Practical No: 01

Exercise:

1. **Develop a JAVA program for multi-client chat server.**

<u>Program:</u>

Server.java

```
package myPack; import
java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

public class Server

{
        private ServerSocket serverSocket;

        public Server (ServerSocket serverSocket)

        {
                this.serverSocket=serverSocket;
        }

        public void startServer()
        {
        try
                {
                        while(!serverSocket.isClosed())
                        {
                                Socket socket=serverSocket.accept();
                                  System.out.println("A new client has connected!");
                                ClientHandler clientHandler = new
ClientHandler(socket);
                                Thread thread=new Thread(clientHandler);
                                thread.start();
                        }
                }
```

```
                catch(IOException e)
                {
                        System.out.println("A new client has connected!");
                }
        }

        public void closeServerSocket()
        {
        try
        {
                        if(serverSocket != null)
                        {
                                serverSocket.close();
                        }
                }
                catch(IOException e)
                {
                        e.printStackTrace();
                }
        }
        public static void main(String[] args) throws IOException
        {
                ServerSocket serverSocket = new ServerSocket(8867);
        Server server = new Server(serverSocket);
                server.startServer();
        }
}
```

Clienthandler.java

```
package myPack;

import java.io.BufferedReader;
import java.io.BufferedWriter; import
java.io.IOException; import
java.io.InvalidObjectException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.Socket; import
java.util.ArrayList;
public class ClientHandler  implements Runnable{
```

```
public static ArrayList<ClientHandler> clientHandlers = new ArrayList<>();
private Socket socket;
private BufferedReader bufferedReader;
private BufferedWriter bufferedwriter;
private String clientUsername;
   private String messageToSend;

   public ClientHandler(Socket socket)
   {
try
   {
   this.socket=socket;
   this.bufferedwriter = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
this.bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
this.clientUsername=bufferedReader.readLine();
   clientHandlers.add(this);
   broadcastMessage("SERVER: "+ clientUsername + " has entered the chat!");
   }
   catch(IOException e)
   {
   closeEverything(socket,bufferedReader, bufferedwriter);
   }
   }
@Override
public void run()
{
String messageFromClient;
while(socket.isConnected())
{
try
{
messageFromClient=bufferedReader.readLine();
broadcastMessage(messageFromClient);

}
catch(IOException e)
{
closeEverything(socket,bufferedReader, bufferedwriter);
break;
}
```

```
}
}
public void broadcastMessage(String messageToSend)
{
for(ClientHandler clientHandler:clientHandlers)
{
try
{
if(!clientHandler.clientUsername.equals(clientUsername))
{
clientHandler.bufferedwriter.write(messageToSend);
clientHandler.bufferedwriter.newLine();
clientHandler.bufferedwriter.flush();
}
}
catch(IOException e)
{
closeEverything(socket,bufferedReader, bufferedwriter);
}
}
}
public void removeClientHandler()
{
clientHandlers.remove(this);
broadcastMessage("SERVER: "+ clientUsername + "has left the chat!");
}
public void closeEverything(Socket socket, BufferedReader bufferedReader,
BufferedWriter bufferedwriter)
{
removeClientHandler()
; try {
if(bufferedReader != null)
{
bufferedReader.close();
}
if(bufferedwriter != null);
{
bufferedwriter.close();
}
if(socket != null)
{
socket.close();
```

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

Client.java

```
package myPack;

import java.io.BufferedReader; import
java.io.BufferedWriter; import
java.io.IOException; import
java.io.InputStreamReader; import
java.io.OutputStreamWriter; import
java.net.Socket; import
java.net.UnknownHostException;
import java.util.Scanner;

public class Client {

private Socket socket; private
BufferedReader bufferedReader;
private BufferedWriter bufferedWriter;
private String username;

public Client (Socket socket, String username)
{
try
{
this.socket=socket;
this.bufferedWriter = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
this.bufferedReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
this.username = username;
}
catch(IOException e)
{
closeEverything(socket, bufferedReader, bufferedWriter);
```

```
}
}
public void sendMessage()
{
try
{
bufferedWriter.write(username); bufferedWriter.newLine();
bufferedWriter.flush();
Scanner scanner = new Scanner(System.in); while(socket.isConnected())
{
String messageToSend = scanner.nextLine();
bufferedWriter.write(username+":"+messageToSend);
bufferedWriter.newLine();
bufferedWriter.flush();
}
}
catch(IOException e)
{
closeEverything(socket, bufferedReader, bufferedWriter);
}
}
public void listenForMessage()
{
new Thread(new Runnable()
{
@Override
public void run()
{
String msgFromGroupChat;
while(socket.isConnected())
{
try
{
msgFromGroupChat = bufferedReader.readLine();
System.out.println(msgFromGroupChat);
}
catch(IOException e)
{
closeEverything(socket, bufferedReader, bufferedWriter);
}
}
}
```

```
}).start();
}
public void closeEverything(Socket socket,BufferedReader bufferedReader,
BufferedWriter bufferedwriter)
{
try
{
if(bufferedReader != null)
{
bufferedReader.close();
}
if(bufferedWriter != null)
{
bufferedWriter.close();
}
if(socket != null)
{
socket.close();
}
}
catch (IOException e)
{
e.printStackTrace();
}
}
public static void main(String[] args)throws UnknownHostException,
IOException
{
Scanner scanner = new Scanner(System.in);
System.out.println("Enter your username for the group chat: ");
String username = scanner.nextLine();
Socket socket = new Socket("localhost", 8867); Client
client = new Client(socket, username);
client.listenForMessage();
client.sendMessage();
}

}
```

Output:

```
Enter your username for the group chat:
Pratibha
Virat Kohli
```

```
A new client has connected!
```

```
Enter your username for the group chat:
Vedika
Rohit
```

```
Enter your username for the group chat:
Pratibha
Virat Kohli
SERVER: Vedika has entered the chat!
Rohit
```

```
Enter your username for the group chat:
Neha
ABD
```

```
Enter your username for the group chat:
Vedika
Rohit
SERVER: Neha has entered the chat!
ABD
```

```
Enter your username for the group chat:
Neha
ABD
Vedika:Hello
Rohit : Hello
```

```
Enter your username for the group chat:
Vedika
Rohit
SERVER: Neha has entered the chat!
ABD
Hello
Neha:Hii
ABD : Hello
```

```
Enter your username for the group chat:
Pratibha
Virat Kohli
SERVER: Vedika has entered the chat!
Rohit
SERVER: Neha has entered the chat!
ABD
Vedika:Hello
Rohit:Hello
Neha:Hii
ABD:Hii
How are you?
```

```
Enter your username for the group chat:
Neha
Vedika:Hello
Hii
Pratibha:Hii! How are you?
```

2. Write a java program to implement mutual exclusion using Token ring algorithm.

Program:

TCP_MyServer.java

```
import java.io.*;     import
java.net.*;     public class
TCP_MyServer {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                try {
                        ServerSocket ss = new ServerSocket(1114);

                        Socket s =ss.accept();
                        DataInputStream in=new DataInputStream(System.in);

                        DataInputStream dis=new DataInputStream(s.getInputStream());

                        DataOutputStream dout=new
DataOutputStream(s.getOutputStream());


                        String str=dis.readUTF();
                        System.out.println("message="+str);
                        System.out.println("enter message for client...");
                        String
str1=in.readLine();
dout.writeUTF(str1);
dout.flush();                   dout.close();
                s.close();
                }
                catch(Exception e){
System.out.println(e);
                }
```

```
        }

}
```

TCP_MyClient.java

```
import java.io.*; import
java.net.*; public class
TCP_MyClient {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                try {
                        Socket s = new Socket("localhost",1114);
                        DataOutputStream dout=new
DataOutputStream(s.getOutputStream());

                        DataInputStream dis=new DataInputStream(s.getInputStream());
                        DataInputStream in=new DataInputStream(System.in);

                        System.out.println("enter message for server...");




                        String str=in.readLine();



                        dout.writeUTF(str);
                        String str1=dis.readUTF();
                        System.out.println("message="+str1);
                        dout.flush();
                dout.close();
        s.close();
                }
                catch(Exception e){
System.out.println(e);
                }


        }
```

}

Output:

```
enter message for server...
hello from pratibha
Hello from Virat Kohli
```

```
message=hello from pratibha
Virat Kohli
enter message for client...
```

```
enter message for server...
hello from pratibha
Hello from Virat Kohli
message=pratibha
message=Virat Kohli
```

Practical No:-02

1.Write a java program to implement a Server calculator using RPC concept. (Make use of datagram)

Program:-

Serverside- package prac_2;

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.ServerSocket;
import java.net.Socket; import
java.util.StringTokenizer; public
class Serverside { int port;
ServerSocket ss;
Socket socket; public
Serverside() {
this.port = 0;
}
public Serverside(int port) { this.port
= port;
}
public double addition(int n1 , int n2) { return
n1+n2;
}
public double substraction(int n1 , int n2) { return
n1-n2;
}
public double multiplication(int n1 , int n2) { return
n1*n2;
}
public double division(int n1 , int n2) { return
n1/n2;
}
```

```
public void listen() {
try {
System.out.print("Server Start"); ss
= new ServerSocket(port); socket =
ss.accept();
DataOutputStream dout = new DataOutputStream(socket.getOutputStream());
DataInputStream dis = new DataInputStream(socket.getInputStream()); double
result = 0.0; while(true) {
String str = dis.readUTF();
StringTokenizer st = new StringTokenizer(str , "-");
int choice = Integer.parseInt(st.nextToken()); int
num1 = Integer.parseInt(st.nextToken()); int num2
= Integer.parseInt(st.nextToken()); Serverside cs =
new Serverside(); switch(choice) { case 1: result =
cs.addition(num1, num2); break; case 2: result =
cs.substraction(num1, num2); break; case 3:
result = cs.multiplication(num1, num2);
break; case 4: result = cs.division(num1,
num2); break;
}
System.out.println("REsult for " + str + " is -");
String res = Double.toString(result);
System.out.println(res);
dout.writeUTF(res); dout.flush();
dis.close(); dout.close();
socket.close();
}
}catch(Exception e) {
System.out.println(e.getMessage());
}
}
```

```
public static void main(String[] args) { Serverside
cs = new Serverside(2000);
cs.listen();
}
}
```

Clientside-

```
package prac_2;
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
//17-10-2023
public class Clientside {
Socket socket;
int port; public
Clientside(int port) {
this.port = port;
}
public void sndReq() throws Exception{ socket=
new Socket("localhost",port);
DataOutputStream dout = new DataOutputStream(socket.getOutputStream());
DataInputStream din = new DataInputStream(socket.getInputStream());
BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
String str=""; int num1,num2;
System.out.println("1.Addition \n2.Substraction \n3.Multiplication\n4.Division \n5.Exit");
System.out.println("\nEnter your Choice : "); int choice = Integer.parseInt(in.readLine());
System.out.println("Val = " +choice);
//int choice = 1;//Integer.parseInt(in.readUTF());
switch(choice) { case 1 :
str += choice+"-";
```

```
System.out.println("Enter 1st Number : ");
num1 = Integer.parseInt(in.readLine()); str
+= num1+"-";
System.out.println("Enter 2nd Number : ");
num2 = Integer.parseInt(in.readLine());
str += num2; break; case 2 :
str += choice+"-";
System.out.println("Enter 1st Number : ");
num1 = Integer.parseInt(in.readLine()); str
+= num1+"-";
System.out.println("Enter 2nd Number : ");
num2 = Integer.parseInt(in.readLine()); str
+= num2; break; case 3:
str += choice+"-";
System.out.println("Enter 1st Number : ");
num1 = Integer.parseInt(in.readLine()); str
+= num1+"-";
System.out.println("Enter 2nd Number : ");
num2 = Integer.parseInt(in.readLine()); str
+= num2; break; case 4:
str += choice+"-";
System.out.println("Enter 1st Number : ");
num1 = Integer.parseInt(in.readLine()); str
+= num1+"-";
System.out.println("Enter 2nd Number : ");
num2 = Integer.parseInt(in.readLine()); str
+= num2;

break; case
5:
System.out.println("Program   Exited!");
break; default :
System.out.println("Invalid Choice!");
```

```
}
System.out.println(str);
dout.writeUTF(str); dout.flush();
String result = din.readUTF();
System.out.println("Result is "+result);
din.close(); dout.close(); socket.close();
}
public static void main(String[] args) {
try {
Clientside cc = new Clientside(2000); cc.sndReq();
}
catch(Exception e) {
System.out.println("Meassage "+e.getMessage());
}
}
}
```

OUTPUT:-

```
Serverside [Java Application] D:\eclipse-jee-2022-12-R-win3
Server Start
```

```
1.Addition
2.Substraction
3.Multiplication
4.Division
5.Exit

Enter your Choice :
1
Val = 1
Enter 1st Number :
9
Enter 2nd Number :
10
```

```
Server StartResult for 1-9-10 is -
19.0
Socket closed
```

2. Write a java to implement a Date Time Server using RPC concept. (Make use of Datagram)

Program:-

UDPServer:-

```
package prac_2; import java.io.IOException;  import
java.net.DatagramPacket; import java.net.DatagramSocket;  import
java.net.SocketException; import java.net.InetAddress; import
java.util.Date;  public class UDPServer {                public static void
main(String[] args) throws SocketException{
                        // TODO Auto-generated method stub
            DatagramPacket dpac;
 DatagramSocket dsoc = new DatagramSocket();
 System.out.print("SeverUp");
 try
{
 while(true) {
System.out.print("Sending");
System.out.println();
Thread.sleep(1000);
String time = new Date().toString();
byte b[] = time.getBytes();  dpac =
new
DatagramPacket(b,b.length,InetAddress.getByName("localhost"),1314);
System.out.println();
dsoc.send(dpac);
 }
 }
catch(IOException | InterruptedException e) {
            System.out.println(e);
 }
dsoc.close();
          }
```

```
 }
```

UDPClient:-

```
package prac_2;



import java.io.IOException; import
java.net.DatagramPacket; import
java.net.DatagramSocket;

public class UDPClient{     public static void main(String[]
args) throws IOException {         int port = 1314; // Use the
same port as the server
        DatagramSocket socket = new DatagramSocket(port);

        System.out.println("Client receiver is up and running on port " + port);

        while (true) {             byte[] buffer
= new byte[1024];
            DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
            socket.receive(packet);

            String message = new String(packet.getData(), 0, packet.getLength());
            System.out.println("We received: " + message);
        }
    }
}
```

OUTPUT:-

```
UDPServer [Java Application] D:\eclipse-jee-2022-12-R-win32-x86_64\eclipse
SeverUpSending

Sending

Sending

Sending

Sending
```

```
UDPClient [Java Application] D:\eclipse-jee-2022-12-R-win32-x86_64\eclipse\plugins\org
Client receiver is up and running on port 1314
We received: Fri Nov 03 00:11:57 IST 2023
We received: Fri Nov 03 00:11:58 IST 2023
We received: Fri Nov 03 00:11:59 IST 2023
We received: Fri Nov 03 00:12:00 IST 2023
We received: Fri Nov 03 00:12:01 IST 2023
We received: Fri Nov 03 00:12:02 IST 2023
We received: Fri Nov 03 00:12:03 IST 2023
```

Practical No. 03

Q.

Server.java

```
package mypack;
 import java.rmi.Naming; import
java.rmi.registry.LocateRegistry;
 public class Server
{     public static void main(String[] args) {
                    try
                    {
                              System.out.println("Calculator Service Started");                    adder
stub= new CalcOperation();
                                        System.out.println("Calculator Service Binding...");
                                   LocateRegistry.createRegistry(5000);


  Naming.rebind("rmi://localhost:5000/CalcOpservice",stub);          System.out.println("Calculator
Service is registered in registery");
                    }
                       catch(Exception e)
                    {
                              System.out.println(e);
                    }
          }
  }
```

Client.java

```
package mypack;   import
java.rmi.Naming; public class Client {


        public static void main(String[] args) {                    try
                {
                                System.out.println("Client Program Started");


                            adder stub =
(adder)Naming.lookup("rmi://localhost:5000/CalcOpservice");
System.out.println("Addition is:
"+stub.getAddition(34,4));
                                System.out.println("Subtraction is:
"+stub.getSubtraction(34,4));
                }
                  catch(Exception e)
                {
                            System.out.println(e);
                }
        }
}
```

CalcOperation.java

```
package mypack;   import
java.rmi.RemoteException;
import    java.rmi.server.UnicastRemoteObject;          public    class    CalcOperation    extends
UnicastRemoteObject implements adder {  private static final long serialVersionUID = 1L;
            CalcOperation() throws RemoteException
        {
super();          }
         @Override
                public int getAddition(int num1,int num2) throws RemoteException
        {
                return num1+num2;      }
```

```
        @Override
                        public int getSubtraction(int num1,int num2) throws RemoteException
        {
                return num1-num2;
        }
}
```

adder.java

```
package mypack;

import java.rmi.Remote; import
java.rmi.RemoteException;  public interface
adder extends Remote {   public int
getAddition(int num1,int num2)
throws RemoteException;

        public int getSubtraction(int num1,int num2)      throws
RemoteException;
}
```

Output :

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console ×
Server (1) [Java Application] C:\Program Files\eclipse_EE\eclipse\plugins\org.eclipse.justj.op
Calculator Service Started
Calculator Service Binding...
Calculator Service is registered in registery
```

```
Markers  Properties  Servers  Data Source Explorer  Snippets  Console ×
<terminated> Client (1) [Java Application] C:\Program Files\eclipse_EE\eclipse\plugins\org.ec
Client Program Started
Addition is: 38
Subtraction is: 30
```

Q.22. Retrieve day, me and date func on from server to client. This program should display server day, date and me. Ans :

• Code : DateTimeClient.java package dateTime;
import
java.rmi.registry.LocateRegistry; import

```
java.rmi.registry.Registry; public class
DateTimeClient { public sta c void
main(String[] args)  throws Excep on{ try {
Registry registry = LocateRegistry.getRegistry("localhost",
5050); DateTimeService dateTimeService =
(DateTimeService) registry.lookup("DateTimeService");
// Invoke remote methods
String day = dateTimeService.getCurrentDay();
String me = dateTimeService.getCurrentTime();
String date = dateTimeService.getCurrentDate();
System.out.println("Server Day: " + day);
System.out.println("Server Time: " + me);
System.out.println("Server Date: " + date);
} catch (Excep on e) { e.printStackTrace();
}
}
}
```

DateTimeServer.java

```
package dateTime; import
java.rmi.registry.LocateRegistry; import
java.rmi.registry.Registry;
import
java.rmi.server.UnicastRemoteObject
; public class DateTimeServer {
public sta c void main(String[] args)
{ try {
DateTimeService dateTimeService = new
DateTimeServiceImpl();
DateTimeService stub = (DateTimeService)
UnicastRemoteObject.exportObject(dateTimeSe rvice,
0); Registry registry =
LocateRegistry.createRegistry(5050);
registry.rebind("DateTimeService", stub);
System.out.println("Server is running..."); } catch
```

```
(Excep on e) { e.printStackTrace();
}
}
}
```

DateTimeService.java

```
package dateTime; import java.rmi.Remote; import
java.rmi.RemoteExcep on; interface
DateTimeService extends Remote {
String getCurrentDay() throws
RemoteExcep on;
String getCurrentTime() throws RemoteExcep on;
String getCurrentDate() throws RemoteExcep on;
}
```

DateTimeServiceImpl.java

```
package dateTime; import java.rmi.RemoteExcep
on; import java.text.SimpleDateFormat; import
java.u l.Date; class DateTimeServiceImpl
implements
DateTimeService {
@Override public String getCurrentDay()
throws RemoteExcep on { SimpleDateFormat
sdf = new SimpleDateFormat("EEEE"); return
sdf.format(new Date());
}
@Override
public String getCurrentTime() throws RemoteExcep on {
SimpleDateFormat sdf = new
SimpleDateFormat("HH:mm:ss"); return sdf.format(new
Date());
}
@Override
```

```
public String getCurrentDate() throws RemoteExcep on {
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MMdd");
return sdf.format(new Date());
}
}
```

- Output :

```
Javadoc  Declaration  Console ×
DateTimeServer [Java Application] E:\ALL\eclipse_EE\eclipse\plugins\org.eclipse.just
Server is running...
```

```
Javadoc  Declaration  Console ×
<terminated> DateTimeClient [Java Application] E:\ALL\eclipse_EE\eclipse\plugi
Server Day: Wednesday
Server Time: 22:30:45
Server Date: 2023-12-13
```

DSCC Practical 4

a) 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.

LibraryDBClient.java import
java.io.BufferedReader; import
java.io.InputStreamReader; import
java.rmi.Naming; public class
LibraryDBClient {
  public static void main(String[] args) {
String sql = "", ch = ""; try {
    LibraryDBInf stub = (LibraryDBInf)
Naming.lookup("rmi://localhost:1901/ROCforLibrary");
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
while (true) {
      System.out.println("Select an Option");
      System.out.println("1: Retrieve Book Information");
      System.out.println("2: Insert Book Information");
      System.out.println("3: Exit");
      System.out.println("Enter your Choice");
ch = br.readLine(); if (ch.equals("1")) {
sql = "SELECT * FROM book"; sql =
stub.getData(sql);
      } else if (ch.equals("2")) {
        // Example insert query, modify as needed
        sql = "INSERT INTO book(Book_id, Book_name, Book_author) VALUES (1, 'Java
Programming', 'John Doe')";
        sql = stub.insertData(sql);
      } else if (ch.equals("3")) {
        System.exit(0);
      } else { sql = "Please
select a valid option";
      }
      System.out.println(sql);
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
 }
}

LibraryDBInf.java import
java.rmi.Remote; import
java.rmi.RemoteException;
public interface LibraryDBInf extends Remote {
   String getData(String strQry) throws RemoteException;
   String insertData(String strQry) throws RemoteException;
}
LibraryDBOperations.java import java.rmi.RemoteException; import
java.rmi.server.UnicastRemoteObject; import java.sql.Connection; import
java.sql.DriverManager; import java.sql.ResultSet; import java.sql.ResultSetMetaData;
import java.sql.Statement; public class LibraryDBOperations extends
UnicastRemoteObject implements LibraryDBInf {   private static final long
serialVersionUID = 1L;
   Connection con;
   Statement stmt;
   ResultSet rs;
   ResultSetMetaData rsmd;
   String colStr, resultStr;   public LibraryDBOperations()
throws RemoteException {   super();   con = null;
stmt = null;   rs = null;   rsmd = null;   colStr = "";
resultStr = "";
   }
   public void setDBCon() {
      try {
         String URL = "jdbc:mysql://localhost:3306/library";
Class.forName("com.mysql.jdbc.Driver");
         con = DriverManager.getConnection(URL, "root", "");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   @Override
   public String getData(String strQry) throws RemoteException {
try {   setDBCon();   stmt = con.createStatement();
rs = stmt.executeQuery(strQry);   rsmd = rs.getMetaData();
for (int i = 1; i <= rsmd.getColumnCount(); i++) {   colStr
= colStr + rsmd.getColumnName(i) + "\t";
         }
         while (rs.next()) {   for (int i = 1; i <=
rsmd.getColumnCount(); i++) {   resultStr =
resultStr + rs.getString(i) + "\t";
            }
            resultStr = resultStr + "\n";

```
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return colStr + "\n\n" + resultStr;
  }
  @Override    public String insertData(String strQry) throws
RemoteException {       try {           setDBCon();           stmt =
con.createStatement();           int recordInserted =
stmt.executeUpdate(strQry);           if (recordInserted != 0)
resultStr = "Record inserted successfully.";           else
            resultStr = "Record not inserted successfully.";
    } catch (Exception e) {
        e.printStackTrace();
    }
    return resultStr;
  }
}
```

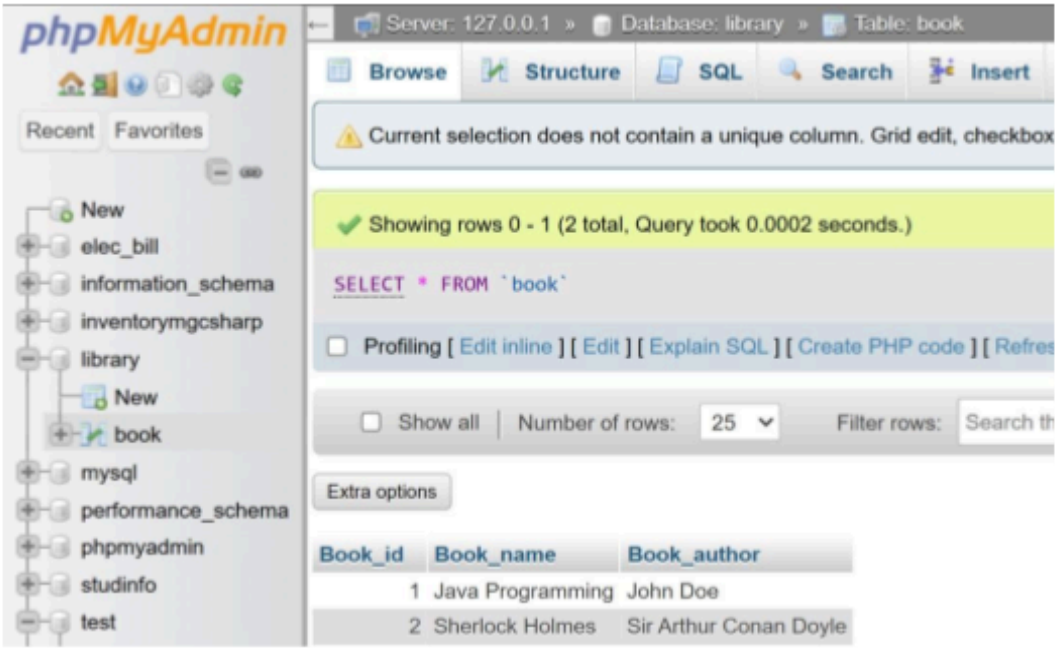
LibraryDBSrv.java import
java.rmi.Naming; import
java.rmi.registry.LocateRegistry; public
class LibraryDBSrv {   public static void
main(String[] args) {   try {

```
        LibraryDBInf skeleton = new LibraryDBOperations();
        LocateRegistry.createRegistry(1901);
        Naming.rebind("rmi://localhost:1901/ROCforLibrary", skeleton);
        System.out.println("Library Server Registered.");
    } catch (Exception e1) {
e1.printStackTrace();
    }
  }
}
```

Output:

```
Console ×
LibraryDBSrv [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\jre\b
Library Server Registered.
```

```
Console ×
LibraryDBClient [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\jre
Select an Option
1: Retrieve Book Information
2: Exit
Enter your Choice
1
Book_id Book_name       Book_author

1       Java Programming        John Doe
2       Sherlock Holmes Sir Arthur Conan Doyle

Select an Option
1: Retrieve Book Information
2: Exit
Enter your Choice
```



b) Using MySQL create Electric_Bill database. Create table Bill (consumer_name, bill_due_date, bill_amount) and retrieve the bill information from the Electric_Bill database using Remote Object Communication concept.

BillDBInf.java import java.rmi.Remote;
import java.rmi.RemoteException; public
interface BillDBInf extends Remote {
   String getData(String strQry) throws RemoteException;
   String insertData(String strQry) throws RemoteException;
}

BillDBOperations.java import java.rmi.RemoteException; import java.rmi.server.UnicastRemoteObject; import java.sql.Connection; import java.sql.DriverManager; import java.sql.ResultSet; import java.sql.ResultSetMetaData; import java.sql.Statement; public class BillDBOperations extends UnicastRemoteObject implements BillDBInf { private static final long serialVersionUID = 1L;

```
   Connection con;
   Statement stmt;
   ResultSet rs;
   ResultSetMetaData rsmd;
   String colStr, resultStr;  public BillDBOperations()
throws RemoteException {  super();  con =
null;  stmt = null;  rs = null;  rsmd = null;
colStr = "";  resultStr = "";
   }
   public void setDBCon() {
      try {
         String URL = "jdbc:mysql://localhost:3306/elec_bill";
Class.forName("com.mysql.jdbc.Driver");
         con = DriverManager.getConnection(URL, "root", "");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
   @Override  public String getData(String strQry) throws
RemoteException {  try {  setDBCon();
         System.out.println("Server Registered.");
stmt = con.createStatement();  rs =
stmt.executeQuery(strQry);  rsmd =
rs.getMetaData();  for (int i = 1; i <=
rsmd.getColumnCount(); i++) {  colStr = colStr +
rsmd.getColumnName(i) + "\t";
         }
         while (rs.next()) {  for (int i = 1; i <=
rsmd.getColumnCount(); i++) {  resultStr =
resultStr + rs.getString(i) + "\t";
            }
            resultStr = resultStr + "\n";
         }
      } catch (Exception e) {
         e.printStackTrace();
      }
      return colStr + "\n\n" + resultStr;
```

```
    }
    @Override     public String insertData(String strQry) throws
RemoteException {        try {            setDBCon();
          System.out.println("Server Registered.");
stmt = con.createStatement();            int recordInserted
= stmt.executeUpdate(strQry);            if (recordInserted
!= 0)                resultStr = "Record inserted
successfully.";            else
              resultStr = "Record not inserted successfully.";
       } catch (Exception e) {
          e.printStackTrace();
       }
       return resultStr;
    }
}
```

ElectricBillDBServiceClient.java

```
import java.io.BufferedReader; import
java.io.InputStreamReader; import
java.rmi.Naming; public class
ElectricBillDBServiceClient {     public
static void main(String[] args) {
String sql = "", ch = "";        try {
          BillDBInf stub = (BillDBInf) Naming.lookup("rmi://localhost:1901/ROCforBillDB");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
while (true) {
              System.out.println("Select an Option");
              System.out.println("1: Retrieve Bill Information");
              System.out.println("2: Insert Bill Information");
              System.out.println("3: Exit");
              System.out.println("Enter your Choice");
ch = br.readLine();                if (ch.equals("1")) {
sql = "SELECT * FROM Bill";                    sql =
stub.getData(sql);
              } else if (ch.equals("2")) {                    sql = "INSERT INTO Bill(consumer_name,
bill_due_date, bill_amount) VALUES
('John Doe', '2023-12-31', 100.00)";
                 sql = stub.insertData(sql);
              } else if (ch.equals("3")) {
                 System.exit(0);
              } else {                    sql = "Please select
a valid Option";
              }
              System.out.println(sql);
```

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

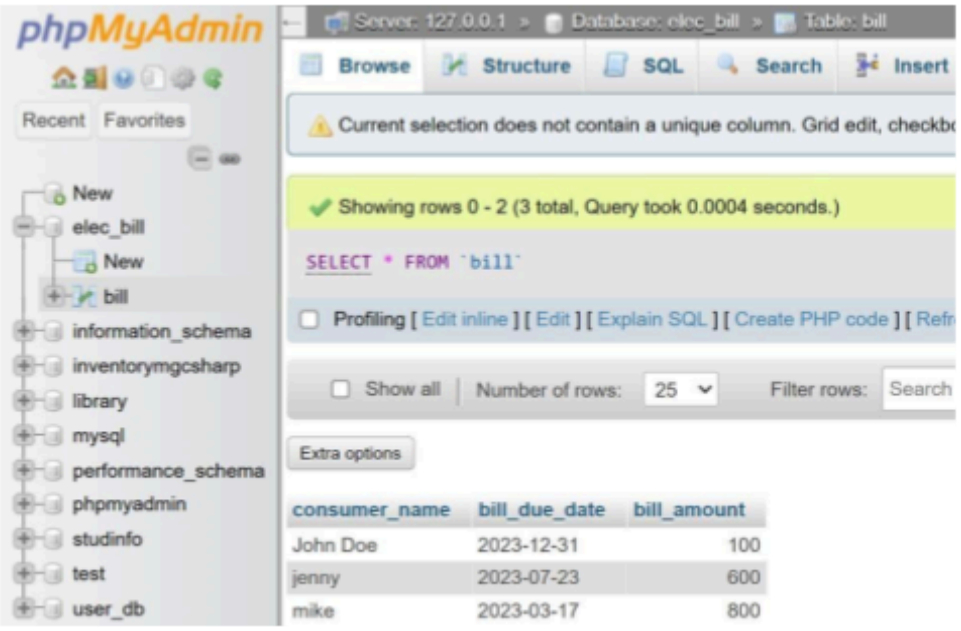
ElectricBillDBServiceSrv.java import
java.rmi.Naming; import
java.rmi.registry.LocateRegistry; public
class ElectricBillDBServiceSrv { public
static void main(String[] args) { try
{

```
        BillDBInf skeleton = new BillDBOperations();
        LocateRegistry.createRegistry(1901);
        Naming.rebind("rmi://localhost:1901/ROCforBillDB", skeleton);
        System.out.println("Server Registered.");
    } catch (Exception e1) {
e1.printStackTrace();
    }
  }
}
```

Output :

```
Console ×
ElectricBillDBServiceSrv [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-
Server Registered.
```

```
Console ×
ElectricBillDBServiceClient [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v202210
Select an Option
1: Retrieve Bill Information
2: Exit
Enter your Choice
1
consumer_name   bill_due_date   bill_amount

John Doe        2023-12-31      100
jenny   2023-07-23      600
mike    2023-03-17      800

Select an Option
1: Retrieve Bill Information
2: Exit
Enter your Choice
```



c) Using MySQL create Student database. Create table student (rollno, studnm, Book_author) and retrieve the Book information from Library database using Remote Object Communication concept. DBServiceClient.java import java.io.BufferedReader; import java.io.InputStreamReader; import java.rmi.Naming; public class DBServiceClient {

```
	public DBServiceClient() {
		super();
	}
	public static void main(String[] args) {
		String sql="", ch="";
		try {
			StudDBInf stub =
(StudDBInf)Naming.lookup("rmi://localhost:1900/ROCforStudDB");
			BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
			while(true) {
				System.out.println("Select an Option");
				System.out.println("1:Retrieve Student Information");
				System.out.println("2:Insert Student Informatin");
				System.out.println("3:Exit");
				System.out.println("Enter your Choice");
				ch = br.readLine();
				if (ch.equals("1")) {
		sql = "SELECT * FROM student";
		sql = stub.getData(sql);
				}
				else if(ch.equals("2")) {
					sql = "INSERT INTO student(rollno, studnm) VALUES
(10,'ss')";
					sql = stub.insertData(sql);
				}
```

```
                    else if(ch.equals("3")) {
                        System.out.println("Exited...");
                        System.exit(0);
                    }
                    else {
                        sql = "Please select valid Option";
                    }
                    System.out.println(sql);
                }
            }
            catch(Exception e)
            {
            e.printStackTrace();
            }
        }
}
```

DBServiceSrv.java import

java.rmi.Naming; import

java.rmi.registry.LocateRegistry; public

class DBServiceSrv {

```
        public DBServiceSrv() {
            // TODO Auto-generated constructor stub
            super();
        }

        public static void main(String[] args) {
            // TODO Auto-generated method stub
            try {
                StudDBInf skeleton = new StudDBOperations();
```

```
                LocateRegistry.createRegistry(1900);
                Naming.rebind("rmi://localhost:1900/ROCforStudDB", skeleton);
                System.out.println("Server Registered.");
            }
            catch (Exception e1) {
                e1.printStackTrace();
            }
        }
}
```

StudDBInf.java import

```
java.rmi.Remote; import
java.rmi.RemoteException;

public interface StudDBInf extends Remote {            public String
getData(String strQry) throws RemoteException;      public String
insertData(String strQry) throws RemoteException;
}
```

StudDBOperations.java import

```
java.rmi.RemoteException; import
java.rmi.server.UnicastRemoteObject; import
java.sql.Connection; import
java.sql.DriverManager; import
java.sql.ResultSet; import
java.sql.ResultSetMetaData; import
java.sql.Statement;
public class StudDBOperations extends UnicastRemoteObject implements StudDBInf {

        /**
```

```
 *
 */
private static final long serialVersionUID = 1L;
Connection con;
Statement stmt;
ResultSet rs;
ResultSetMetaData rsmd;
String colStr, resultStr;

public StudDBOperations() throws RemoteException {
	// TODO Auto-generated constructor stub
	super();
	con = null;
	stmt = null;
	rs = null;
	rsmd = null;
	colStr = "";
	resultStr = "";
}

public void setDBCon() {
	try {
		String URL="jdbc:mysql://localhost:3306/studinfo";
Class.forName("com.mysql.jdbc.Driver");
		con = DriverManager.getConnection(URL,"root","");
	}
	catch(Exception e) {
		e.printStackTrace();
	}
```

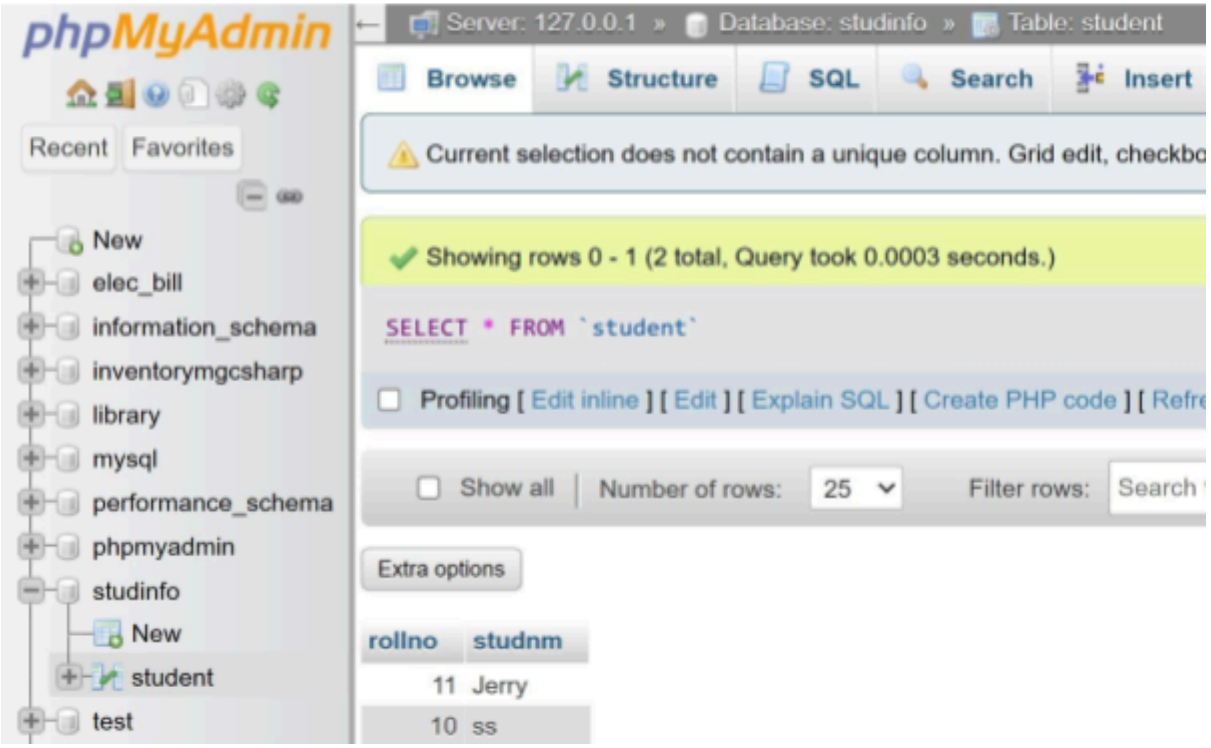
```
}

@Override
public String getData(String strQry) throws RemoteException {
	// TODO Auto-generated method stub
	try {
setDBCon();
		System.out.println("Server Registered.");
		//System.out.println(con.toString());
		stmt = con.createStatement();
	rs = stmt.executeQuery(strQry);
rsmd = rs.getMetaData();
		for(int i=1; i<=rsmd.getColumnCount();i++) {
		colStr = colStr + rsmd.getColumnName(i) + "\t";
		}

		while(rs.next()) {
			for(int i=1; i<=rsmd.getColumnCount(); i++) {
			resultStr = resultStr + rs.getString(i) + "\t";
			}
			resultStr = resultStr + "\n";
		}
	}
	catch(Exception e) {
		e.printStackTrace();
	}
	return colStr + "\n\n" +resultStr;
}
```

```
	@Override
	public String insertData(String strQry) throws RemoteException {
		// TODO Auto-generated method stub
		try {
	setDBCon();
			System.out.println("Server Registered.");
			//System.out.println(con.toString());
			stmt = con.createStatement();
			int recordInserted = stmt.executeUpdate(strQry);
			if(recordInserted != 0)
				resultStr = "Record inserted successfully.";
			else
				resultStr = "Record not inserted successfully.";
		}
		catch(Exception e) {
			e.printStackTrace();
		}
		return resultStr;
	}
}
```

Output:

```
Console ×
DBServiceSrv [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\jre\b
Server Registered.
```

```
Console ×
DBServiceClient [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\jr
Select an Option
1:Retrieve Student Information
2:Insert Student Informatin
3:Exit
Enter your Choice
2
Record inserted successfully.
Select an Option
1:Retrieve Student Information
2:Insert Student Informatin
3:Exit
Enter your Choice
1
rollno  studnm  rollno  studnm

Record inserted successfully.11 Jerry
10      ss

Select an Option
1:Retrieve Student Information
2:Insert Student Informatin
3:Exit
Enter your Choice
```



PRACTICAL NO. 5

Mutual Exclusion

LOB5: Understand the mechanism of mutual exclusion using token ring algorithm.
LO5: Implement program based on mutual exclusion using token ring algorithm.

Mutual Exclusion:

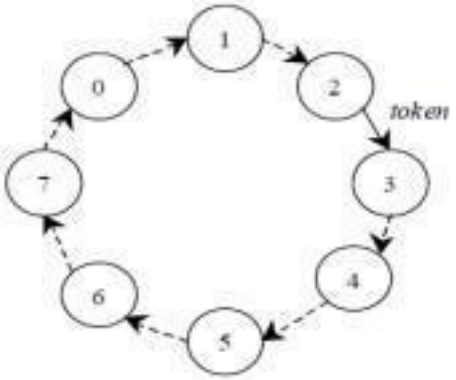
A mutual exclusion (mutex) is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource. Only one thread owns the mutex at a time, thus a mutex with a unique name is created when a program starts. When a thread holds a resource, it has to lock the mutex from other threads to prevent concurrent access of the resource. Upon releasing the resource, the thread unlocks the mutex.

Mutex comes into the picture when two threads work on the same data at the same time. It acts as a lock and is the most basic synchronization tool. When a thread tries to acquire a mutex, it gains the mutex if it is available, otherwise the thread is set to sleep condition. Mutual exclusion reduces latency and busy-waits using queuing and context switches. Mutex can be enforced at both the hardware and software levels.

Disabling interrupts for the smallest number of instructions is the best way to enforce mutex at the kernel level and prevent the corruption of shared data structures. If multiple processors share the same memory, a flag is set to enable and disable the resource acquisition based on availability. The busy-wait mechanism enforces mutex in the software areas.

Token Ring Algorithm for Mutual Exclusion:

For this algorithm, we assume that there is a group of processes with no inherent ordering of processes, but that some ordering can be imposed on the group. For example, we can identify each process by its machine address and process ID to obtain an ordering. Using this imposed ordering, a logical ring is constructed in software. Each process is assigned a position in the ring and each process must know who is next to it in the ring (figure)



the ring and each process must know who is next to it in the ring (Figure).

Figure: Token Ring example

1. The ring is initialized by giving a token to process 0. The token circulates around the ring (process n passes it to (n+1) mod ring_size.
2. When a process acquires the token, it checks to see if it is attempting to enter the critical section. If so, it enters and does its work. On exit, it passes the token to its neighbor.
3. If a process isn't interested in entering a critical section, it simply passes the token along.

Only one process has the token at a time and it must have the token to work on a critical section, so mutual exclusion is guaranteed. Order is also well-defined, so starvation cannot occur.

The biggest Drawback of this algorithm is that if a token is lost, it will have to be generated. Determining that a token is lost can be difficult.

Exercise:

a) Write a java program to implement mutual exclusion using Token ring algorithm.
Program:

Server.java import
java.net.DatagramPacket; import
java.net.DatagramSocket;
class TokenServer { public static void main(String[] args)
throws Exception {
// TODO Auto-generated method stub
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 bu[]=new byte[256];
DatagramSocket ds;
        DatagramPacket dp;   String str;
ds=new DatagramSocket(recPort);     dp=new
DatagramPacket(bu, bu.length);
        ds.receive(dp);
        ds.close();
        str=new String(dp.getData(),0,dp.getLength());
System.out.println("This message is"+str);
}
        public static void main(String[] args) {
                // TODO Auto-generated method stub
        }}
```

TokenClient1.java import

```
java.io.BufferedReader; import
java.io.InputStreamReader; import
java.net.DatagramPacket; import
java.net.DatagramSocket; import
java.net.InetAddress; public class
TokenClient1 {
        public static void main(String arg[]) throws Exception
        {
                InetAddress lclhost;
                BufferedReader br;
                String str="";
                TokenClient21 tkcl,tkser;
                boolean hasToken;
        boolean setSendData;
                while(true)
                {
```

```
                        lclhost=InetAddress.getLocalHost();
                tkcl = new TokenClient21(lclhost);
        tkser = new TokenClient21(lclhost);
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 am in else");
                                        tkcl.hasToken=false;
                        tkcl.sendData();
                                        tkcl.recData();
                                        System.out.println(" i am leaving else");
                                }
                        }else
                        {
                                System.out.println("ENTERING RECEIVING MODE...");
                                tkcl.recData();
                                tkcl.hasToken=true;
                        }}
                }}
class TokenClient21
{
        InetAddress lclhost;    int
sendport, recport;      boolean
hasToken =  true;       boolean
setSendData = false;    TokenClient21
tkcl,tkser;
        TokenClient21(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);
                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();
        }}
```

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();              tkcl.hasToken=false;
tkcl.sendData(); tkcl.recData();              hasToken=false;
         }
      } else
      {
         System.out.println("entering recieving mode");
tkcl.recData();
         hasToken=true;
      }
   }}}
```

Output:

```
Problems  @ Javadoc  Declaration  Console
TokenClient1 [Java Application] C:\Users\Dell\.p2\pool\plugins\
Do you want to enter the data -> YES/NO
YES
ready to send
Sending
Enter the Data
HELLO..!!!
```

```
Problems  @ Javadoc  Declaration  Console
TokenClient2 [Java Application] C:\Users\Dell\.p2\pool\plugins\org
entering if
entering recieving mode
```

Practical 6

Write a java program to access the files from your Google Drive account and read and write the file content from your program. DemoGdriveApplication.java package in.ac.famt.demoGdrive; import com.google.api.client.auth.oauth2.Credential; import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp; import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver; import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow; import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets; import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport; import com.google.api.client.http.AbstractInputStreamContent; import com.google.api.client.http.InputStreamContent; import com.google.api.client.http.javanet.NetHttpTransport; import com.google.api.client.json.JsonFactory; import com.google.api.client.json.jackson2.JacksonFactory; import com.google.api.client.util.store.FileDataStoreFactory; import com.google.api.services.drive.Drive; import com.google.api.services.drive.DriveScopes; import com.google.api.services.drive.model.File; import com.google.api.services.drive.model.FileList;

import java.io.FileInputStream; import

java.io.FileNotFoundException; import

java.io.IOException; import

java.io.InputStream; import

```
java.io.InputStreamReader; import

java.security.GeneralSecurityException; import

java.util.Collections; import java.util.List;



//@SpringBootApplication public class DemoGdriveApplication {              private static final

JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();


   // Directory to store user credentials for this application.

  //private static final java.io.File CREDENTIALS_FOLDER = new
java.io.File(System.getProperty("user.home"), "credentials");

  private static final java.io.File CREDENTIALS_FOLDER = new
java.io.File("C:\\Users\\MRUDH\\Documents\\demoGdrive (1)\\demoGdrive");


  // Global instance of the scopes required by this program.

   private static final List<String> SCOPES = Collections.singletonList(DriveScopes.DRIVE);

   //https://developers.google.com/resources/api-
libraries/documentation/drive/v2/java/latest/com/google/api/services/drive/DriveScopes.html


   private static Credential getCredentials(final NetHttpTransport HTTP_TRANSPORT) throws
IOException {

      java.io.File clientSecretFilePath = new
java.io.File("C:\\Users\\MRUDH\\Downloads\\DatteBayo.json");


      if (!clientSecretFilePath.exists()) {

         throw new FileNotFoundException("Please copy credentials.");

      }


      // Load client secrets.

      InputStream in = new FileInputStream(clientSecretFilePath);


      GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(JSON_FACTORY, new
InputStreamReader(in));
```

```
    // Build flow and trigger user authorization request.

    GoogleAuthorizationCodeFlow flow = new
GoogleAuthorizationCodeFlow.Builder(HTTP_TRANSPORT, JSON_FACTORY,

            clientSecrets, SCOPES).setDataStoreFactory(new
FileDataStoreFactory(CREDENTIALS_FOLDER))

                    .setAccessType("offline").build();

    //System.out.println("Flow info - " + flow.toString());


    return new AuthorizationCodeInstalledApp(flow, new LocalServerReceiver()).authorize("user");

  }

          public static void main(String[] args)throws IOException, GeneralSecurityException {

                        // 1: Build a new authorized API client service.        final NetHttpTransport

HTTP_TRANSPORT = GoogleNetHttpTransport.newTrustedTransport();


    // 2: Read client_secret.json file & create Credential object.

    Credential credential = getCredentials(HTTP_TRANSPORT);


    // 3: Create Google Drive Service.

    Drive service = new Drive.Builder(HTTP_TRANSPORT, JSON_FACTORY,
credential).setApplicationName("GDrive Access").build();

    System.out.println("----" + service.getApplicationName() + "----");


    // Print the names and IDs for up to 10 files.

   FileList result = service.files().list().setPageSize(2).setFields("nextPageToken, files(id,
name)").execute();

    String lastFile = "";

    List<File> files = result.getFiles();

if (files == null || files.isEmpty()) {

        System.out.println("No files found.");

    } else {

        System.out.println("Files:");
```

```
        for (File file : files) {
            System.out.printf("%s (%s)\n", file.getName(), file.getId());
            lastFile = file.getId();
        }
    }



    //Create Folder on Google Drive
File fileMetadata = new File();
    fileMetadata.setName("MyFolderUsingJava");
fileMetadata.setMimeType("application/vnd.google-apps.folder");
//fileMetadata.setParents(folderIdParent);

    File file = service.files().create(fileMetadata).setFields("id, name").execute();
    if(file != null)
        System.out.println("Folder Created..");





    //Creating a file on GDrive
    java.io.File uploadFileContent = new java.io.File("C:\\Users\\MRUDH\\Documents\\demoGdrive
(1)\\demoGdrive\\kimon.txt");
    String contentType = "text/plain";

    AbstractInputStreamContent uploadStreamContent = new
InputStreamContent(contentType,new FileInputStream(uploadFileContent));
    fileMetadata = new File();
fileMetadata.setName("MyGDriveJavaFile.txt");
    file = service.files().create(fileMetadata, uploadStreamContent).setFields("id, webContentLink,
webViewLink, parents").execute();

    if(file != null) {
```

```
        System.out.println("File Created..");

        System.out.println("WebContentLink: " + file.getWebContentLink() );
System.out.println("WebViewLink: " + file.getWebViewLink() );

    }


        }


}
```

Pom.xml

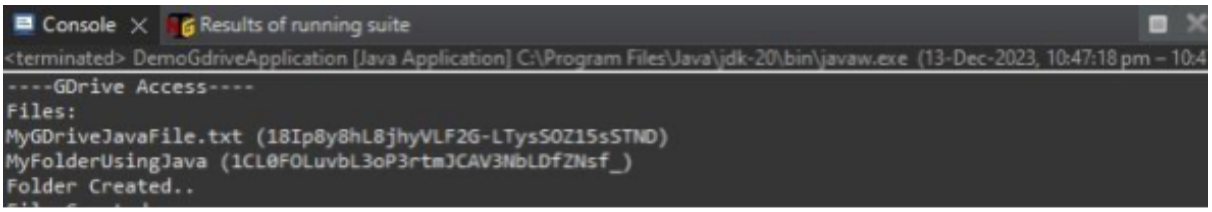
```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>

        <parent>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-starter-parent</artifactId>

                <version>3.2.1-SNAPSHOT</version>

                <relativePath/> <!-- lookup parent from repository -->

        </parent>

        <groupId>in.ac.famt</groupId>

        <artifactId>demoGdrive</artifactId>

        <version>0.0.1-SNAPSHOT</version>

        <name>demoGdrive</name>

        <description>Gdrive Access</description>

        <properties>

                <java.version>17</java.version>

        </properties>

        <dependencies>

                <dependency>
```

```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
        <groupId>com.google.apis</groupId>
         <artifactId>google-api-services-drive</artifactId>
         <version>v3-rev105-1.23.0</version>
        </dependency>

         <!-- https://mvnrepository.com/artifact/com.google.api-client/google-api-client -->
        <dependency>
         <groupId>com.google.api-client</groupId>
         <artifactId>google-api-client</artifactId>
         <version>1.23.0</version>
        </dependency>

<!-- https://mvnrepository.com/artifact/com.google.oauth-client/google-oauthclient-jetty -->
        <dependency>
         <groupId>com.google.oauth-client</groupId>
         <artifactId>google-oauth-client-jetty</artifactId>
         <version>1.23.0</version>
        </dependency>



    </dependencies>
```

```
<build>
	<plugins>
		<plugin>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-maven-plugin</artifactId>
		</plugin>
	</plugins>
</build>
<repositories>
	<repository>
		<id>spring-milestones</id>
		<name>Spring Milestones</name>
		<url>https://repo.spring.io/milestone</url>
		<snapshots>
			<enabled>false</enabled>
		</snapshots>
	</repository>
	<repository>
		<id>spring-snapshots</id>
		<name>Spring Snapshots</name>
		<url>https://repo.spring.io/snapshot</url>
		<releases>
			<enabled>false</enabled>
		</releases>
	</repository>
</repositories>
<pluginRepositories>
	<pluginRepository>
		<id>spring-milestones</id>
		<name>Spring Milestones</name>
```

```
                <url>https://repo.spring.io/milestone</url>
                <snapshots>
                        <enabled>false</enabled>
                </snapshots>
        </pluginRepository>
        <pluginRepository>
                <id>spring-snapshots</id>
                <name>Spring Snapshots</name>
                <url>https://repo.spring.io/snapshot</url>
                <releases>
                        <enabled>false</enabled>
                </releases>
        </pluginRepository>
</pluginRepositories>
```

</project> OUTPUT:



PRACTICAL NO. 7

App Development using Cloud Computing

LOB7: Understand use of various tools and techniques to develop cloud applications.
LO7: Implement basic cloud applications using various tools.

Google App Engine is a scalable runtime environment mostly devoted to executing Web applications. It is a PaaS solution that enables users to host their own applications on the same or similar infrastructure as Google Docs, Google Maps, and other popular Google services. These take advantage of the large computing infrastructure of Google to dynamically scale as the demand varies over time.

App Engine provides both a secure execution environment and a collection of services that simplify the development of scalable and high-performance Web applications. These services include inmemory caching, scalable data store, job queues, messaging, and corn tasks. Developers can build and test applications on their own machines using the App Engine software development kit (SDK), which replicates the production runtime environment and helps test and profile applications. Once development is complete, developers can easily migrate their application to App Engine, set quotas to contain the costs generated, and make the application available to the world. The languages currently supported are Python, Java, and Go.

Benefits of GAE

Ease of setup and use. GAE is fully managed, so users can write code without considering IT operations and back-end infrastructure. The built-in APIs enable users to build different types of applications. Access to application logs also facilitates debugging and monitoring in production.

Pay-per-use pricing. GAE's billing scheme only charges users daily for the resources they use. Users can monitor their resource usage and bills on a dashboard.

Scalability. Google App Engine automatically scales as workloads fluctuate, adding and removing application instances or application resources as needed.

Security. GAE supports the ability to specify a range of acceptable Internet Protocol (IP) addresses. Users can allow list specific networks and services and blocklist specific IP addresses.

GAE challenges

Lack of control. Although a managed infrastructure has advantages, if a problem occurs in the backend infrastructure, the user is dependent on Google to fix it.

Performance limits. CPU-intensive operations are slow and expensive to perform using GAE. This is because one physical server may be serving several separate, unrelated app engine users at once who need to share the CPU.

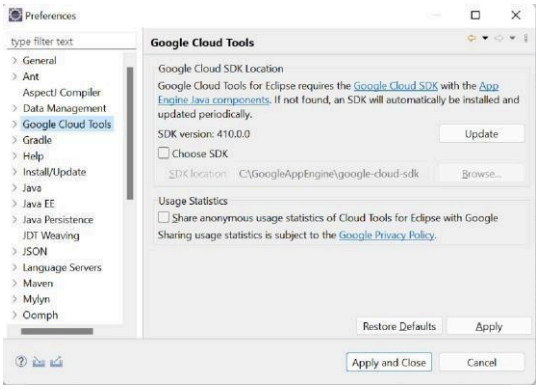
Limited access. Developers have limited, read-only access to the GAE filesystem.

Java limits. Java apps cannot create new threads and can only use a subset of the Java runtime environment standard edition classes.

Steps for Configuration of Google SDK with Eclipse –

1. Open the Eclipse IDE for installation of Google SDK plugin. Check "Java Build Path" for "JRE System Library" which must be jdk1.8.0 (jdk8). The higher versions of JDKs are not supported by Google Cloud Tools.
2. To add Google SDK solution; click Help-> Eclipse Marketplace and then search for "Google Cloud Tools for Eclipse 1.8.4" in the window. Click on Install button to install soution to the Eclipse IDE.

3. Check the Google Cloud Tools location by clicking the menu Window->Preferences-> Google Cloud Tools which will display following window –



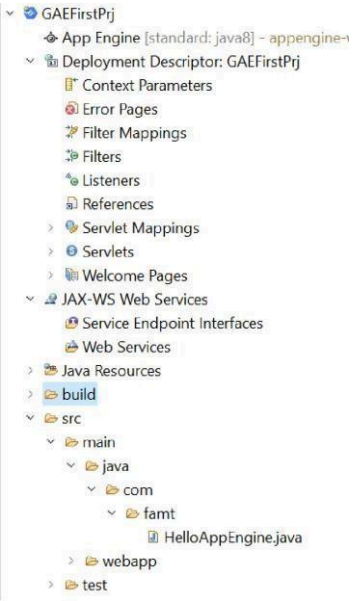
4. If the "Google Cloud Tools" are not properly configured then you may run GoogleCloudSDKInstaller.exe for installation of Google SDK cloud tools. (You may download it from – https://dl.google.com/dl/cloudsdk/channels/rapid/GoogleCloudSDKInstaller.exe
5. The screen looks like –

6. Default SDK install doesn't include some extra components like App Engine extensions for Java, which you can separately install using Google Cloud SDK shell.
7. To install java component use – "gcloud components install app-engine-java" command. After running this command, it will start installation in new command prompt and complete it with the help of Internet.

Creating a Google App Engine Standard Java Project –

Create a project using **File->New->Project->Google Cloud Platform->Google App Engine Standard Java Project** and click on next for assigning **project name and java package**. Click next and add **App Engine API** from next window and click Finish.

The project explorer shows following structure –



To run the Google App Engine project click Run->Run As->App Engine which will after compilation; opens the welcome page in the default browser and displays the Welcome message.

Department of MCA

Course: - MCAL32 Distributed System and Cloud Computing Lab

Exercise:

1. **Write a java program using Google App Engine for checking entered number is Odd or Even.**

Ans :

Program:

HelloAppEngine.java

```
package com.example; import
java.io.IOException; import
java.util.Scanner;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;
@WebServlet(
    name = "HelloAppEngine",
urlPatterns = {"/hello"}
)
public class HelloAppEngine extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
      throws IOException {
    response.setContentType("text/plain");
response.setCharacterEncoding("UTF-8");
response.setContentType("text/html");
response.getWriter().println("<html><body>");
        response.getWriter().println("<h2>Check Odd or Even</h2>");
response.getWriter().println("<form method='post'>");
response.getWriter().println("Enter a number: <input type='text'
name='number'><br>");
        response.getWriter().println("<input type='submit'
value='Check'>");
        response.getWriter().println("</form></body></html>");
    }
@Override
    public void doPost(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        String numberStr = request.getParameter("number");
try {
            int number = Integer.parseInt(numberStr);
```

```
            String result = (number % 2 == 0) ? "Even" : "Odd";
response.setContentType("text/html");
response.getWriter().println("<html><body>");
```

```
            response.getWriter().println("<h2>Result</h2>");
            response.getWriter().println("The number " + number + " is "
+ result + ".");
            response.getWriter().println("</body></html>");
        } catch (NumberFormatException e) {
            response.getWriter().println("Please enter a valid
number.");
        }

    }
    }
```

Output:

2. **Write a java program using Google App Engine for checking entered number is Prime or not.**

Ans:

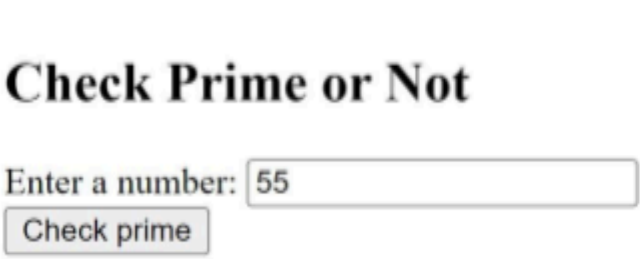
Program:

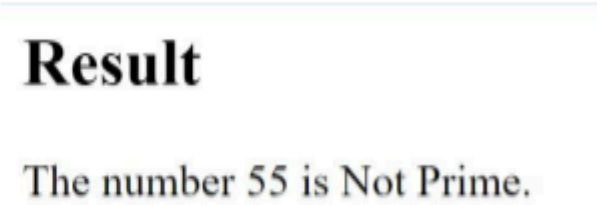
HelloAppEngine.java

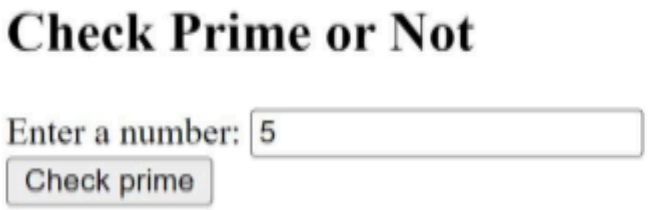
```
package com.example;
import java.io.IOException;
import java.util.Scanner;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet(
    name = "HelloAppEngine",
    urlPatterns = {"/hello"}
)
```

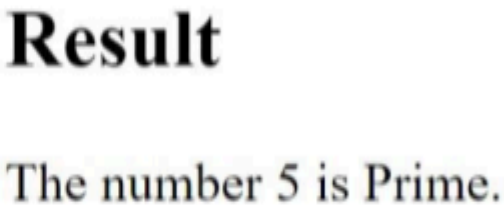
```
public class HelloAppEngine extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
      throws IOException {
response.setContentType("text/plain");
response.setCharacterEncoding("UTF-8");
response.setContentType("text/html");
response.getWriter().println("<html><body>");
response.getWriter().println("<h2>Check Prime or Not</h2>");
response.getWriter().println("<form method='post'>");
response.getWriter().println("Enter a number: <input type='text'
name='number'><br>");
             response.getWriter().println("<input type='submit'
value='Check prime'>");
             response.getWriter().println("</form></body></html>");
         }
         @Override
         public void doPost(HttpServletRequest request,
HttpServletResponse response) throws IOException {
             String numberStr = request.getParameter("number");
try {
                 int number = Integer.parseInt(numberStr);
boolean isPrime = checkPrime(number);
                 String result = (isPrime) ? "Prime" : "Not Prime";
response.setContentType("text/html");
response.getWriter().println("<html><body>");
response.getWriter().println("<h2>Result</h2>");
response.getWriter().println("The number " + number + " is " + result +
".");
                 response.getWriter().println("</body></html>");
             } catch (NumberFormatException e) {
             response.getWriter().println("Please enter a valid
number.");
             }
         }
             private boolean checkPrime(int number) {
                 if (number <= 1) {
                 }
                 for (int i = 2; i <= Math.sqrt(number); i++) {
                     if (number % i == 0) {
return false;
                     }
}                 return true;
             }  }
```

Output:


Check Prime or Not
Enter a number: 55
Check prime


Result
The number 55 is Not Prime.


Check Prime or Not
Enter a number: 5
Check prime


Result
The number 5 is Prime.

Q.3 Google App Engine

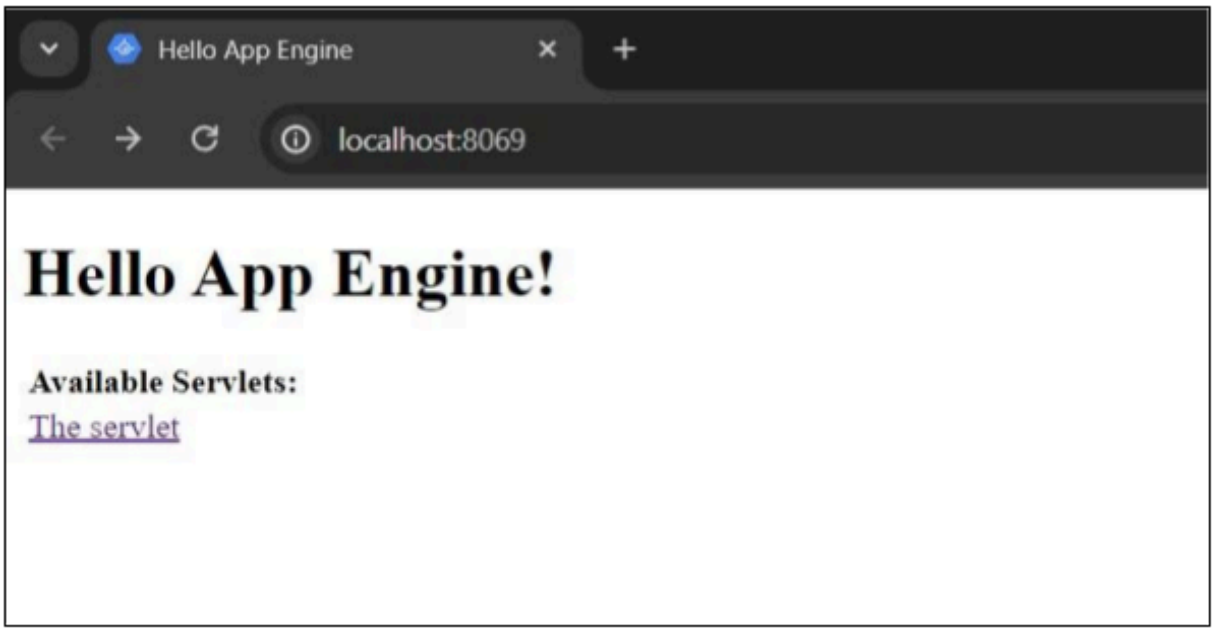
HelloAppEngine.java

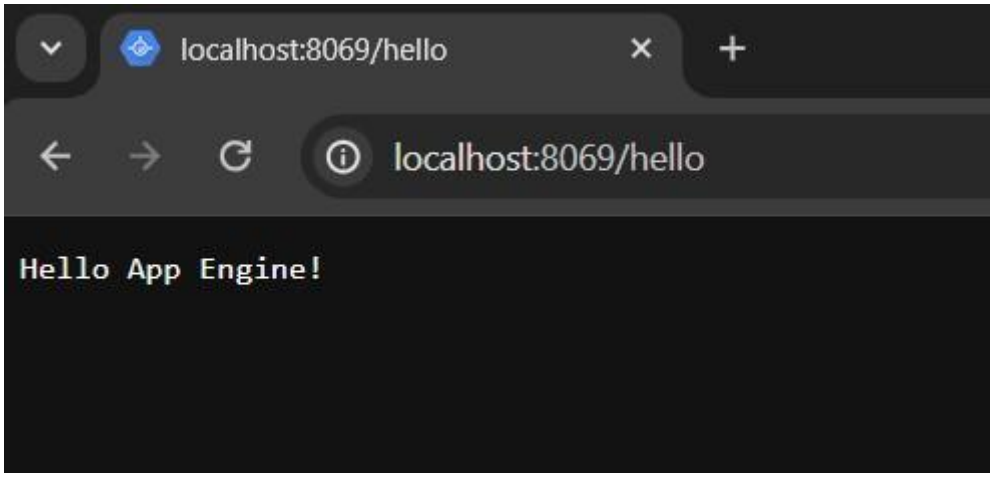
```
import java.io.IOException;
 import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet; import
javax.servlet.http.HttpServletRequest; import
javax.servlet.http.HttpServletResponse;

@WebServlet(
    name = "HelloAppEngine",
urlPatterns = {"/hello"}
)
public class HelloAppEngine extends HttpServlet {

  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse
response)
      throws IOException {
     response.setContentType("text/plain");
response.setCharacterEncoding("UTF-8");

    response.getWriter().print("Hello App Engine!\r\n");

  }
}
```

HelloAppEngineTest.java

```
import java.io.IOException;
 import
org.junit.Assert; import
org.junit.Test;

public class HelloAppEngineTest {

@Test
  public void test() throws IOException {
    MockHttpServletResponse response = new MockHttpServletResponse();
new HelloAppEngine().doGet(null, response);
    Assert.assertEquals("text/plain", response.getContentType());
    Assert.assertEquals("UTF-8", response.getCharacterEncoding());
    Assert.assertEquals("Hello App Engine!\r\n",
response.getWriterContent().toString());
  }
}
```

Output:


Hello App Engine
localhost:8069
Hello App Engine!
Available Servlets:
The servlet


localhost:8069/hello
localhost:8069/hello
Hello App Engine!

DSCC Prac cal 6

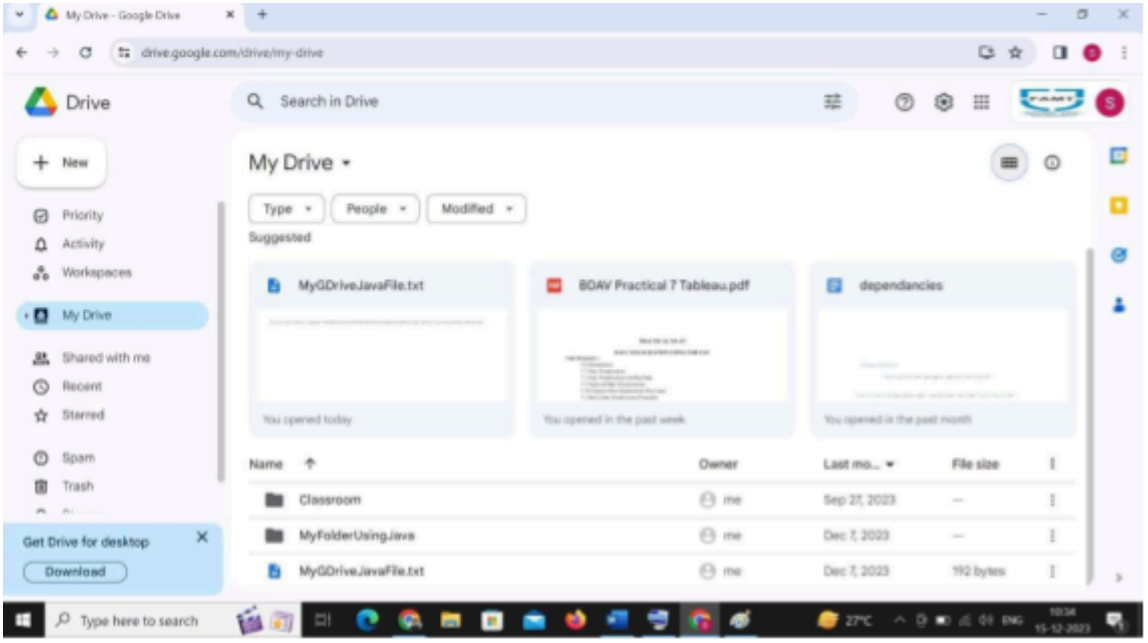
1. Write a java program to access the files from your Google drive account and read and write the file contents from your program.

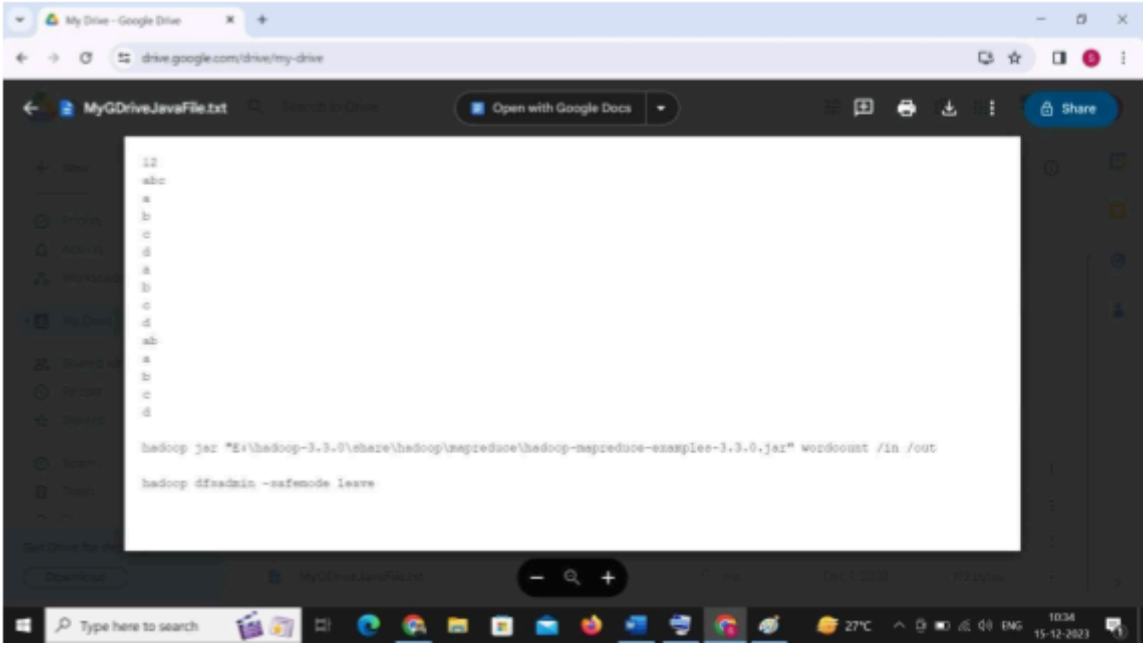
Code:

```
package in.ac.famt.prac cal6; import com.google.api.client.auth.oauth2.Creden al;
import
com.google.api.client.extensions.java6.auth.oauth2.Authoriza onCodeInstalledApp;
import com.google.api.client.extensions.je y.auth.oauth2.LocalServerReceiver; import
com.google.api.client.googleapis.auth.oauth2.GoogleAuthoriza onCodeFlow; import
com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets; import
com.google.api.client.googleapis.javanet.GoogleNetH pTransport; import
com.google.api.client.h p.AbstractInputStreamContent; import com.google.api.client.h
p.FileContent; import com.google.api.client.h p.H pResponse; import
com.google.api.client.h p.InputStreamContent; import com.google.api.client.h
p.javanet.NetH pTransport; import com.google.api.client.json.JsonFactory; import
com.google.api.client.json.jackson2.JacksonFactory; import com.google.api.client.u
l.store.FileDataStoreFactory; import com.google.api.services.drive.Drive; import
com.google.api.services.drive.DriveScopes; import
com.google.api.services.drive.model.File; import
com.google.api.services.drive.model.FileList; import java.io.*; import
java.nio.charset.StandardCharsets; import java.security.GeneralSecurityExcep on;
import java.u l.Collec ons; import java.u l.List;

public class Q1 {     private sta c final JsonFactory JSON_FACTORY =
JacksonFactory.getDefaultInstance();     private sta c final java.io.File
CREDENTIALS_FOLDER = new java.io.File("D:\\SYMCA\\Semester-III\\L2
DSCC\\DSCC Prac cals");     private sta c final List<String> SCOPES = Collec
ons.singletonList(DriveScopes.DRIVE);     private sta c Creden al getCreden als(final
NetH pTransport HTTP_TRANSPORT) throws IOExcep on {         java.io.File
clientSecretFilePath = new java.io.File("D:\\SYMCA\\Semester-III\\L2
DSCC\\DSCC Prac cals\\creden als.json");         if
(!clientSecretFilePath.exists()) {             throw new
FileNotFoundExcep on("Please copy creden als.");
        }
        InputStream in = new FileInputStream(clientSecretFilePath);
        GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(JSON_FACTORY,
new InputStreamReader(in));
        GoogleAuthoriza onCodeFlow flow = new
```

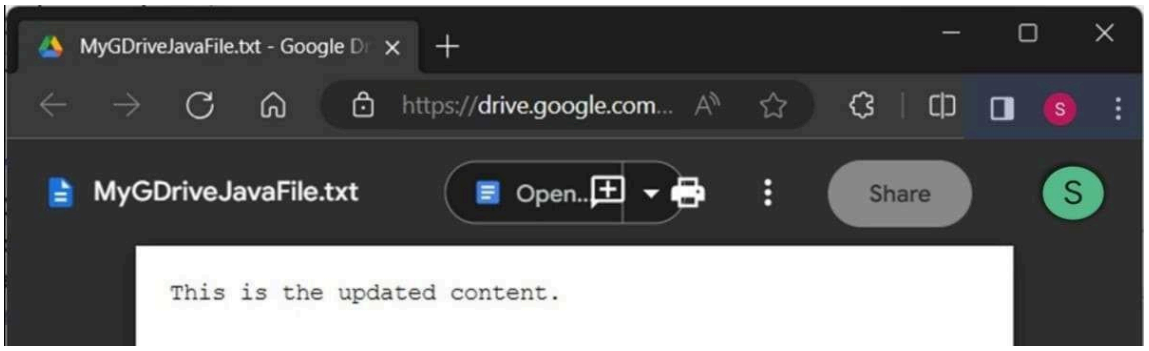
```
GoogleAuthoriza onCodeFlow.Builder(HTTP_TRANSPORT,
JSON_FACTORY, clientSecrets,
SCOPES).setDataStoreFactory(new
FileDataStoreFactory(CREDENTIALS_FOLDER))
        .setAccessType("offline").build();
    return new Authoriza onCodeInstalledApp(flow, new
LocalServerReceiver()).authorize("user");
  }
  public sta c void main(String[] args) throws IOExcep on, GeneralSecurityExcep on {
final NetH pTransport HTTP_TRANSPORT =
GoogleNetH pTransport.newTrustedTransport();
    Creden al creden al = getCreden als(HTTP_TRANSPORT);
    Drive service = new Drive.Builder(HTTP_TRANSPORT,
JSON_FACTORY, creden al).setApplica onName("GDrive
Access").build();
    // Example: Reading file contents
    String fileId = "1hBt_BZ1z_49uUJeYQ9aYwhmomIAqlA0J";  // Replace with
actual file ID    ByteArrayOutputStream outputStream = new
ByteArrayOutputStream();
service.files().get(fileId).executeMediaAndDownloadTo(outputStream);
    String fileContent = new String(outputStream.toByteArray(),
StandardCharsets.UTF_8);
    System.out.println("File Content:\n\n" + fileContent);
    // Example: Wri ng file contents
    String updatedContent = "This is the updated content.";
    InputStreamContent mediaContent = new InputStreamContent("text/plain", new
ByteArrayInputStream(updatedContent.getBytes(StandardCharsets.UTF_8)));
File fileMetadata = new File();    fileMetadata.setName("MyGDriveJavaFile.txt");
// Replace with the actual file name    service.files().update(fileId, fileMetadata,
mediaContent).execute();    System.out.println("File Content updated
successfully.");
  }
}
```

Output:






```
Console
<terminated> Q1 [Java Application] C:\Users\HP\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20221004-1257\j
File Content:

This is the updated content.
File Content updated successfully.
```


MyGDriveJavaFile.txt - Google Dr
https://drive.google.com...
MyGDriveJavaFile.txt
Open..
Share
S
This is the updated content.

Practical No 5

Write a java program to implement mutual exclusion using token ring algorithm.

Server.java package

```
MyPack;  import
java.net.*; import
java.io.*; class
TokenServer {

                                        public static void main(String[] args) throws Exception {
                        while(true)
                         {
                         Server sr=new Server();
                sr.recPort(8000);
sr.recData();

                         }
                }
}
public class Server {
        boolean hasToken=false;
        boolean sendData=false;
        int recport;

                void recPort(int recport) {
this.recport=recport;
                }
  void recData() throws Exception{     byte bu[]=new byte[256];
        DatagramSocket ds;
                                DatagramPacket dp;
```

```
                                        String str;


                        ds=new DatagramSocket(recport);

dp=new DatagramPacket(bu,bu.length);
```

```
                                        ds.receive(dp);
 ds.close();
                                                        str=new String(dp.getData(),0,dp.getLength());
                                                System.out.println("This message is "+str);
                        }
}
```

TokenClient1.java package

```
MyPack; import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

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(9004);
tkcl.setRecPort(8002);

InetAddress.getLocalHost(); tSendPort(9000);
{
if(tkcl.hasToken

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 am in else");
        tkcl.hasToken=false;
tkcl.sendData();        tkcl.recData();
                                System.out.println(" i am leaving else");
}
```

```
                    }
                    else
                    {

                        System.out.println("ENTERING RECEIVING MODE...");
            tkcl.recData(); tkcl.hasToken=true;
                    }
            }

        }

}
```

```
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);
            ds.send(dp); ds.close(); setSendData = false;  hasToken =
            false;
    }
            Data() 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

```
package MyPack;
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 TokenClient 21(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();              tkcl.hasToken=false; tkcl.sendData();
tkcl.recData();              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.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;
        }
    }
}
```

Output:

```
Console  Results of running suite
TokenClient1 [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (01-Dec-2023, 4:52:01 pm) [pid: 11488]
Do you want to enter the data -> YES/NO
Yes
ready to send
Sending
Enter the Data
Hii You Adorable!!
how  Sending
Do you want to enter the data -> YES/NO
```

```
Console  Results of running suite
<terminated> TokenClient2 [Java Application] C:\Program Files\Java\jdk-20\bin\javaw.exe (0
entering if
entering recieving mode
```