = new Date();
        response = "Current Date and Time: " + formatter.format(date);
        } else {
        response = "Invalid request. Send 'datetime' to get the current date and       time.";
        }
        byte[] responseData = response.getBytes();
        DatagramPacket responsePacket = new DatagramPacket(responseData,
        responseData.length, clientAddress, clientPort);
        serverSocket.send(responsePacket);
        }} catch (Exception e) { e.printStackTrace();
        } finally {
        if (serverSocket != null && !serverSocket.isClosed())         {
serverSocket.close();}}}}
```

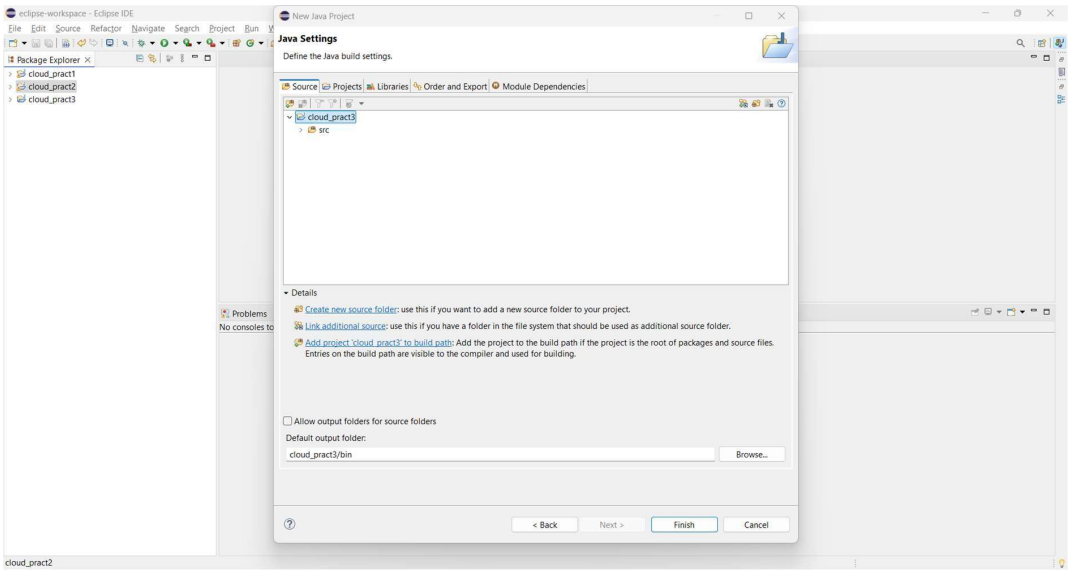
DateTimeClient:-

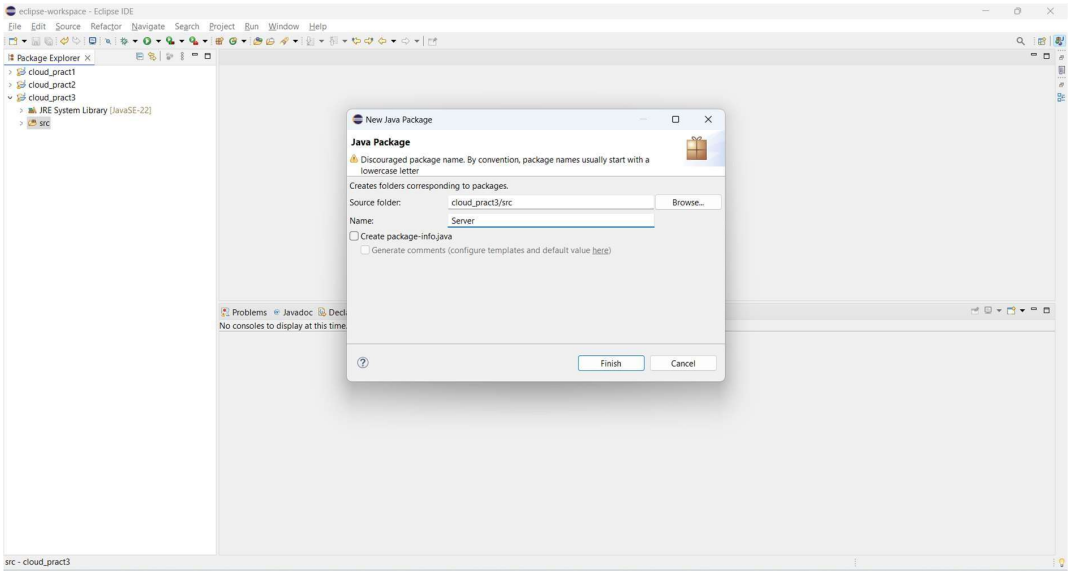
```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class DateTimeClient {
public static void main(String[] args) { DatagramSocket clientSocket = null; try {
clientSocket = new DatagramSocket();
InetAddress serverAddress = InetAddress.getByName("localhost"); Scanner scanner = new
Scanner(System.in);
System.out.println("Enter 'datetime' to get the current date and time or 'exit' to quit:");
String request = scanner.nextLine();
if (request.equalsIgnoreCase("exit")) { System.out.println("Exiting..."); return;}
byte[] sendData = request.getBytes();
DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
```

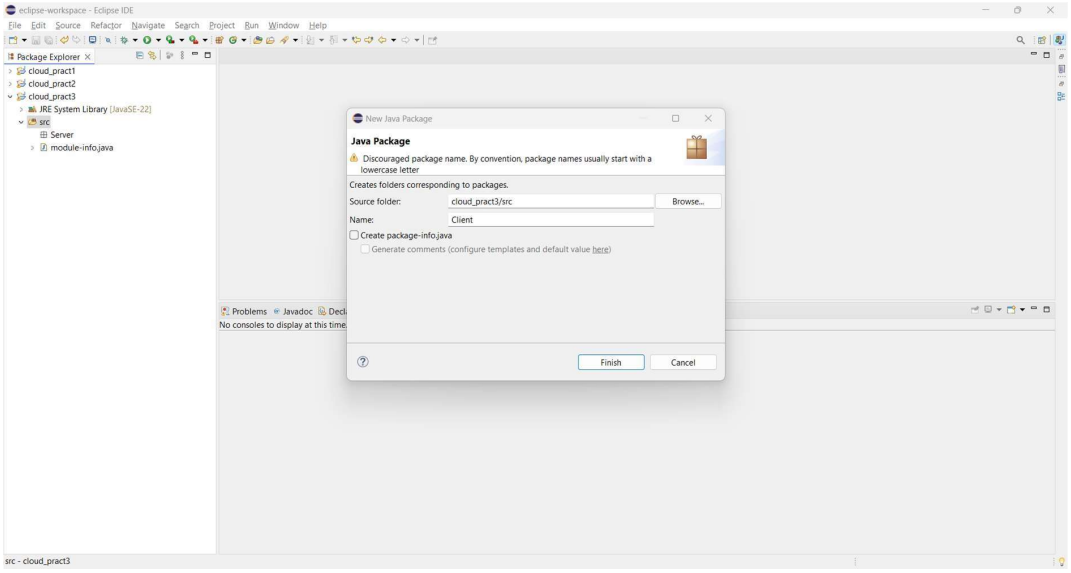
serverAddress, 9876);
clientSocket.send(sendPacket); byte[] receiveData = new byte[1024];
DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
clientSocket.receive(receivePacket);
String response = new String(receivePacket.getData(), 0, receivePacket.getLength());
System.out.println("Server Response: " + response);
} catch (Exception e) { e.printStackTrace();
} finally {
if (clientSocket != null && !clientSocket.isClosed()) { clientSocket.close();}}}}

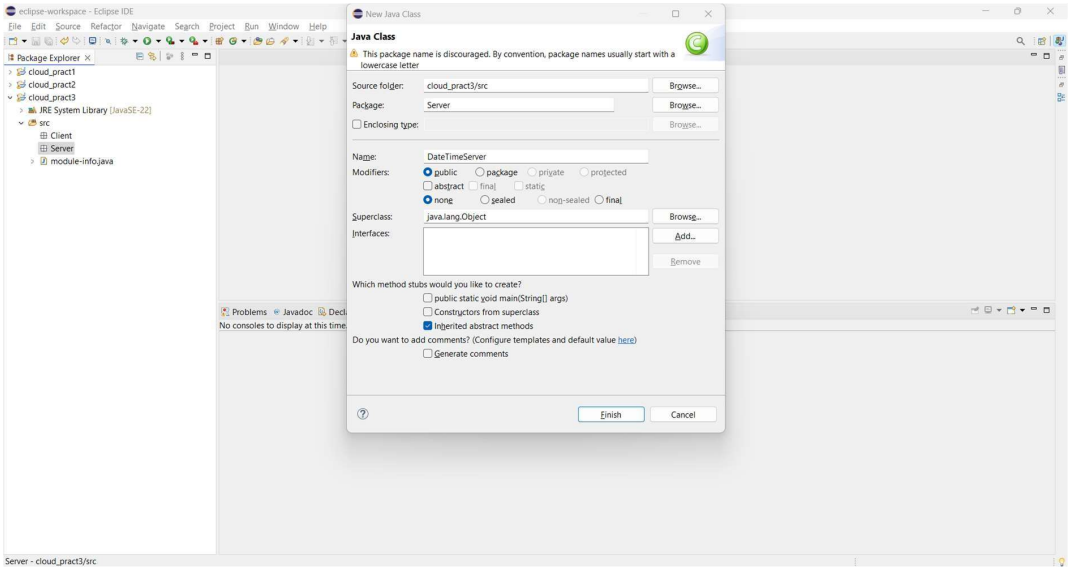
Execution/ Compilation Screenshot:-

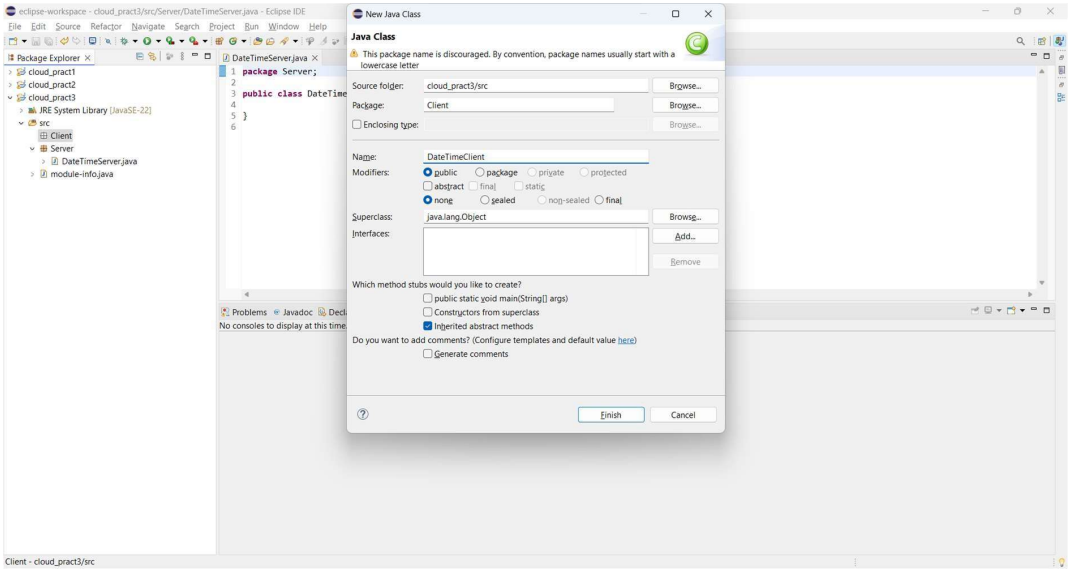




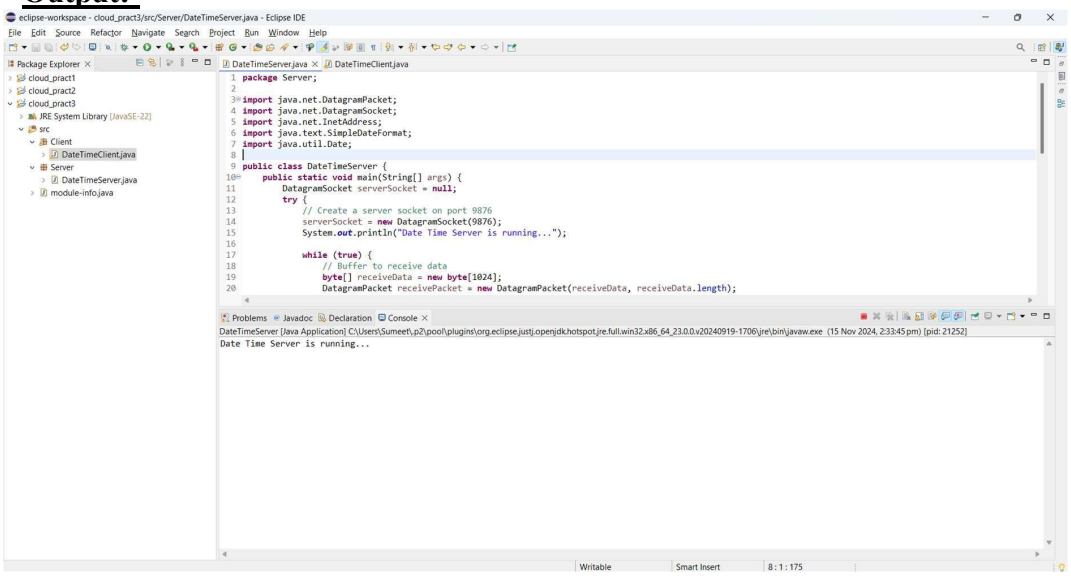

eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
cloud_pract2
cloud_pract3
JRE System Library [JavaSE-22]
src
New Java Package
Java Package
Discouraged package name. By convention, package names usually start with a lowercase letter
Creates folders corresponding to packages.
Source folder: cloud_pract3/src
Browse...
Name: Server
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel
Problems
Javadoc
No consoles to display at this time.
src - cloud_pract3

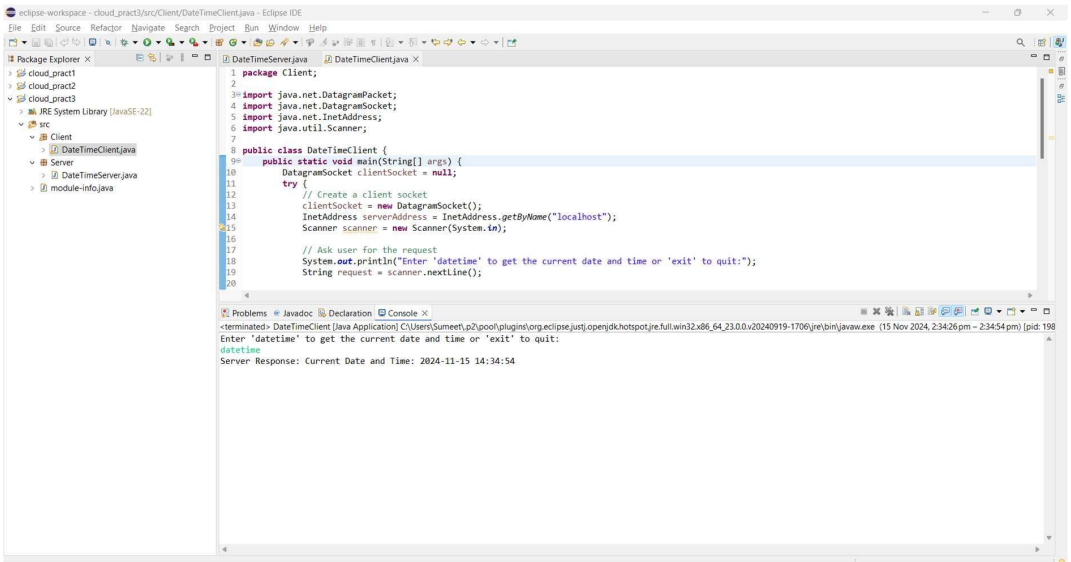

eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
cloud_pract2
cloud_pract3
JRE System Library [JavaSE-22]
src
Server
module-info.java
New Java Package
Java Package
Discouraged package name. By convention, package names usually start with a lowercase letter
Creates folders corresponding to packages.
Source folder: cloud_pract3/src
Browse...
Name: Client
Create package-info.java
Generate comments (configure templates and default value here)
Finish
Cancel
Problems
Javadoc
No consoles to display at this time.
src - cloud_pract3

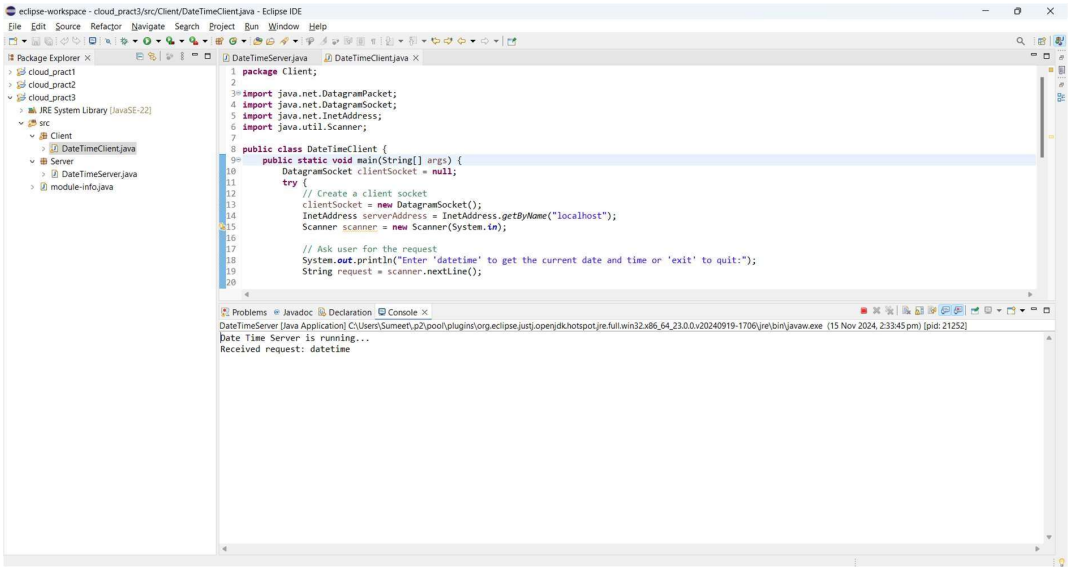

eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
cloud_pract1
cloud_pract2
cloud_pract3
JRE System Library [JavaSE-22]
src
Client
Server
module-info.java
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract3/src
Browse...
Package: Server
Enclosing type:
Name: DateTimeServer
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Add...
Remove
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Problems
Javadoc
No consoles to display at this time.
Server - cloud_pract3/src


New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract3/src
Package: Client
Enclosing type:
Name: DateTimeClient
Modifiers: public package private protected abstract final static none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

Output:-


DateTimeServer [Java Application] C:\Users\Sumeet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.exe (15 Nov 2024, 2:33:45 pm) [pid: 21252]
Date Time Server is running...


Enter 'datetime' to get the current date and time or 'exit' to quit:
datetime
Server Response: Current Date and Time: 2024-11-15 14:34:54



Reference:-

https://www.geeksforgeeks.org/working-udp-datagramsockets-java/

https://stackoverflow.com/questions/51497871/is-it-possible-to-receive-datagrampackets-from-many-clients-in-one-socket-in-the

Practical No. :- 4

Aim:- To retrieve day, time and date function from server to client. This program should display server day, time and date. (Use Concept of JDBC and RMI for accessing multiple data access objects)

Theory:-

This program allows a **client** to retrieve the **current day, time, and date** from a **server** using **JDBC (Java Database Connectivity)** and **RMI (Remote Method Invocation)**. The server acts as a remote service that exposes a method to fetch system information such as the current day, time, and date.
JDBC is used to interact with the database if additional data retrieval from a database is required. RMI facilitates the communication between the client and the server by enabling the client to call methods on the remote server object as if it were a local method. RMI abstracts the complexity of network communication and provides a straightforward mechanism for remote procedure calls. The **cloud computing** implementation of this application can scale the server to handle multiple client requests, ensuring high availability and reliability.

Algorithm/ Concept:-

□ **Server-Side Setup:**

- **Create a Remote Interface:**
  - Define a remote interface with methods like getDay(), getTime(), and getDate().
- **Implement the Interface:**
  - Implement the interface in a server class, where each method retrieves the current day, time, and date.
  - Use JDBC if needed to retrieve additional data from a database (e.g., historical data related to dates/times).
- **Bind the Remote Object:**
  - Create an instance of the server class and bind it to the RMI registry using Naming.rebind().

□ **Client-Side Setup:**

- **Look up Remote Object:**
  - Use Naming.lookup() to get the reference to the remote object from the RMI registry.
- **Invoke Methods:**
  - Call the remote methods getDay(), getTime(), and getDate() to fetch the day, time, and date from the server.

□ **Display Results:**

- The client displays the retrieved values (day, time, and date).

Code/ Program:-

Database

```
CREATE TABLE server_time (
id INT PRIMARY KEY
AUTO_INCREMENT, day
VARCHAR(20),
time TIME,
date DATE );
INSERT INTO server_time (day, time,
date) VALUES ('Monday', '10:30:00',
'2024-11-15');
```

Server

```
import
java.rmi.RemoteException;
import
java.rmi.registry.LocateRegistry
; import
java.rmi.registry.Registry;

public class Server {
  public static void main(String[] args) {

    try {
      ServerTimeService service = new
      ServerTimeServiceImpl(); Registry registry =
      LocateRegistry.createRegistry(1099);
      registry.rebind("ServerTimeService", service);
      System.out.println("Server is running");

    } catch (RemoteException e)
      { e.printStackTrace();}
  }
}
```

Client

```
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", 1099);
      ServerTimeService service = (ServerTimeService)
      registry.lookup("ServerTimeService");
      System.out.println("Day: " +
      service.getDay()); System.out.println("Time:
      " + service.getTime());
      System.out.println("Date: " +
      service.getDate());
```

```
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

[Interface] ServerTimeService

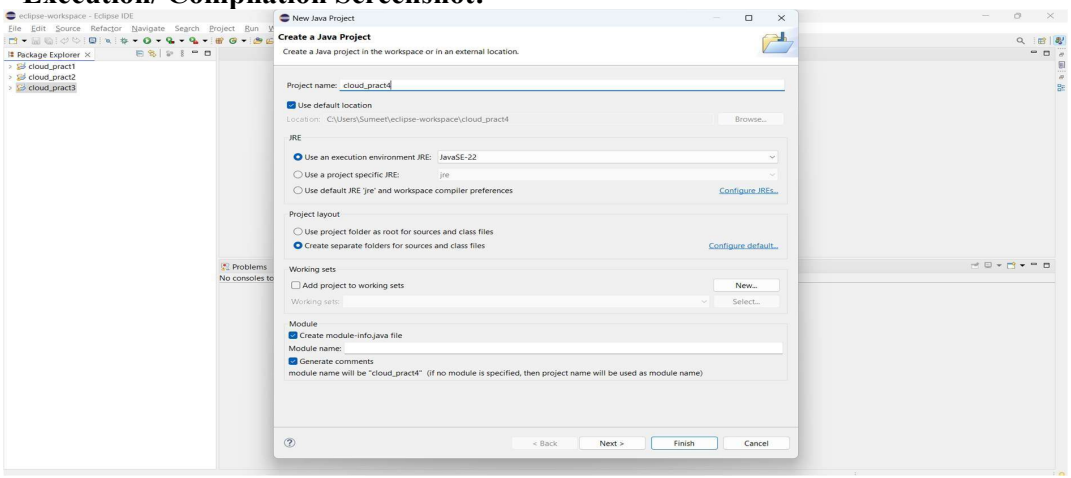
```
import java.rmi.Remote;
import  java.rmi.RemoteException;
public interface ServerTimeService extends Remote {
String getDay() throws RemoteException;
String getTime() throws RemoteException;
String getDate() throws RemoteException;
}
```

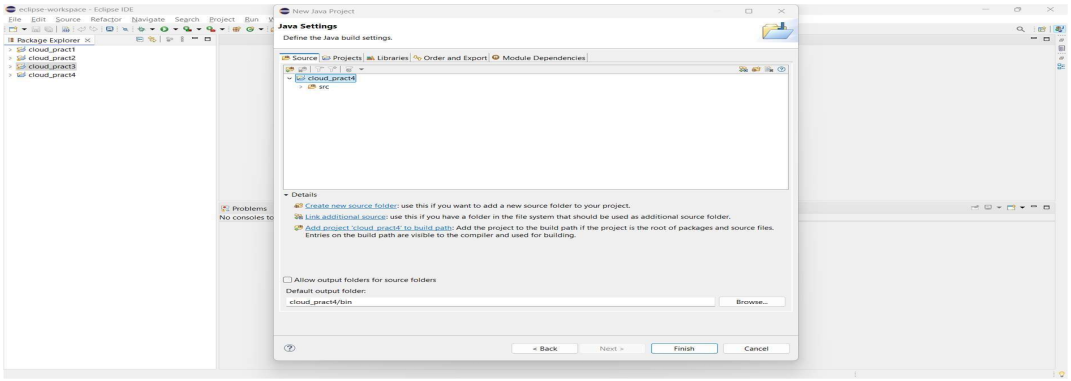
ServerTimeServiceImpl

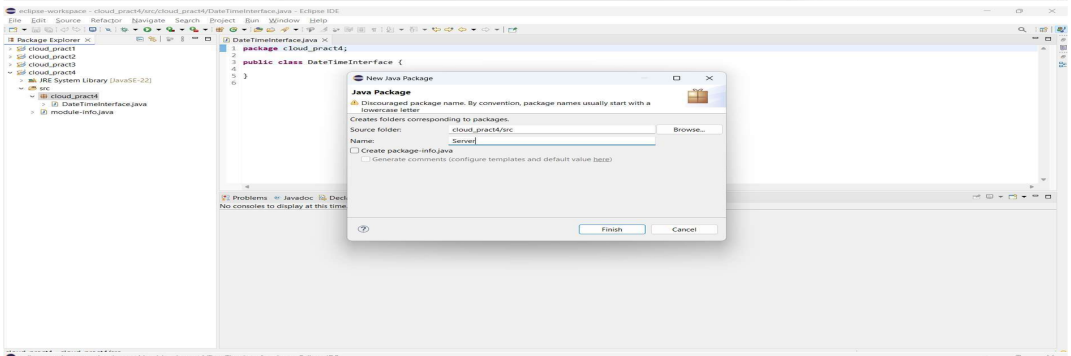
```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.*;
public class ServerTimeServiceImpl extends UnicastRemoteObject implements
ServerTimeService {
private static final String DB_URL = "jdbc:mysql://localhost:3306/cloud";
private static final String DB_USER = "root";
private static final String DB_PASSWORD = "gnims";
protected ServerTimeServiceImpl() throws RemoteException {
super();
}
private Connection getConnection() throws SQLException{
return DriverManager.getConnection(DB_URL,DB_USER,DB_PASSWORD);
}
@Override
public String getDay() throws RemoteException {
return fetchColumn("day");
}
@Override
public String getTime() throws RemoteException {
return fetchColumn("time");
}
@Override
public String getDate() throws RemoteException {
return fetchColumn("date");
}
private String fetchColumn(String columnName){
String query = "SELECT "+columnName+" FROM server_time";
try(Connection con = getConnection(); Statement statement = con.createStatement();
ResultSet rs = statement.executeQuery(query)){
if (rs.next()){
return rs.getString(columnName);
}
}catch (SQLException e){ e.printStackTrace();
```

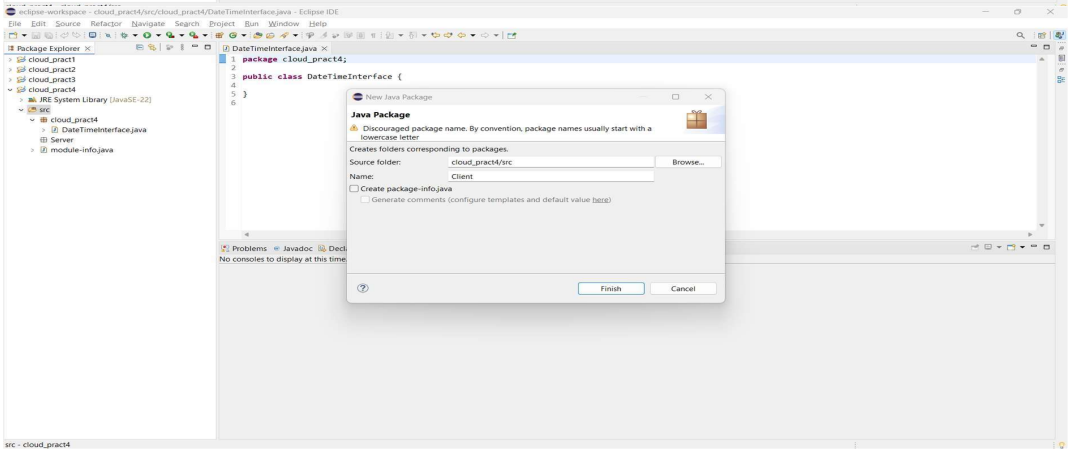
}
return "Error retrieving " + columnName;}}

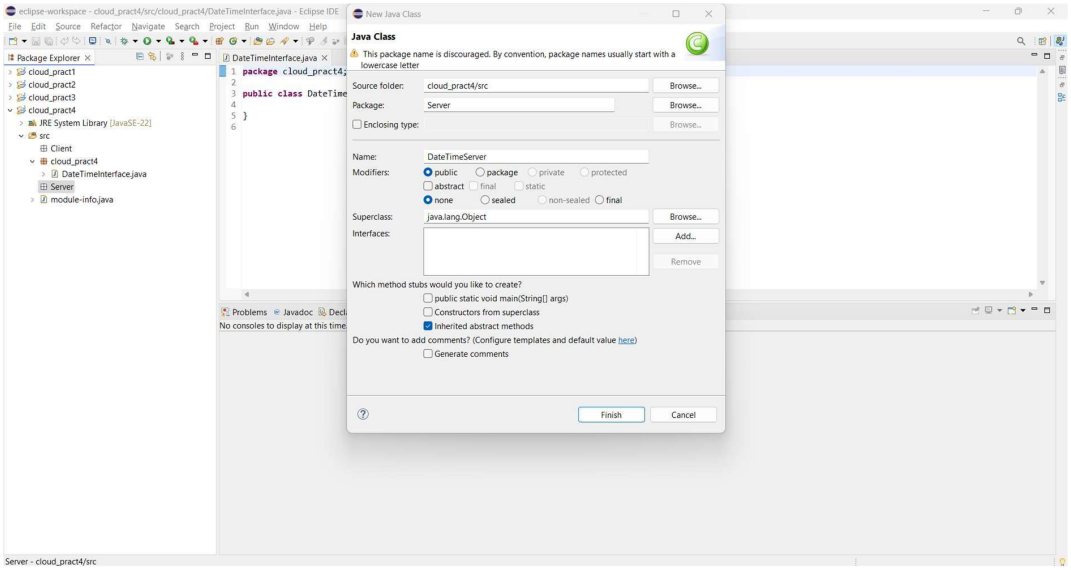
Execution/ Compilation Screenshot:-

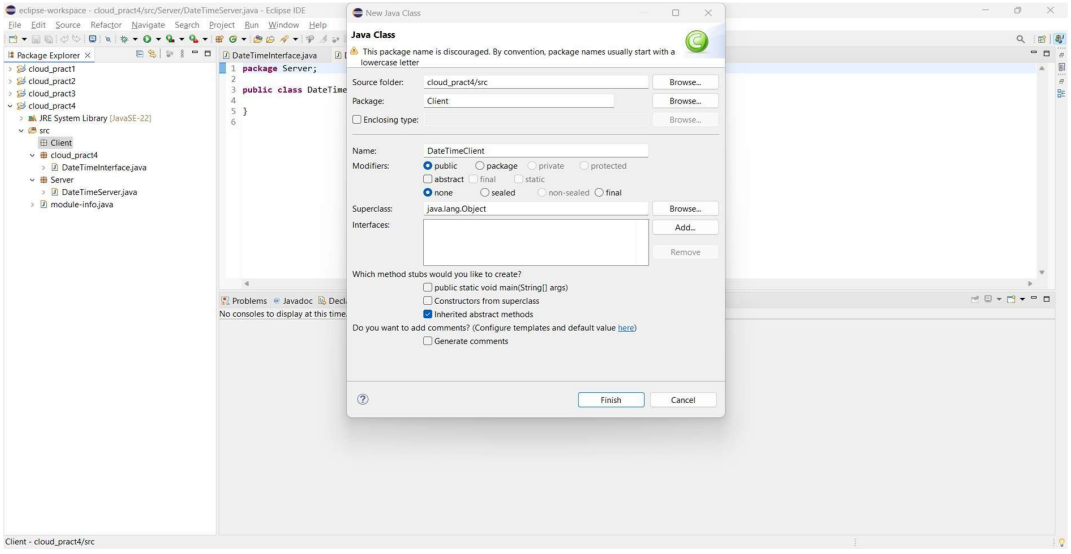




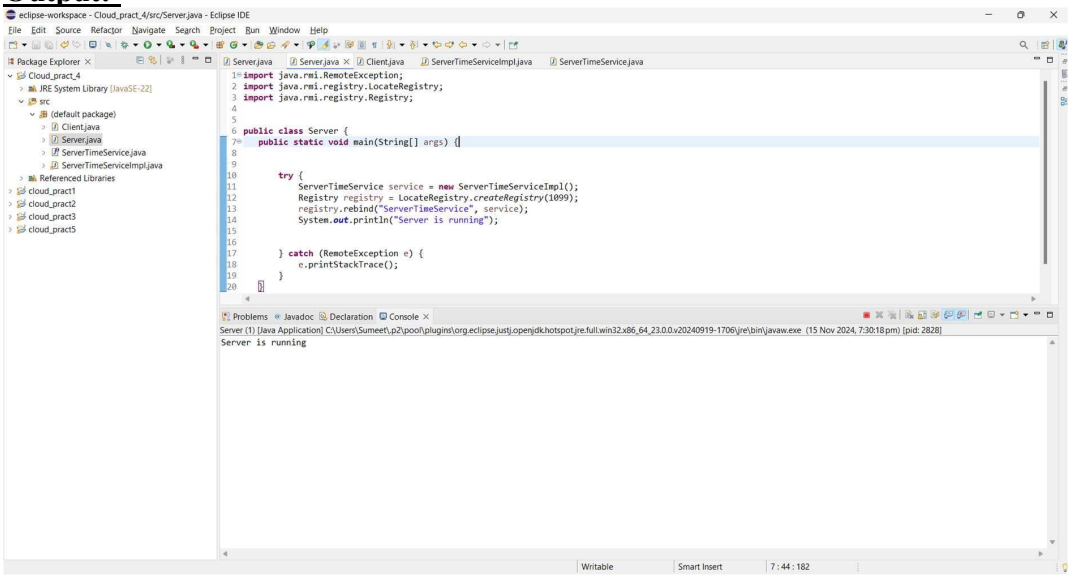


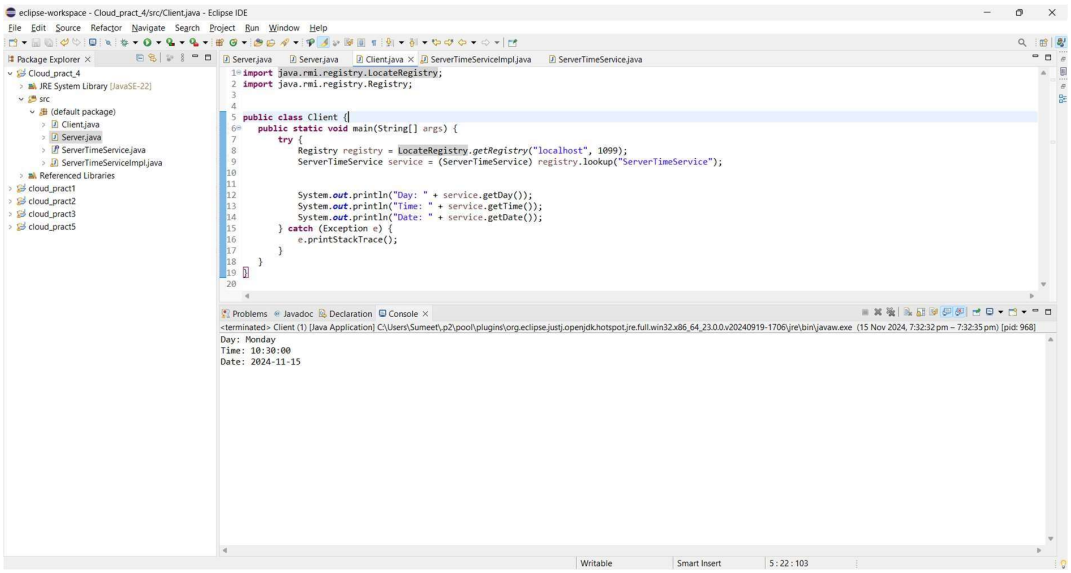


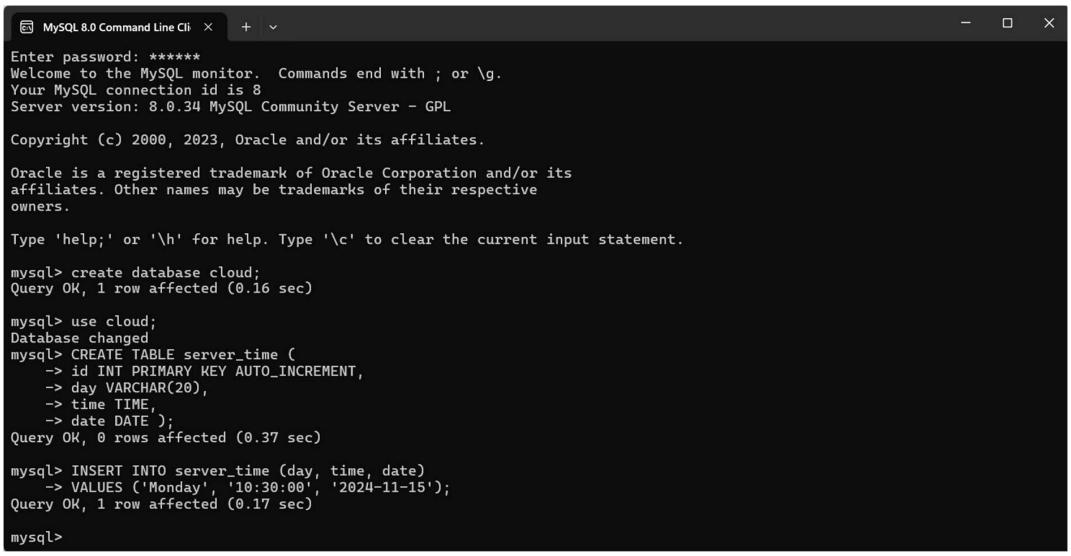

eclipse-workspace - cloud_pract4/src/cloud_pract4/DateTimeInterface.java - Eclipse IDE
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract4/src
Package: Server
Enclosing type:
Name: DateTimeServer
Modifiers: public package private protected abstract final static none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Server - cloud_pract4/src


eclipse-workspace - cloud_pract4/src/Server/DateTimeServer.java - Eclipse IDE
New Java Class
Java Class
This package name is discouraged. By convention, package names usually start with a lowercase letter
Source folder: cloud_pract4/src
Package: Client
Enclosing type:
Name: DateTimeClient
Modifiers: public package private protected abstract final static none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
Client - cloud_pract4/src

Output:-


eclipse-workspace - Cloud_pract_4/src/Server.java - Eclipse IDE
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Server {
public static void main(String[] args) {
try {
ServerTimeService service = new ServerTimeServiceImpl();
Registry registry = LocateRegistry.createRegistry(1099);
registry.rebind("ServerTimeService", service);
System.out.println("Server is running");
} catch (RemoteException e) {
e.printStackTrace();
}
}
Server is running





Reference:-

https://www.javatpoint.com/RMI
https://stackoverflow.com/questions/11296606/java-jdbc-dates-consistently-two-days-off

Practical No. :-5

Aim:- The client should provide an equation to the server through an interface. The server will solve the expression given by the client.

Theory:-

In this program, a **client** sends a mathematical equation to a **server** through a remote interface using **RMI (Remote Method Invocation)**. The server receives the equation, processes it, and sends back the solution. The RMI framework allows the client to invoke methods on the server remotely, abstracting the complexities of network communication. This setup involves creating a remote interface with a method to accept the equation as a string, evaluate it, and return the result. The server uses a parsing mechanism (e.g., JavaScript engine or custom logic) to solve the equation. Deploying this in a **cloud computing** environment can scale the server to handle multiple simultaneous requests, ensuring high availability and responsiveness. This approach is efficient for distributed applications where clients need computational services from a central server, utilizing the cloud's computational resources.

Algorithm/ Concept:-

□ **Server-Side Setup:**

- Define a remote interface with a method solveEquation(String equation).
- Implement the interface in a server class, where the method parses and evaluates the equation.
- Bind the server object to the RMI registry.

□ **Client-Side Setup:**

- Look up the remote object using the RMI registry.
- Provide an equation as input through the client interface.
- Invoke the solveEquation() method with the provided equation.

□ **Solve Equation:**

- The server parses the input string and evaluates the equation.
- It computes the result and returns it to the client.

□ **Display Result:**

- The client receives and displays the solution.

□ **Termination:**

- Close the connection and terminate the client-server session.

Code/ Program:-

[Interface] EquationInterface

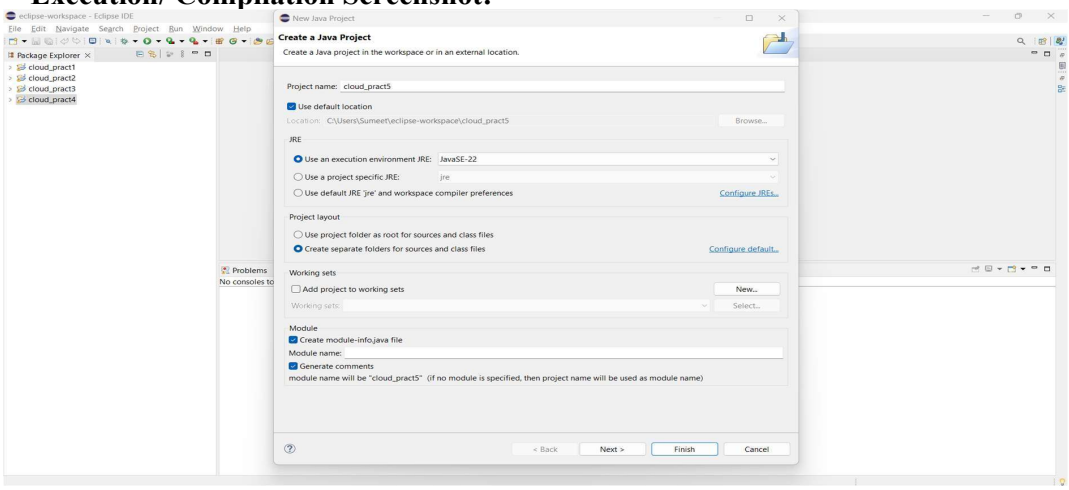
```
package cloud_pract5;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface EquationInterface extends Remote {
double solveEquation(String equation) throws RemoteException;
}
```

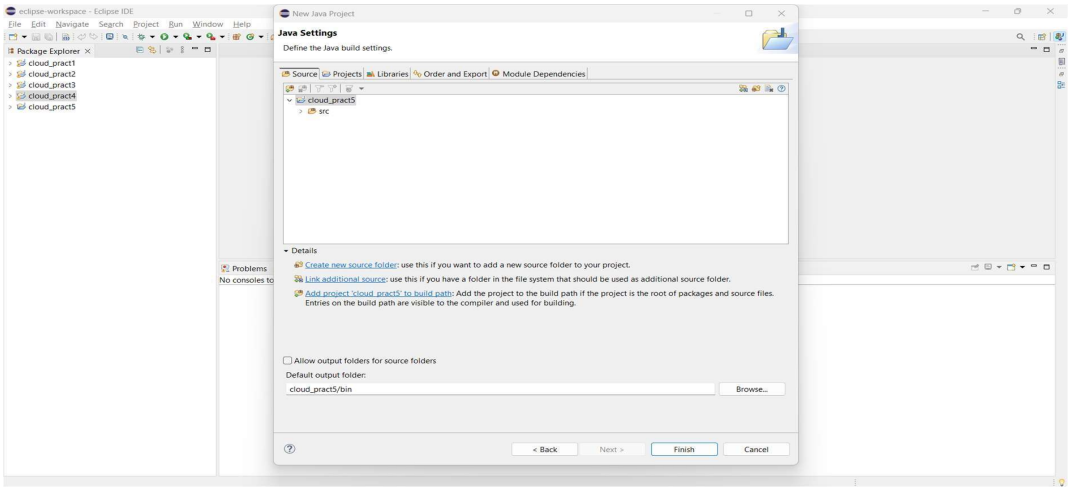
EquationServer

```
package cloud_pract5;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry; import
java.rmi.server.UnicastRemoteObject; import net.objecthunter.exp4j.Expression;
import net.objecthunter.exp4j.ExpressionBuilder;
import net.objecthunter.exp4j.tokenizer.UnknownFunctionOrVariableException;
public class EquationServer extends UnicastRemoteObject implements
EquationInterface {protected EquationServer() throws RemoteException {
super();
}@Override
public double solveEquation(String equation) throws RemoteException {
try {
Expression expression = new ExpressionBuilder(equation).build();
return expression.evaluate();
} catch (UnknownFunctionOrVariableException e) { System.err.println("Error
evaluating expression: " + e.getMessage()); throw new RemoteException("Invalid
equation: " + e.getMessage());
} catch (Exception e) {
System.err.println("General error: " + e.getMessage());
throw new RemoteException("Error evaluating the equation.");}}
public static void main(String[] args) {
try {
LocateRegistry.createRegistry(2024); EquationServer server = new EquationServer();
Naming.rebind("rmi://localhost:2024/EquationService", server);
System.out.println("Equation Server is running on port 2024...");
} catch (Exception e) {
System.err.println("Server exception: " + e.getMessage()); e.printStackTrace();}}}
EquationClient
package cloud_pract5;
import java.rmi.Naming;
import java.util.Scanner;
public class EquationClient {
public static void main(String[] args) {
try {
EquationInterface equationService = (EquationInterface)
Naming.lookup("rmi://localhost:2024/EquationService");
Scanner scanner = new Scanner(System.in);
System.out.println("Enter a mathematical expression (e.g., 3 + 4 * 2): "); String
```

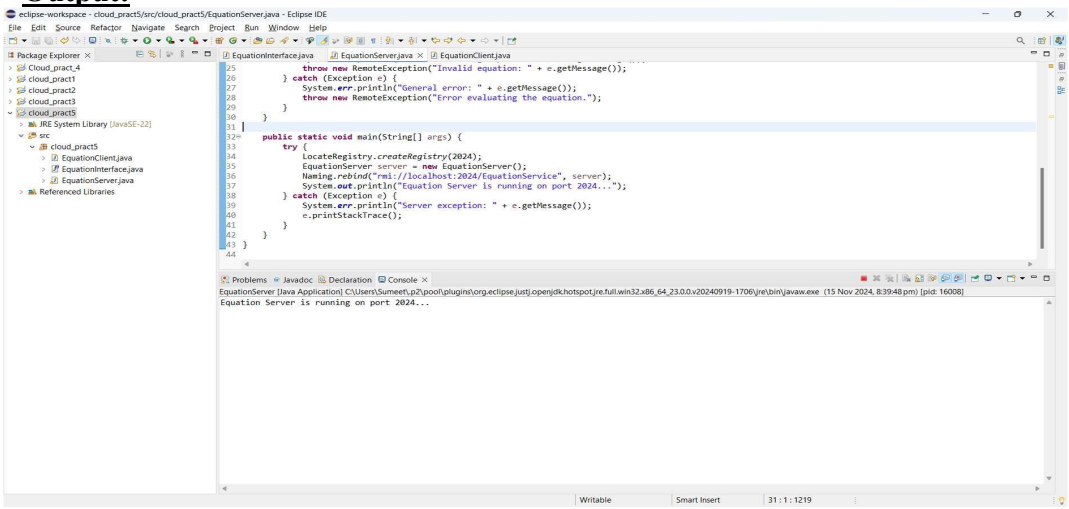
```
equation = scanner.nextLine();
double result = equationService.solveEquation(equation); System.out.println("Result:
" + result);
scanner.close();
} catch (Exception e) {
System.err.println("Client exception: " + e.getMessage()); e.printStackTrace();}}}
```

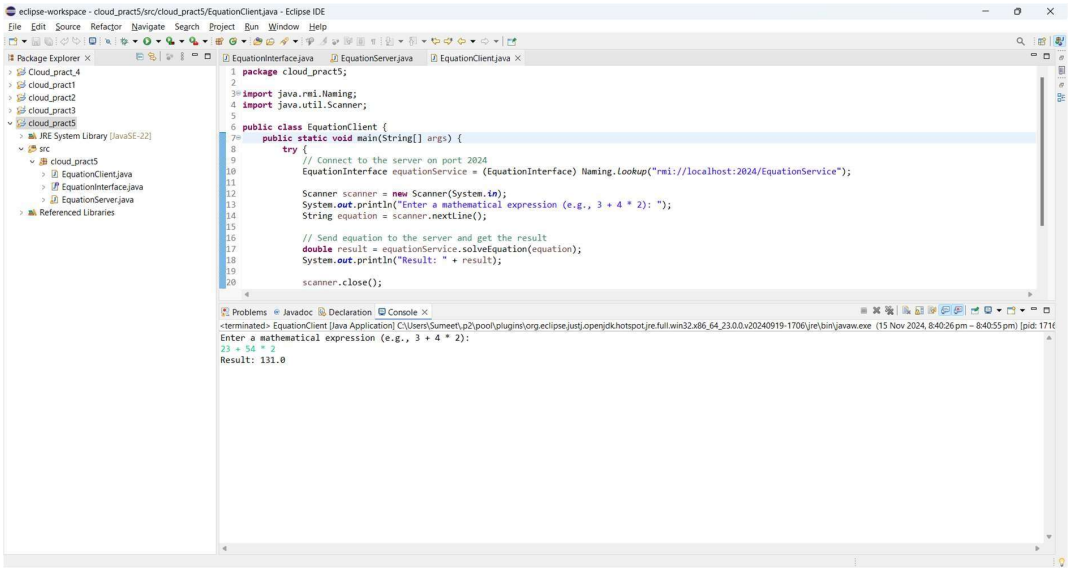
Execution/ Compilation Screenshot:





Output:





Reference:-

https://mscitdcspracs.blogspot.com/p/practical-no-3-remote-method-invocation.html

https://stackoverflow.com/questions/32593242/some-questions-about-rmi-remote-method-invocation?rq=1

Practical No. 6

Aim:- Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.

Theory:-

This practical involves creating a Library database using MySQL and implementing a Book table to store information about books, including Book_id, Book_name, and Book_author. The goal is to retrieve book information using the Remote Object Communication concept with RMI (Remote Method Invocation). In this setup, the server acts as an intermediary that connects to the MySQL database using JDBC (Java Database Connectivity). The server retrieves data based on client requests and sends the book information back. The client remotely calls methods on the server using RMI, abstracting the database connection and query execution process. This architecture is useful in cloud computing environments, allowing multiple clients to access the centralized library database concurrently through remote communication, leveraging the cloud's scalability and efficiency in handling distributed requests.

Algorithm/ Concept:-

- Create MySQL Database:
- Create a Library database in MySQL.
- Create a Book table with columns: Book_id, Book_name, and Book_author.
- Insert sample book records into the table.

- Server-Side Setup:
- Define a remote interface with a method getBookInfo() to retrieve book information.
- Implement the interface in a server class, using JDBC to connect to the MySQL database.
- Execute a SQL query to fetch all book records from the Book table.
- Bind the server object to the RMI registry.

- Client-Side Setup:
- Look up the remote object from the RMI registry.
- Call the getBookInfo() method to request book information.

- Retrieve and Display Data:
- The server fetches data from the database and returns the list of books to the client.
- The client displays the retrieved book information.

- Termination:
  - Close database connections and terminate the client-server session.

Code/ Program:-

Database

```
CREATE DATABASE Library;
USE Library;
```

```
CREATE TABLE Book (
Book_id INT PRIMARY KEY AUTO_INCREMENT, Book_name VARCHAR(100)
NOT NULL,
Book_author VARCHAR(100) NOT NULL
);
INSERT INTO Book (Book_name, Book_author) VALUES ('Effective Java', 'Joshua
Bloch'),
('Clean Code', 'Robert C. Martin'), ('Design Patterns', 'Erich Gamma');
```

LibraryInterface

```
package cloud_pract6;
import java.rmi.Remote;
import java.rmi.RemoteException; import java.util.List;
public interface LibraryInterface extends Remote { List<String> getBookDetails()
throws RemoteException;
}
```

LibraryServer

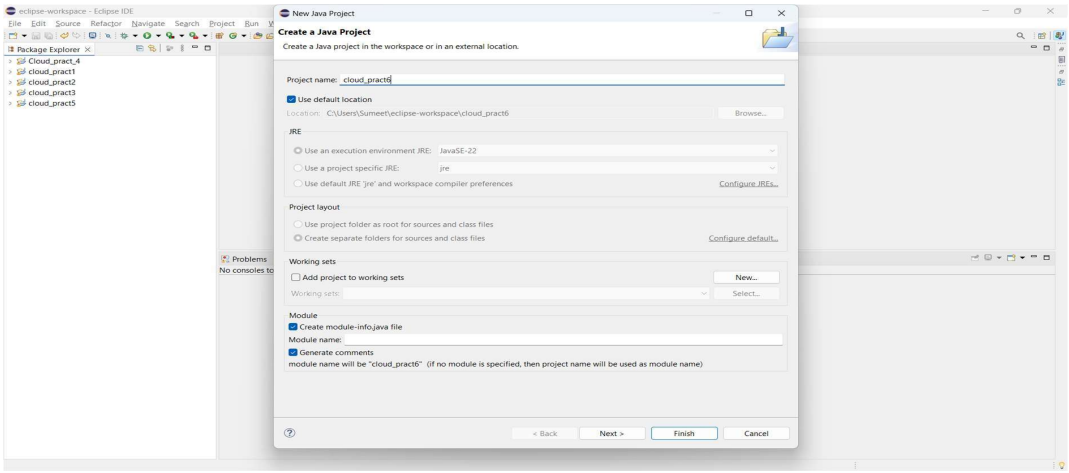
```
package cloud_pract6;
import java.rmi.Naming;
import java.rmi.RemoteException; import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject; import java.sql.Connection;
import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.Statement;
import java.util.ArrayList; import java.util.List;
public class LibraryServer extends UnicastRemoteObject implements LibraryInterface
{ protected LibraryServer() throws RemoteException {
super();
}
@Override
public List<String> getBookDetails() throws RemoteException { List<String>
bookList = new ArrayList<>();
try {
String url = "jdbc:mysql://localhost:3306/Library"; String user = "root";
String password = "gnims";
Connection con = DriverManager.getConnection(url, user, password); Statement stmt =
con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Book"); while (rs.next()) {
String bookInfo = "ID: " + rs.getInt("Book_id") +
", Name: " + rs.getString("Book_name") + ", Author: " + rs.getString("Book_author");
bookList.add(bookInfo);}
rs.close();
stmt.close();
con.close();
} catch (Exception e) {
System.err.println("Database Error: " + e.getMessage()); throw new
RemoteException("Error fetching book details.");
}
return bookList;
}
public static void main(String[] args) { try {
```

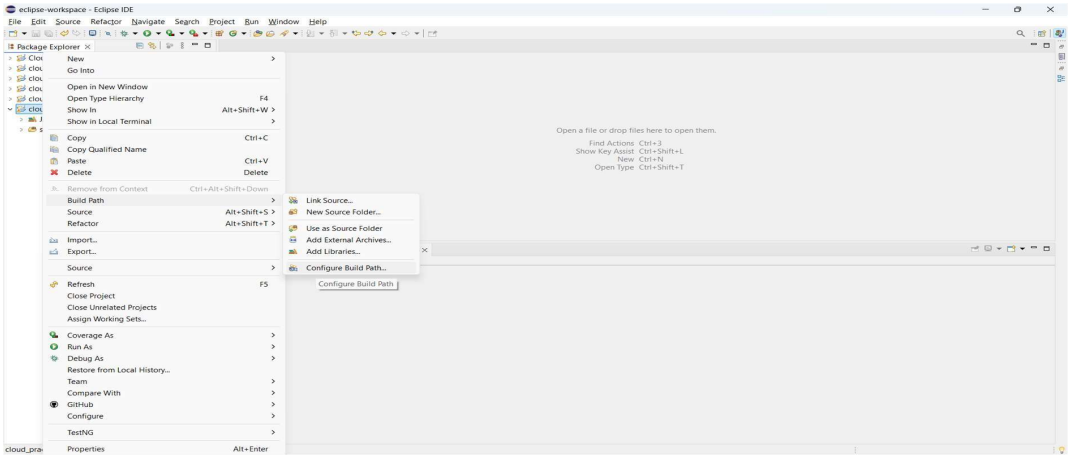
```
LocateRegistry.createRegistry(2024); LibraryServer server = new LibraryServer();
Naming.rebind("rmi://localhost:2024/LibraryService", server);
System.out.println("Library Server is running on port 2024...");
} catch (Exception e) {
System.err.println("Server exception: " + e.getMessage()); e.printStackTrace();
}}}
```

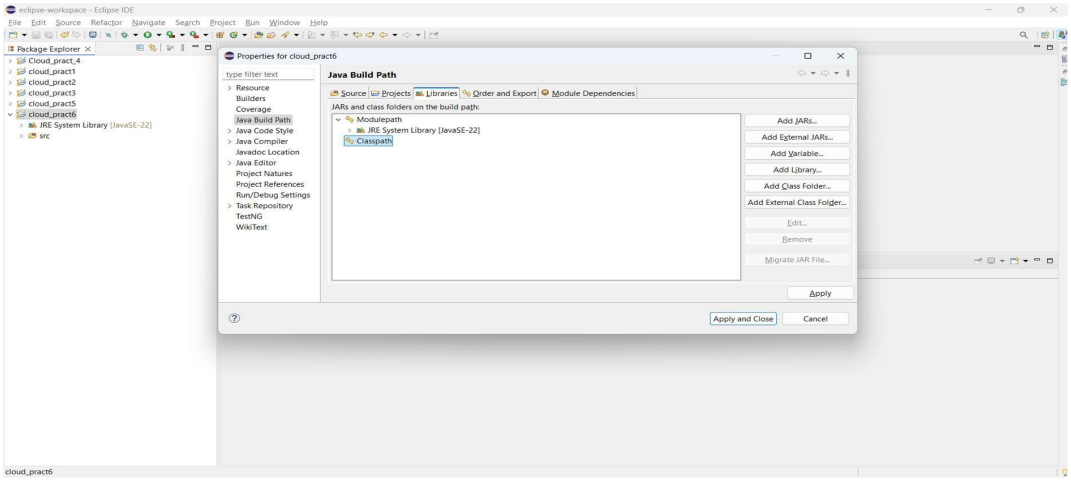
LibraryClient

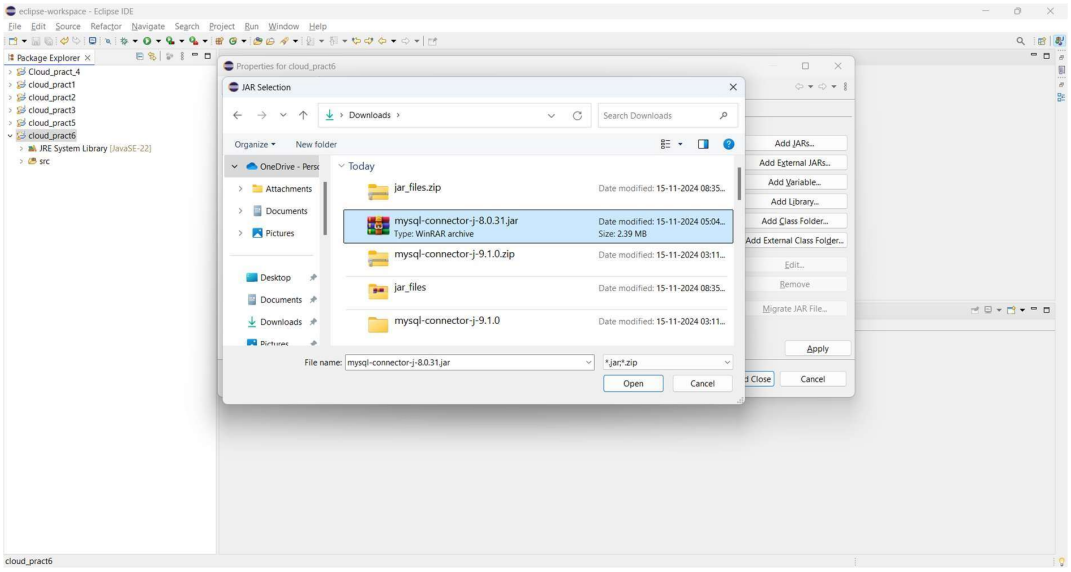
```
package cloud_pract6;
import java.rmi.Naming;
import java.util.List;
public class LibraryClient {
public static void main(String[] args) { try {
LibraryInterface libraryService = (LibraryInterface)
Naming.lookup("rmi://localhost:2024/LibraryService");
List<String> books = libraryService.getBookDetails();
System.out.println("Book Details from Library Database:"); for (String book : books) {
System.out.println(book);
}} catch (Exception e) {
System.err.println("Client exception: " + e.getMessage()); e.printStackTrace();
}}}
```

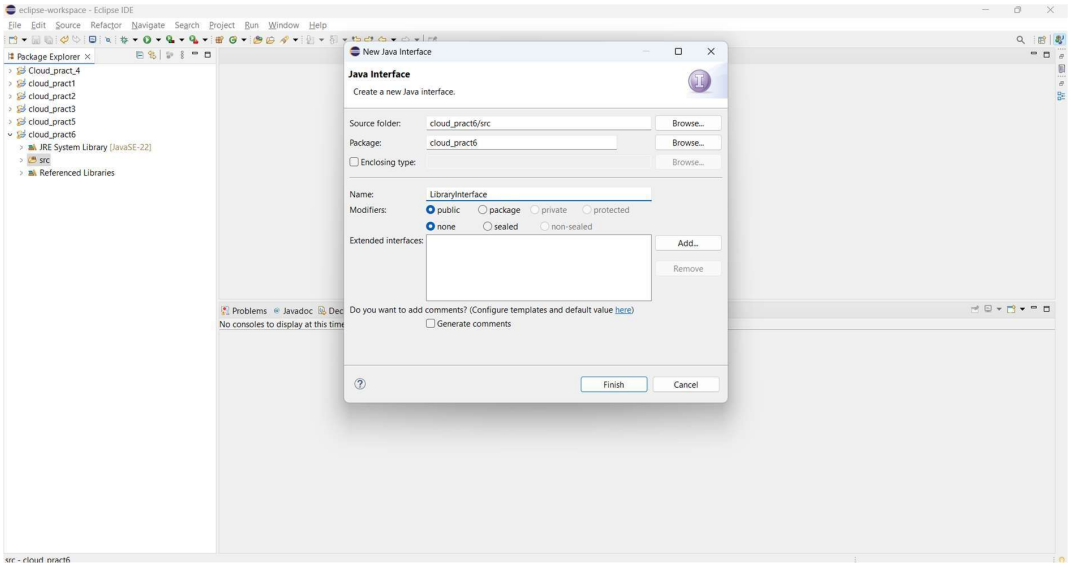
Execution/ Compilation Screenshot:-

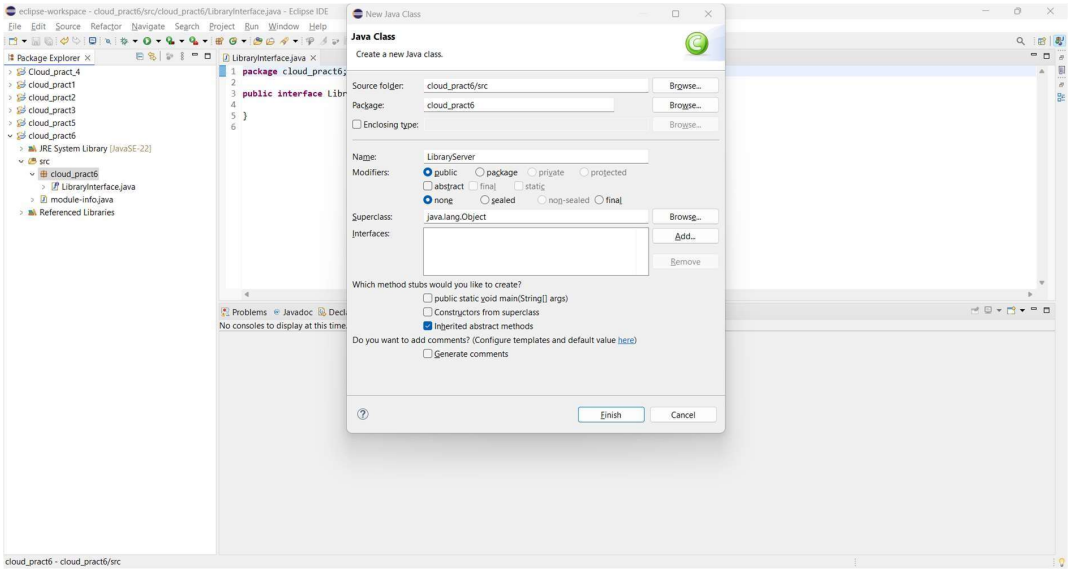


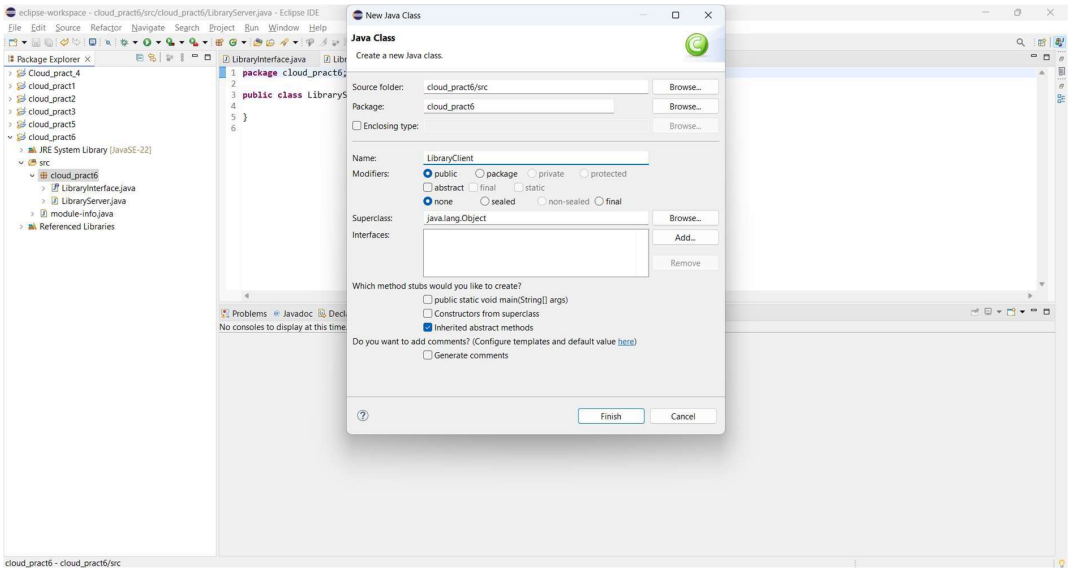



Properties for cloud_pract6
Java Build Path
Source
Projects
Libraries
Order and Export
Module Dependencies
JARs and class folders on the build path:
Modulepath
JRE System Library [JavaSE-22]
Classpath
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel

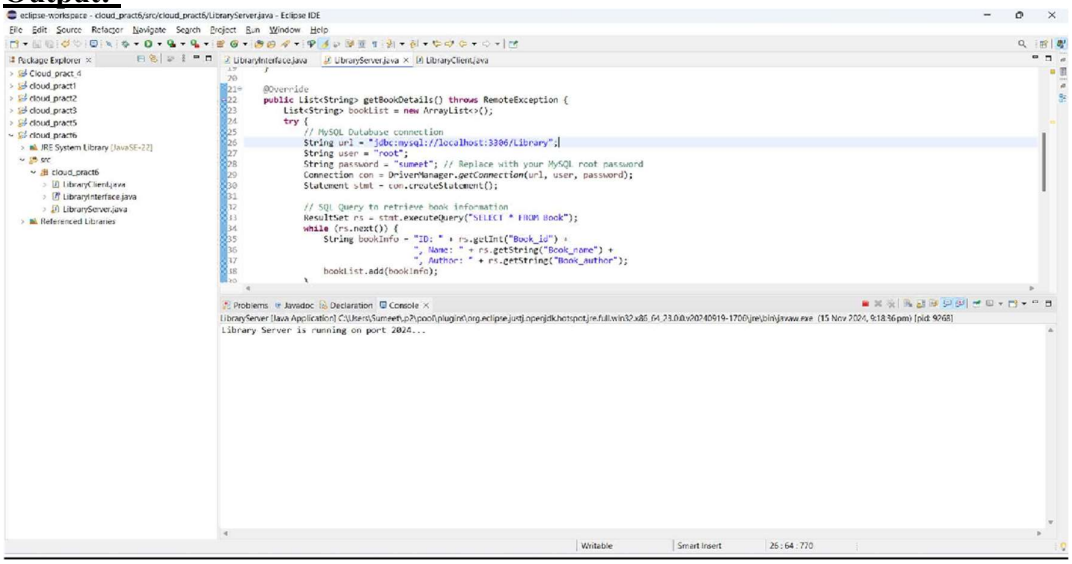

JAR Selection
Downloads
Search Downloads
Organize
New folder
Today
jar_files.zip
mysql-connector-j-8.0.31.jar
Type: WinRAR archive
Size: 2.39 MB
mysql-connector-j-9.1.0.zip
jar_files
mysql-connector-j-9.1.0
File name: mysql-connector-j-8.0.31.jar
.jar;.zip
Open
Cancel

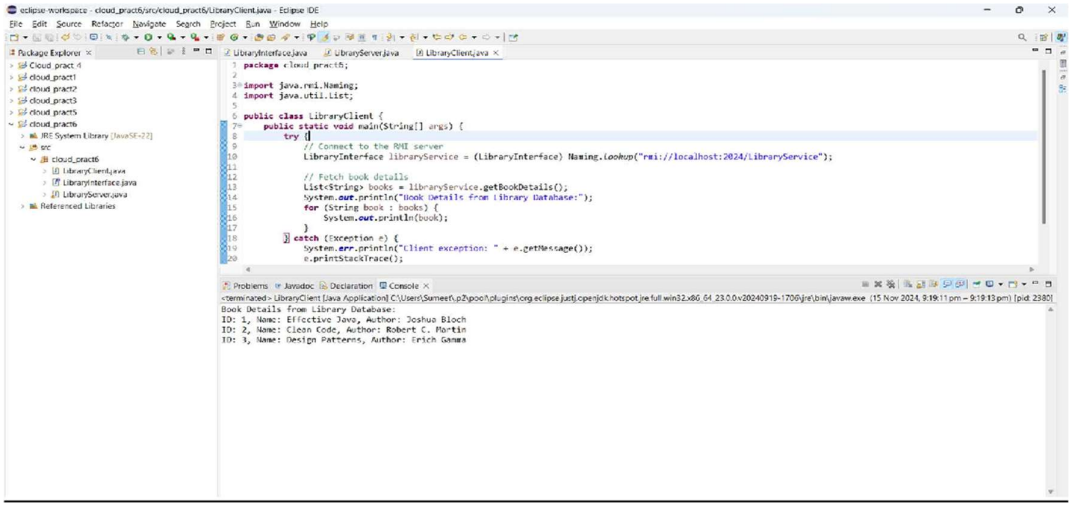

New Java Interface
Java Interface
Create a new Java interface.
Source folder: cloud_pract6/src
Package: cloud_pract6
Enclosing type:
Name: LibraryInterface
Modifiers: public package private protected
none sealed non-sealed
Extended interfaces:
Add...
Remove
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel

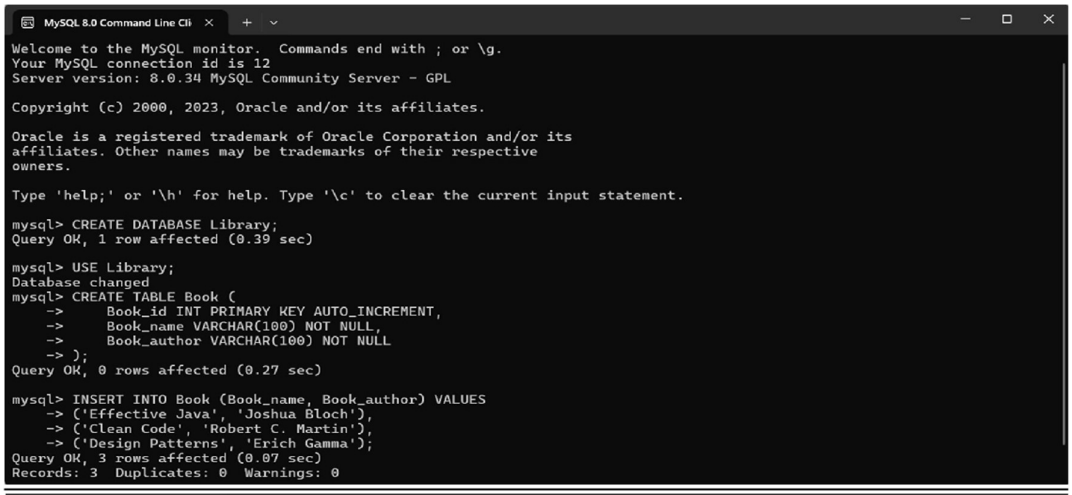

eclipse-workspace - cloud_pract6/src/cloud_pract6/LibraryInterface.java - Eclipse IDE
New Java Class
Java Class
Create a new Java class.
Source folder: cloud_pract6/src
Package: cloud_pract6
Enclosing type:
Name: LibraryServer
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
cloud_pract6 - cloud_pract6/src

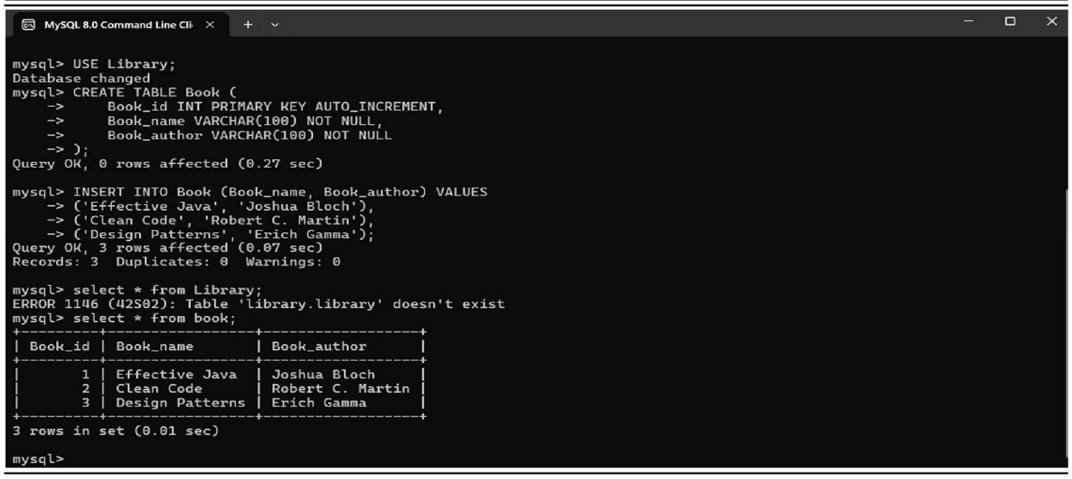

eclipse-workspace - cloud_pract6/src/cloud_pract6/LibraryServer.java - Eclipse IDE
New Java Class
Java Class
Create a new Java class.
Source folder: cloud_pract6/src
Package: cloud_pract6
Enclosing type:
Name: LibraryClient
Modifiers: public package private protected
abstract final static
none sealed non-sealed final
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel
cloud_pract6 - cloud_pract6/src

Output:-


eclipse-workspace - cloud_pract6/src/cloud_pract6/LibraryServer.java - Eclipse IDE
LibraryServer is running on port 2024...







Reference:-

https://www.tutorialspoint.com/java_rmi/java_rmi_database_application.htm https://www.scribd.com/doc/246765817/my-sql

Practical No. :-7

Aim:- Using MySQL create Elecrtic_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the Elecrtic_Bill database using Remote Object Communication concept.

Theory:-

This practical involves creating an **Electric_Bill** database in **MySQL** with a table Bill that stores the consumer_name, bill_due_date, and bill_amount for electric bills. The goal is to retrieve bill information from this database using **Remote Object Communication (RMI)**. In this setup, the **server** acts as a remote service that connects to the MySQL database via **JDBC (Java Database Connectivity)**, fetches the bill data, and sends it back to the **client**. The client communicates with the server through RMI, invoking remote methods to retrieve the required bill information. This approach is well-suited for cloud computing environments, where the server can be hosted on a cloud platform, enabling multiple clients to retrieve bill information concurrently. The scalability of cloud services ensures that large amounts of data and simultaneous requests are handled efficiently.

Algorithm/ Concept:-

□ **Create MySQL Database:**

- Create a Electric_Bill database in MySQL.
- Create a Bill table with columns: consumer_name, bill_due_date, and bill_amount.
- Insert sample bill records into the table.

□ **Server-Side Setup:**

- Define a remote interface with a method getBillInfo() to retrieve bill details.
- Implement the interface in a server class, where JDBC is used to connect to the MySQL database.
- Execute an SQL query to fetch bill records from the Bill table.
- Bind the server object to the RMI registry.

□ **Client-Side Setup:**

- Look up the remote object from the RMI registry.
- Call the getBillInfo() method to retrieve bill data.

□ **Retrieve and Display Data:**

- The server fetches data from the database and returns the list of bills to the client.
- The client displays the retrieved bill information.

□ **Termination:**

- Close database connections and terminate the client-server communication.

Code/ Program:-

Database

```
CREATE DATABASE Electric_Bill; USE Electric_Bill;
CREATE TABLE Bill (
Bill_id INT PRIMARY KEY AUTO_INCREMENT,
consumer_name VARCHAR(100) NOT NULL, bill_due_date DATE NOT NULL,
bill_amount DECIMAL(10, 2) NOT NULL
);
INSERT INTO Bill (consumer_name, bill_due_date, bill_amount) VALUES ('John
Doe', '2024-11-30', 150.75),
('Jane Smith', '2024-12-15', 200.50),
('Alice Johnson', '2024-11-25', 120.00);
```

ElectricBillInterface

```
package cloud_pract7;
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
public interface ElectricBillInterface extends Remote { List<String> getBillDetails()
throws RemoteException;
}
```

ElectricBillServer

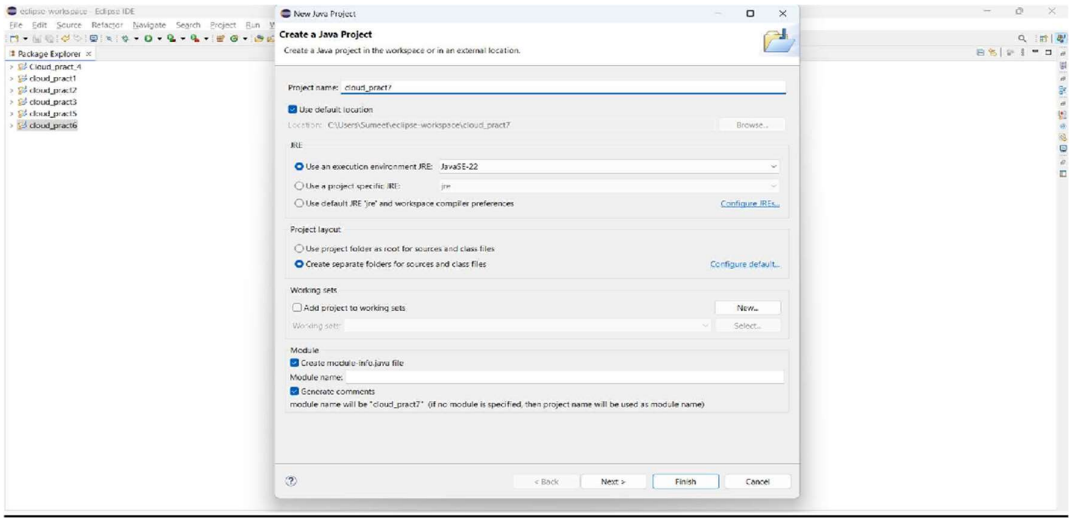
```
package cloud_pract7;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
public class ElectricBillServer extends UnicastRemoteObject implements
ElectricBillInterface {
protected ElectricBillServer() throws RemoteException {
super();
}
@Override
public List<String> getBillDetails() throws RemoteException { List<String> billList =
new ArrayList<>();
try {
String url = "jdbc:mysql://localhost:3306/Electric_Bill"; String user = "root";
String password = "gnims";
Connection con = DriverManager.getConnection(url, user, password); Statement stmt =
con.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Bill");
while (rs.next()) {
String billInfo = "ID: " + rs.getInt("Bill_id") +
```

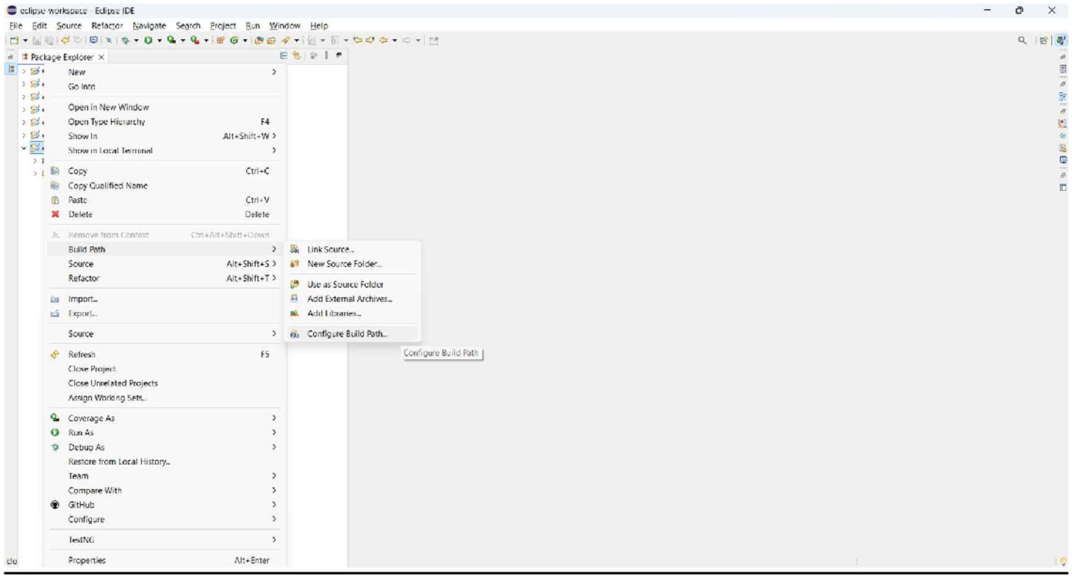
", Consumer: " + rs.getString("consumer_name") + ", Due Date: " +
rs.getDate("bill_due_date") +
", Amount: $" + rs.getDouble("bill_amount"); billList.add(billInfo);}
rs.close();
stmt.close();
con.close();
} catch (Exception e) {
System.err.println("Database Error: " + e.getMessage());
throw new RemoteException("Error fetching bill details.");}
return billList;}
public static void main(String[] args) {
try {
LocateRegistry.createRegistry(2025); ElectricBillServer server = new
ElectricBillServer();
Naming.rebind("rmi://localhost:2025/ElectricBillService", server);
System.out.println("Electric Bill Server is running on port 2025...");
} catch (Exception e) {System.err.println("Server exception: " + e.getMessage());
e.printStackTrace();}}}

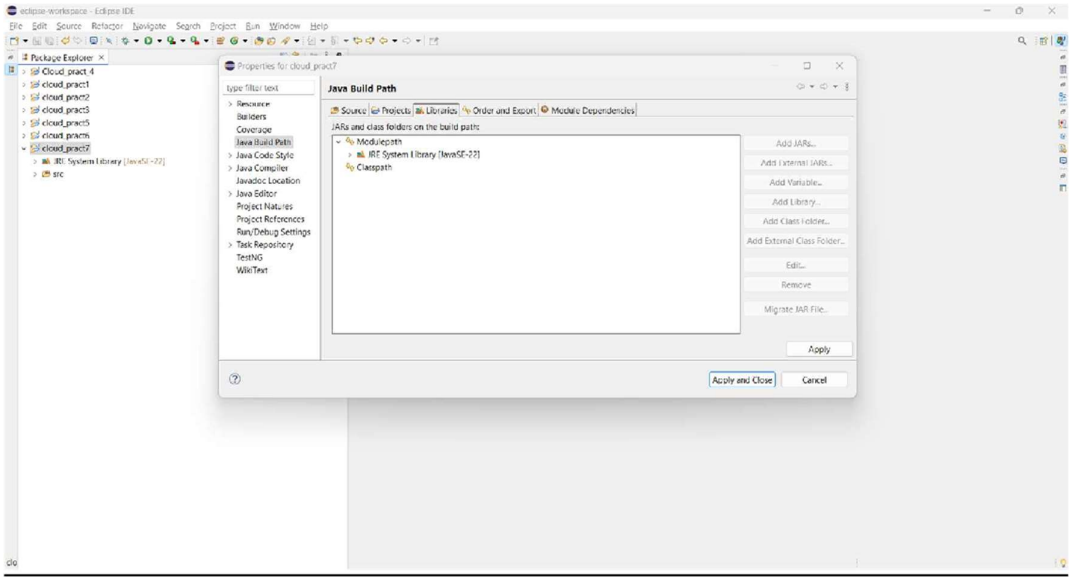
ElectricBillClient

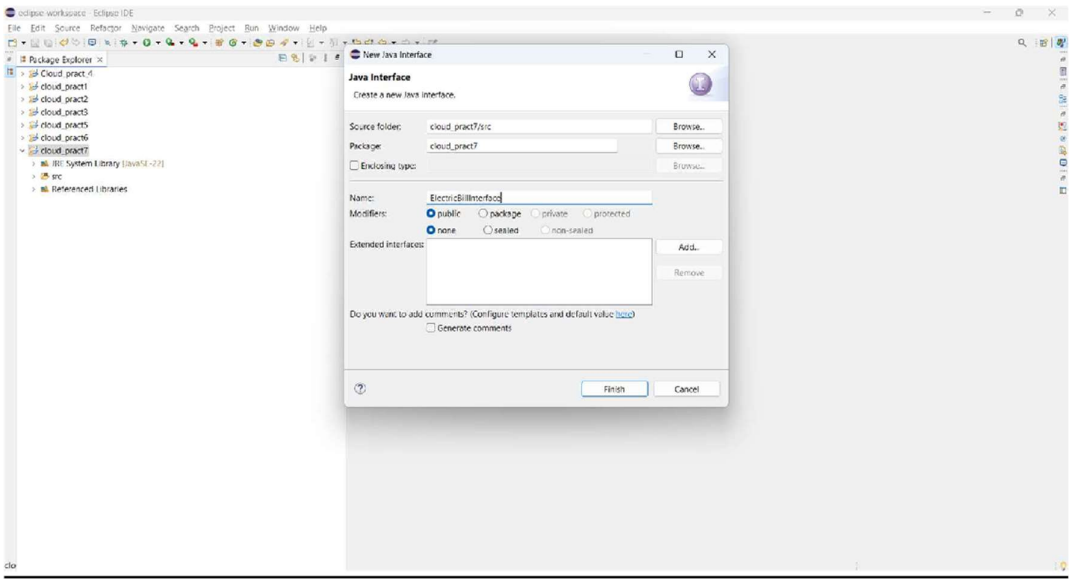
```
package cloud_pract7;
import java.rmi.Naming;
import java.util.List;
public class ElectricBillClient {
public static void main(String[] args) {
try {
ElectricBillInterface billService = (ElectricBillInterface)
Naming.lookup("rmi://localhost:2025/ElectricBillService");
List<String> bills = billService.getBillDetails(); System.out.println("Bill Details from
Electric_Bill Database:"); for (String bill : bills) {
System.out.println(bill);
}} catch (Exception e) {
System.err.println("Client exception: " + e.getMessage()); e.printStackTrace();
}}}
```

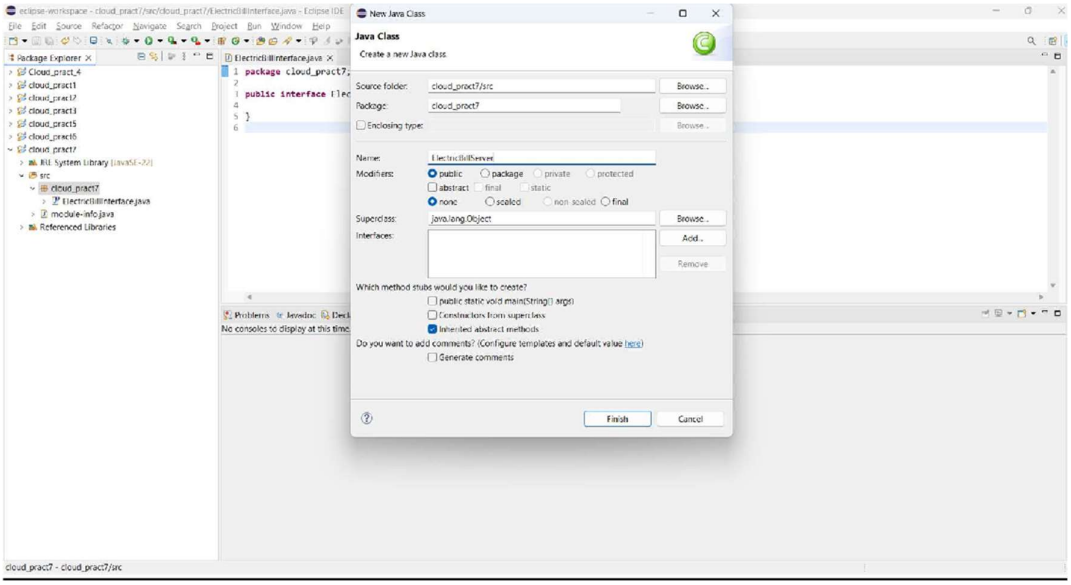
Execution/ Compilation Screenshot:-

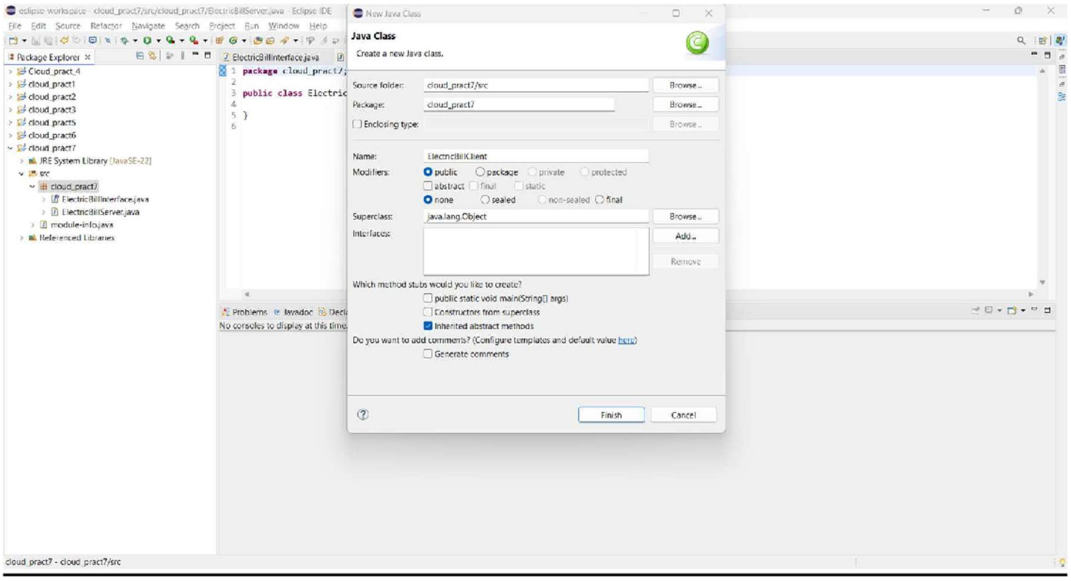



eclipse-workspace - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Source
Refresh F5
Close Project
Close Unrelated Projects
Assign Working Sets...
Coverage As
Run As
Debug As
Restore from Local History...
Team
Compare With
GitHub
Configure
TestNG
Properties Alt+Enter
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
Configure Build Path

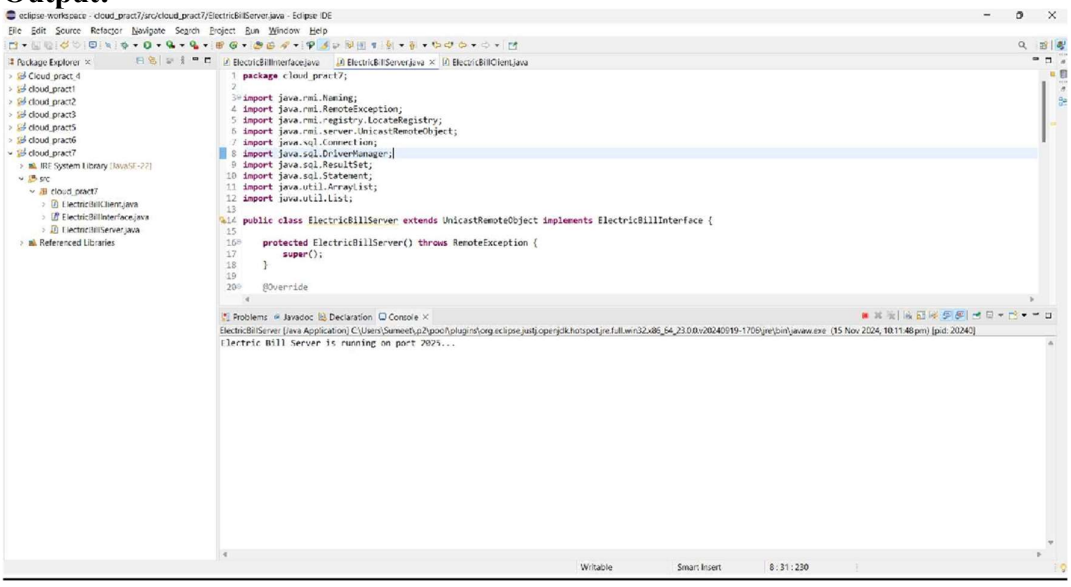

Properties for cloud_pract7
type filter text
Resource
Builders
Coverage
Java Build Path
Java Code Style
Java Compiler
Javadoc Location
Java Editor
Project Natures
Project References
Run/Debug Settings
Task Repository
TestNG
WikiText
Java Build Path
Source Projects Libraries Order and Export Module Dependencies
JARs and class folders on the build path:
Modulepath
JRE System Library [JavaSE-22]
Classpath
Add JARs...
Add External JARs...
Add Variable...
Add Library...
Add Class Folder...
Add External Class Folder...
Edit...
Remove
Migrate JAR File...
Apply
Apply and Close
Cancel

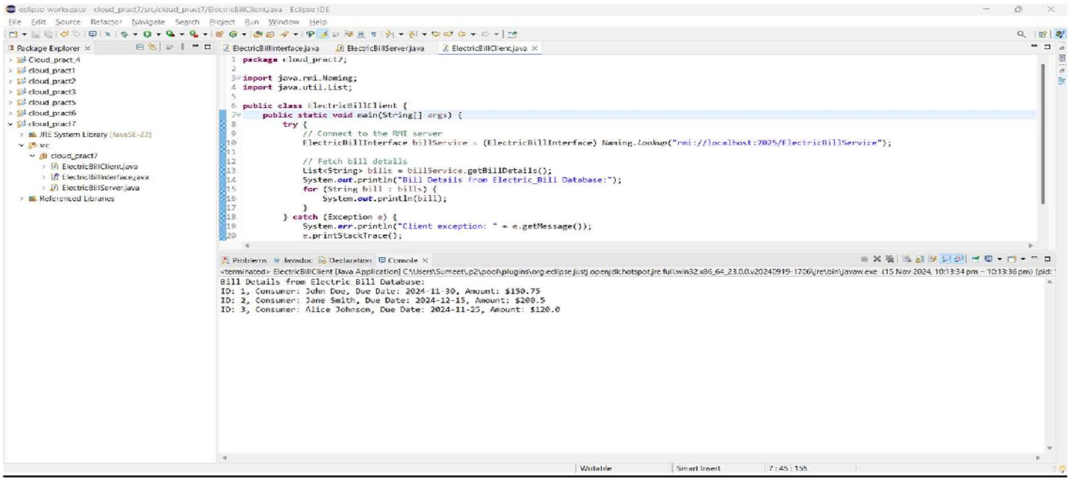

New Java Interface
Java Interface
Create a new Java interface.
Source folder: cloud_pract7/src
Package: cloud_pract7
Enclosing type:
Name: ElectricBillInterface
Modifiers: public package private protected
none sealed non-sealed
Extended interfaces:
Add...
Remove
Do you want to add comments? (Configure templates and default value here)
Generate comments
Finish
Cancel



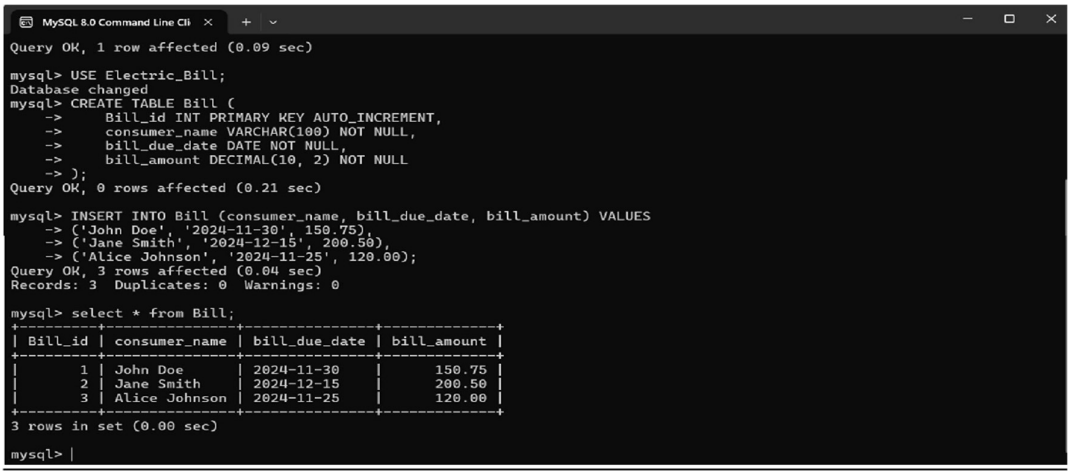


Output:-

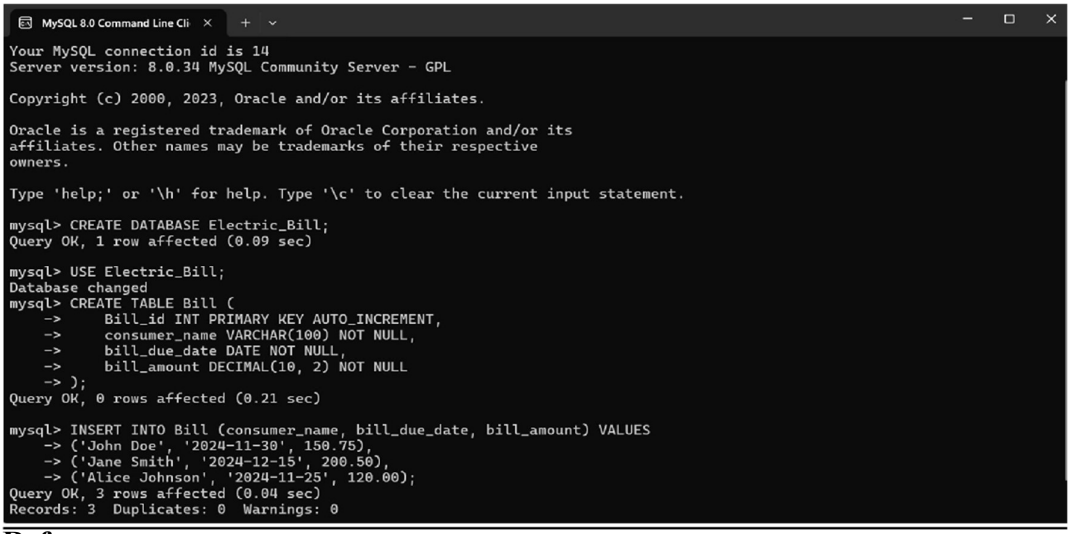




```
Bill Details from Electric_Bill Database:
ID: 1, Consumer: John Doe, Due Date: 2024-11-30, Amount: $150.75
ID: 2, Consumer: Jane Smith, Due Date: 2024-12-15, Amount: $200.5
ID: 3, Consumer: Alice Johnson, Due Date: 2024-11-25, Amount: $120.0
```



```
MySQL 8.0 Command Line Cli
Query OK, 1 row affected (0.09 sec)

mysql> USE Electric_Bill;
Database changed
mysql> CREATE TABLE Bill (
    ->     Bill_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     consumer_name VARCHAR(100) NOT NULL,
    ->     bill_due_date DATE NOT NULL,
    ->     bill_amount DECIMAL(10, 2) NOT NULL
    -> );
Query OK, 0 rows affected (0.21 sec)

mysql> INSERT INTO Bill (consumer_name, bill_due_date, bill_amount) VALUES
    -> ('John Doe', '2024-11-30', 150.75),
    -> ('Jane Smith', '2024-12-15', 200.50),
    -> ('Alice Johnson', '2024-11-25', 120.00);
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from Bill;
+---------+---------------+---------------+-------------+
| Bill_id | consumer_name | bill_due_date | bill_amount |
+---------+---------------+---------------+-------------+
|       1 | John Doe      | 2024-11-30    |      150.75 |
|       2 | Jane Smith    | 2024-12-15    |      200.50 |
|       3 | Alice Johnson | 2024-11-25    |      120.00 |
+---------+---------------+---------------+-------------+
3 rows in set (0.00 sec)

mysql>
```



```
MySQL 8.0 Command Line Cli
Your MySQL connection id is 14
Server version: 8.0.34 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> CREATE DATABASE Electric_Bill;
Query OK, 1 row affected (0.09 sec)

mysql> USE Electric_Bill;
Database changed
mysql> CREATE TABLE Bill (
    ->     Bill_id INT PRIMARY KEY AUTO_INCREMENT,
    ->     consumer_name VARCHAR(100) NOT NULL,
    ->     bill_due_date DATE NOT NULL,
    ->     bill_amount DECIMAL(10, 2) NOT NULL
    -> );
Query OK, 0 rows affected (0.21 sec)

mysql> INSERT INTO Bill (consumer_name, bill_due_date, bill_amount) VALUES
    -> ('John Doe', '2024-11-30', 150.75),
    -> ('Jane Smith', '2024-12-15', 200.50),
    -> ('Alice Johnson', '2024-11-25', 120.00);
Query OK, 3 rows affected (0.04 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

Reference:-

https://www.scribd.com/doc/255516553/10739977-828642320526947-512557484-n

https://www.scribd.com/doc/255516553/10739977-828642320526947-512557484-n

Practical No. :-8

Aim:- Implementation of mutual exclusion using Token ring algorithm.

Theory:-

Mutual exclusion is a critical concept in distributed systems to ensure that multiple processes or nodes do not access shared resources simultaneously. The **Token Ring Algorithm** is a solution for mutual exclusion in a distributed system, where a logical token circulates in a closed loop among processes. A process can access the shared resource only if it holds the token. This prevents conflicts and ensures that only one process can access the resource at any given time. In cloud computing, this algorithm can be implemented across multiple distributed nodes in the cloud infrastructure to ensure safe access to resources like databases or files. The cloud environment's elasticity and scalability ensure that the token ring algorithm can handle large numbers of nodes efficiently. This algorithm reduces the need for global synchronization and avoids deadlocks, making it suitable for cloud-based applications requiring coordination among distributed entities.

Algorithm/ Concept:-

□ **Initialization:**

- Assign a token to a node in the system.
- Each node has a unique identifier and is connected to a neighboring node.

□ **Token Passing:**

- The token circulates through the system in a fixed order.
- A node can only enter the critical section when it holds the token.

□ **Critical Section:**

- A node holding the token enters the critical section to access the shared resource.
- After finishing, the node passes the token to its neighboring node.

□ **Token Passing Continues:**

- The token circulates until the process completes its task, ensuring mutual exclusion.

□ **Termination:**

- The process can continue indefinitely, with the token continually circulating through the system.

Code/ Program:-

```
package cloud_pract8;

import java.util.concurrent.locks.Lock;
```

```
import java.util.concurrent.locks.ReentrantLock;
class TokenRing {
private final int numberOfProcesses; private final Process[] processes;
private int tokenHolder = 0;
public TokenRing(int numberOfProcesses)
{ this.numberOfProcesses = numberOfProcesses;
this.processes = new Process[numberOfProcesses];
// Initialize processes
for (int i = 0; i < numberOfProcesses; i++) { processes[i] = new Process(i);}}
public void start() {
// Start each process
for (Process p : processes) {
new Thread(p).start();}}
public synchronized void passToken() {
tokenHolder = (tokenHolder + 1) % numberOfProcesses; notifyAll();}
class Process implements Runnable {
private final int id; // The unique identifier for each process
private final Lock lock = new ReentrantLock();
public Process(int id) {
this.id = id;}
@Override
public void run() {
while (true) {
try {
synchronized (TokenRing.this) { while (tokenHolder != id) { TokenRing.this.wait();}}
criticalSection(); passToken(); nonCriticalSection();
} catch (InterruptedException e) { e.printStackTrace();}}}
private void criticalSection() { lock.lock();
try {
System.out.println("Process " + id + " is entering the critical section."); Thread.sleep(1000);
System.out.println("Process " + id + " is leaving the critical section.");
```

} catch (InterruptedException e) { e.printStackTrace();

} finally { lock.unlock();}}

private void nonCriticalSection() {

try {

System.out.println("Process " + id + " is performing non-critical work."); Thread.sleep(500); // Simulate work done outside the critical section

} catch (InterruptedException e) { e.printStackTrace();}}}

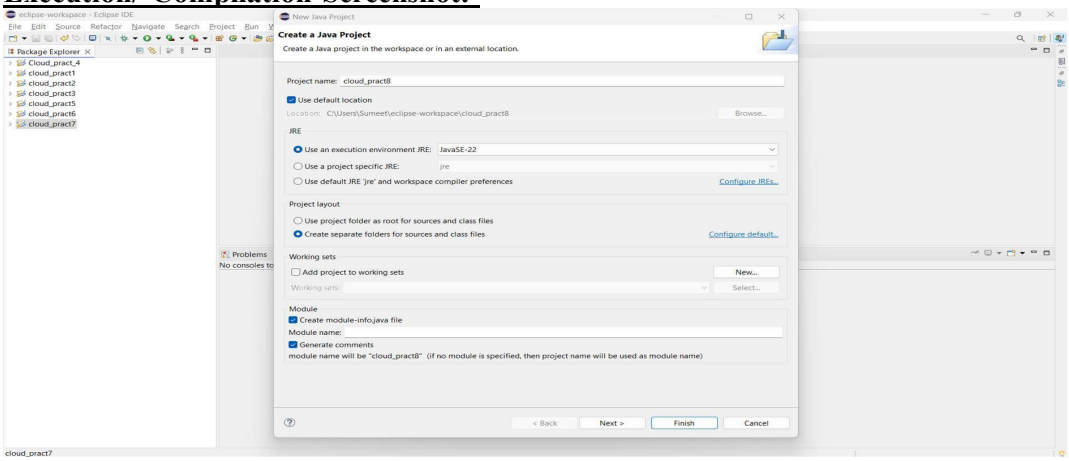
public static void main(String[] args) {

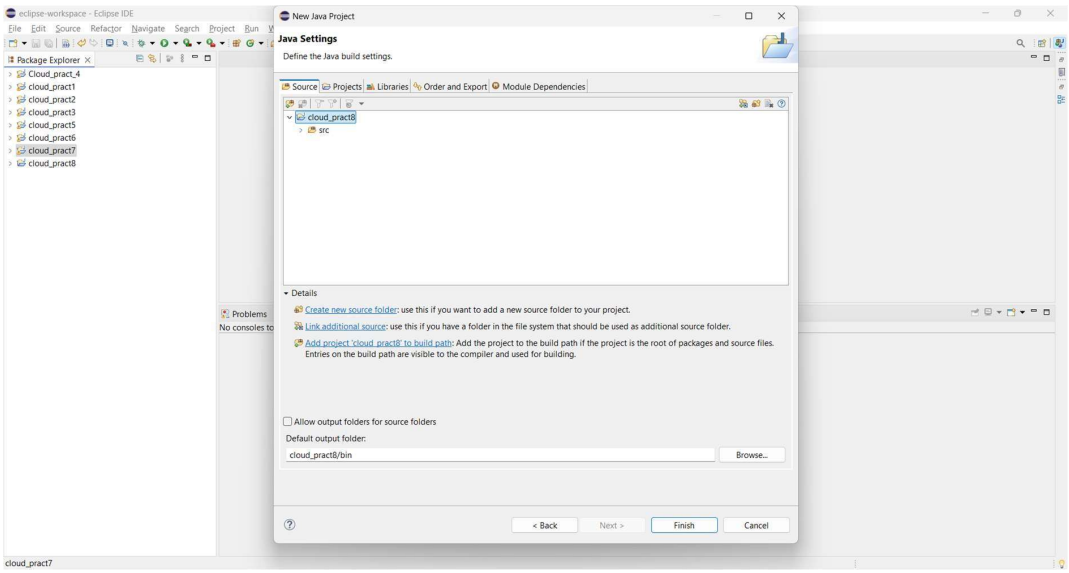
int numberOfProcesses = 5;

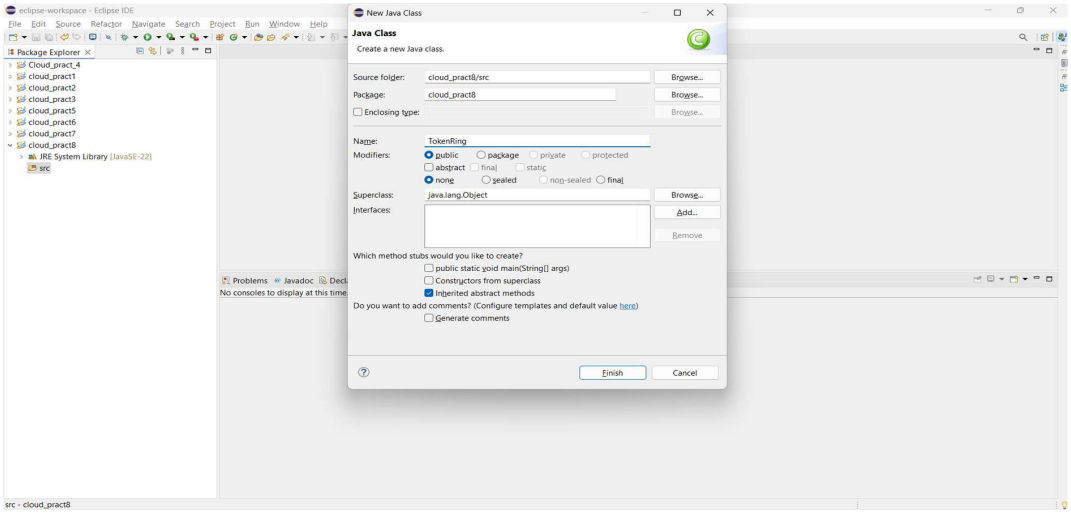
TokenRing tokenRing = new TokenRing(numberOfProcesses);

// Start the Token Ring System tokenRing.start();}}

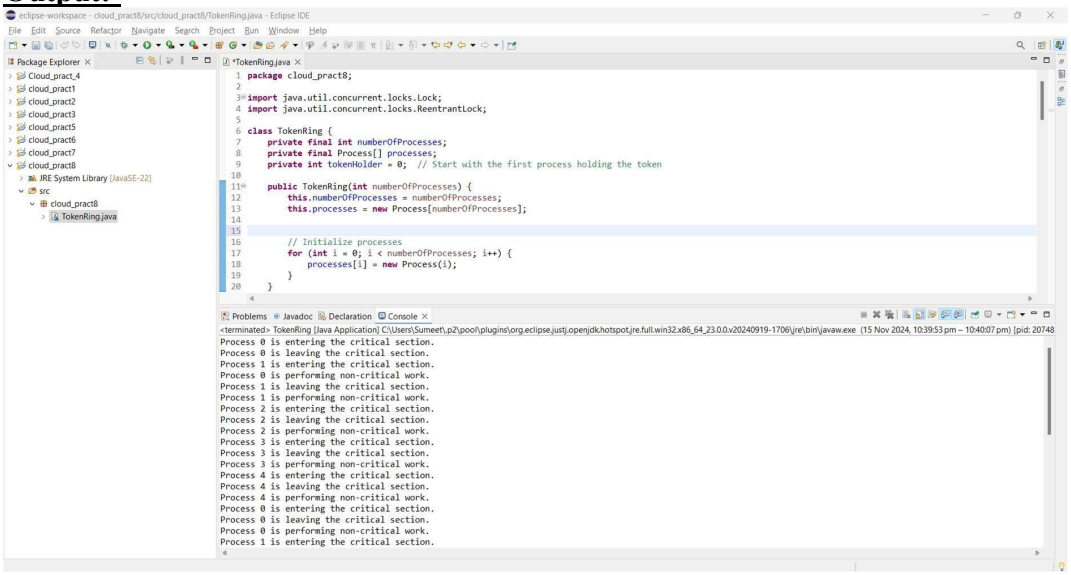
Execution/ Compilation Screenshot:-







Output:-



Reference:-
https://www.geeksforgeeks.org/mutual-exclusion-in-distributed-system/
https://stackoverflow.com/questions/27471838/explain-why-this-algorithm-does-not-guarantee- mutual-exclusi

Practical No. :-9

Aim:- Implementation of Storage as a Service using Google Docs.

Theory:-

Storage as a Service (SaaS) is a cloud computing model where data storage is provided to users over the internet, allowing access, management, and sharing of data remotely. **Google Docs** is an online word processor that allows users to store, share, and collaborate on documents. Implementing **Storage as a Service using Google Docs** involves using Google's cloud platform (Google Drive) for storing files and managing them through APIs like the **Google Drive API**. This implementation allows users to upload, retrieve, update, and delete documents in Google Docs and manage permissions and sharing settings. By integrating the Google Docs API into your application, users can access their stored documents and interact with them programmatically. The solution is scalable, ensuring that users can store large amounts of data, with Google handling infrastructure and ensuring high availability, security, and real-time collaboration features.

Algorithm/ Concept:-

- API Setup:
- Create a Google Cloud project.
- Enable the Google Drive API.
- Set up OAuth 2.0 for authentication and authorization.
- Authentication:
- Authenticate users using OAuth 2.0 credentials to securely access their Google Docs.
- Obtain an access token to allow access to Google Drive.
- File Operations:
- Upload Document: Use Google Docs API to upload documents to Google Drive by sending file data.
- Retrieve Document: Use the API to fetch the document from Google Drive and display it.
- Update Document: Modify an existing document in Google Docs and save the changes to Google Drive.
- Delete Document: Use the API to remove a document from Google Drive.
- Sharing & Permissions:
- Set permissions to allow specific users to view, comment, or edit documents using the API.
- Termination:
- Close the connection and log out the user after completing the required operations.

Execution/ Compilation:-

/// Steps to execute for the following practical

1- Installation of Oracle VIRTUAL BOX.
2- Ubuntu server OS iso file
3- config. ubuntu in virtual box

4- network settings in virtual box - Bridged Adapter

5- installation of SAMBA server sudo apt update
sudo apt-get upgrade sudo apt install samba

6- setting directory at SAMBA server for sharing DOCUMENTS sudo mkdir /srv/samba/shared

7- Granting read, write and execute permission for shared folder sudo chmod 777 /srv/samba/shared

8- Creating backup of smb.conf for recover incase of misconfiguration sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.bak

9- Updating smb.conf file
sudo nano /etc/samba/smb.conf

10- Add the following lines at the end of smb.conf file [Shared]
path = /srv/samba/shared available = yes
valid users = nobody read only = no browsable = yes public = yes writable = yes

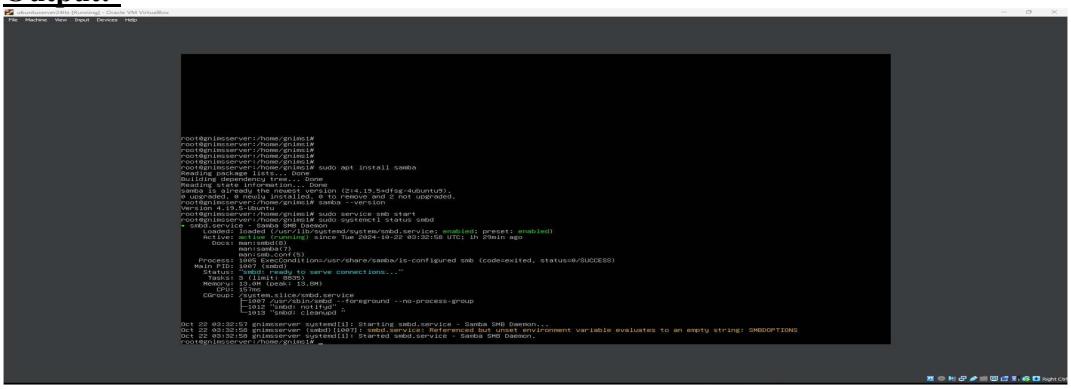
11- Create a Samba User named "nobody" sudo smbpasswd -a nobody

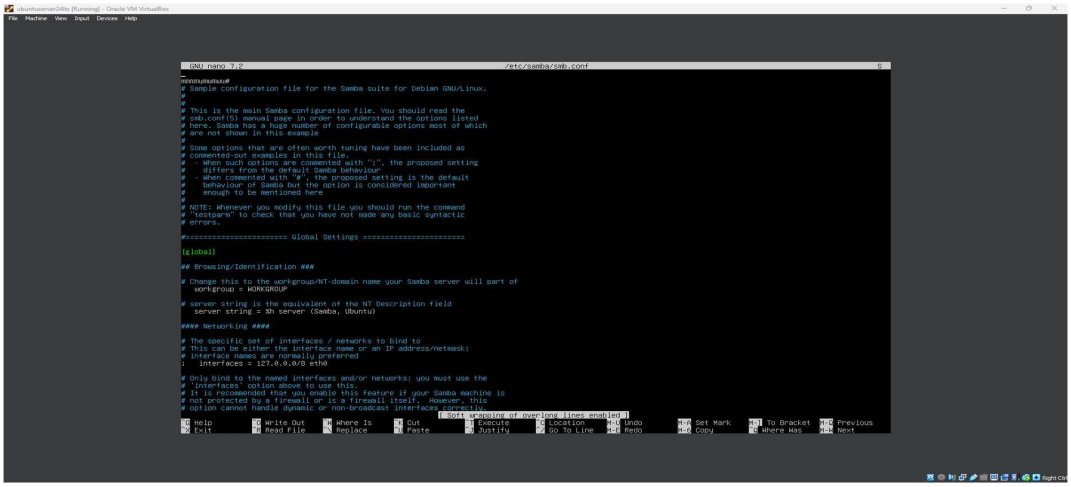
12- Restart Samba Service
sudo systemctl restart smbd

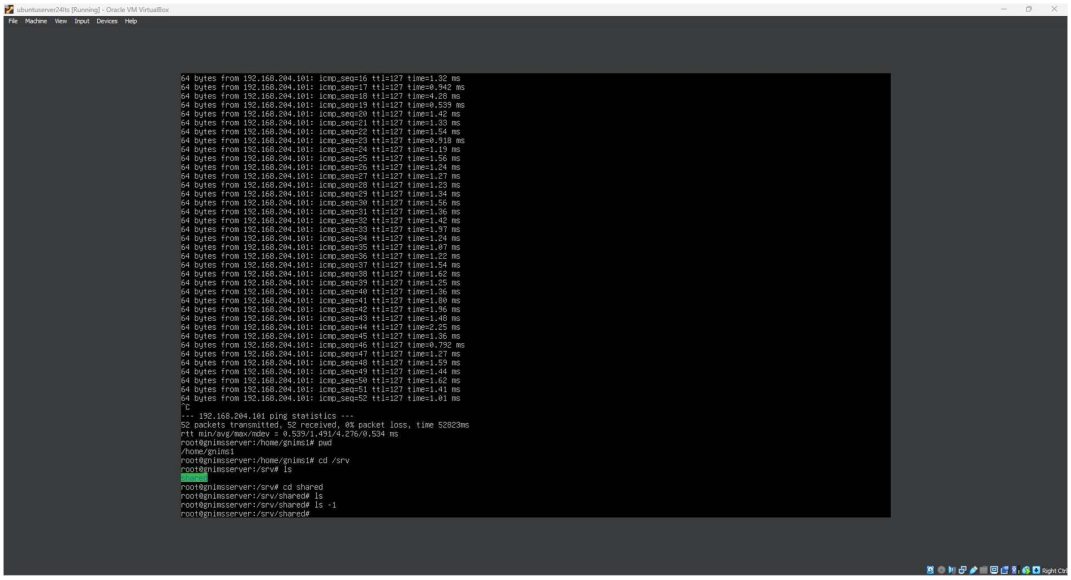
13- Access shared folder of Linux from windows or other operating system from same or different computer

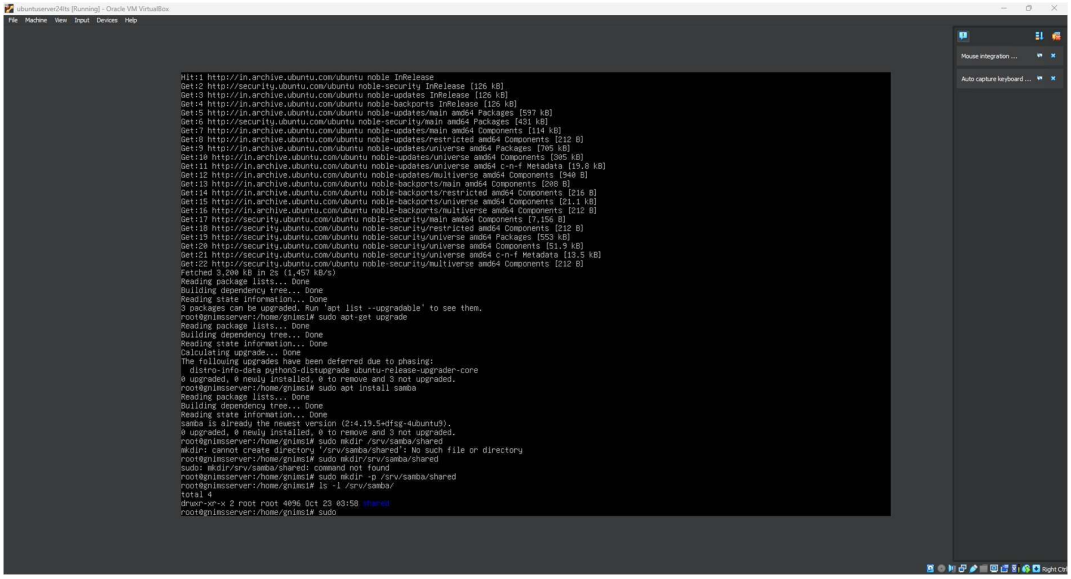
- open file explorer and type:
\\IP NUMBER OF SAMBA MACHINE\shared

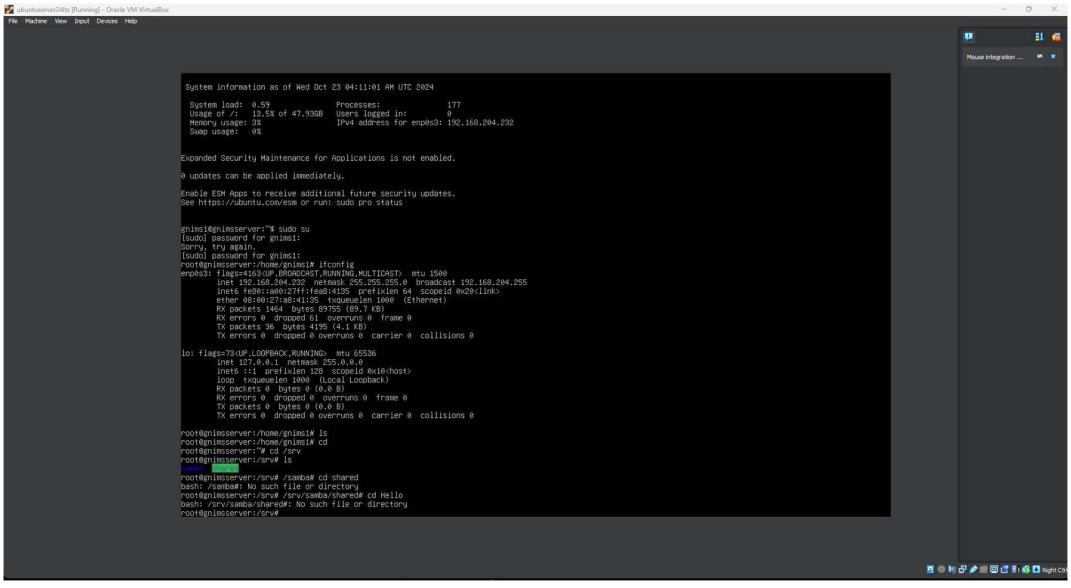
Output:-

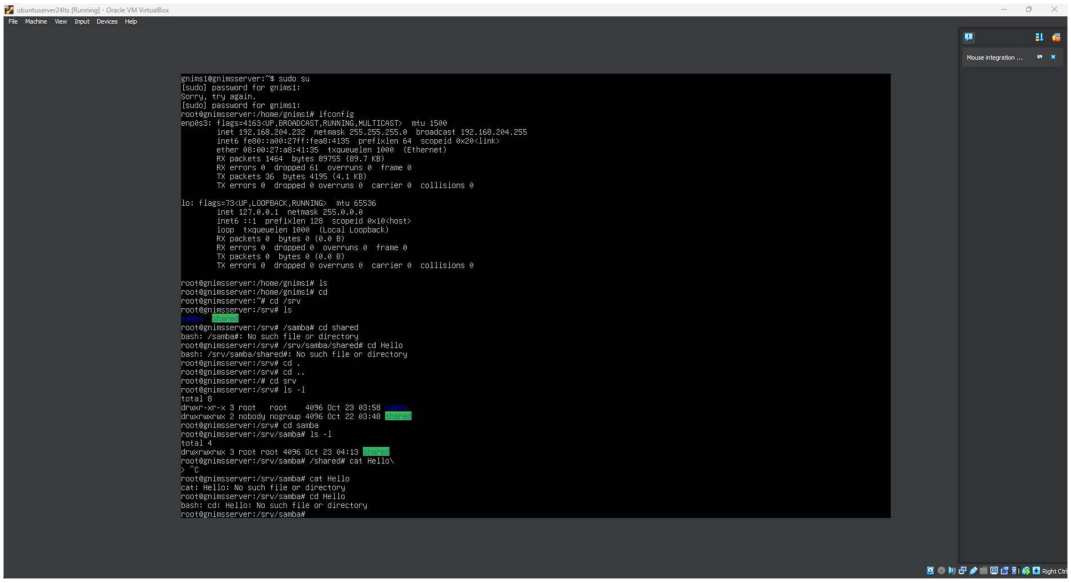




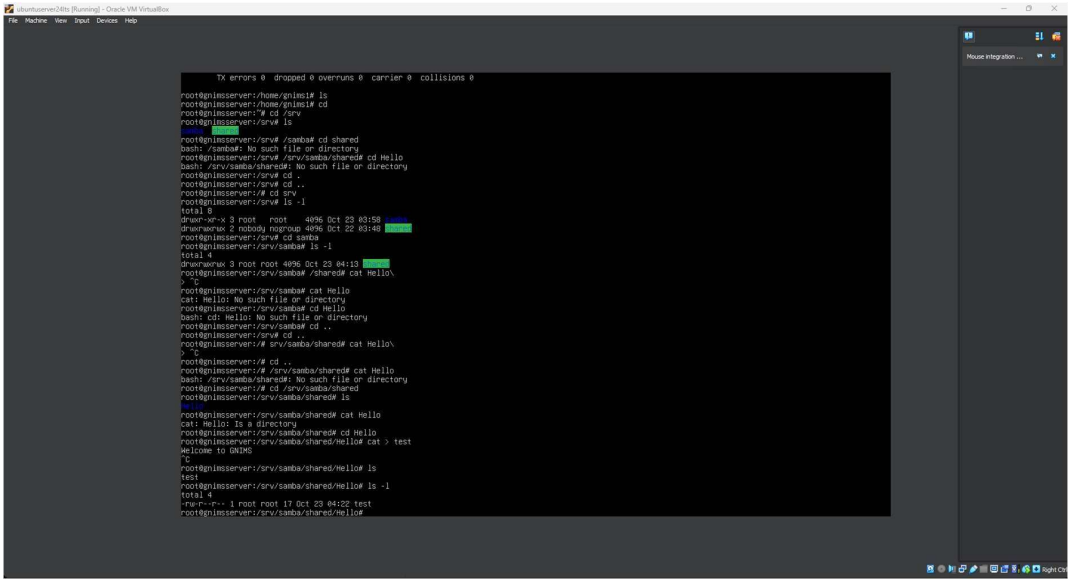

ubuntuserver24lts [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
64 bytes from 192.168.204.101: icmp_seq=16 ttl=127 time=1.32 ms
64 bytes from 192.168.204.101: icmp_seq=17 ttl=127 time=0.942 ms
64 bytes from 192.168.204.101: icmp_seq=18 ttl=127 time=4.28 ms
64 bytes from 192.168.204.101: icmp_seq=19 ttl=127 time=0.539 ms
64 bytes from 192.168.204.101: icmp_seq=20 ttl=127 time=1.42 ms
64 bytes from 192.168.204.101: icmp_seq=21 ttl=127 time=1.33 ms
64 bytes from 192.168.204.101: icmp_seq=22 ttl=127 time=1.64 ms
64 bytes from 192.168.204.101: icmp_seq=23 ttl=127 time=0.918 ms
64 bytes from 192.168.204.101: icmp_seq=24 ttl=127 time=1.19 ms
64 bytes from 192.168.204.101: icmp_seq=25 ttl=127 time=1.56 ms
64 bytes from 192.168.204.101: icmp_seq=26 ttl=127 time=1.24 ms
64 bytes from 192.168.204.101: icmp_seq=27 ttl=127 time=1.27 ms
64 bytes from 192.168.204.101: icmp_seq=28 ttl=127 time=1.53 ms
64 bytes from 192.168.204.101: icmp_seq=29 ttl=127 time=1.34 ms
64 bytes from 192.168.204.101: icmp_seq=30 ttl=127 time=1.56 ms
64 bytes from 192.168.204.101: icmp_seq=31 ttl=127 time=1.36 ms
64 bytes from 192.168.204.101: icmp_seq=32 ttl=127 time=1.42 ms
64 bytes from 192.168.204.101: icmp_seq=33 ttl=127 time=1.97 ms
64 bytes from 192.168.204.101: icmp_seq=34 ttl=127 time=1.24 ms
64 bytes from 192.168.204.101: icmp_seq=35 ttl=127 time=1.07 ms
64 bytes from 192.168.204.101: icmp_seq=36 ttl=127 time=1.22 ms
64 bytes from 192.168.204.101: icmp_seq=37 ttl=127 time=1.54 ms
64 bytes from 192.168.204.101: icmp_seq=38 ttl=127 time=1.62 ms
64 bytes from 192.168.204.101: icmp_seq=39 ttl=127 time=1.25 ms
64 bytes from 192.168.204.101: icmp_seq=40 ttl=127 time=1.36 ms
64 bytes from 192.168.204.101: icmp_seq=41 ttl=127 time=1.80 ms
64 bytes from 192.168.204.101: icmp_seq=42 ttl=127 time=1.96 ms
64 bytes from 192.168.204.101: icmp_seq=43 ttl=127 time=1.48 ms
64 bytes from 192.168.204.101: icmp_seq=44 ttl=127 time=2.25 ms
64 bytes from 192.168.204.101: icmp_seq=45 ttl=127 time=1.86 ms
64 bytes from 192.168.204.101: icmp_seq=46 ttl=127 time=0.792 ms
64 bytes from 192.168.204.101: icmp_seq=47 ttl=127 time=1.27 ms
64 bytes from 192.168.204.101: icmp_seq=48 ttl=127 time=1.59 ms
64 bytes from 192.168.204.101: icmp_seq=49 ttl=127 time=1.44 ms
64 bytes from 192.168.204.101: icmp_seq=50 ttl=127 time=1.62 ms
64 bytes from 192.168.204.101: icmp_seq=51 ttl=127 time=1.41 ms
64 bytes from 192.168.204.101: icmp_seq=52 ttl=127 time=1.01 ms
^C
--- 192.168.204.101 ping statistics ---
52 packets transmitted, 52 received, 0% packet loss, time 52823ms
rtt min/avg/max/mdev = 0.539/1.491/4.276/0.534 ms
root@gnimsserver:/home/gnims1# pwd
/home/gnims1
root@gnimsserver:/home/gnims1# cd /srv
root@gnimsserver:/srv# ls
samba shared
root@gnimsserver:/srv# cd shared
root@gnimsserver:/srv/shared# ls
root@gnimsserver:/srv/shared# ls -1
root@gnimsserver:/srv/shared#

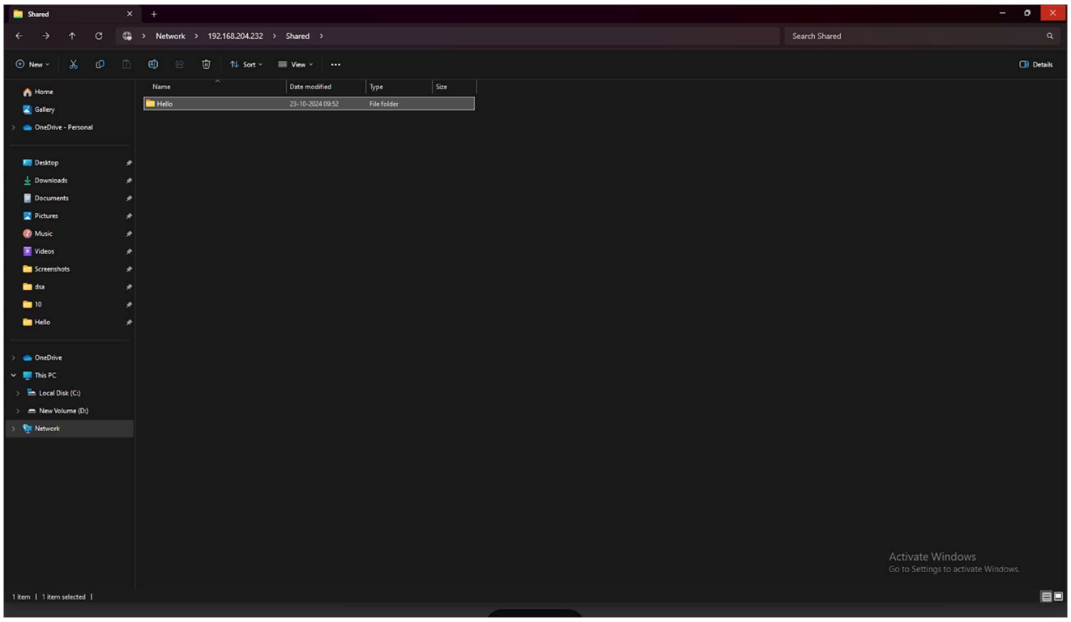

ubuntuserver24lts [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Mouse integration ...
Auto capture keyboard ...
Hit:1 http://in.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:5 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [597 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [431 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [114 kB]
Get:8 http://in.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:9 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [705 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [345 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu noble-updates/universe amd64 c-n-f Metadata [19.8 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:13 http://in.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [208 B]
Get:14 http://in.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:15 http://in.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [21.1 kB]
Get:16 http://in.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [7,156 B]
Get:18 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:19 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [553 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:21 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [13.5 kB]
Get:22 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 3,200 kB in 2s (1,457 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
3 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@gnimsserver:/home/gnims1# sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following upgrades have been deferred due to phasing:
  distro-info-data python3-distupgrade ubuntu-release-upgrader-core
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
root@gnimsserver:/home/gnims1# sudo apt install samba
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
samba is already the newest version (2:4.19.5+dfsg-4ubuntu9).
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
root@gnimsserver:/home/gnims1# sudo mkdir /srv/samba/shared
mkdir: cannot create directory '/srv/samba/shared': No such file or directory
root@gnimsserver:/home/gnims1# sudo mkdir/srv/samba/shared
sudo: mkdir/srv/samba/shared: command not found
root@gnimsserver:/home/gnims1# sudo mkdir -p /srv/samba/shared
root@gnimsserver:/home/gnims1# ls -l /srv/samba/
total 4
drwxr-xr-x 2 root root 4096 Oct 23 03:58 shared
root@gnimsserver:/home/gnims1# sudo


ubuntuserver24lts [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Mouse integration ...
System information as of Wed Oct 23 04:11:01 AM UTC 2024
System load: 0.59 Processes: 177
Usage of /: 15.5% of 47.93GB Users logged in: 0
Memory usage: 3% IPv4 address for enp0s3: 192.168.204.232
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
gnims1@gnimsserver:~$ sudo su
[sudo] password for gnims1:
Sorry, try again.
[sudo] password for gnims1:
root@gnimsserver:/home/gnims1# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.204.232 netmask 255.255.255.0 broadcast 192.168.204.255
inet6 fe80::a00:27ff:fea8:4135 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:a8:41:35 txqueuelen 1000 (Ethernet)
RX packets 1464 bytes 89755 (89.7 KB)
RX errors 0 dropped 61 overruns 0 frame 0
TX packets 36 bytes 4195 (4.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@gnimsserver:/home/gnims1# ls
root@gnimsserver:/home/gnims1# cd
root@gnimsserver:~# cd /srv
root@gnimsserver:/srv# ls
samba shared
root@gnimsserver:/srv# /samba# cd shared
bash: /samba#: No such file or directory
root@gnimsserver:/srv# /srv/samba/shared# cd Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/srv#


gnims1@gnimsserver:~$ sudo su
[sudo] password for gnims1:
Sorry, try again.
[sudo] password for gnims1:
root@gnimsserver:/home/gnims1# ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 192.168.204.232 netmask 255.255.255.0 broadcast 192.168.204.255
inet6 fe80::a00:27ff:fe8d:4135 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:8d:41:35 txqueuelen 1000 (Ethernet)
RX packets 1464 bytes 89755 (89.7 KB)
RX errors 0 dropped 61 overruns 0 frame 0
TX packets 36 bytes 4195 (4.1 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@gnimsserver:/home/gnims1# ls
root@gnimsserver:/home/gnims1# cd
root@gnimsserver:~# cd /srv
root@gnimsserver:/srv# ls
samba shared
root@gnimsserver:/srv# /samba# cd shared
bash: /samba#: No such file or directory
root@gnimsserver:/srv# /srv/samba/shared# cd Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/srv# cd .
root@gnimsserver:/srv# cd ..
root@gnimsserver:/# cd srv
root@gnimsserver:/srv# ls -l
total 8
drwxr-xr-x 3 root   root   4096 Oct 23 03:58 samba
drwxrwxrwx 2 nobody nogroup 4096 Oct 22 03:48 shared
root@gnimsserver:/srv# cd samba
root@gnimsserver:/srv/samba# ls -l
total 4
drwxrwxrwx 3 root root 4096 Oct 23 04:13 shared
root@gnimsserver:/srv/samba# /shared# cat Hello\
> ^C
root@gnimsserver:/srv/samba# cat Hello
cat: Hello: No such file or directory
root@gnimsserver:/srv/samba# cd Hello
bash: cd: Hello: No such file or directory
root@gnimsserver:/srv/samba#


inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@gnimsserver:/home/gnims1# ls
root@gnimsserver:/home/gnims1# cd
root@gnimsserver:~# cd /srv
root@gnimsserver:/srv# ls
samba shared
root@gnimsserver:/srv# /samba# cd shared
bash: /samba#: No such file or directory
root@gnimsserver:/srv# /srv/samba/shared# cd Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/srv# cd .
root@gnimsserver:/srv# cd ..
root@gnimsserver:/# cd srv
root@gnimsserver:/srv# ls -l
total 8
drwxr-xr-x 3 root   root   4096 Oct 23 03:58 samba
drwxrwxrwx 2 nobody nogroup 4096 Oct 22 03:48 shared
root@gnimsserver:/srv# cd samba
root@gnimsserver:/srv/samba# ls -l
total 4
drwxrwxrwx 3 root root 4096 Oct 23 04:13 shared
root@gnimsserver:/srv/samba# /shared# cat Hello\
> ^C
root@gnimsserver:/srv/samba# cat Hello
cat: Hello: No such file or directory
root@gnimsserver:/srv/samba# cd Hello
bash: cd: Hello: No such file or directory
root@gnimsserver:/srv/samba# cd ..
root@gnimsserver:/srv# cd ..
root@gnimsserver:/# srv/samba/shared# cat Hello\
> ^C
root@gnimsserver:/# cd ..
root@gnimsserver:/# /srv/samba/shared# cat Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/# cd /srv/samba/shared
root@gnimsserver:/srv/samba/shared# ls
Hello
root@gnimsserver:/srv/samba/shared# cat Hello
cat: Hello: Is a directory
root@gnimsserver:/srv/samba/shared# cd Hello
root@gnimsserver:/srv/samba/shared/Hello# cat > test
Welcome to GNIMS


TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
root@gnimsserver:/home/gnims1# ls
root@gnimsserver:/home/gnims1# cd
root@gnimsserver:~# cd /srv
root@gnimsserver:/srv# ls
samba shared
root@gnimsserver:/srv# /samba# cd shared
bash: /samba#: No such file or directory
root@gnimsserver:/srv# /srv/samba/shared# cd Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/srv# cd .
root@gnimsserver:/srv# cd ..
root@gnimsserver:/# cd srv
root@gnimsserver:/srv# ls -l
total 8
drwxr-xr-x 3 root   root   4096 Oct 23 03:58 samba
drwxrwxrwx 2 nobody nogroup 4096 Oct 22 03:48 shared
root@gnimsserver:/srv# cd samba
root@gnimsserver:/srv/samba# ls -l
total 4
drwxrwxrwx 3 root root 4096 Oct 23 04:13 shared
root@gnimsserver:/srv/samba# /shared# cat Hello\
> ^C
root@gnimsserver:/srv/samba# cat Hello
cat: Hello: No such file or directory
root@gnimsserver:/srv/samba# cd Hello
bash: cd: Hello: No such file or directory
root@gnimsserver:/srv/samba# cd ..
root@gnimsserver:/srv# cd ..
root@gnimsserver:/# srv/samba/shared# cat Hello\
> ^C
root@gnimsserver:/# cd ..
root@gnimsserver:/# /srv/samba/shared# cat Hello
bash: /srv/samba/shared#: No such file or directory
root@gnimsserver:/# cd /srv/samba/shared
root@gnimsserver:/srv/samba/shared# ls
Hello
root@gnimsserver:/srv/samba/shared# cat Hello
cat: Hello: Is a directory
root@gnimsserver:/srv/samba/shared# cd Hello
root@gnimsserver:/srv/samba/shared/Hello# cat > test
Welcome to GNIMS
^C
root@gnimsserver:/srv/samba/shared/Hello# ls
test
root@gnimsserver:/srv/samba/shared/Hello# ls -l
total 4
-rw-r--r-- 1 root root 17 Oct 23 04:22 test
root@gnimsserver:/srv/samba/shared/Hello#



Reference:-

https://cloud.google.com/storage/docs

https://workspace.google.com/intl/en_in/products/drive/

Practical No. :-10

Aim:- To develop applications using Google App Engine by using Eclipse IDE.

Theory:-

Google App Engine (GAE) is a platform-as-a-service (PaaS) offering by Google Cloud that allows developers to build and deploy scalable web applications. It supports multiple programming languages such as Java, Python, Node.js, and others. By using **Google App Engine** with **Eclipse IDE**, developers can efficiently create, test, and deploy cloud-based applications. Eclipse, an integrated development environment (IDE), simplifies the development process with its robust features and plugins for Google Cloud. The App Engine provides automatic scaling, load balancing, and a managed environment, allowing developers to focus on writing code without worrying about server management. This setup is ideal for building web applications that require high availability, elasticity, and seamless deployment to the cloud, leveraging the capabilities of Google's infrastructure.

Algorithm/ Concept:-

- ☐ Setup Environment:
- Install Eclipse IDE and set up the Google Cloud SDK.
- Install the Google Cloud Tools for Eclipse plugin.
- Create a Google Cloud project and enable Google App Engine.
- ☐ Create a New App Engine Project:
- Open Eclipse and create a new project (File > New > Project > Google App Engine Standard Project).
- Choose the preferred programming language (e.g., Java) and configure the project settings.
- ☐ Develop the Application:
- Implement the application logic using Servlets or frameworks like Spring Boot.
- Define the application's behavior and response using appropriate API endpoints.
- ☐ Configure App Engine:
- Set up the app.yaml or WEB-INF configuration files for deployment.
- Specify runtime details, environment variables, and other App Engine configurations.
- ☐ Local Testing:
- Run the application locally using the Eclipse App Engine Local Server.
- Debug and test the application for errors.
- ☐ Deploy to Google App Engine:
- Right-click the project in Eclipse and select Deploy > Deploy to App Engine.
- Authenticate with Google Cloud and deploy the application.
- ☐ Access the Application:
- Open the provided App Engine URL to access the deployed application.
- Monitor the app using Google Cloud Console for performance and logs.
- ☐ Termination:
- Shut down local services and close the Eclipse IDE after successful deployment.

Code/ Program:-

app.yaml File:

```
application: ae-01-trivial version: 1
runtime: python api_version: 1 handlers:
- url: /.*
script: index.py
```

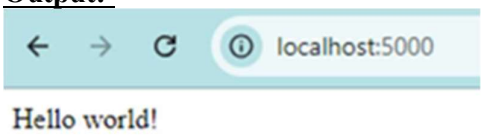
index.py File:

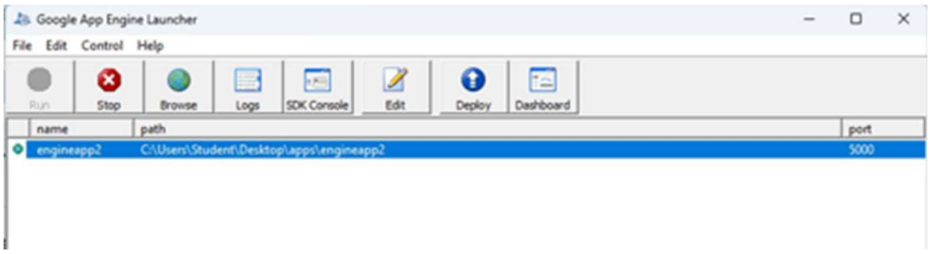
```
print 'Content-Type: text/plain'
print ' '
print 'Hello there!'
```

Execution/ Compilation Screenshot:-

- **Local Testing**:
  - Open the Google App Engine Launcher.
  - Add your application folder to the launcher.
  - Click "Run" to start the server.
  - Access the application at http://localhost:8080.
- **Editing & Debugging**:
  - Modify index.py to update the output text.
  - Refresh the browser to see changes immediately.
- **Deployment**:
  - Use the "Deploy" option in the Google App Engine Launcher to upload the application to the cloud.

Output:-





Reference:-

https://cloud.google.com/eclipse/docs/creating-new-webapp

https://marketplace.eclipse.org/free-tagging/app-engine