MCA A.Y. 2023-2024

Journal of Distributed System and Cloud Computing LAB

Audyogik Shikshan Mandal's
Institute of Management & Computer Studies, Thane

Candidate Full Name:	**Omkar Bharat Khandare**
Roll No.:	**MC2223044**
Course:	**Master of Computer Applications (MCA)**
Semester:	**Semester-III**
Subject Faculty Incharge:	**Prof. Mahesh Mahajan**

Audyogik Shikshan Mandal's Institute of

Management & Computer Studies,Thane

Certificate

This is to certify that Mr.**Omkar Bharat Khandare** Student of **MCA** Course, First year, **Semester 3,** Roll No. **MC2223044** has successfully completed the required number of practical in the subject of **Distributed System and Cloud Computing LAB** as prescribed by the **University of Mumbai** under our supervision during the academic year 2023-2024.

Practical In-Charge **Internal Examiner**
Date: **Date:**

External Examiner **Director**
Date: **IMCOST**

College Seal:

INDEX

Practical No.	Practical Name
1	To develop a program for multi-client chat server using Socket.
2	To implement a Server calculator using RPC concept. (Make use of datagram).
3	To implement a Date Time Server using RPC concept. (Make use of datagram).
4	To retrieve day, time and date function from server to client. This program should display server day, time and date. (Use Concept RMI for accessing multiple data access objects).
5	The client should provide an equation to the server through an interface. The server will solve the expression given by the client.
6	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.
7	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.
8	Implementation of mutual exclusion using Token ring algorithm.
9	Implementation of Identity Management.
10	Implementation of Storage as a Service using Google Docs.

PRACTICAL 1

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

To develop this experiment we are using the Thread concepts for developing a multi chat application using JDK.

There are two ways to create thread in java:

a) Extending the Thread Class.
b) And by implementing the Runnable interface

I. Multithreading in java is a process where multiple threads are executed simultaneously.
II. In a multi-threaded program, the program consist of two or more process that can run threads concurrently and each process(thread) can handle a different task at the same time where it make optimal use of resources.
III. The process of executing multiple threads simultaneously is known as multithreading.
IV. In a Single Thread Socket program where only one clients can communicate with the server.
V. If you want to connect or communicate with more than one client then we have to write the code using Multithreaded Socket Programming.

Server Side Programming(Server.java):

1. Server class : The main server implementation is easy and similar to the previous article. The following points will help understand Server implementation :

1. The server runs an infinite loop to keep accepting incoming requests.
2. When a request comes, it assigns a new thread to handle the communication part.
3. The server also stores the client name into a vector, to keep a track of connected devices. The vector stores the thread object corresponding to the current request. The helper class uses this vector to find the name of recipient to which message is to be delivered. As this vector holds all the streams, handler class can use it to successfully deliver messages to specific clients.
4. Invoke the start() method.

2. ClientHandler class : Similar to previous article, we create a helper class for handling various requests. This time, along with the socket and streams, we introduce a name variable. This will hold the name of the client that is connected to the server. The following points will help understand ClientHandler implementation

MultithreadedSocketServer.java

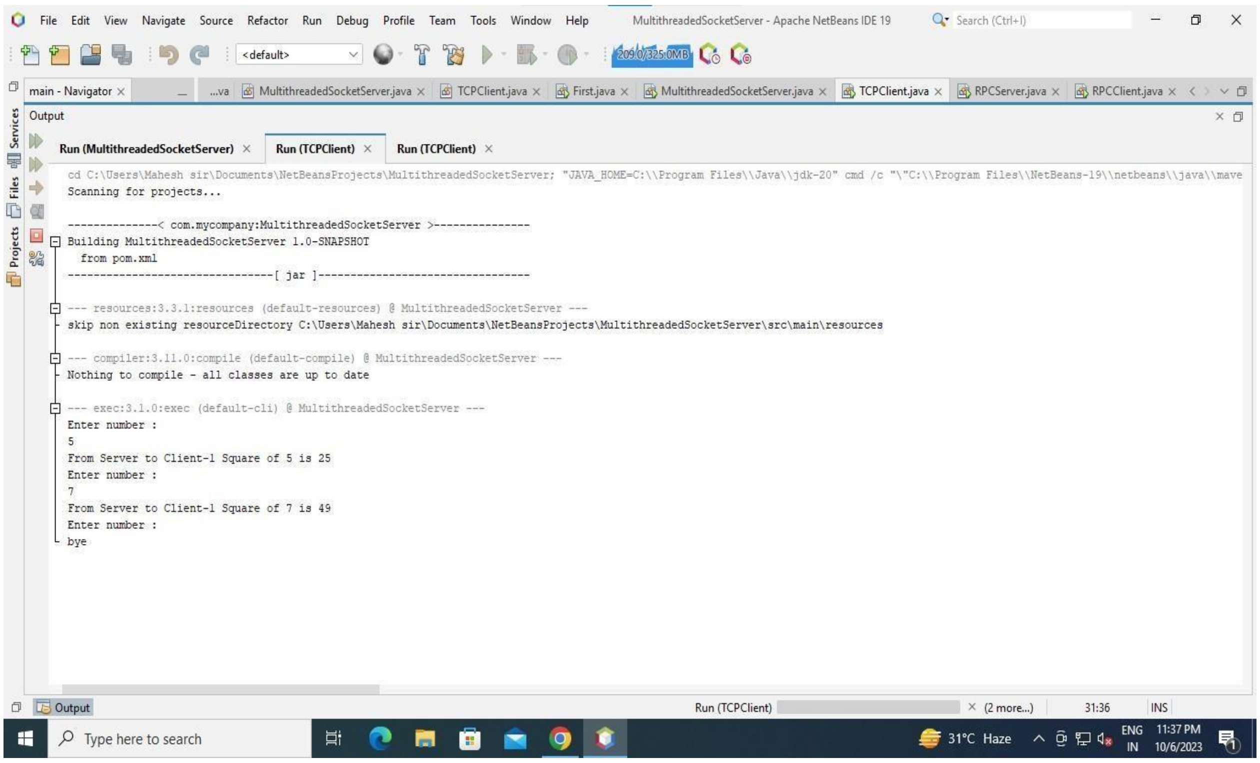
```
import java.net.*; import
java.io.*;
public class MultithreadedSocketServer
{ public static void main(String[] args) throws
 Exception
{
try
{
 ServerSocket server=new ServerSocket(8888); int
counter=0;
 System.out.println("Server Started....."); while(true)
{ counter++;
 Socket serverClient=server.accept(); //server accept the client connection request
 System.out.println(" >> " + "Client No:" + counter + " started!");
 ServerClientThread sct = new ServerClientThread(serverClient,counter); //send the request to
a separate thread sct.start();
}
 } catch(Exception
e)
{
 System.out.println(e);
 }
 } }
class ServerClientThread extends Thread
{
 Socket serverClient; int
    clientNo;        int
squre;
 ServerClientThread(Socket inSocket,int counter)
{ serverClient =
 inSocket;
clientNo=counter;
 } public void
 run()
{ try
{
DataInputStream inStream = new DataInputStream(serverClient.getInputStream());
DataOutputStream outStream = new DataOutputStream(serverClient.getOutputStream());
```

```
 String clientMessage="", serverMessage=""; while(!clientMessage.equals("bye"))
{ clientMessage=inStream.readUTF();
System.out.println("From Client-" +clientNo+ ": Number is :"+clientMessage); squre =
 Integer.parseInt(clientMessage) * Integer.parseInt(clientMessage); serverMessage="From
 Server to Client-" + clientNo + " Square of " + clientMessage + " is "
+squre; outStream.writeUTF(serverMessage);
 outStream.flush();
 }
 inStream.close();
outStream.close(); serverClient.close();
 } catch(Exception
ex)
{
 System.out.println(ex);
 } finally
{
 System.out.println("Client -" + clientNo + " exit!! ");
}
 }
}
```

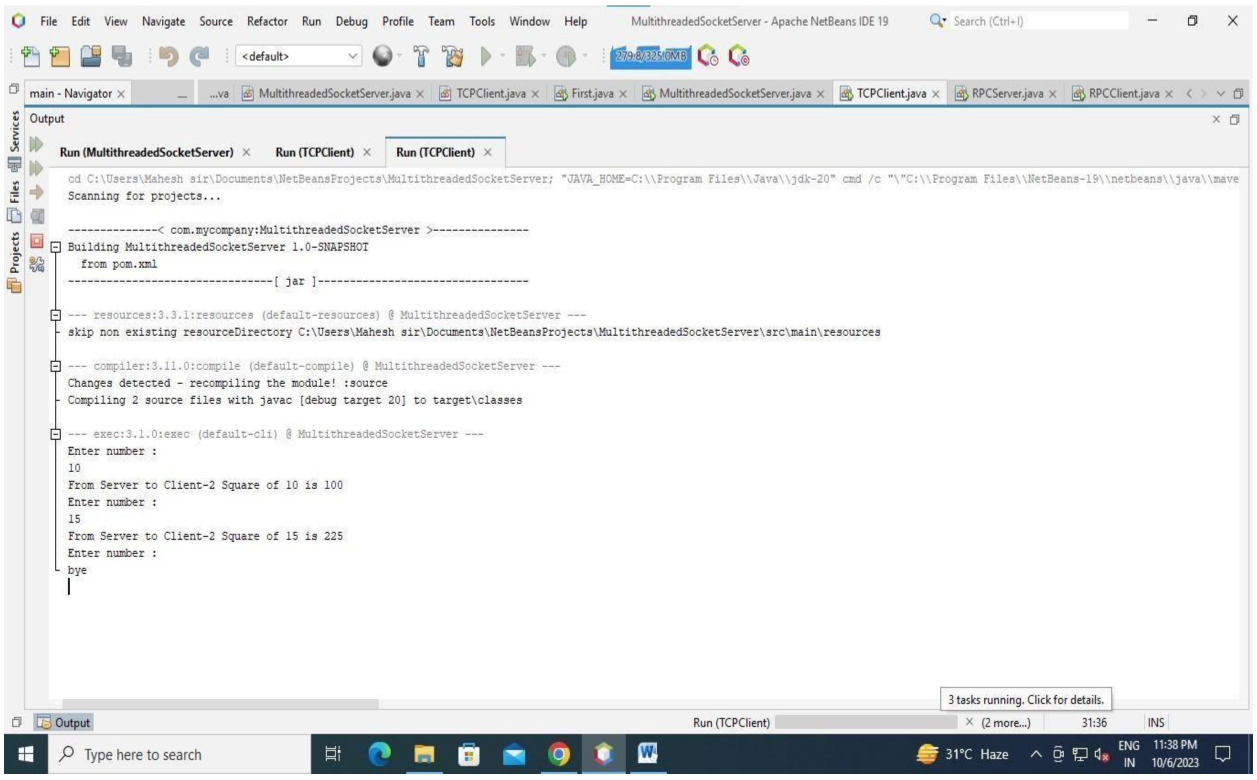
TCPClient.java

```
import     java.net.*; import
   java.io.*; public class
TCPClient
{ public static void main(String[] args) throws Exception
{
try
{
 Socket socket=new Socket("127.0.0.1",8888);
 DataInputStream inStream=new DataInputStream(socket.getInputStream());
 DataOutputStream outStream=new DataOutputStream(socket.getOutputStream());
 BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
 String clientMessage="",serverMessage=""; while(!clientMessage.equals("bye"))
{
 System.out.println("Enter number :");
clientMessage=br.readLine();
outStream.writeUTF(clientMessage); outStream.flush();
serverMessage=inStream.readUTF();
System.out.println(serverMessage);
 } outStream.close();
outStream.close(); socket.close();
 } catch(Exception
e)
{
 System.out.println(e);
}
 }
}
```

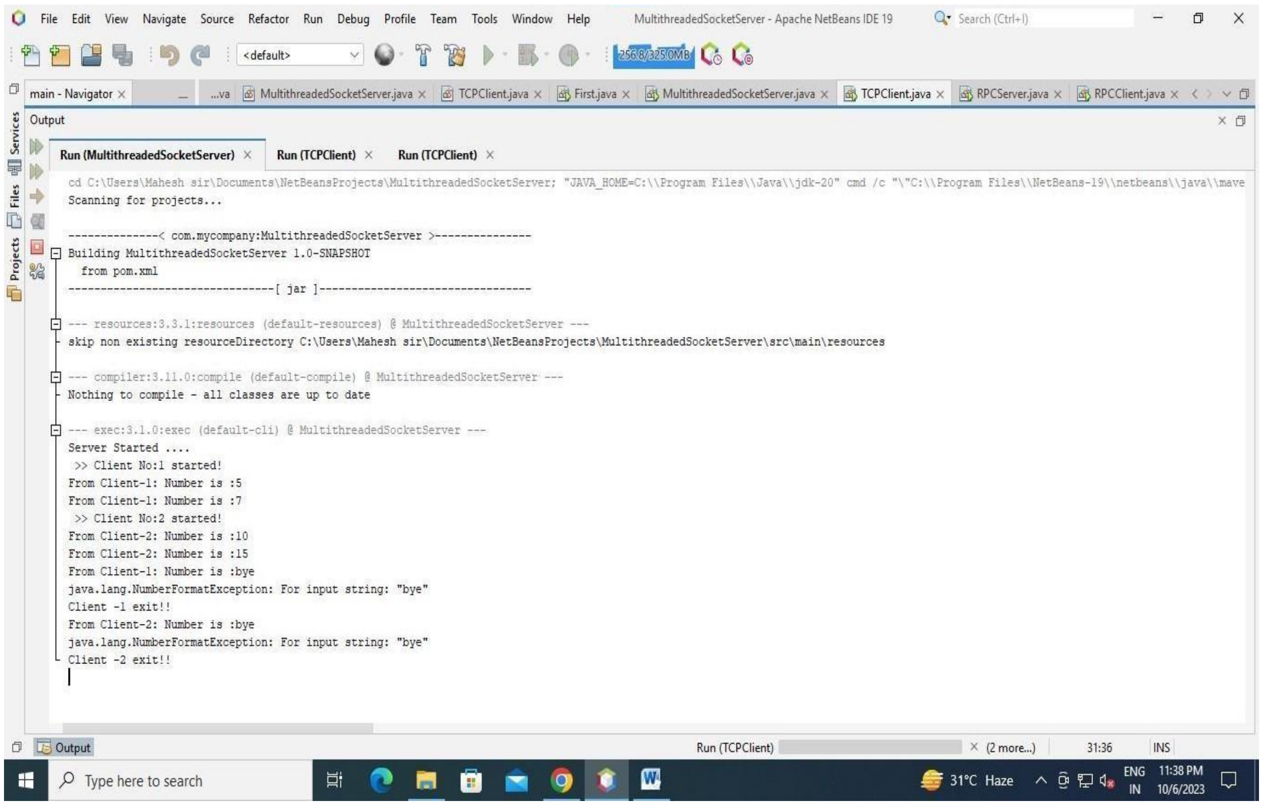
OUTPUT


MultithreadedSocketServer - Apache NetBeans IDE 19
Output
Run (MultithreadedSocketServer)
Run (TCPClient)
Run (TCPClient)

```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\mave
Scanning for projects...

--------------< com.mycompany:MultithreadedSocketServer >---------------
Building MultithreadedSocketServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ MultithreadedSocketServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ MultithreadedSocketServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ MultithreadedSocketServer ---
Enter number :
5
From Server to Client-1 Square of 5 is 25
Enter number :
7
From Server to Client-1 Square of 7 is 49
Enter number :
bye
```


MultithreadedSocketServer - Apache NetBeans IDE 19
Run (MultithreadedSocketServer)
Run (TCPClient)
Run (TCPClient)
3 tasks running. Click for details.

```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\mave
Scanning for projects...

--------------< com.mycompany:MultithreadedSocketServer >---------------
Building MultithreadedSocketServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ MultithreadedSocketServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ MultithreadedSocketServer ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 20] to target\classes

--- exec:3.1.0:exec (default-cli) @ MultithreadedSocketServer ---
Enter number :
10
From Server to Client-2 Square of 10 is 100
Enter number :
15
From Server to Client-2 Square of 15 is 225
Enter number :
bye
```


MultithreadedSocketServer - Apache NetBeans IDE 19
Run (MultithreadedSocketServer)
Run (TCPClient)
Run (TCPClient)

```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\mave
Scanning for projects...

--------------< com.mycompany:MultithreadedSocketServer >---------------
Building MultithreadedSocketServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ MultithreadedSocketServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\MultithreadedSocketServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ MultithreadedSocketServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ MultithreadedSocketServer ---
Server Started ....
 >> Client No:1 started!
From Client-1: Number is :5
From Client-1: Number is :7
 >> Client No:2 started!
From Client-2: Number is :10
From Client-2: Number is :15
From Client-1: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -1 exit!!
From Client-2: Number is :bye
java.lang.NumberFormatException: For input string: "bye"
Client -2 exit!!
```

PRACTICAL 2

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

A remote procedure call is an interprocess communication technique that is used for clientserver based applications. It is also known as a subroutine call or a function call. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

The sequence of events in a remote procedure call are given as follows:

- The client stub is called by the client.
- The client stub makes a system call to send the message to the server and puts the parameters in the message.
- The message is sent from the client to the server by the client's operating system.
- The message is passed to the server stub by the server operating system.
- The parameters are removed from the message by the server stub.
- Then, the server procedure is called by the server stub.

Stub: The stub is a client-side object that functions as a gateway. It is via which all outbound requests are routed. It is a client-side object that represents a distant object. The caller does the following duties when calling a method on the stub object:

1) It establishes a connection with a remote Virtual Machine (JVM), then writes and sends (marshals) the parameters to the remote Virtual Machine (JVM).

2) It sits and waits for the outcome. It reads (un-marshals) the return value or exception after receiving the result, and then returns the value to the caller.

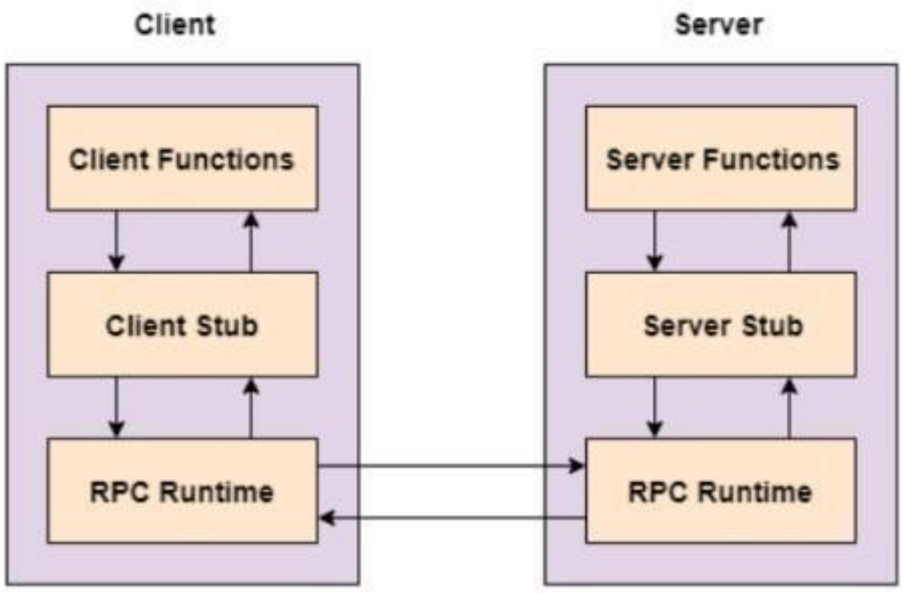


Figure 1: working of RPC

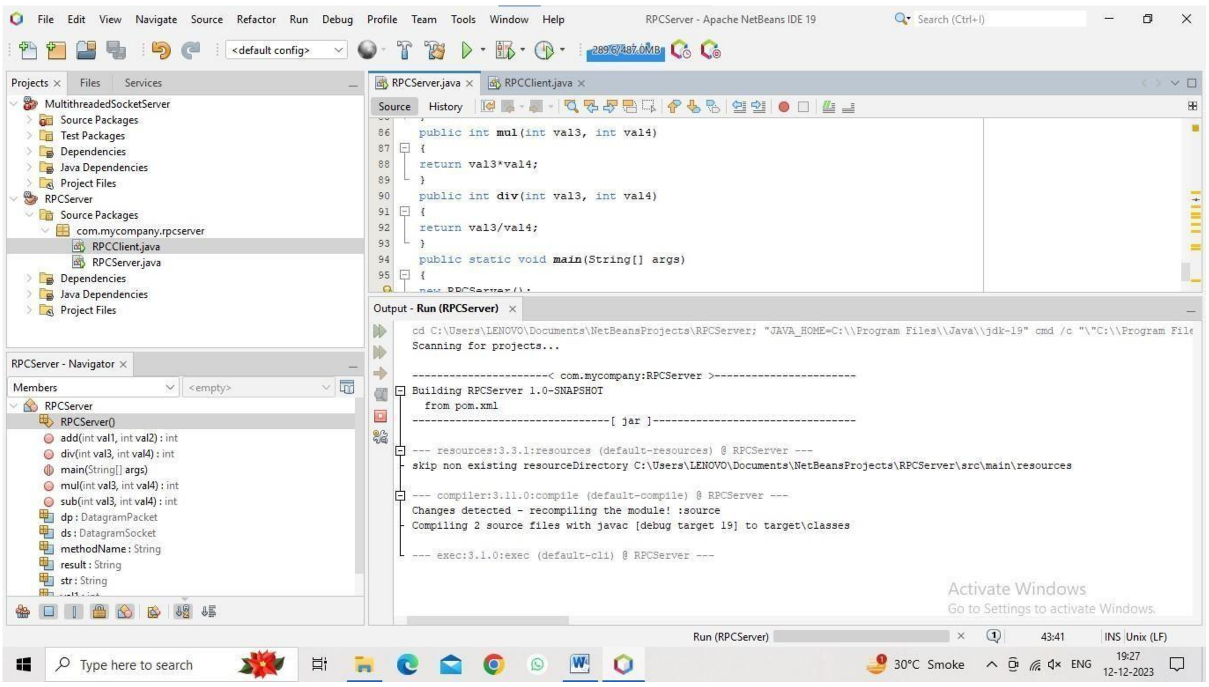
RPCServer.Java

```
import java.util.*; import
java.net.*;
class RPCServer
{
DatagramSocket ds;
DatagramPacket    dp;    String
str,methodName,result;         int
val1,val2;
RPCServer()
{ try {
ds=new DatagramSocket(1200); byte
   b[]=new          byte[4096];
while(true) {
dp=new DatagramPacket(b,b.length);
ds.receive(dp);
str=new String(dp.getData(),0,dp.getLength()); if(str.equalsIgnoreCase("q")) {
System.exit(1);
} else
{
StringTokenizer st = new StringTokenizer(str," "); int
i=0; while(st.hasMoreTokens())
{
String token=st.nextToken(); methodName=token;
val1 = Integer.parseInt(st.nextToken()); val2 =
Integer.parseInt(st.nextToken());
}
}
System.out.println(str);
InetAddress ia = InetAddress.getLocalHost(); if(methodName.equalsIgnoreCase("add"))
{
result= "" + add(val1,val2);
}
else if(methodName.equalsIgnoreCase("sub"))
{
result= "" + sub(val1,val2);
}
else if(methodName.equalsIgnoreCase("mul"))
{
result= "" + mul(val1,val2);
}
else if(methodName.equalsIgnoreCase("div"))
{
result= "" + div(val1,val2);
}
byte b1[]=result.getBytes();
DatagramSocket ds1 = new DatagramSocket();
DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
System.out.println("result : "+result+"\n"); ds1.send(dp1);
} }
catch (Exception e)
{
e.printStackTrace();
} }
public int add(int val1, int val2)
{
return val1+val2;
}
public int sub(int val3, int val4)
{
return val3-val4;
}
public int mul(int val3, int val4)
{
return val3*val4;
}
public int div(int val3, int val4)
{
return val3/val4;
}
public static void main(String[] args)
{
new RPCServer();
}
}
```

RPCClient.Java

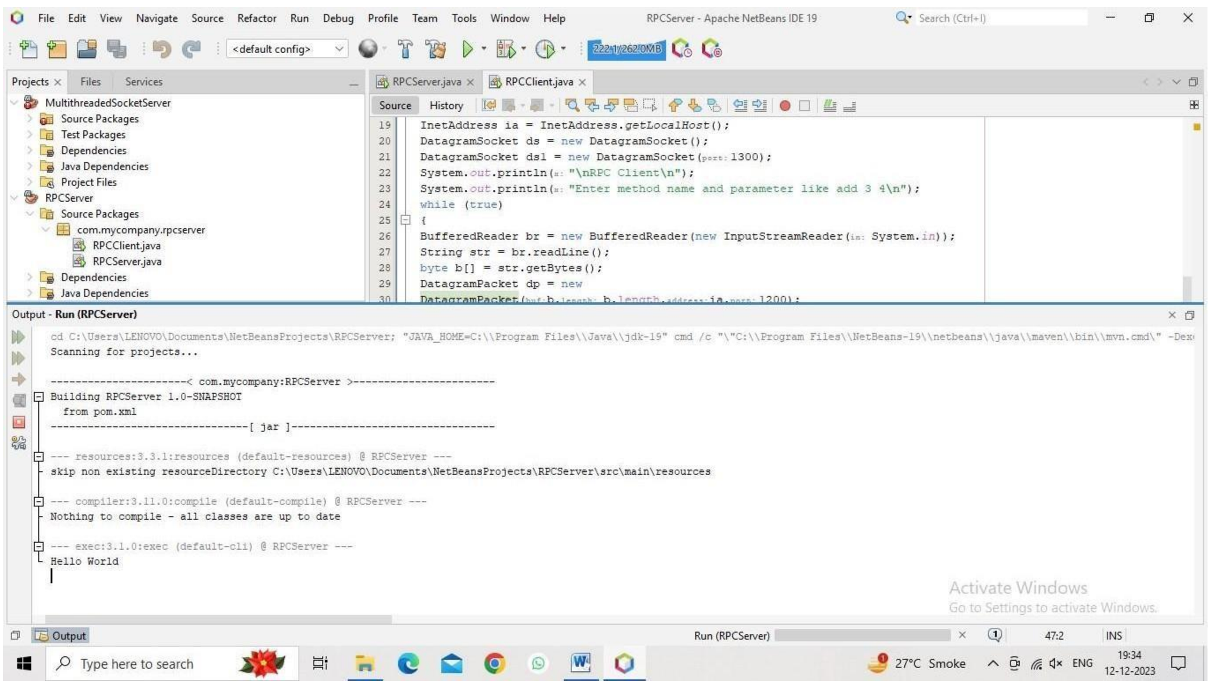
```
import     java.io.*;
import java.net.*; class
RPCClient
{
RPCClient()
{ try
{
InetAddress ia = InetAddress.getLocalHost();
DatagramSocket ds = new DatagramSocket();
DatagramSocket ds1 = new DatagramSocket(1300);
System.out.println("\nRPC Client\n");
System.out.println("Enter method name and parameter like add 3 4\n"); while
(true)
{
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String str = br.readLine(); byte b[]
= str.getBytes();
DatagramPacket dp = new
DatagramPacket(b,b.length,ia,1200);  ds.send(dp);  dp  =
new DatagramPacket(b,b.length); ds1.receive(dp);
String s = new String(dp.getData(),0,dp.getLength());
System.out.println("\nResult = " + s + "\n");
} }
catch (Exception e)
{
e.printStackTrace();
} }
public static void main(String[] args)
{
new RPCClient();
}
}
```

OUTPUT



```
public int mul(int val3, int val4)
{
return val3*val4;
}
public int div(int val3, int val4)
{
return val3/val4;
}
public static void main(String[] args)
{
```

```
cd C:\Users\LENOVO\Documents\NetBeansProjects\RPCServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-19" cmd /c "\"C:\\Program File
Scanning for projects...

----------------------< com.mycompany:RPCServer >-----------------------
Building RPCServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ RPCServer ---
skip non existing resourceDirectory C:\Users\LENOVO\Documents\NetBeansProjects\RPCServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ RPCServer ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 19] to target\classes

--- exec:3.1.0:exec (default-cli) @ RPCServer ---
```



```
InetAddress ia = InetAddress.getLocalHost();
DatagramSocket ds = new DatagramSocket();
DatagramSocket ds1 = new DatagramSocket(port: 1300);
System.out.println(x: "\nRPC Client\n");
System.out.println(x: "Enter method name and parameter like add 3 4\n");
while (true)
{
BufferedReader br = new BufferedReader(new InputStreamReader(in: System.in));
String str = br.readLine();
byte b[] = str.getBytes();
DatagramPacket dp = new
```

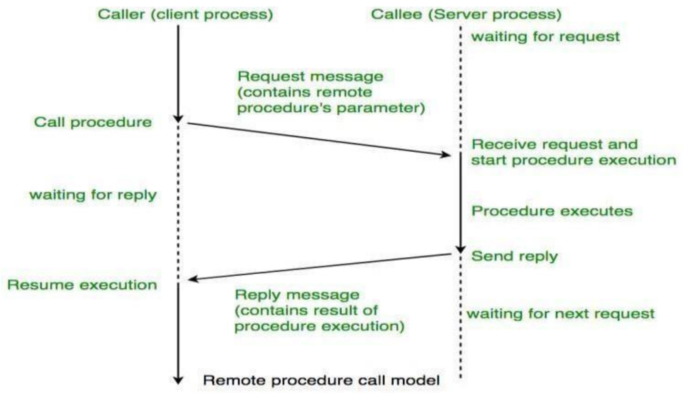
```
cd C:\Users\LENOVO\Documents\NetBeansProjects\RPCServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-19" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\maven\\bin\\mvn.cmd\" -Dex
Scanning for projects...

----------------------< com.mycompany:RPCServer >-----------------------
Building RPCServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ RPCServer ---
skip non existing resourceDirectory C:\Users\LENOVO\Documents\NetBeansProjects\RPCServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ RPCServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ RPCServer ---
Hello World
```

PRACTICAL 3

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

In this example both client and server program run on different command prompt. The time and date of server machine's is displayed on client machine.

Remote Procedure Call (RPC) is a powerful technique for constructing **distributed, client-server based applications**. It is based on extending the conventional local procedure calling so that the **called procedure need not exist in the same address space as the calling procedure**. The two processes may be on the same system, or they may be on different systems with a network connecting them. **When making a Remote Procedure Call:**

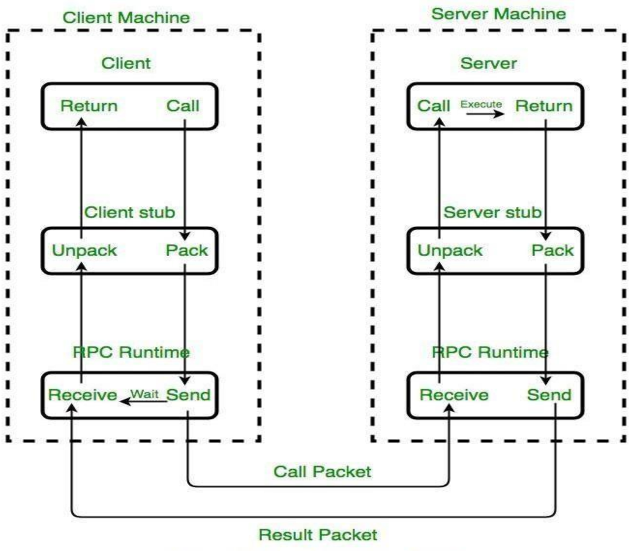

Caller (client process)
Callee (Server process)
waiting for request
Request message (contains remote procedure's parameter)
Call procedure
Receive request and start procedure execution
waiting for reply
Procedure executes
Send reply
Resume execution
Reply message (contains result of procedure execution)
waiting for next request
Remote procedure call model

1. The calling environment is suspended, procedure parameters are transferred across the network to the environment where the procedure is to execute, and the procedure is executed there.

2. When the procedure finishes and produces its results, its results are transferred back to the calling environment, where execution resumes as if returning from a regular procedure call.

NOTE: RPC is especially well suited for client-server **(e.g. queryresponse)** interaction in which the flow of control **alternates between the caller and callee**. Conceptually, the client and server do not both execute at the same time. Instead, the thread of execution jumps from the caller to the callee and then back again.

Working of RPC



Implementation of RPC mechanism

The following steps take place during a RPC:

1. A client invokes a **client stub procedure**, passing parameters in the usual way. The client stub resides within the client's own address space.
2. The client stub **marshalls(pack)** the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a server stub, which **demarshalls(unpack)** the parameters and calls the desired server routine using the regular procedure call mechanism.
5. When the server procedure completes, it returns to the server stub **(e.g., via a normal procedure call return)**, which marshalls the return values into a message. The server stub then hands the message to the transport layer.
6. The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.
7. The client stub demarshalls the return parameters and execution returns to the caller.

Key Considerations for Designing and Implementing RPC Systems are:

- **Security:** Since RPC involves communication over the network, security is a major concern. Measures such as authentication, encryption, and authorization must be implemented to prevent unauthorized access and protect sensitive data.

- **Scalability:** As the number of clients and servers increases, the performance of the RPC system must not degrade. Load balancing techniques and efficient resource utilization are important for scalability.
- **Fault tolerance:** The RPC system should be resilient to network failures, server crashes, and other unexpected events. Measures such as redundancy, failover, and graceful degradation can help ensure fault tolerance.
- **Standardization:** There are several RPC frameworks and protocols available, and it is important to choose a standardized and widely accepted one to ensure interoperability and compatibility across different platforms and programming languages.
- **Performance tuning:** Fine-tuning the RPC system for optimal performance is important. This may involve optimizing the network protocol, minimizing the data transferred over the network, and reducing the latency and overhead associated with RPC calls.

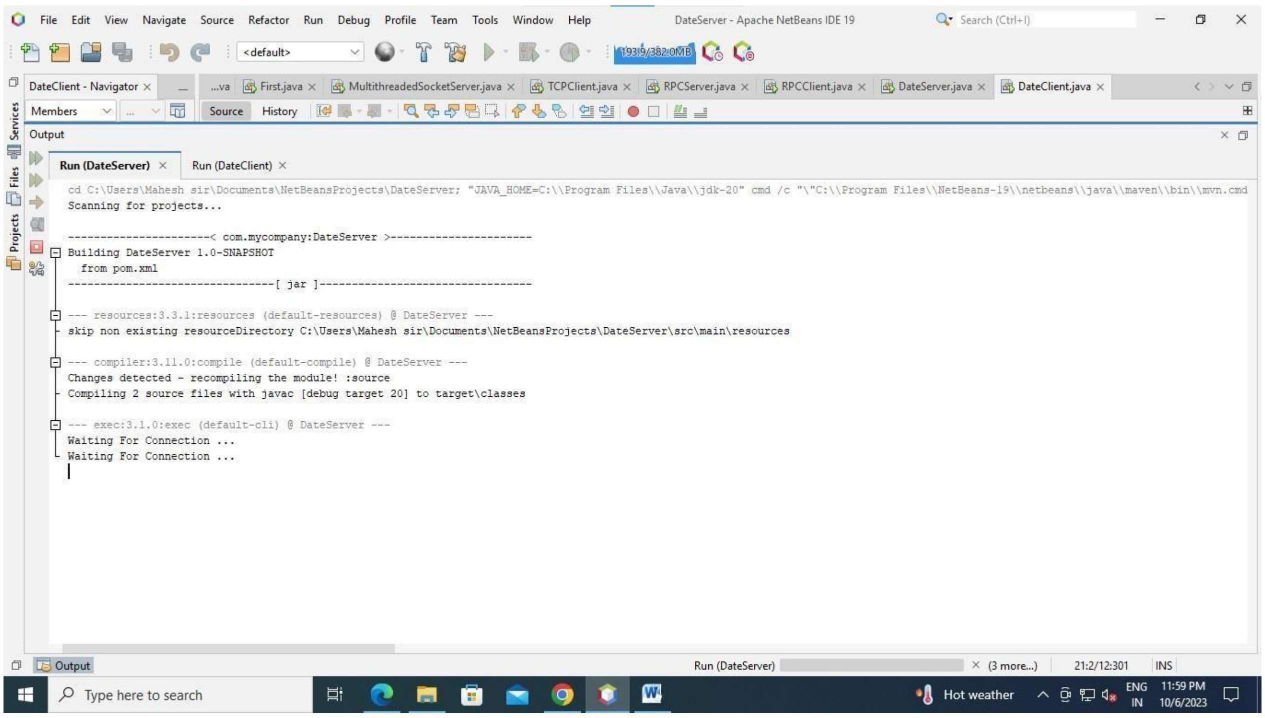
DateServer.java

```
import      java.net.*;
import       java.io.*;
import      java.util.*;
class DateServer
{ public static void main(String args[]) throws
 Exception
 {
 ServerSocket s=new ServerSocket(5217);
 while(true)
 {
 System.out.println("Waiting For Connection ...");
 Socket soc=s.accept();
 DataOutputStream                                    out=new
DataOutputStream(soc.getOutputStream());
out.writeBytes("Server Date: " + (new Date()).toString() + "\n");
out.close(); soc.close();
 }
 }
```

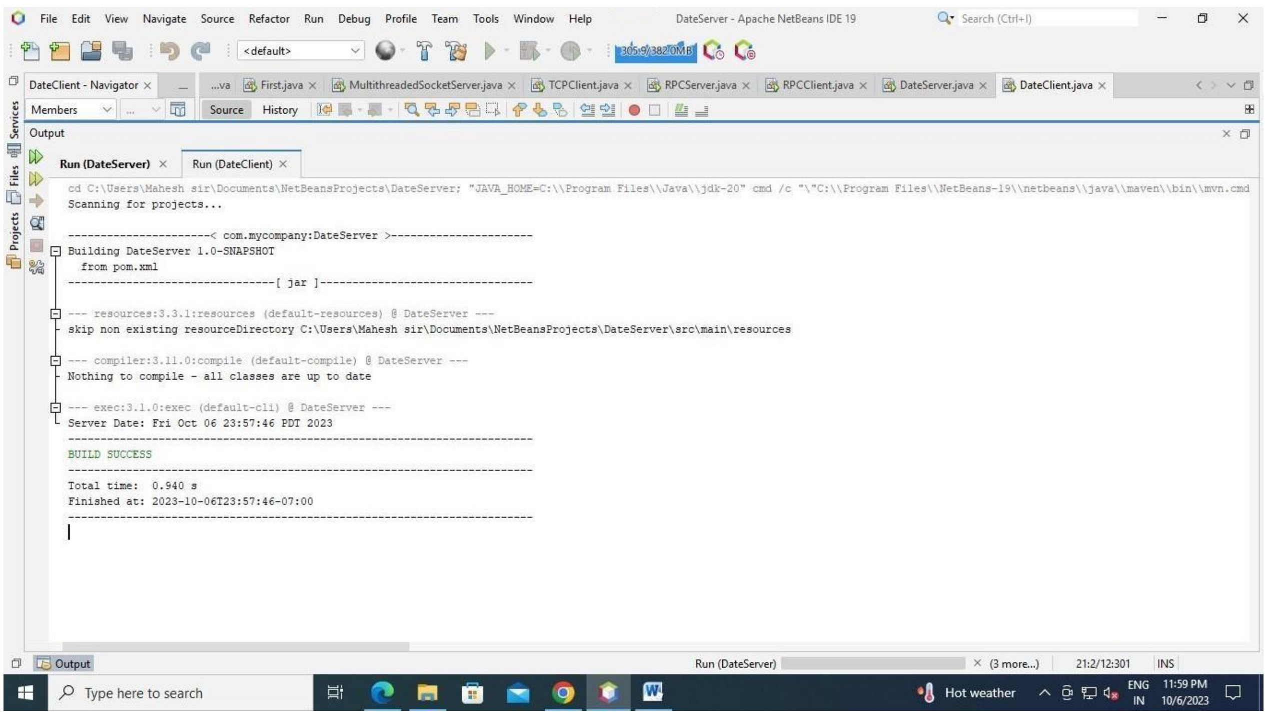
DateClient.java

```
import   java.io.*;
import java.net.*; class
DateClient
{ public static void main(String args[]) throws Exception
 {
 Socket soc=new Socket(InetAddress.getLocalHost(),5217);
BufferedReader in=new BufferedReader(new
InputStreamReader(soc.getInputStream() ));
System.out.println(in.readLine()); }
}
```

OUTPUT:



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\DateServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\maven\\bin\\mvn.cmd
Scanning for projects...

----------------------< com.mycompany:DateServer >----------------------
Building DateServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ DateServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\DateServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ DateServer ---
Changes detected - recompiling the module! :source
Compiling 2 source files with javac [debug target 20] to target\classes

--- exec:3.1.0:exec (default-cli) @ DateServer ---
Waiting For Connection ...
Waiting For Connection ...
```



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\DateServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java\\maven\\bin\\mvn.cmd
Scanning for projects...

----------------------< com.mycompany:DateServer >----------------------
Building DateServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ DateServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\DateServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ DateServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ DateServer ---
Server Date: Fri Oct 06 23:57:46 PDT 2023
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.940 s
Finished at: 2023-10-06T23:57:46-07:00
------------------------------------------------------------------------
```

PRACTICAL 4

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

The client application needs only two files, remote interface and client application. In the rmi application, both client and server interact with the remote interface. The client application invokes methods on the proxy object; RMI sends the request to the remote JVM. The return value is sent back to the proxy object and then to the client application.

1) create the remote interface: For creating the remote interface, extend the Remote interface and declare the RemoteException with all the methods of the remote interface. Here, we are creating a remote interface that extends the Remote interface. There is only one method named add() and it declares RemoteException.
2) Provide the implementation of the remote interface: Now provide the implementation of the remote interface. For providing the implementation of the Remote interface, we need to
   - Either extend the UnicastRemoteObject class,
   - or use the exportObject() method of the UnicastRemoteObject class In case, you extend the UnicastRemoteObject class, you must define a constructor that declares RemoteException.
3) create the stub and skeleton objects using the rmic tool: Next step is to create stub and skeleton objects using the rmi compiler. The rmic tool invokes the RMI compiler and creates stub and skeleton objects.
4) Start the registry service by the rmiregistry tool: Remote Method Invocation Now start the registry service by using the rmiregistry tool. If you don't specify the port number, it uses a default port number. In this example, we are using the port number 5000. rmiregistry 5000
5) Create and run the server application: Now rmi services need to be hosted in a server process. The Naming class provides methods to get and store the remote object.

Java is a most powerful programming language, by which we can do many things and Java is an industry preferable language. So it has a huge field of features. Here we discuss one of the best features of Java, that is how to represent the **current date and time using Java**.
There are many ways to do this, there are many classes by which it can be possible to display the current Date and Time.

Method 1: Using Date class

There is a class called **Date** class which can represent the current date and time in **GMT** form. We can get the **IST** form by adding 5 hours and 30 minutes to the GMT form. This class comes under the **util** package of Java.

Method 2: Using LocalDateTime class

There is a class called **LocalDateTime** class which can also represent the current date and time in GMT form. We can get the IST form by adding 5 hours and 30 minutes to the GMT form. This class comes under the **Time** package of Java.

Method 3: Using Clock class

There is a class called **Clock** class which can also represent the current date and time in UTC form. This class comes under the **Time** package of Java.

DateClient.java

```
import java.rmi.*;
public class DateClient
{ public static void main(String[] args)
{ try
{

IDate intf=(IDate)Naming.lookup("rmi://localhost:1099/DateServer");

System.out.println("The Date On Server is: "+intf.getDate());
} catch(Exception
e){ e.printStackTra
ce();
} }
}
```

DateImpl.java

```
import java.rmi.*; import
java.rmi.server.*;  import
java.util.*;
public class DateImpl extends UnicastRemoteObject implements IDate
{
public DateImpl() throws RemoteException{}
public String getDate()
{
Date d=new Date(); return(d.toString());
}
}
```

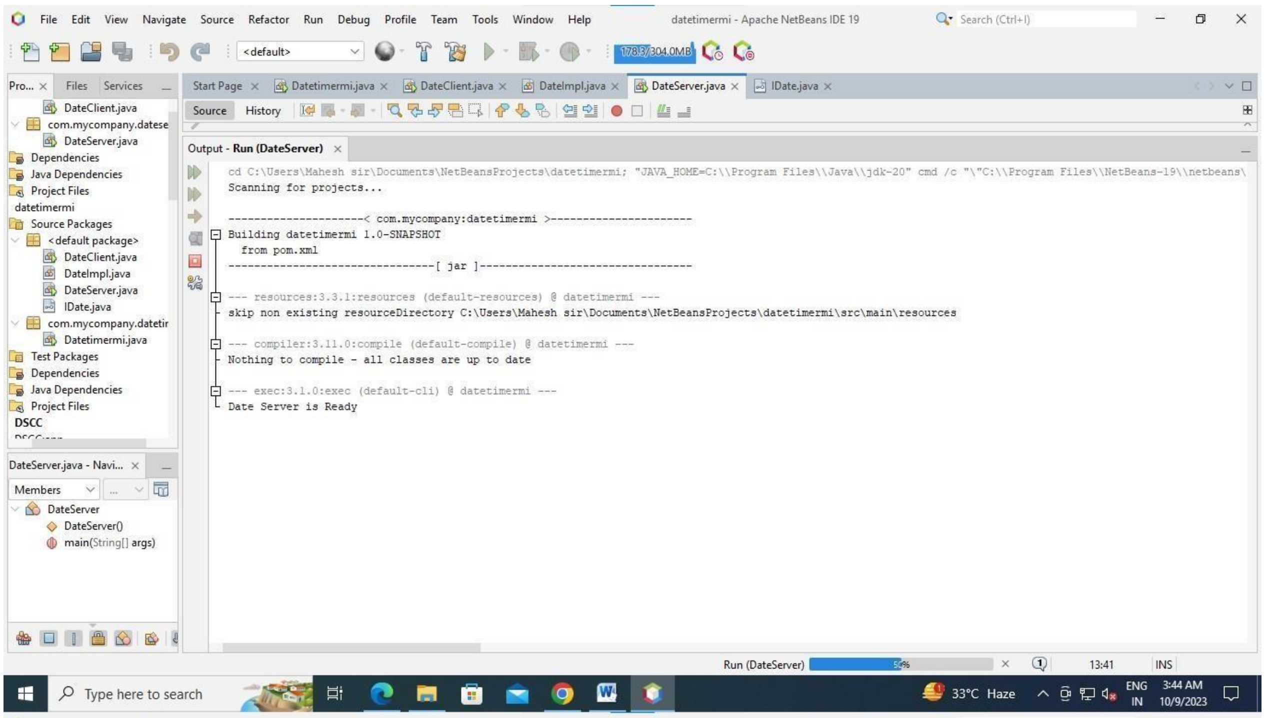
DateServer.java

```
import   java.rmi.*;      import
java.rmi.registry.LocateRegistry; import
java.rmi.registry.Registry;       public  class
DateServer
{ public static void main(String[] args)
  {
try
{
        Registry r = java.rmi.registry.LocateRegistry.createRegistry(1099);

        DateImpl di=new DateImpl();
        Naming.rebind("DateServer", (Remote) di);
        System.out.println("Date Server is Ready");
}
catch(Exception e)
{
e.printStackTrace();
}
} }
```

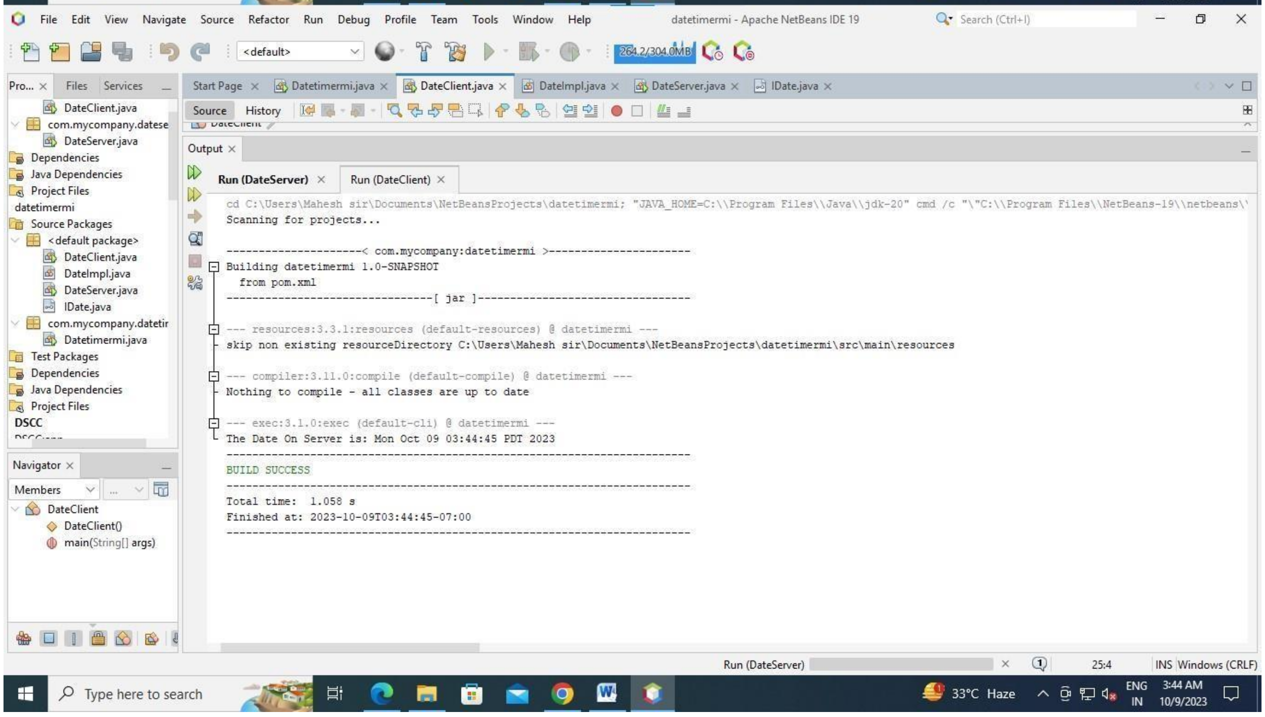
IDate.java import

```
java.rmi.*;

public interface IDate extends Remote
{
String getDate() throws RemoteException; }
```

OUTPUT:



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\datetimermi; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\
Scanning for projects...

--------------------< com.mycompany:datetimermi >---------------------
Building datetimermi 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ datetimermi ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\datetimermi\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ datetimermi ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ datetimermi ---
Date Server is Ready
```



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\datetimermi; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\'
Scanning for projects...

--------------------< com.mycompany:datetimermi >---------------------
Building datetimermi 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ datetimermi ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\datetimermi\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ datetimermi ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ datetimermi ---
The Date On Server is: Mon Oct 09 03:44:45 PDT 2023
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  1.058 s
Finished at: 2023-10-09T03:44:45-07:00
------------------------------------------------------------------------
```

PRACTICAL 5

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

In this example using Remote Method Invocation, how a client should provide an equation to the server through an interface. The server will solve the expression given by the client. The client application needs only two files, remote interface and client application. In the rmi application, both client and server interact with the remote interface. The client application invokes methods on the proxy object; RMI sends the request to the remote JVM. The return value is sent back to the proxy object and then to the client application.

1) create the remote interface: For creating the remote interface, extend the Remote interface and declare the RemoteException with all the methods of the remote interface. Here, we are creating a remote interface that extends the Remote interface. There is only one method named add() and it declares RemoteException.
2) Provide the implementation of the remote interface: Now provide the implementation of the remote interface. For providing the implementation of the Remote interface, we need to
   - Either extend the UnicastRemoteObject class,
   - or use the exportObject() method of the UnicastRemoteObject class In case, you extend the UnicastRemoteObject class, you must define a constructor that declares RemoteException.
3) create the stub and skeleton objects using the rmic tool: Next step is to create stub and skeleton objects using the rmi compiler. The rmic tool invokes the RMI compiler and creates stub and skeleton objects.
4) Start the registry service by the rmiregistry tool: Remote Method Invocation Now start the registry service by using the rmiregistry tool. If you don't specify the port number, it uses a default port number. In this example, we are using the port number 5000. rmiregistry 5000
5) Create and run the server application: Now rmi services need to be hosted in a server process. The Naming class provides methods to get and store the remote object.

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects *stub* and *skeleton*.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM), 3. It

waits for the result

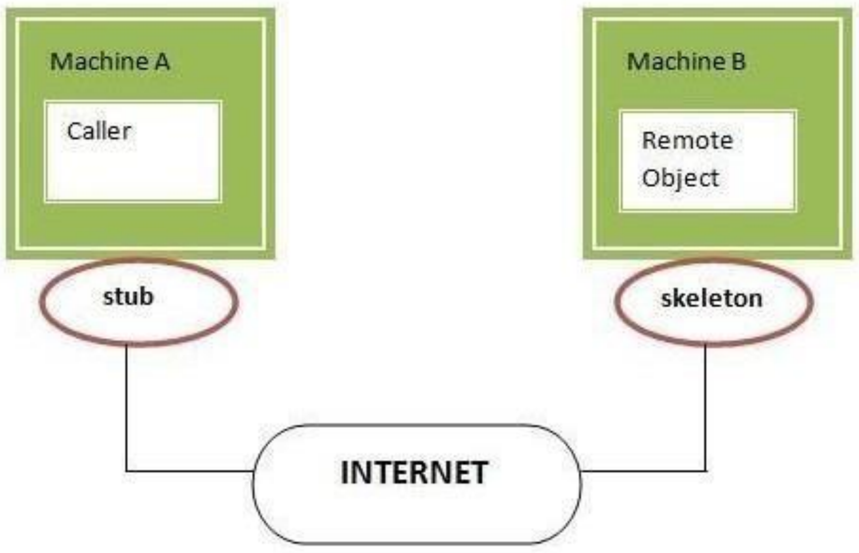
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



EServer.java

```
import  java.rmi.*;  import  java.rmi.registry.*;
public
class EServer
{
public static void main(String args[])throws Exception
{
EInt m1=new EImpl();
Runtime.getRuntime().exec("rmiregistry 8700");
Naming.rebind("rmi://localhost:8700/Testing",m1);
System.out.println("Server is ready....... ");
}
}
```

EImpl.java

```
import java.rmi.*;
import java.rmi.server.*;
public class EImpl extends UnicastRemoteObject implements EInt
{
public EImpl()throws RemoteException
{ super(); } public String MyEquation(int a,int b,int c)throws
RemoteException
{
if((b*b-4*a*c)<0)
{ return "x = Irrational number.";
} else
{ double d = (-b+Math.sqrt(b*b-4*a*c))/(2*a);
double e = (-b-Math.sqrt(b*b-4*a*c))/(2*a);      return
"x = "+d+" or x = "+e; }
}
}
```

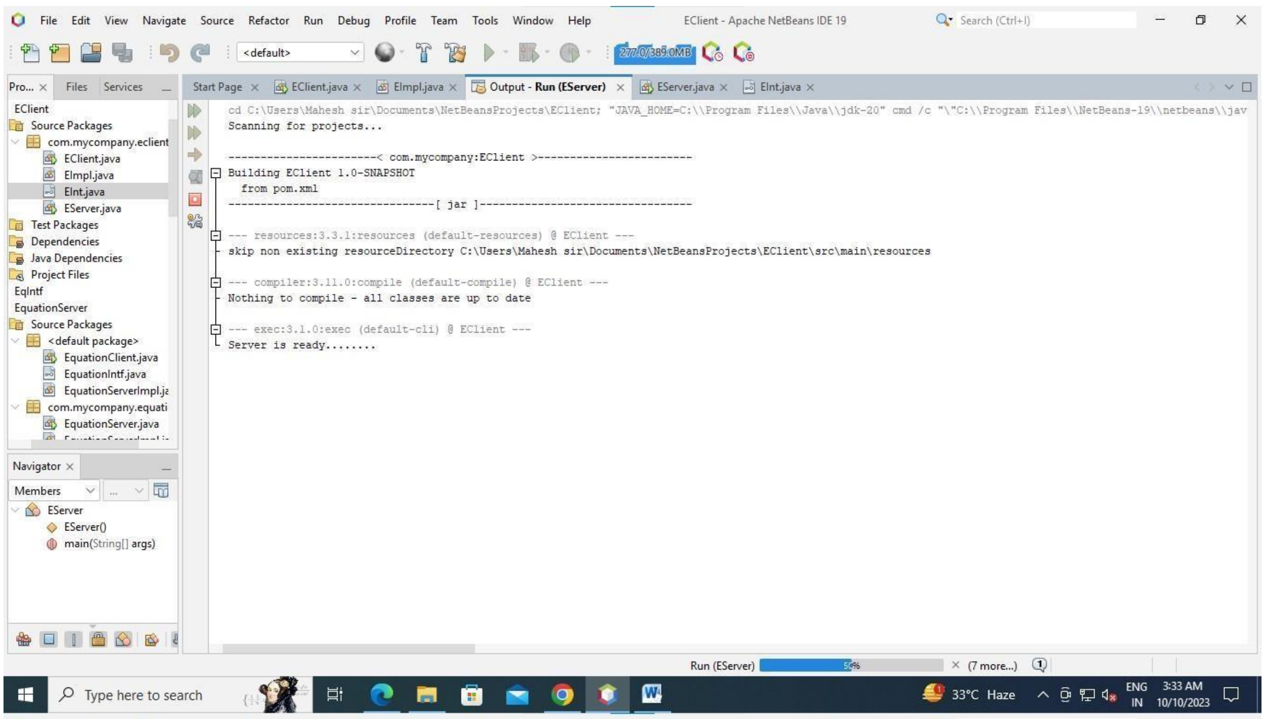
EInt.java

```
import java.rmi.*;
public interface EInt extends Remote
{
String MyEquation(int a, int b, int c) throws RemoteException;
}
```

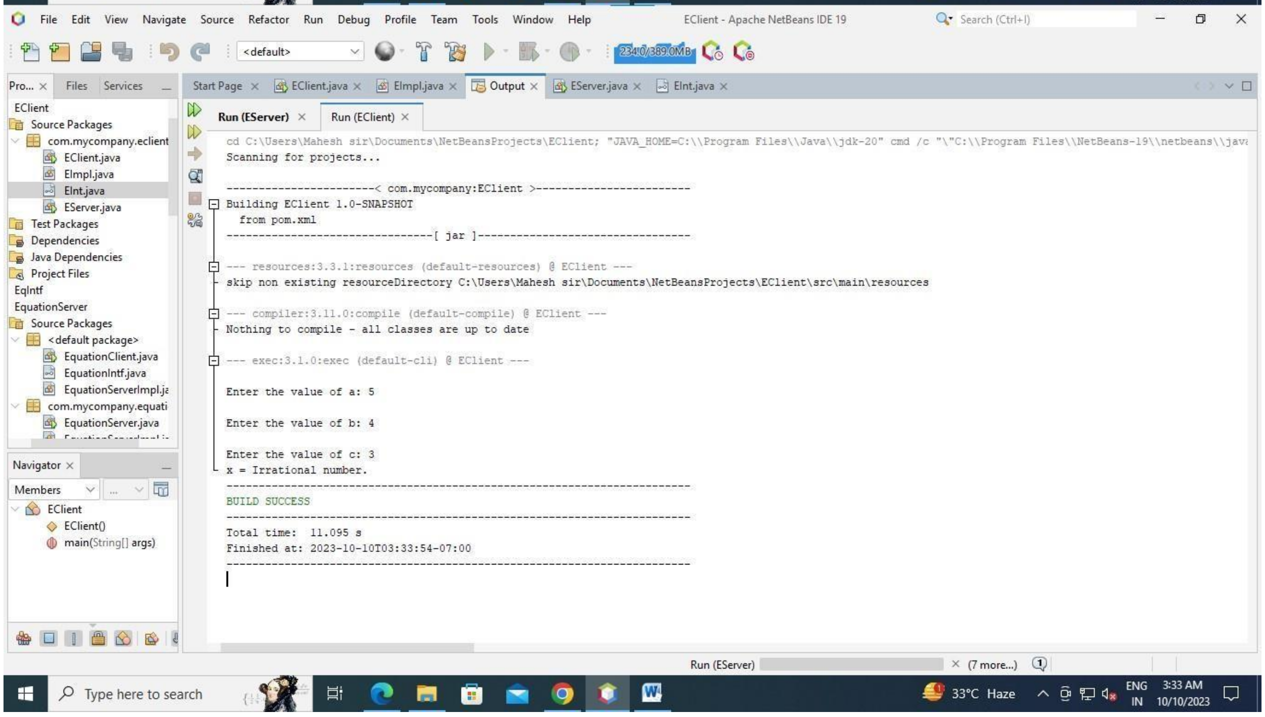
EClient.java import java.rmi.*; import java.io.*;

```
 public class EClient
{
public static void main(String args[])throws Exception
{
EInt o1=(EInt)Naming.lookup("rmi://localhost:8700/Testing");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in)); int
a,b,c;
System.out.print("\nEnter    the    value    of    a:    ");
a=Integer.parseInt(br.readLine()); System.out.print("\nEnter
the value of b: "); b=Integer.parseInt(br.readLine());
System.out.print("\nEnter the value of c: ");
c=Integer.parseInt(br.readLine());
System.out.println(o1.MyEquation(a,b,c));
}
}
```

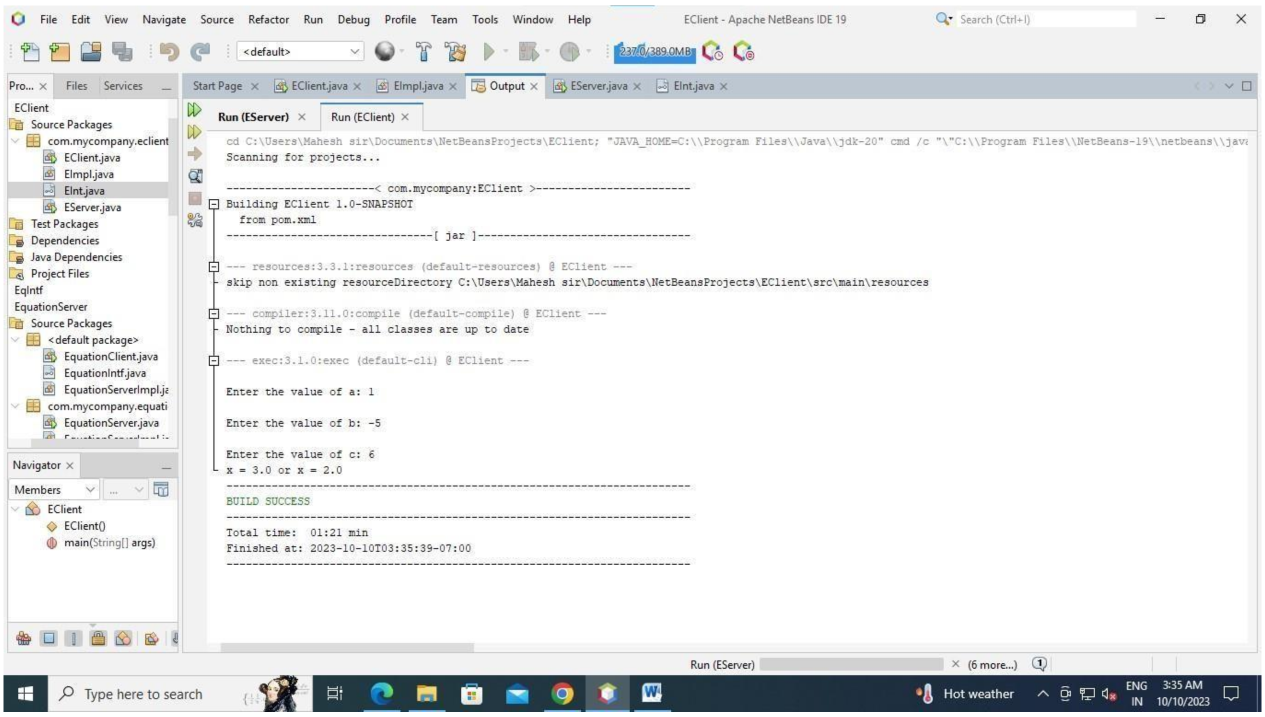
OUTPUT:



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\jav
Scanning for projects...

-----------------------< com.mycompany:EClient >------------------------
Building EClient 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ EClient ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ EClient ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ EClient ---
Server is ready........
```



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java
Scanning for projects...

-----------------------< com.mycompany:EClient >------------------------
Building EClient 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ EClient ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ EClient ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ EClient ---

Enter the value of a: 5

Enter the value of b: 4

Enter the value of c: 3
x = Irrational number.
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  11.095 s
Finished at: 2023-10-10T03:33:54-07:00
------------------------------------------------------------------------
```

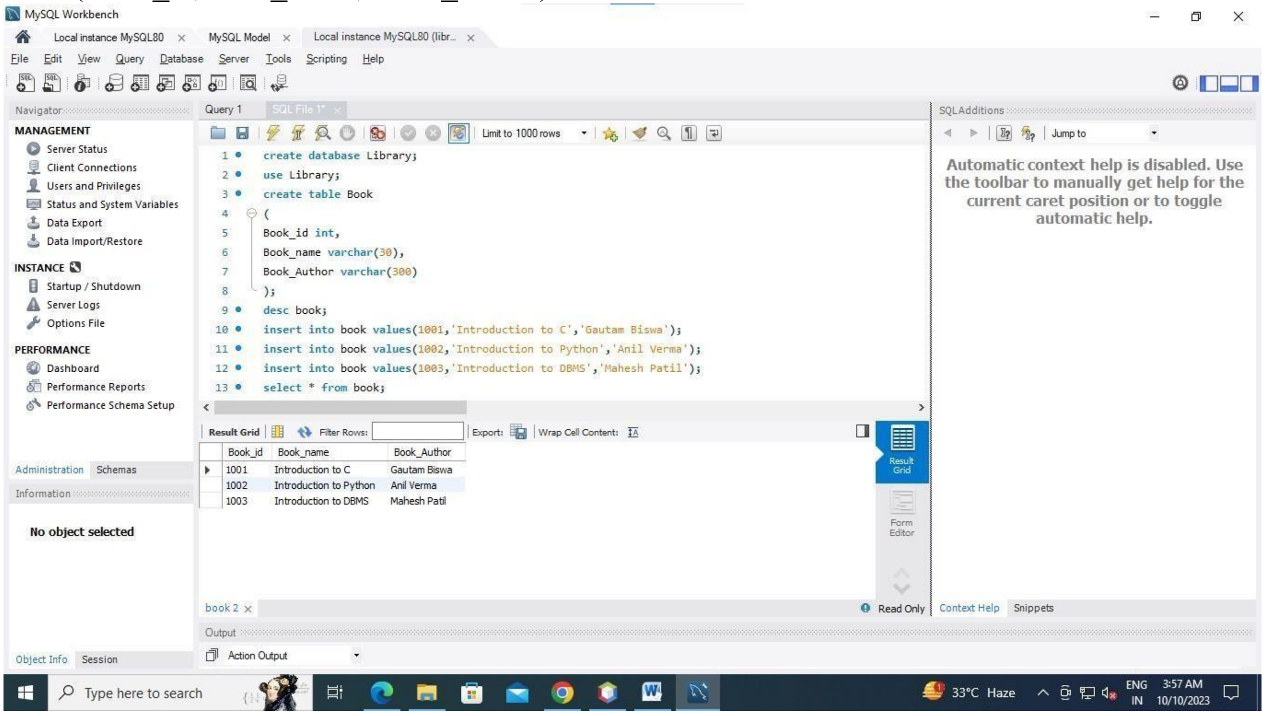


```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netbeans\\java
Scanning for projects...

-----------------------< com.mycompany:EClient >------------------------
Building EClient 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ EClient ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\EClient\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ EClient ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ EClient ---

Enter the value of a: 1

Enter the value of b: -5

Enter the value of c: 6
x = 3.0 or x = 2.0
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  01:21 min
Finished at: 2023-10-10T03:35:39-07:00
------------------------------------------------------------------------
```

PRACTICAL 6

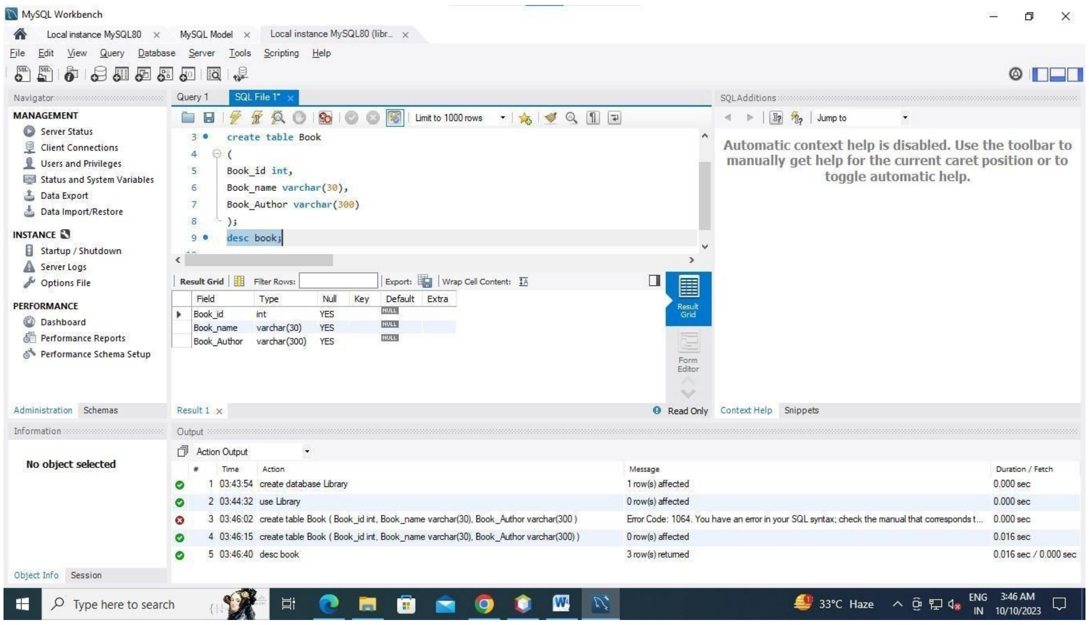
AIM: Using MySQL create Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept.

Before starting with coding in java,create Libray database in MySql.In Library Database Create Book(Book_id,Book_name, Book_author) table.



```
create database Library;
use Library;
create table Book
(
Book_id int,
Book_name varchar(30),
Book_Author varchar(300)
);
desc book;
insert into book values(1001,'Introduction to C','Gautam Biswa');
insert into book values(1002,'Introduction to Python','Anil Verma');
insert into book values(1003,'Introduction to DBMS','Mahesh Patil');
select * from book;
```

Book_id	Book_name	Book_Author
1001	Introduction to C	Gautam Biswa
1002	Introduction to Python	Anil Verma
1003	Introduction to DBMS	Mahesh Patil



```
create table Book
(
Book_id int,
Book_name varchar(30),
Book_Author varchar(300)
);
desc book;
```

Field	Type	Null	Key	Default	Extra
Book_id	int	YES		NULL	
Book_name	varchar(30)	YES		NULL	
Book_Author	varchar(300)	YES		NULL	

```
#  Time      Action                                                                        Message                                                                          Duration / Fetch
1  03:43:54  create database Library                                                       1 row(s) affected                                                                0.000 sec
2  03:44:32  use Library                                                                   0 row(s) affected                                                                0.000 sec
3  03:46:02  create table Book ( Book_id int, Book_name varchar(30), Book_Author varchar(300 )   Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds t...   0.000 sec
4  03:46:15  create table Book ( Book_id int, Book_name varchar(30), Book_Author varchar(300) )   0 row(s) affected                                                                0.016 sec
5  03:46:40  desc book                                                                     3 row(s) returned                                                                0.016 sec / 0.000 sec
```

Note: for MYSQL connectivity, you need to add following dependency in pom.xml file.

```
<dependencies>
      <dependency>
         <groupId>mysql</groupId>
         <artifactId>mysql-connector-java</artifactId>
         <version>8.1.0</version>
       </dependency>
   </dependencies>
```

Publishers.java

```
import java.io.Serializable;
public class Publishers implements Serializable
{

   String id;
   String bname;
   String bauthor;


}
```

Book_Server.java

```
import java.io.*; import java.sql.*; import
java.net.*;  import  java.util.*;  import
java.util.logging.Level;           import
java.util.logging.Logger;          import
javax.swing.table.DefaultTableModel;
import java.sql.Connection;
import java.sql.DriverManager;


public class Book_Server extends Thread
{

   Statement stmt=null;
   Vector records = new Vector(10,10);
   ResultSet rs = null;
   ServerSocket server = null;
   Socket client = null;
   Connection con = null;
   ObjectOutputStream out =null;
   String str = null;
```

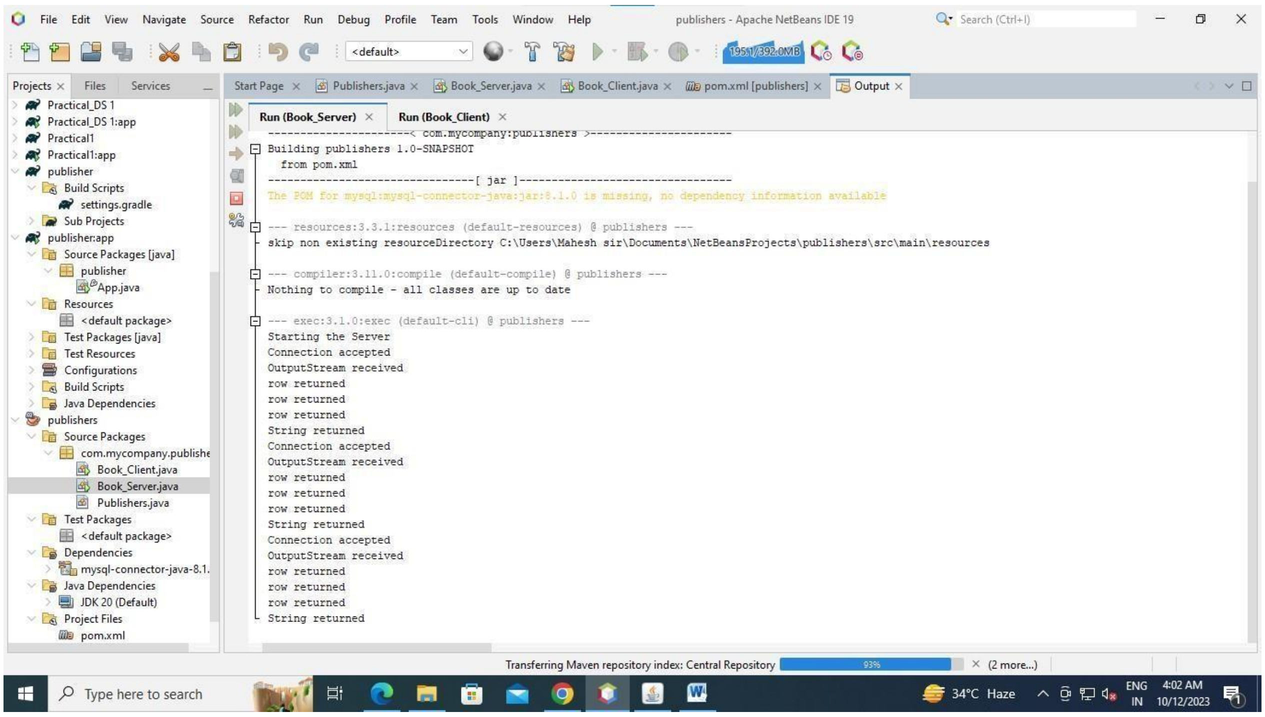
```
    Publishers pub = null;

    public Book_Server()
    {
  try
        { server = new ServerSocket(1700);
  System.out.println("Starting the Server");            start();
        }
        catch (IOException ex)
        {
           Logger.getLogger(Book_Server.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

    public void run()
    { while(true)
        {    try        {
   int
  CC; client = server.accept();
               System.out.println("Connection   accepted");  out  =  new
               ObjectOutputStream(client.getOutputStream());
               System.out.println("OutputStream received");
  try               {
                          Class.forName("com.mysql.cj.jdbc.Driver");

  Con=DriverManager.getConnection("jdbc:mysql://localhost:3306/library?zeroDateTimeBeha
  vior=CONVERT_TO_NULL", "root","Yashwant@55");
                  stmt = con.createStatement();
                  rs = stmt.executeQuery("select * from book");
  records.removeAllElements();

            ResultSetMetaData RSMD = rs.getMetaData(); CC
            = RSMD.getColumnCount();

                  while(rs.next())
                  { pub = new Publishers();
  pub.id = rs.getString(1); pub.bname
  = rs.getString(2);
```

```
                    pub.bauthor = rs.getString(3);

                    records.addElement(pub);
                    System.out.println("row returned"); }

                 out.writeObject(records);
                  out.close();
                  System.out.println("String returned");


                }
                catch (ClassNotFoundException ex)
                {
                  Logger.getLogger(Book_Server.class.getName()).log(Level.SEVERE, null, ex);
                }
                catch (SQLException ex)
                {
                  Logger.getLogger(Book_Server.class.getName()).log(Level.SEVERE, null, ex);
                }

              }
              catch (IOException ex)
              {
                Logger.getLogger(Book_Server.class.getName()).log(Level.SEVERE, null, ex);
              }

            }
          }
          public static void main(String args[])
        {   new
        Book_Server();

        }
        }
```

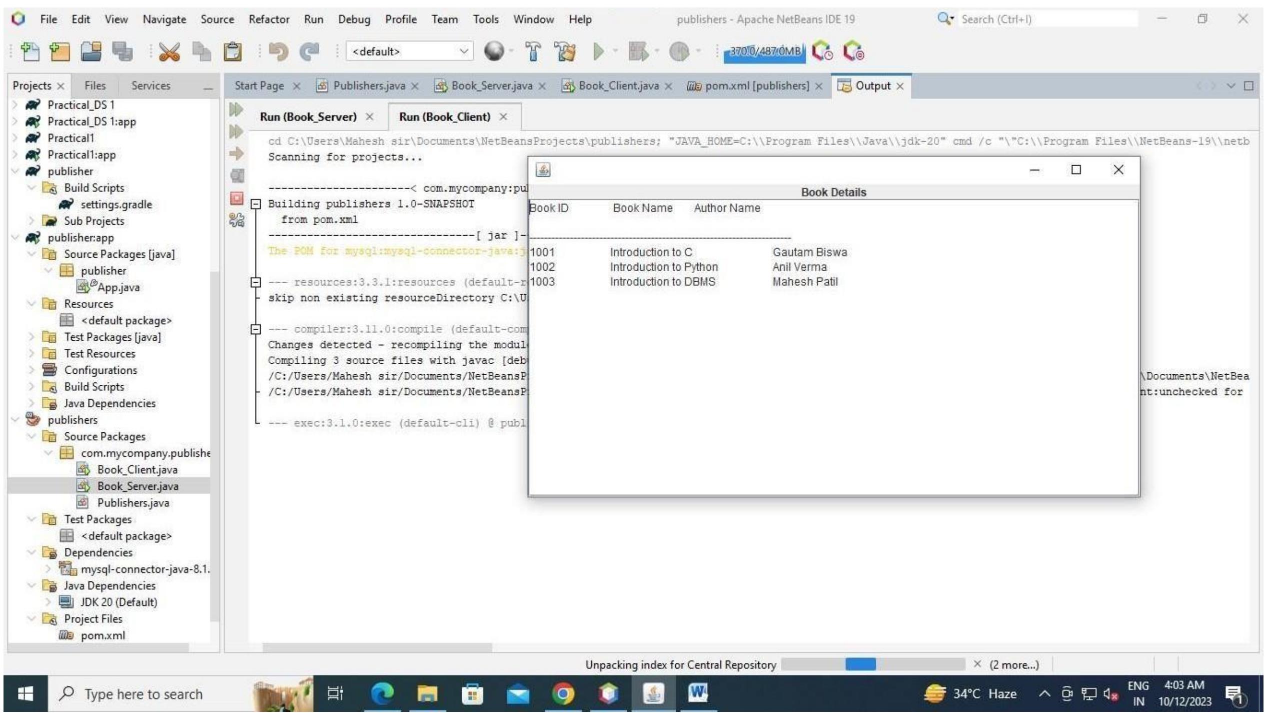
Book_Client.java

```
import      java.net.*;      import
java.io.*;    import    java.awt.*;
import      java.util.*;      import
java.sql.*;                   import
java.util.logging.Level;   import
java.util.logging.Logger; import
javax.swing.*;

public class Book_Client extends JFrame
{

    String str;
    ResultSet rs;
    Vector records;
    GridBagLayout gb1;
    GridBagConstraints gbc;
    JScrollPane sp;
    JTextArea result;
    JLabel
  label; Publishers
  pub; int i=0;
    ObjectInputStream br = null;
    Socket clientSocket = null;

    public Book_Client()
    { label = new JLabel("Book Details");
        result = new JTextArea(20,60);
str = ""; pub = null;      records = new
Vector();           gb1 = new
GridBagLayout();        gbc = new
GridBagConstraints();
getContentPane().setLayout(gb1);
        gbc.gridx = 0;
gbc.gridy = 0;
        getContentPane().add(label,gbc);
        gbc.gridx = 0;       gbc.gridy
= 1;        sp = new
JScrollPane(result);
getContentPane().add(sp,gbc); try
        { clientSocket = new Socket("localhost",1700);br =
new ObjectInputStream(clientSocket.getInputStream());
            records    =(Vector)br.readObject();
            br.close();  result.setText("");  int  i
            =0;

            result.append("Book ID\t Book Name\t Author Name\n");
            result.append("\n.........................................................................\n");
while(i < records.size())
            { pub = (Publishers)records.elementAt(i);
str = pub.id;               result.append(str + "\t");
str = pub.bname;                 result.append(str +
"\t");                     str = pub.bauthor;
result.append(str + "\n"); i++;

            }
            records.removeAllElements();
        }
        catch (IOException ex)
        {
          Logger.getLogger(Book_Client.class.getName()).log(Level.SEVERE, null, ex);
        }
        catch (ClassNotFoundException ex)
        {
        Logger.getLogger(Book_Client.class.getName()).log(Level.SEVERE, null, ex);
      }

    }
    public static void main(String[ ] args)
        {
                Book_Client  server1=new  Book_Client();  server1.setSize(500,
            300);
            server1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
server1.pack();
            server1.setVisible(true);

            }
}
```

OUTPUT:



```
---------------------< com.mycompany:publishers >---------------------
Building publishers 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------
The POM for mysql:mysql-connector-java:jar:8.1.0 is missing, no dependency information available

--- resources:3.3.1:resources (default-resources) @ publishers ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\publishers\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ publishers ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ publishers ---
Starting the Server
Connection accepted
OutputStream received
row returned
row returned
row returned
String returned
Connection accepted
OutputStream received
row returned
row returned
row returned
String returned
Connection accepted
OutputStream received
row returned
row returned
row returned
String returned
```


publishers - Apache NetBeans IDE 19
Run (Book_Server)
Run (Book_Client)
Unpacking index for Central Repository
4:03 AM 10/12/2023

```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\publishers; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\netb
Scanning for projects...

---------------------< com.mycompany:pu
Building publishers 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]-
The POM for mysql:mysql-connector-java:j

--- resources:3.3.1:resources (default-r
skip non existing resourceDirectory C:\U

--- compiler:3.11.0:compile (default-com
Changes detected - recompiling the modul
Compiling 3 source files with javac [deb
/C:/Users/Mahesh sir/Documents/NetBeansP                                  \Documents\NetBea
/C:/Users/Mahesh sir/Documents/NetBeansP                                  nt:unchecked for

--- exec:3.1.0:exec (default-cli) @ publ
```

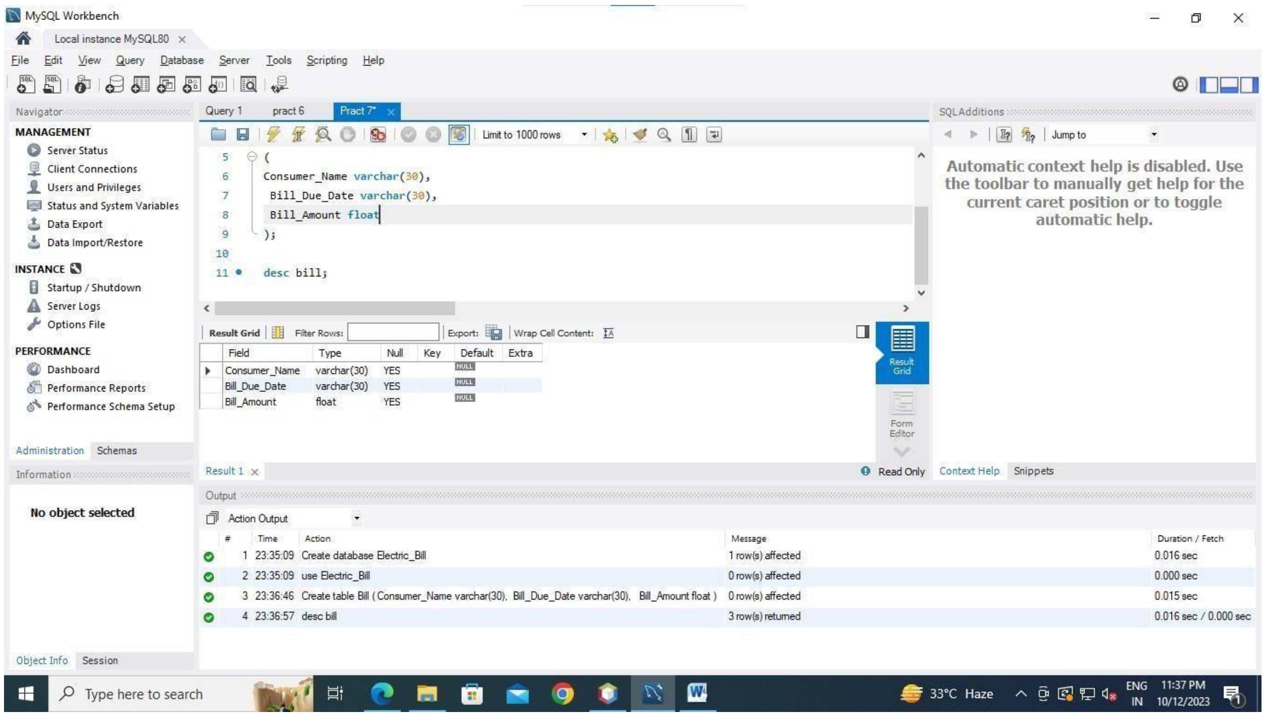
Book Details

```
Book ID       Book Name       Author Name
----------------------------------------------
1001          Introduction to C          Gautam Biswa
1002          Introduction to Python     Anil Verma
1003          Introduction to DBMS       Mahesh Patil
```

PRACTICAL 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.

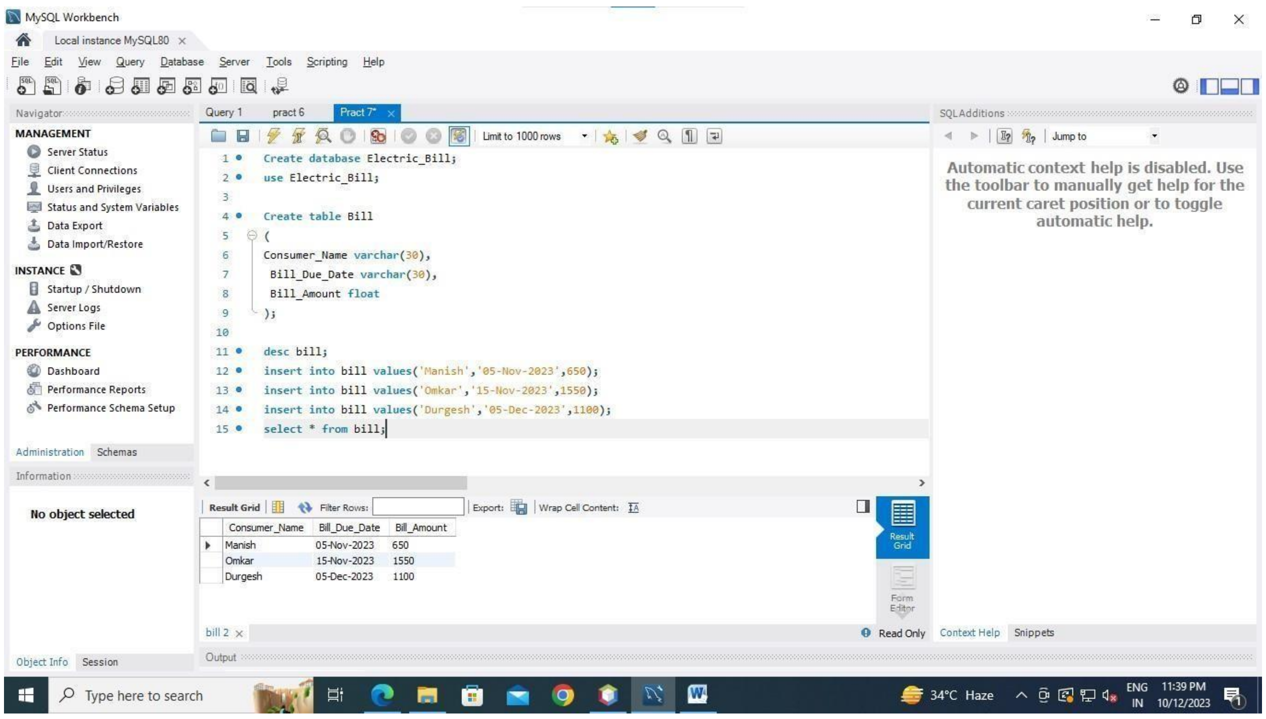
Before starting with coding in java, create ElectricBill database in MySql.In ElectricBill Database Create Bill(Consumer_Name, Bill_Due_Date, Bill_Amount) table.



```
(
Consumer_Name varchar(30),
 Bill_Due_Date varchar(30),
 Bill_Amount float
);

desc bill;
```

Field	Type	Null	Key	Default	Extra
Consumer_Name	varchar(30)	YES		NULL	
Bill_Due_Date	varchar(30)	YES		NULL	
Bill_Amount	float	YES		NULL	

```
#  Time      Action                                                                                  Message             Duration / Fetch
1  23:35:09  Create database Electric_Bill                                                           1 row(s) affected   0.016 sec
2  23:35:09  use Electric_Bill                                                                       0 row(s) affected   0.000 sec
3  23:36:46  Create table Bill ( Consumer_Name varchar(30),  Bill_Due_Date varchar(30),  Bill_Amount float )  0 row(s) affected   0.015 sec
4  23:36:57  desc bill                                                                               3 row(s) returned   0.016 sec / 0.000 sec
```


```
Create database Electric_Bill;
use Electric_Bill;

Create table Bill
(
Consumer_Name varchar(30),
 Bill_Due_Date varchar(30),
 Bill_Amount float
);

desc bill;
insert into bill values('Manish','05-Nov-2023',650);
insert into bill values('Omkar','15-Nov-2023',1550);
insert into bill values('Durgesh','05-Dec-2023',1100);
select * from bill;
```

Consumer_Name	Bill_Due_Date	Bill_Amount
Manish	05-Nov-2023	650
Omkar	15-Nov-2023	1550
Durgesh	05-Dec-2023	1100

Note: for MYSQL connectivity, you need to add following dependency in pom.xml file.

```
<dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.1.0</version>
         </dependency>
    </dependencies>
```

Bill_publisher.java

```
import java.io.Serializable;
public class Bill_publisher implements Serializable {

    String cname;
    String bdate;
    String bamount; }
```

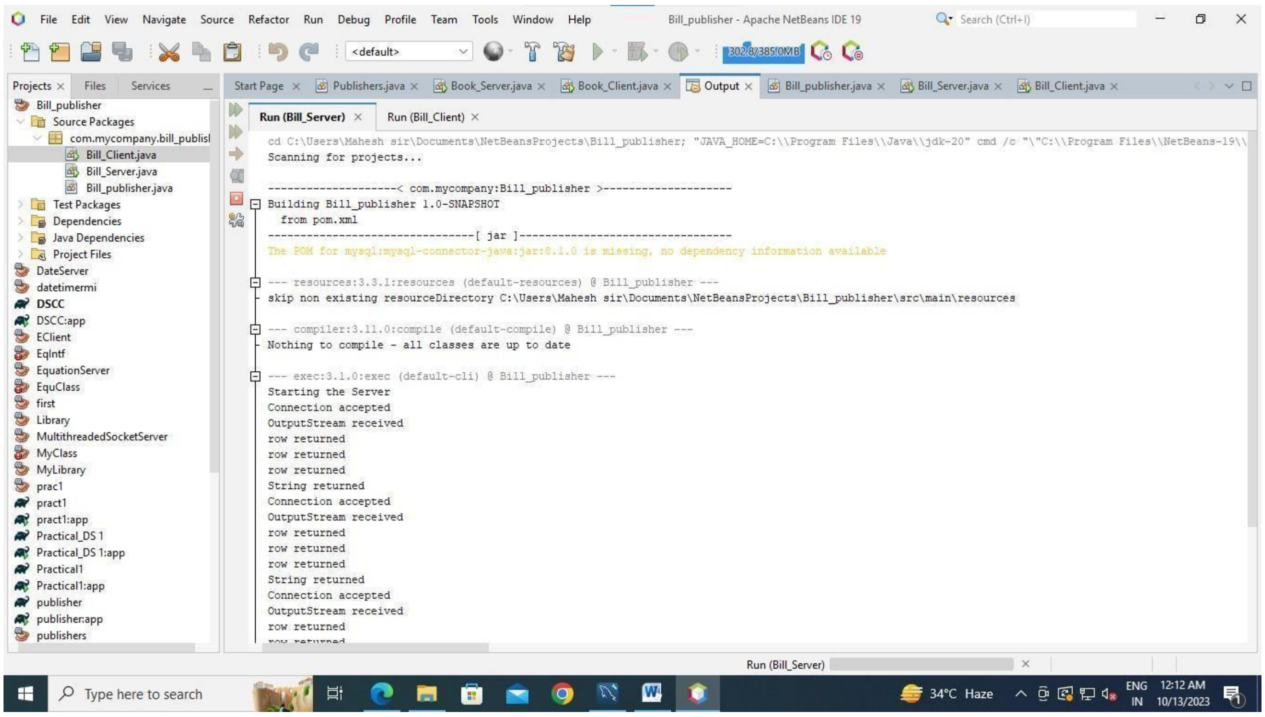
Bill_Server.java

```
import java.io.*; import java.sql.*; import
java.net.*; import java.util.*; import
java.util.logging.Level; import
java.util.logging.Logger; import
javax.swing.table.DefaultTableModel;
import java.sql.Connection; import
java.sql.DriverManager; public class
Bill_Server extends Thread {
    Statement stmt=null;
    Vector records = new Vector(10,10);
    ResultSet rs = null;
    ServerSocket server = null;
    Socket client = null;
    Connection con = null;
    ObjectOutputStream out =null;
    String str = null;
Bill_publisher pub = null; public
            Bill_Server()
     {
try
        { server = new ServerSocket(1900);
System.out.println("Starting the Server");             start();
        }
        catch (IOException ex)
        {
           Logger.getLogger(Bill_Server.class.getName()).log(Level.SEVERE, null, ex); }
    }
    public void run()
    { while(true)
       {    try         {
  int
CC; client = server.accept();
             System.out.println("Connection   accepted");   out   =   new
             ObjectOutputStream(client.getOutputStream());
             System.out.println("OutputStream received");
try               {
Class.forName("com.mysql.cj.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/electric_bill?zeroDateTime
Behavior=CONVERT_TO_NULL", "root","Yashwant@55"); stmt
                = con.createStatement();
                rs = stmt.executeQuery("select * from bill");
records.removeAllElements();

          ResultSetMetaData RSMD = rs.getMetaData(); CC
          = RSMD.getColumnCount();

                while(rs.next())
                { pub         =         new
                     Bill_publisher();
pub.cname             =             rs.getString(1);
pub.bdate             =             rs.getString(2);
pub.bamount = rs.getString(3);

                   records.addElement(pub);
                   System.out.println("row returned");
                } out.writeObject(records);
                out.close();
                System.out.println("String returned");


             }
             catch (ClassNotFoundException ex)
             {
                Logger.getLogger(Bill_Server.class.getName()).log(Level.SEVERE, null, ex); }
             catch (SQLException ex)
             {
                Logger.getLogger(Bill_Server.class.getName()).log(Level.SEVERE, null, ex);
             }

          }
          catch (IOException ex)
          {
             Logger.getLogger(Bill_Server.class.getName()).log(Level.SEVERE, null, ex);
          }

       }
    }
    public static void main(String args[])
{ new Bill_Server();

}
}
```

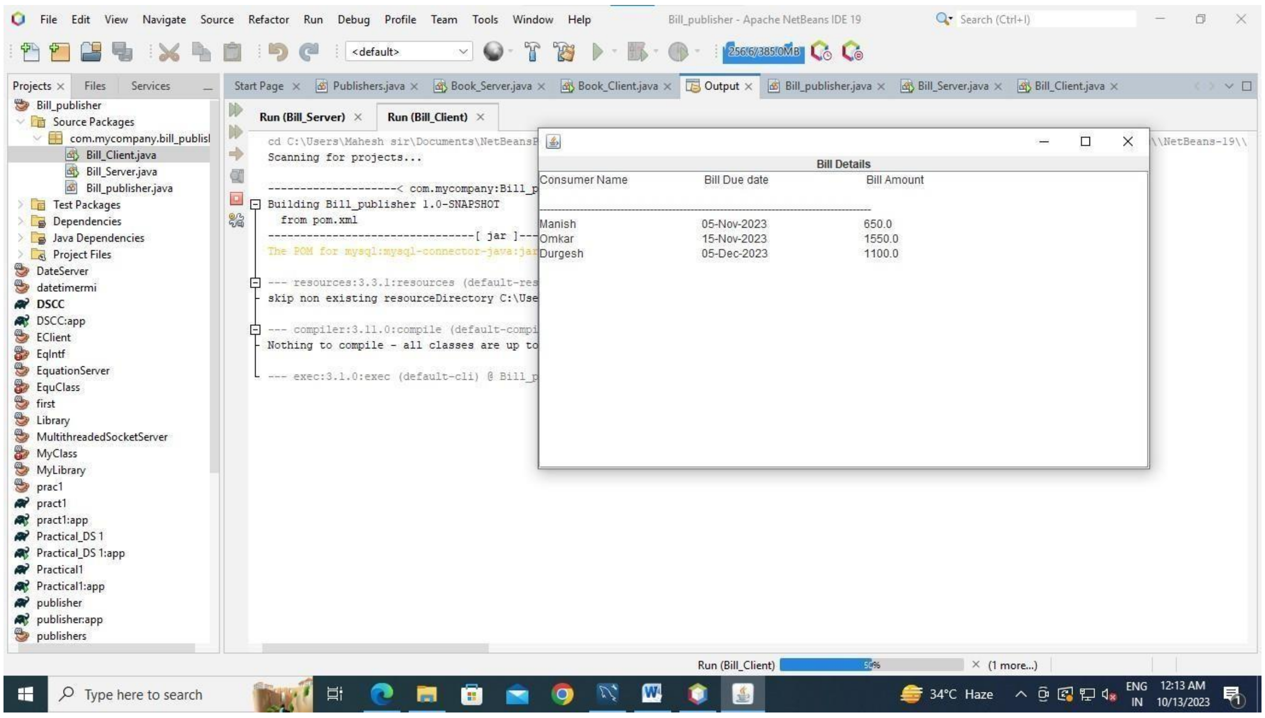
Bill_Client.java

```
    import java.net.*; import java.io.*;
    import      java.awt.*;      import
    java.util.*;    import    java.sql.*;
    import      java.util.logging.Level;
    import    java.util.logging.Logger;
    import javax.swing.*;
    public class Bill_Client extends JFrame
    {
       String str;
       ResultSet rs;
       Vector records;
       GridBagLayout gb1;
       GridBagConstraints gbc;
       JScrollPane sp;
       JTextArea result;
       JLabel label;
    Bill_publisher  pub;  int
       i=0;
       ObjectInputStream  br  =  null;  Socket
       clientSocket = null;

       public Bill_Client()
       {
            label = new JLabel("Bill
            Details"); result = new
            JTextArea(20,60); str = "";
                  pub = null;
    records = new Vector();         gb1 =
    new GridBagLayout();        gbc = new
    GridBagConstraints();
    getContentPane().setLayout(gb1);
            gbc.gridx = 0;
    gbc.gridy = 0;
            getContentPane().add(label,gbc);
            gbc.gridx = 0;       gbc.gridy
    = 1;         sp = new
    JScrollPane(result);
    getContentPane().add(sp,gbc);
    try
                { clientSocket = new Socket("localhost",1900);          br =
    new ObjectInputStream(clientSocket.getInputStream());
                      records =(Vector)br.readObject();
                      br.close();
    result.setText("");
    int                i                =0;
    result.append("Consumer
    Name\t Bill Due date\t\t Bill
    Amount\n"); result.append("\n
                      ..........................................................................................................
          ..............\n"); while(i <
                      records.size())
                      { pub = (Bill_publisher)records.elementAt(i);
    str = pub.cname;           result.append(str + "\t\t"); str =
    pub.bdate;         result.append(str + "\t\t"); str =
    pub.bamount;    result.append(str +
    "\n");
                          i++;
                      }
                      records.removeAllElements();
                           } catch
                (IOException ex)
                {
                    Logger.getLogger(Bill_Client.class.getName()).log(Level.SEVERE, null, ex);
                }
                catch (ClassNotFoundException ex)
                {
                Logger.getLogger(Bill_Client.class.getName()).log(Level.SEVERE, null, ex);
              }

            }
            public static void main(String[ ] args)
                {
                        Bill_Client server1=new Bill_Client();

                    server1.setSize(500, 300);
        server1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);        server1.pack();
                    server1.setVisible(true);

                    }
        }
```

OUTPUT:



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\Bill_publisher; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\
Scanning for projects...

--------------------< com.mycompany:Bill_publisher >--------------------
Building Bill_publisher 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------
The POM for mysql:mysql-connector-java:jar:8.1.0 is missing, no dependency information available

--- resources:3.3.1:resources (default-resources) @ Bill_publisher ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\Bill_publisher\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ Bill_publisher ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ Bill_publisher ---
Starting the Server
Connection accepted
OutputStream received
row returned
row returned
row returned
String returned
Connection accepted
OutputStream received
row returned
row returned
row returned
String returned
Connection accepted
OutputStream received
row returned
```



```
Bill Details
Consumer Name             Bill Due date             Bill Amount
-------------------------------------------------------------------
Manish                    05-Nov-2023               650.0
Omkar                     15-Nov-2023               1550.0
Durgesh                   05-Dec-2023               1100.0
```

PRACTICAL 8

AIM: Implementation of mutual exclusion using Token ring algorithm.

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e. only one process is allowed to execute the critical section at any given instance of time.

Mutual exclusion is a property of process synchronization which states that ―no two processes can exist in the critical section at any given point of time‖. The term was first coined by Dijkstra. Any process synchronization technique being used must satisfy the property of mutual exclusion, without which it would not be possible to get rid of a race condition.

Token Based Algorithm:

I. A unique token is shared among all the sites.

II. If a site possesses the unique token, it is allowed to enter its critical section III. This approach uses sequence number to order requests for the critical section.

IV. Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.

V. This approach insures Mutual exclusion as the token is unique.

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process can not enter its critical section while another concurrent process is currently present or executing in its critical section i.e only one process is allowed to execute the critical section at any given instance of time.

Mutual exclusion in single computer system Vs. distributed system:

In single computer system, memory and other resources are shared between different processes. The status of shared resources and the status of users is easily available in the shared memory so with the help of shared variable (For example: Semaphores) mutual exclusion problem can be easily solved. In Distributed systems, we neither have shared memory nor a common physical clock and there for we can not solve mutual exclusion problem using shared variables. To eliminate the mutual exclusion problem in distributed system approach based on message passing is used. A site in distributed system do not have complete information of state of the system due to lack of shared memory and a common physical clock.

Requirements of Mutual exclusion Algorithm:

- **No Deadlock:** Two or more site should not endlessly wait for any message that will never arrive.
- **No Starvation:** Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site are repeatedly executing critical section
- **Fairness:** Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e Critical

section execution requests should be executed in the order of their arrival in the system.
- **Fault Tolerance:** In case of failure, it should be able to recognize it by itself in order to continue functioning without any disruption.

Solution to distributed mutual exclusion:

As we know shared variables or a local kernel can not be used to implement mutual exclusion in distributed systems. Message passing is a way to implement mutual exclusion. Below are the three approaches based on message passing to implement mutual exclusion in distributed systems:

1. Token Based Algorithm:
- A unique **token** is shared among all the sites.
- If a site possesses the unique token, it is allowed to enter its critical section This approach uses sequence number to order requests for the critical section.
- Each requests for critical section contains a sequence number. This sequence number is used to distinguish old and current requests.
- This approach insures Mutual exclusion as the token is unique Example : Suzuki–Kasami Algorithm

2. Non-token based approach:
- A site communicates with other sites in order to determine which sites should execute critical section next. This requires exchange of two or more successive round of messages among sites.
- This approach use timestamps instead of sequence number to order requests for the critical section.
- When ever a site make request for critical section, it gets a timestamp. Timestamp is also used to resolve any conflict between critical section requests.
- All algorithm which follows non-token based approach maintains a logical clock. Logical clocks get updated according to Lamport's scheme

Example : Ricart–Agrawala Algorithm

3. Quorum based approach:
- Instead of requesting permission to execute the critical section from all other sites, Each site requests only a subset of sites which is called a **quorum**. Any two subsets of sites or Quorum contains a common site.
- This common site is responsible to ensure mutual exclusion Example :

Maekawa's Algorithm

TokenServer.java

```
import  java.io.*;  import  java.net.*;
public class
TokenServer
{
public static void main(String agrs[])throws Exception
{ while(true)
{
Server sr=new Server(); sr.recPort(8000);
sr.recData();
}
}     }      class      Server
{  boolean hasToken=false;
boolean sendData=false; int
recport;    void recPort(int
recport) {
this.recport=recport;
}
void recData()throws Exception
{ byte buff[]=new byte[256];
DatagramSocket
ds;
DatagramPacket dp;
String str; ds=new
DatagramSocket(recport); dp=new
DatagramPacket(buff,buff.length);
ds.receive(dp); ds.close(); str=new
String(dp.getData(),0,dp.getLength());
System.out.println("The message is "+str);
}
}
```

TokenClient1.java

```
import   java.io.*;   import   java.net.*;
public class
TokenClient1
{
public static void main(String arg[]) throws Exception
{
InetAddress lclhost;
BufferedReader br;
String str="";
TokenClient12      tkcl,tkser;
boolean  hasToken;  boolean
setSendData; while(true) {
lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient12(lclhost); tkser
= new TokenClient12(lclhost);
//tkcl.setSendPort(9001);
tkcl.setSendPort(9004); tkcl.setRecPort(8002);
lclhost=InetAddress.getLocalHost();
tkser.setSendPort(9000);
if(tkcl.hasToken == true) {

System.out.println("Do you want to enter the Data –> YES/NO"); br=new
BufferedReader(new InputStreamReader(System.in)); str=br.readLine();
if(str.equalsIgnoreCase("yes"))
{
System.out.println("ready to send"); tkser.setSendData =
true;
tkser.sendData(); tkser.setSendData
= false; }
else if(str.equalsIgnoreCase("no"))
{
System.out.println("i  m  in  else");  //tkcl.hasToken=false;
tkcl.sendData();

tkcl.recData();
System.out.println("i m leaving else");
} }
else
{
System.out.println("ENTERING RECEIVING MODE…"); tkcl.recData();
}
}
}
}

class TokenClient12
{
InetAddress        lclhost; int
sendport,recport; boolean
hasToken = true; boolean
setSendData = false;
TokenClient12 tkcl,tkser;
TokenClient12(InetAddress lclhost)
{
this.lclhost = lclhost;
}
void setSendPort(int sendport)
{
this.sendport = sendport;
}
void setRecPort(int recport)
{
this.recport = recport;
}

void sendData() throws Exception
{
BufferedReader br;
String str="Token";
DatagramSocket  ds;
DatagramPacket  dp; if(setSendData
== true)
{
System.out.println("sending "); System.out.println("Enter the
Data"); br=new BufferedReader(new
InputStreamReader(System.in)); str = "ClientOne….." +
br.readLine();
System.out.println("now sending");
}
ds = new DatagramSocket(sendport); dp = new
DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
ds.send(dp); ds.close(); setSendData = false; hasToken = false;
}
```

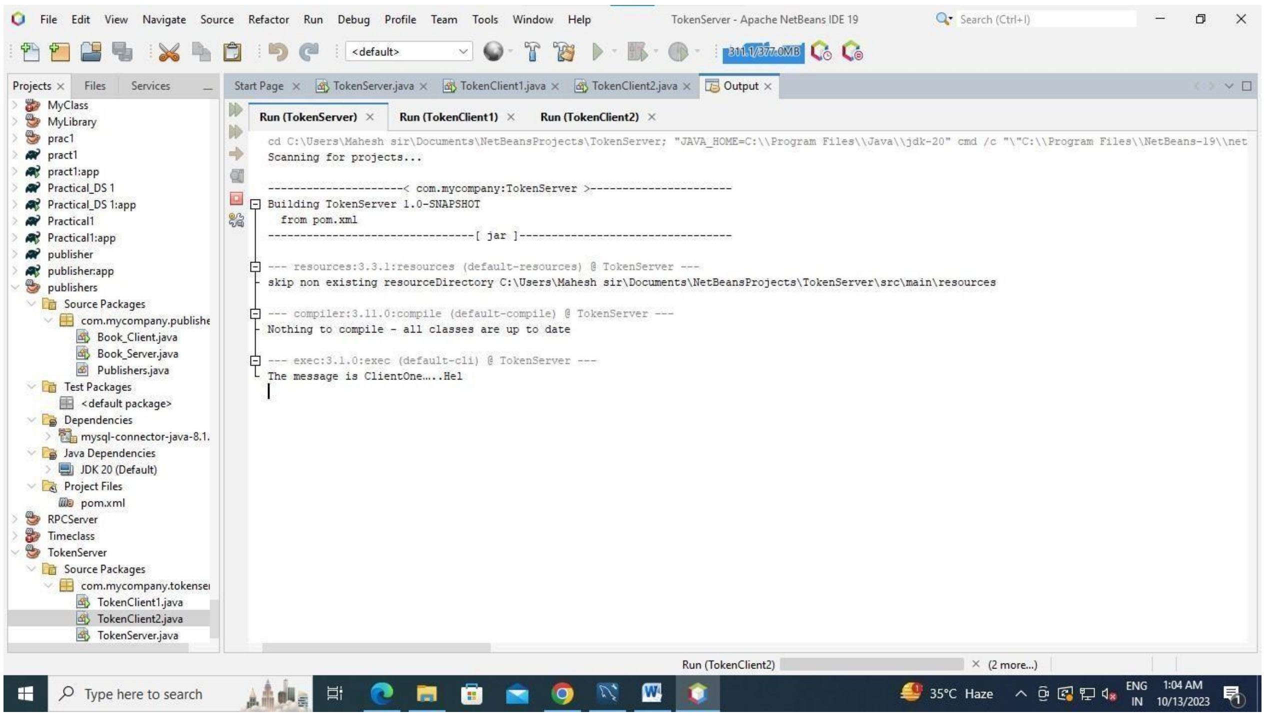
```
void recData()throws Exception
{
String msgstr; byte buffer[] = new
byte[256];
DatagramSocket ds; DatagramPacket dp; ds =
new    DatagramSocket(recport);    dp    =    new
DatagramPacket(buffer,buffer.length); ds.receive(dp);
ds.close(); msgstr = new
String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+msgstr);
if(msgstr.equals("Token"))
{ hasToken = true;
}
} }
```

TokenClient2.java

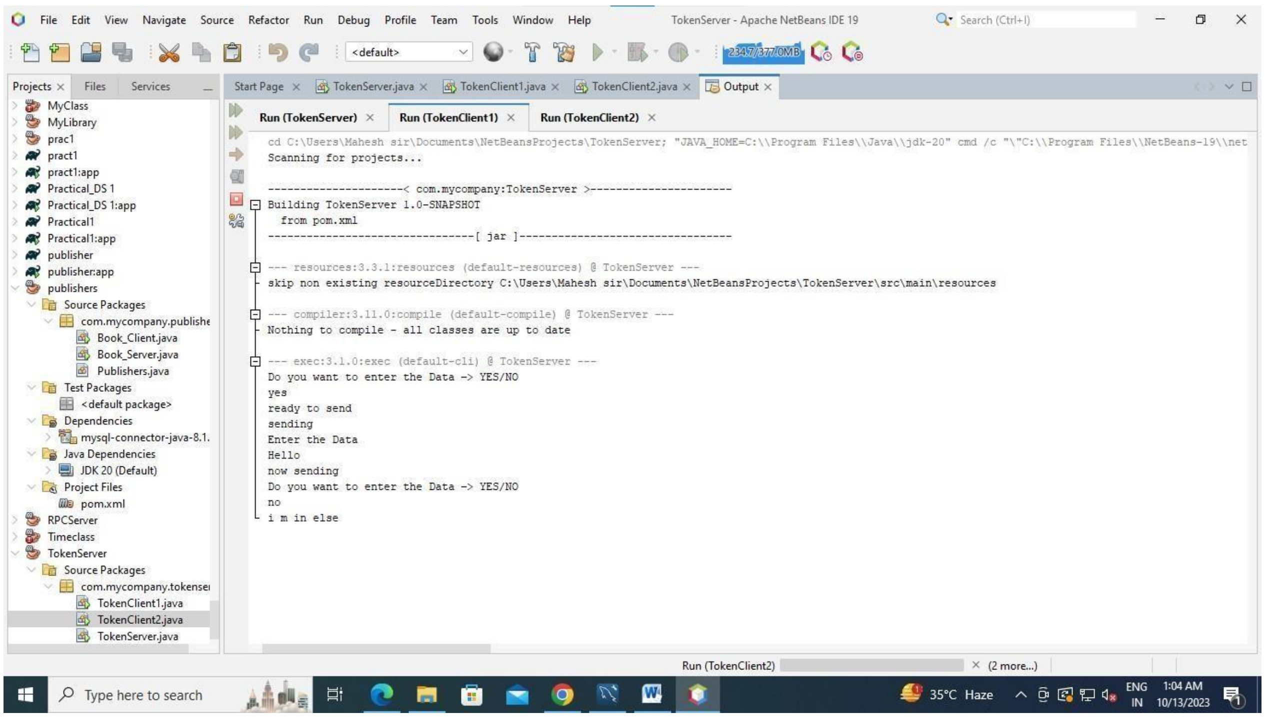
```
import   java.io.*;   import
java.net.*;   public   class
TokenClient2
{ static boolean setSendData ;
static
boolean hasToken ;
public static void main(String arg[]) throws Exception
{
InetAddress lclhost;
BufferedReader br;
String str1;
TokenClient21 tkcl; TokenClient21 ser;
while(true)
{ lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient21(lclhost);
tkcl.setRecPort(8004);
tkcl.setSendPort(9002);
lclhost=InetAddress.getLocalHost();
ser = new TokenClient21(lclhost);
ser.setSendPort(9000);
System.out.println("entering if");
if(hasToken == true)
{
System.out.println("Do you want to enter the Data –> YES/NO");
br=new BufferedReader(new InputStreamReader(System.in));
str1=br.readLine(); if(str1.equalsIgnoreCase("yes"))
{
System.out.println("ignorecase");
ser.setSendData = true; ser.sendData();
}
else if(str1.equalsIgnoreCase("no"))
{ tkcl.sendData(); hasToken=false;
} }
else
{
System.out.println("entering recieving mode"); tkcl.recData();
hasToken=true;
}
}
}
}
class TokenClient21
{
```

```
InetAddress      lclhost;int
sendport,recport;          boolean
setSendData = false; boolean
hasToken = false; TokenClient21
tkcl;
TokenClient21 ser;
TokenClient21(InetAddress lclhost)
{
this.lclhost = lclhost;
}
void setSendPort(int sendport)
{
this.sendport = sendport;
}
void setRecPort(int recport)
{
this.recport = recport;
}
void sendData() throws Exception
{
System.out.println("case");
BufferedReader br;
String str="Token";
DatagramSocket          ds;
DatagramPacket          dp;
if(setSendData == true)
{
System.out.println("Enter the Data"); br=new
BufferedReader(new InputStreamReader(System.in)); str =
"ClientTwo......" + br.readLine();
} ds = new DatagramSocket(sendport); dp =
new
DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
ds.send(dp); ds.close();
System.out.println("Data                    Sent");
setSendData = false; hasToken
= false;
}

void recData()throws Exception
{
String msgstr;
byte buffer[] = new byte[256];
DatagramSocket                    ds;
DatagramPacket  dp;  ds  =  new
DatagramSocket(recport);  //ds  =
new DatagramSocket(4000); dp =
new
DatagramPacket(buffer,buffer.lengt
h);
ds.receive(dp);     ds.close();     msgstr     =     new
String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+msgstr); if(msgstr.equals("Token"))
{ hasToken =
true;
}
}
}
```

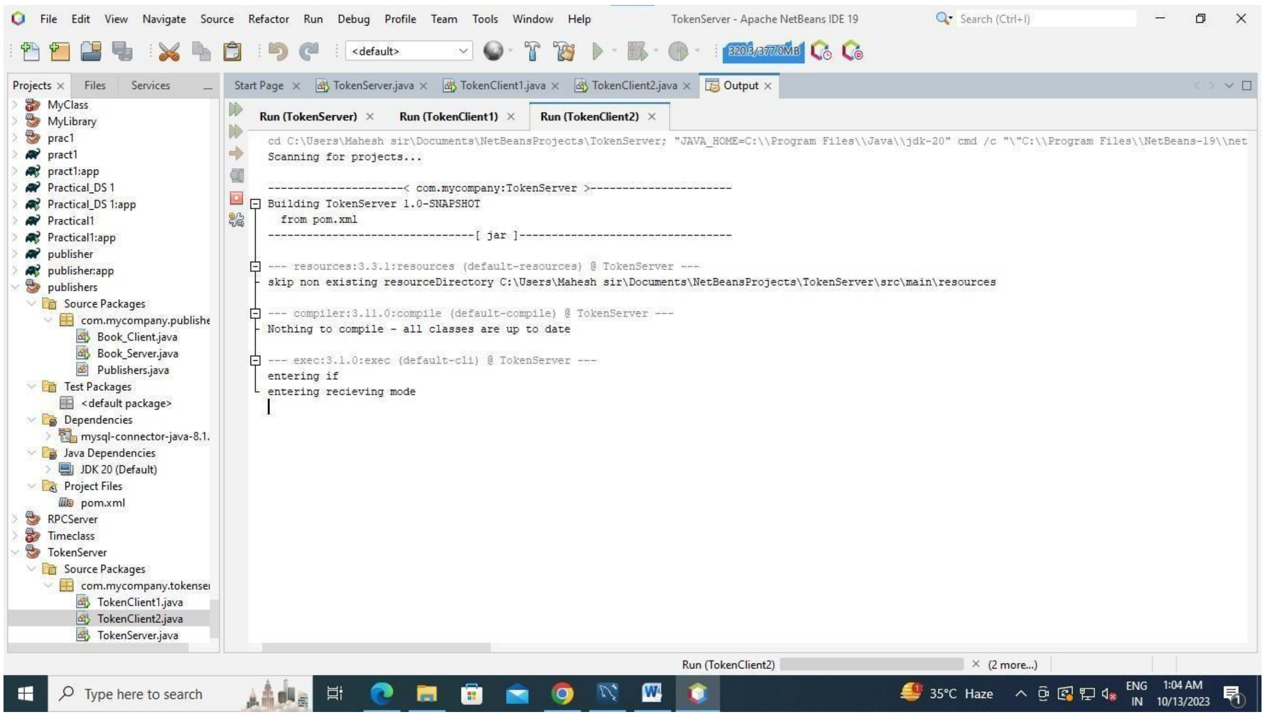
OUTPUT:



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\net
Scanning for projects...

--------------------< com.mycompany:TokenServer >---------------------
Building TokenServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ TokenServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ TokenServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ TokenServer ---
The message is ClientOne.....Hel
```



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\net
Scanning for projects...

--------------------< com.mycompany:TokenServer >---------------------
Building TokenServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ TokenServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ TokenServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ TokenServer ---
Do you want to enter the Data -> YES/NO
yes
ready to send
sending
Enter the Data
Hello
now sending
Do you want to enter the Data -> YES/NO
no
i m in else
```



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\net
Scanning for projects...

--------------------< com.mycompany:TokenServer >---------------------
Building TokenServer 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ TokenServer ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\TokenServer\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ TokenServer ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ TokenServer ---
entering if
entering recieving mode
```

PRACTICAL 9

AIM: Implementation of Identity Management.

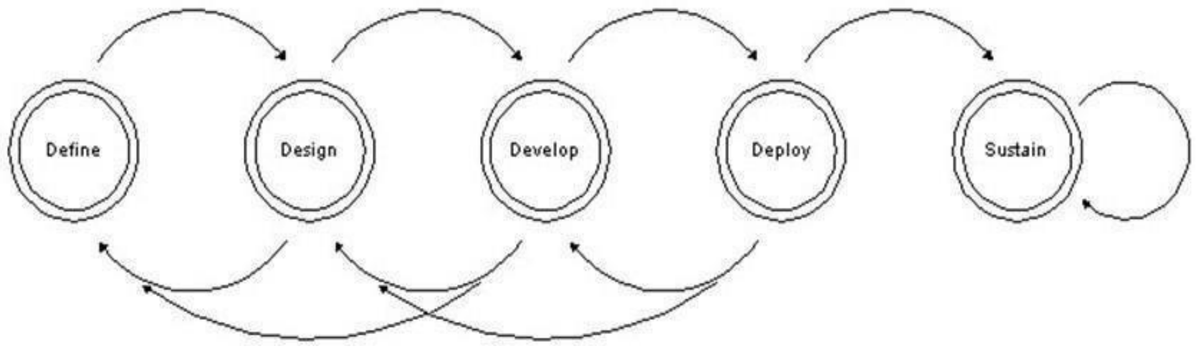
The main goal of identity management is to ensure only authenticated users are granted access to the specific applications, systems or IT environments for which they are authorized. Identity management in cloud computing is the subsequent step of identity and access management (IAM) solutions. However, it is a lot more than merely a straightforward web app single signon (SSO) solution. This next generation of IAM solution is a holistic move of the identity provider right to the cloud. Innovations in the user identity management space have been a trend in the past couple of years. Most of these developments across business and technology fronts have been around identity management in cloud computing, enabling the authentication and authorization processes right in the cloud. The primary goal of identity management in cloud computing is dealing with personal identity information so that a user's access to data, computer resources, applications, and services is controlled accurately. Identity management in cloud computing is the subsequent step of identity and access management (IAM) solutions. However, it is a lot more than merely a straightforward web app single sign-on (SSO) solution. This next generation of IAM solution is a holistic move of the identity provider right to the cloud. Known as Directory-as-a-Service (DaaS), this particular service is the advanced version of the conventional and on-premises solutions, including Lightweight Directory Access Protocol (LDAP) as well as Microsoft Active Directory (AD).

Identity Management solutions have taken many guises as it has become a popular term, with many vendors claiming their solutions meet the criteria to be called such. Identity Management is a new and rapidly evolving market that has not achieved the level of maturity whereby we can say definitively what an Identity Management solution "must" contain in terms of functionality and services. Rather, as we have discussed in the previous chapters, the breakdown of Identity Management terms and components allows for flexibility, which in turn, makes implementations of Identity Management solutions unique to each organization.

Identity Management implementations have historically been undertaken as part of an organization's security initiative or as a set of components built primarily on existing security infrastructure. However, although the implementation of Identity Management in an organization is strongly tied to security requirements, the strategic drivers should be, and are, at a higher level, tied to business requirements. The reality is that the security component is only a small part of Identity Management, and that much more process and technology lies beneath the surface. This chapter is about how you can go about implementing Identity Management in your organization.

Planning—Where Do I Start?

One of the most important aspects of any project is the methodology you employ to actually plan and implement a solution. For my own projects, I often use what's known as the 4DS planning methodology, which Figure 4.1 illustrates.



This book isn't designed to describe this particular methodology in great detail, and many organizations already have a methodology or even project management specialists. However, for the purposes of this discussion, the 4DS process provides a framework, so let's take a few moments now to discuss the basic steps and terminology associated with this concept so that we may refer back to it later in the book.

- **Define**—Determine the scope and deliverables that you want to provide and in what environment. Ensure that all must have, may have, and desired functionality is clearly defined and prioritized and that any expectations of vendor products is agreed upon and documented.
- **Design**—Set down the details of what you plan to deliver, then document and create a higher-level project plan.
- **Develop**—Create the code and identify the tools you need, and deliver a detailed project plan. At this stage, you should also be ready to pilot or run your project through a quality assurance (QA) or testing phase. This part of the process is essential—you must QA or test what you are developing throughout. Does it meet the needs you defined earlier? Does it match documented requirements, and do vendor solutions meet expectations? This testing also includes validating your delivery and deployment scenarios.
- **Deploy**—Deliver the solution according to your plan.
- **Sustain**—As with any product, you need a plan to sustain or maintain the product through its various life cycles, including new deployments or updates.

Features of a Modern Cloud Identity Management Solution:

The following are a few advantages of identity management in cloud computing:

- It offers a consistent access control interface: Applicable for all cloud platform services; Cloud IAM solutions provide a clean and single access control interface.
- It offers superior security levels: If needed, we can easily define increased security levels for crucial applications.
- It lets businesses access resources at diverse levels: Businesses can define roles and grant permissions to explicit users for accessing resources at diverse granularity levels.

Why Do You Need Cloud IAM?

Identity management in cloud computing incorporates all categories of user-base who can operate in diverse scenarios and with specific devices. A modern cloud Identity and Access Management (IAM) solution helps to:

- Connect professionals, employees, IT applications, and devices securely either on-premise or the cloud and through involved networks.
- It makes it easy to share the network abilities with the entire grid of users who were precisely connected with it.
- It offers zero management overhead, enhanced security levels, and easy management of diverse users with directory service in a SaaS solution.
- It is utterly known that cloud-based services are enabled, configured, and hosted by external providers. This scenario may also get the least hassle, either for users or clients. As a result, many organizations can enhance their productivity with cloud IAM.
- SaaS protocol is created and used as a hub for connecting with all virtual networks of distributors, suppliers, and partners.
- Business users can deal with all services and programs in one place with cloud services, and Identity management can be enabled with a click on a single dashboard.
- Easily connect your cloud servers, which are virtually hosted at Google Cloud, AWS, or elsewhere right next to your current LDAP or AD user store.
- Widen and extend your present LDAP or AD directory right to the cloud.

LoginFormDemo.java

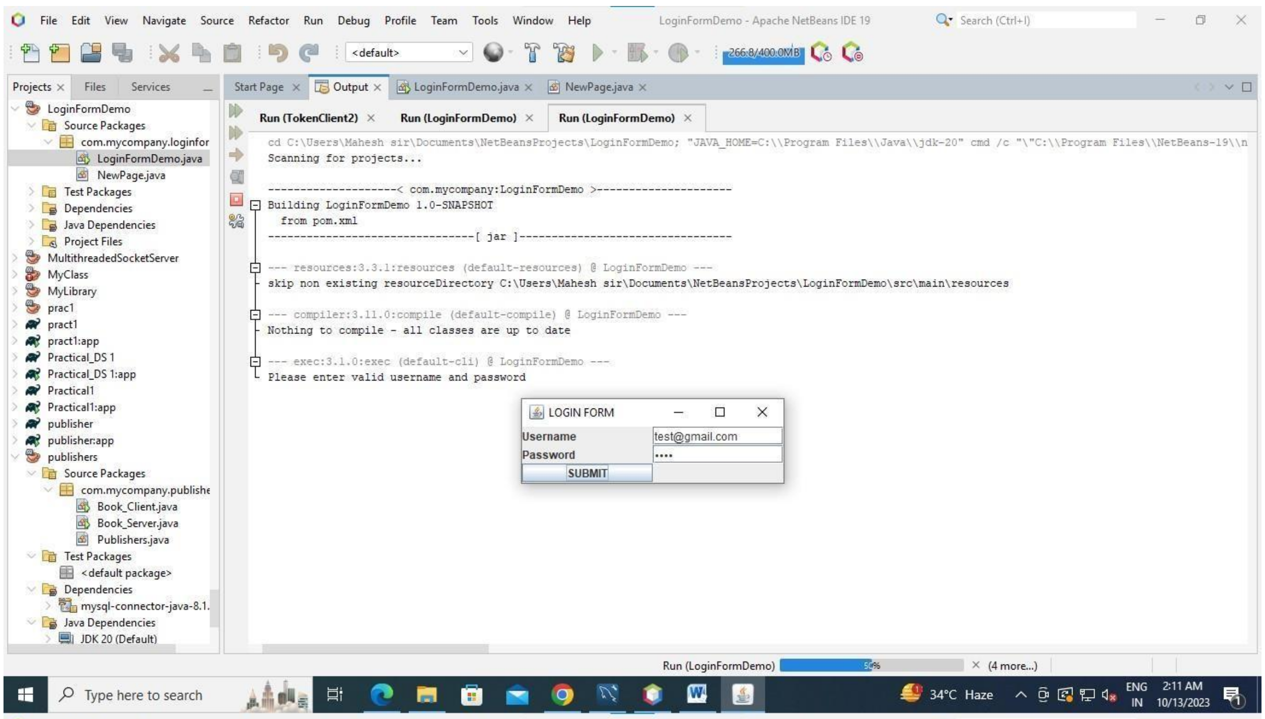
```
import javax.swing.*;
import java.awt.*; import
java.awt.event.*;  import
java.lang.Exception;
//create CreateLoginForm class to create login form
 //class extends JFrame to create a window where our component add
//class implements ActionListener to perform an action on button click
class CreateLoginForm extends JFrame implements ActionListener
 {
 //initialize button, panel, label, and text field
JButton b1;
 JPanel newPanel;
 JLabel userLabel, passLabel; final
JTextField textField1, textField2;
//calling constructor
 CreateLoginForm()
{
  //create label for username userLabel
= new JLabel();
 userLabel.setText("Username"); //set label value for textField1
//create text field to get username from the user textField1
= new JTextField(15); //set length of the text

 //create label for password
passLabel = new JLabel();
 passLabel.setText("Password"); //set label value for textField2
//create text field to get password from the user textField2 = new
 JPasswordField(15); //set length for the password
//create submit button
b1 = new JButton("SUBMIT"); //set label to button
  //create panel to put form elements newPanel = new
JPanel(new       GridLayout(3, 1)); newPanel.add(userLabel); //set username
label to panel newPanel.add(textField1);       //set   text   field   to
   panel newPanel.add(passLabel); //set password label to panel
newPanel.add(textField2);      //set   text   field   to     panel
newPanel.add(b1); //set button to panel
//set     border     to     panel     add(newPanel,
 BorderLayout.CENTER); //perform
action on button click
 b1.addActionListener(this); //add action listener to button
setTitle("LOGIN FORM"); //set title to the login form
 }
  //define abstract method actionPerformed() which will be called on button click public
void actionPerformed(ActionEvent ae) //pass action listener as a parameter
```

```
 {
String userValue = textField1.getText(); //get user entered username from the textField1
String passValue = textField2.getText(); //get user entered pasword from the textField2
//check whether the credentials are authentic or not if
 (userValue.equals("test1@gmail.com") && passValue.equals("test")) {
//if authentic, navigate user to a new page
//create instance of the NewPage
 NewPage page = new NewPage(); //make
page visible to the user
page.setVisible(true);
  //create a welcome label and set it to the new page JLabel
wel_label = new JLabel("Welcome: "+userValue);
page.getContentPane().add(wel_label);
}
else{
 //show error message
 System.out.println("Please enter valid username and password");
 }
 }
}
 //create the main class
 class LoginFormDemo
 {
 //main() method start public static
void main(String arg[])
 {
try
 {
 //create instance of the CreateLoginForm
CreateLoginForm form = new CreateLoginForm();
form.setSize(300,100); //set size of the frame
form.setVisible(true); //make form visible to the user
} catch(Exception e)
 {
//handle exception
JOptionPane.showMessageDialog(null, e.getMessage());
 }
 }
 }
```

NewPage.java

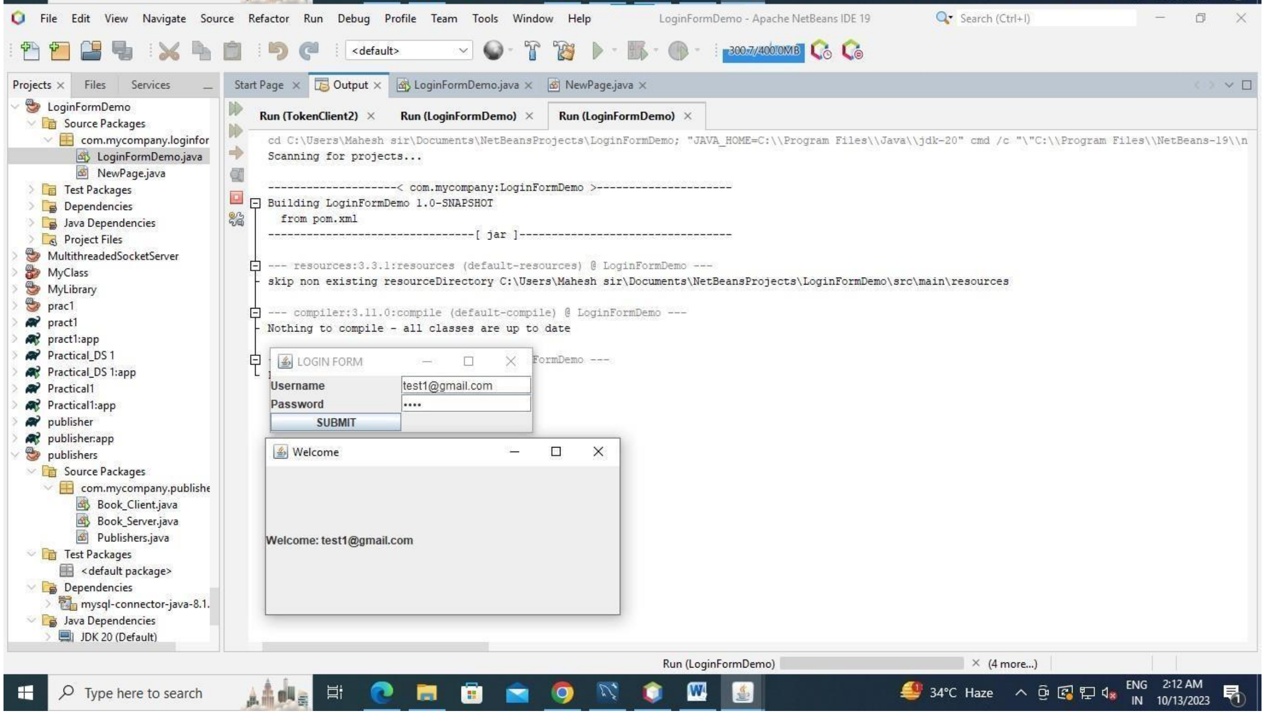
```
import javax.swing.*; import
java.awt.*;

 //create NewPage class to create a new page on which user will navigate
 class NewPage extends JFrame
 {
 //constructor
 NewPage()
 {
 setDefaultCloseOperation(javax.swing.
 WindowConstants.DISPOSE_ON_CLOSE); setTitle("Welcome"); setSize(400,
 200);
 }
}
```

OUTPUT



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\LoginFormDemo; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\n
Scanning for projects...

--------------------< com.mycompany:LoginFormDemo >--------------------
Building LoginFormDemo 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LoginFormDemo ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\LoginFormDemo\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LoginFormDemo ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ LoginFormDemo ---
Please enter valid username and password
```

LOGIN FORM
Username: test@gmail.com
Password: ••••
SUBMIT



```
cd C:\Users\Mahesh sir\Documents\NetBeansProjects\LoginFormDemo; "JAVA_HOME=C:\\Program Files\\Java\\jdk-20" cmd /c "\"C:\\Program Files\\NetBeans-19\\n
Scanning for projects...

--------------------< com.mycompany:LoginFormDemo >--------------------
Building LoginFormDemo 1.0-SNAPSHOT
  from pom.xml
--------------------------------[ jar ]---------------------------------

--- resources:3.3.1:resources (default-resources) @ LoginFormDemo ---
skip non existing resourceDirectory C:\Users\Mahesh sir\Documents\NetBeansProjects\LoginFormDemo\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ LoginFormDemo ---
Nothing to compile - all classes are up to date
```

LOGIN FORM
Username: test1@gmail.com
Password: ••••
SUBMIT

Welcome

Welcome: test1@gmail.com

PRACTICAL 10

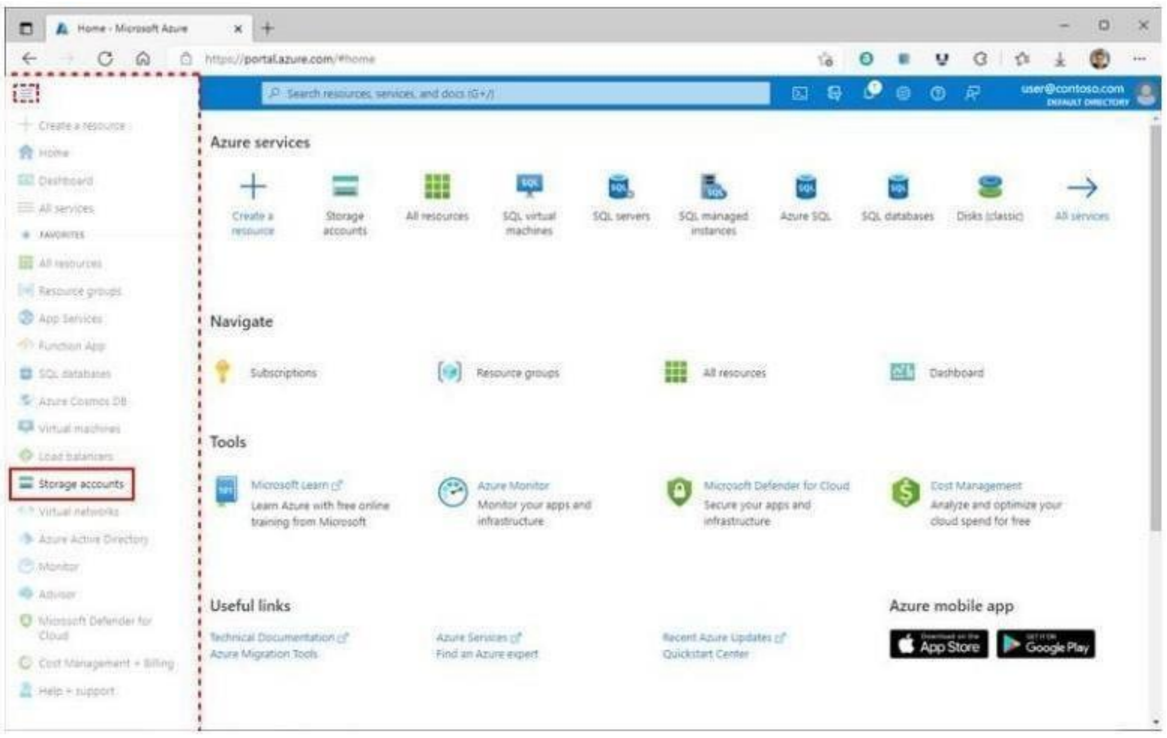
AIM: Implementation of Storage as a Service using Google Docs.

Create a storage account:

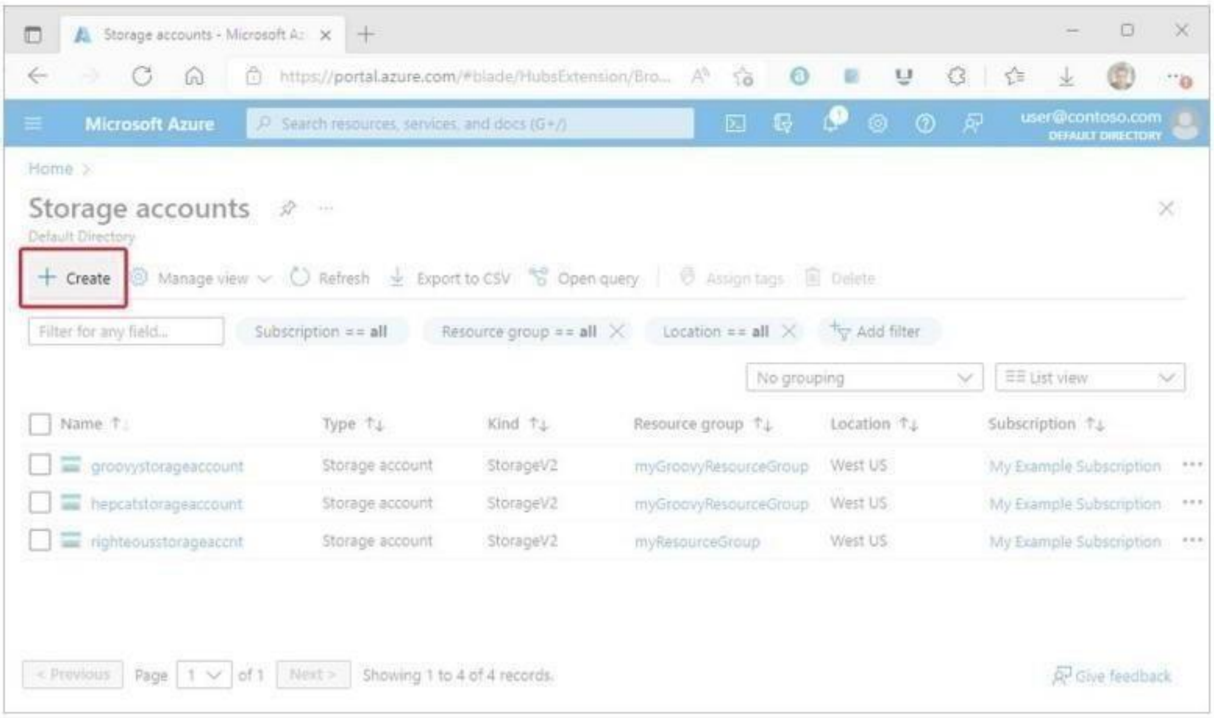
A storage account is an Azure Resource Manager resource. Resource Manager is the deployment and management service for Azure. For more information, see Azure Resource Manager overview. Every Resource Manager resource, including an Azure storage account, must belong to an Azure resource group. A resource group is a logical container for grouping your Azure services. When you create a storage account, you have the option to either create a new resource group, or use an existing resource group. This how-to shows how to create a new resource group.

To create an Azure storage account with the Azure portal, follow these steps:

1. From the left portal menu, select Storage accounts to display a list of your storage accounts. If the portal menu isn't visible, click the menu button to toggle it on.

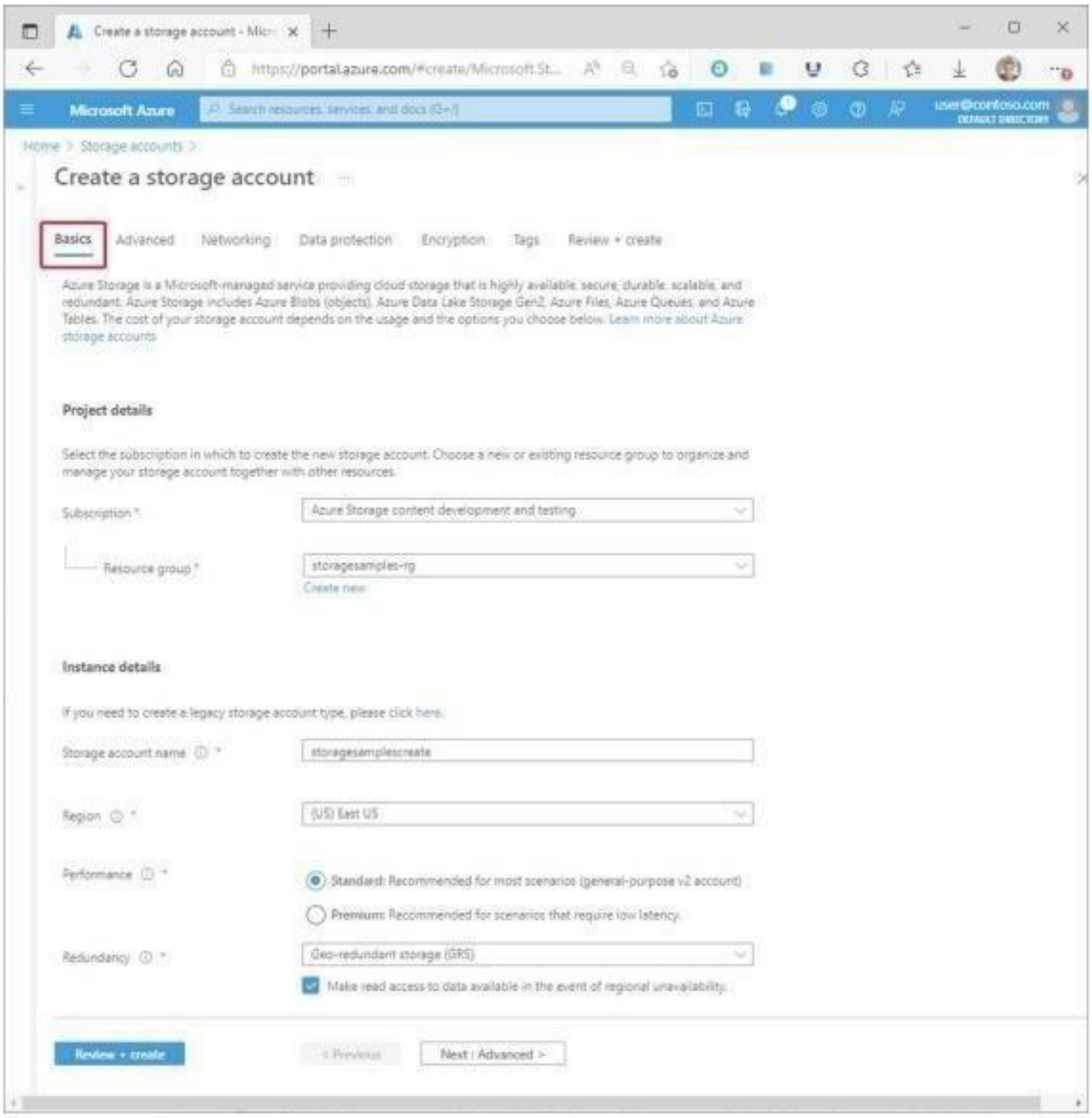


2. On the Storage accounts page, select Create.



Name	Type	Kind	Resource group	Location	Subscription
groovystorageaccount	Storage account	StorageV2	myGroovyResourceGroup	West US	My Example Subscription
hepcatstorageaccount	Storage account	StorageV2	myGroovyResourceGroup	West US	My Example Subscription
righteousstorageaccnt	Storage account	StorageV2	myResourceGroup	West US	My Example Subscription

The following image shows a standard configuration of the basic properties for a new storage account.


Create a storage account - Micr
https://portal.azure.com/#create/Microsoft.St...
Microsoft Azure
Search resources, services, and docs (G+/)
user@contoso.com
DEFAULT DIRECTORY
Home > Storage accounts >

Create a storage account

Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. Learn more about Azure storage accounts

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *: Azure Storage content development and testing

Resource group *: storagesamples-rg

Create new

Instance details

If you need to create a legacy storage account type, please click here.

Storage account name *: storagesamplescreate

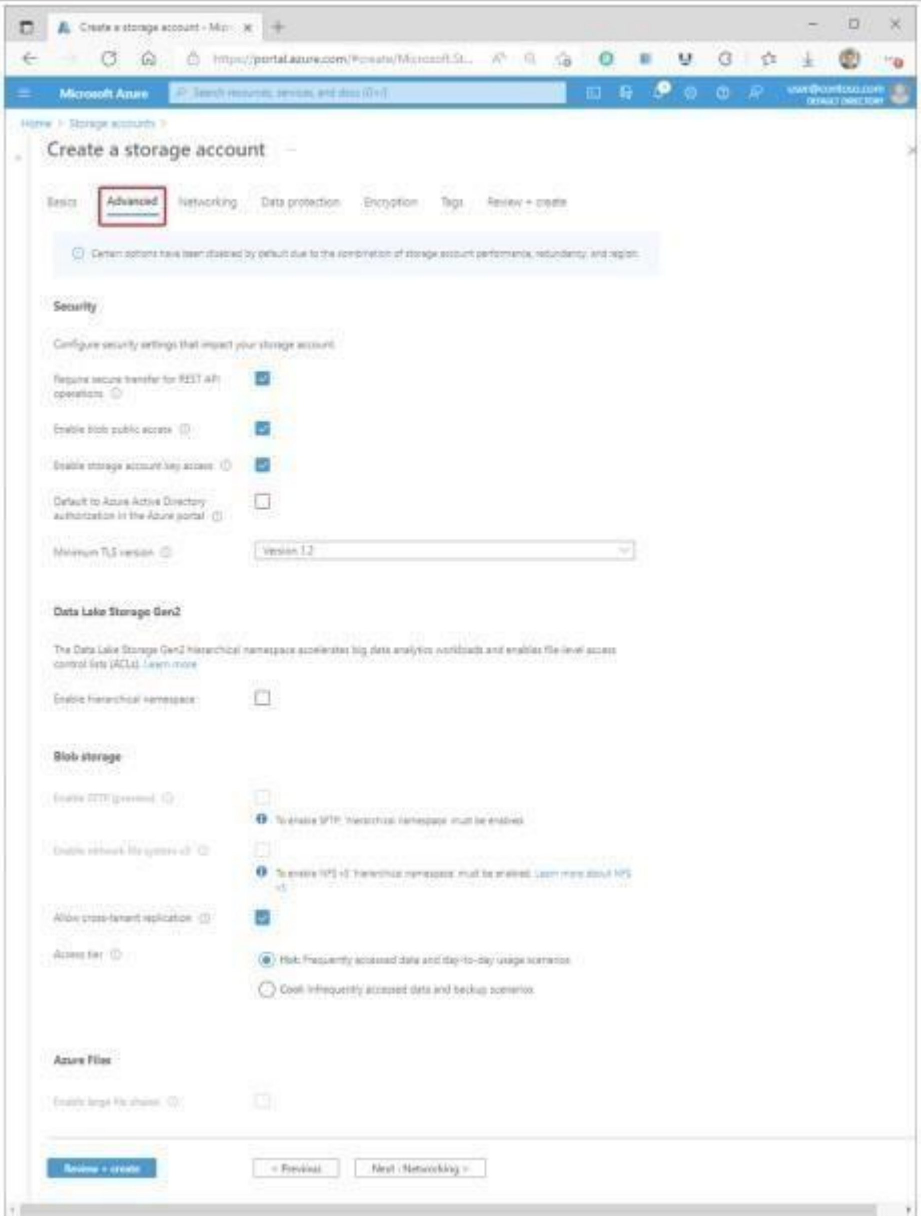
Region *: (US) East US

Performance *: [✓ **Standard:** Recommended for most scenarios (general-purpose v2 account)] **Premium:** Recommended for scenarios that require low latency.

Redundancy *: Geo-redundant storage (GRS)

[✓] Make read access to data available in the event of regional unavailability.

The following image shows a standard configuration of the advanced properties for a new storage account.


Create a storage account - Microsoft
https://portal.azure.com/#create/Microsoft.St...
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Storage accounts >
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Review + create
< Previous
Next : Networking >

Create a storage account

Certain options have been disabled by default due to the combination of storage account performance, redundancy, and region.

Security

Configure security settings that impact your storage account.

Require secure transfer for REST API operations	✓
Enable blob public access	✓
Enable storage account key access	✓
Default to Azure Active Directory authorization in the Azure portal	☐
Minimum TLS version	Version 1.2

Data Lake Storage Gen2

The Data Lake Storage Gen2 hierarchical namespace accelerates big data analytics workloads and enables file-level access control lists (ACLs). Learn more

Enable hierarchical namespace	☐

Blob storage

Enable SFTP (preview)	☐ To enable SFTP, hierarchical namespace must be enabled.
Enable network file system v3	☐ To enable NFS v3 hierarchical namespace must be enabled. Learn more about NFS v3
Allow cross-tenant replication	✓
Access tier	[✓ Hot: Frequently accessed data and day-to-day usage scenarios] Cool: Infrequently accessed data and backup scenarios

Azure Files

Enable large file shares	☐