Assignment 1

Title: Socket Programming

Code:

MyClient.java

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

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

      //The socket object takes ip and port number of the server which client wants to connect
      Socket s = new Socket("127.0.0.1",5555);
      System.out.println("Connected to Server, Please type your message and hit Enter to send");

      //Reading input from KeyBoard
      BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

      //OutputStream object to write to Server
      OutputStream ostream = s.getOutputStream();

      //PrintWriter object to send the data to the outputstream
      PrintWriter pw = new PrintWriter(ostream, true);

      //InputStream objects to recieve from Server
      InputStream istream = s.getInputStream();

      //Reading receieved message from Server
      BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));
```

```
//Client Message and Server Message objects
String clientmessage = "";
String servermessage = "";

while(true)
{
   //Input Message to be sent to Server
   System.out.print("Client: ");
   clientmessage = br.readLine();

   //print writer object sending the message to the socket through outputstream
   pw.println(clientmessage);

   //if the message is bye end the communication here
   if(clientmessage.equals("bye"))
   {
      break;
   }

   //Read the inputstream of the server from the socket
   servermessage = recieve.readLine();
   System.out.println("Server: "+servermessage);

   //if the message is bye end the communication here
   if(servermessage.equals("bye"))
   {
      break;
   }
}
```

```
        //closing all the streams and sockets
        s.close();
        istream.close();
        ostream.close();

        System.out.println("Connection Terminated");
    }
}
```

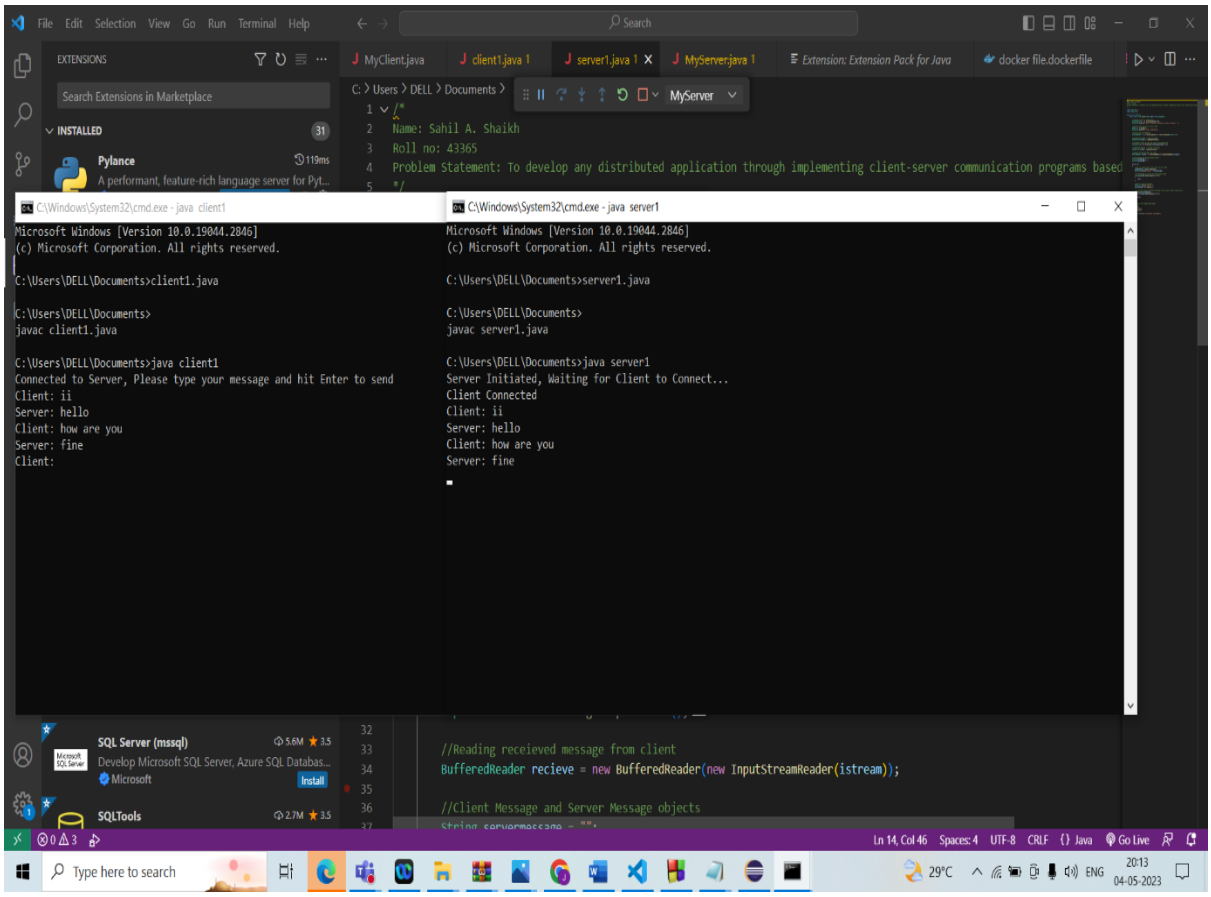
MyServer.java

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

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

        //Creating a port for communication
        ServerSocket ss = new ServerSocket(5555);
        System.out.println("Server Initiated, Waiting for Client to Connect...");

        //Binding Client and Server on port 5555
        Socket s = ss.accept();
        System.out.println("Client Connected");

        //Reading input from KeyBoard
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        //OutputStream object to write to clients
```

```
OutputStream ostream = s.getOutputStream();

//PrintWriter object to send the data to the outputstream
PrintWriter pw = new PrintWriter(ostream,true);

//InputStream objects to recieve from Client
InputStream istream = s.getInputStream();

//Reading receieved message from client
BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));

//Client Message and Server Message objects
String servermessage = "";
String clientmessage = "";

while(true)
{
   //Read the inputstream of the client from the socket
   clientmessage = recieve.readLine();
   System.out.println("Client: "+clientmessage);

   //if the message is bye end the communication here
   if(clientmessage.equals("bye"))
   {
      break;
   }

   //Server writing its message
   System.out.print("Server: ");
   servermessage = br.readLine();
```

```
            //print writer object sending the message to the socket through outputstream
            pw.println(servermessage);
            if(servermessage.equals("bye"))
            {
                break;
            }
        }

        //closing all the streams and sockets
        s.close();
        ss.close();
        istream.close();
        ostream.close();

        System.out.println("Connection Terminated");
    }
}
```

Output:


C:\Windows\System32\cmd.exe - java client1
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.
C:\Users\DELL\Documents>client1.java
C:\Users\DELL\Documents>
javac client1.java
C:\Users\DELL\Documents>java client1
Connected to Server, Please type your message and hit Enter to send
Client: ii
Server: hello
Client: how are you
Server: fine
Client:
C:\Windows\System32\cmd.exe - java server1
Microsoft Windows [Version 10.0.19044.2846]
(c) Microsoft Corporation. All rights reserved.
C:\Users\DELL\Documents>server1.java
C:\Users\DELL\Documents>
javac server1.java
C:\Users\DELL\Documents>java server1
Server Initiated, Waiting for Client to Connect...
Client Connected
Client: ii
Server: hello
Client: how are you
Server: fine
//Reading receieved message from client
BufferedReader recieve = new BufferedReader(new InputStreamReader(istream));
//Client Message and Server Message objects

Assignment 2

Title: Remote Method Invocation

Code:

Server.java

```
import java.rmi.*;
import java.net.*;

public class Server {
   public static void main(String[] args) {
      try {
         Servant s = new Servant();
         Naming.rebind("Server", s);
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Servant.java

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

public class Servant extends UnicastRemoteObject implements ServerInterface {
   protected Servant() throws RemoteException {
      super();
   }
```

```
    @Override
    public String concat(String a, String b) throws RemoteException {
        return a + b;
    }
}
```

ServerInterface.java

```
import java.rmi.*;

public interface ServerInterface extends Remote {
    String concat(String a, String b) throws RemoteException;
}
```

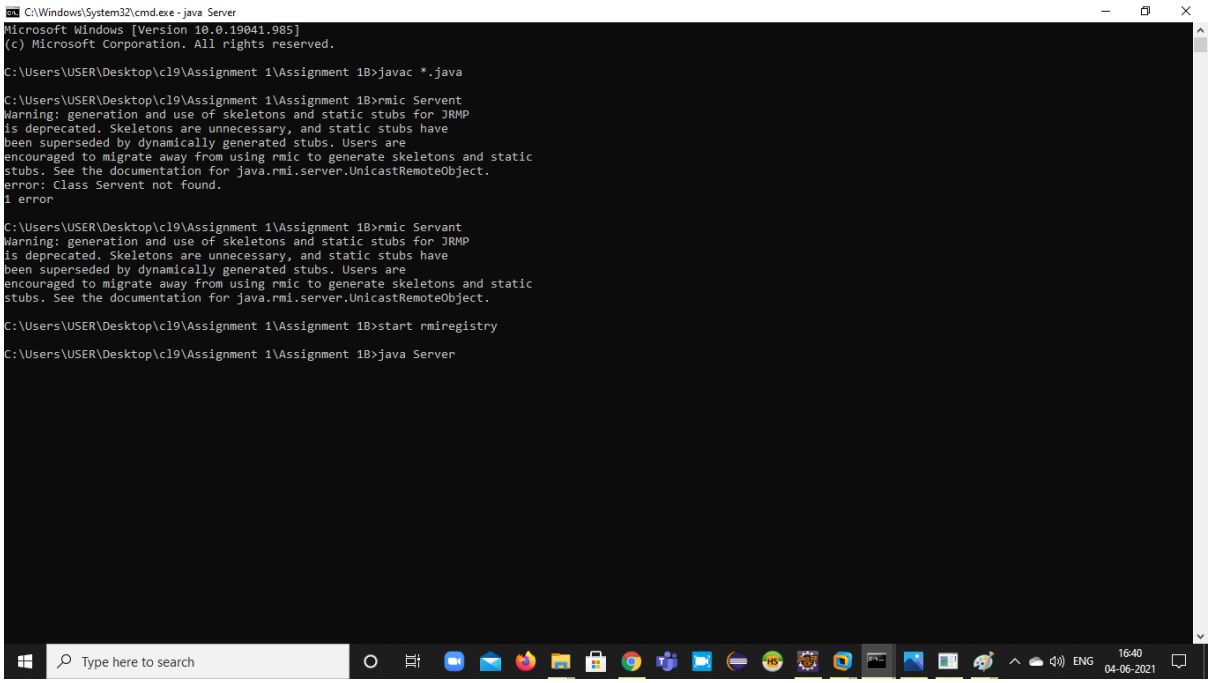
Client.java

```
import java.rmi.*;
import java.util.Scanner;

public class Client {
    public static void main(String args[]) {
        try {
            Scanner s = new Scanner(System.in);
            System.out.println("Enter the Server address : ");
            String server = s.nextLine();
            ServerInterface si = (ServerInterface) Naming.lookup("rmi://" + server + "/Server");
            System.out.println("Enter first string : ");
```

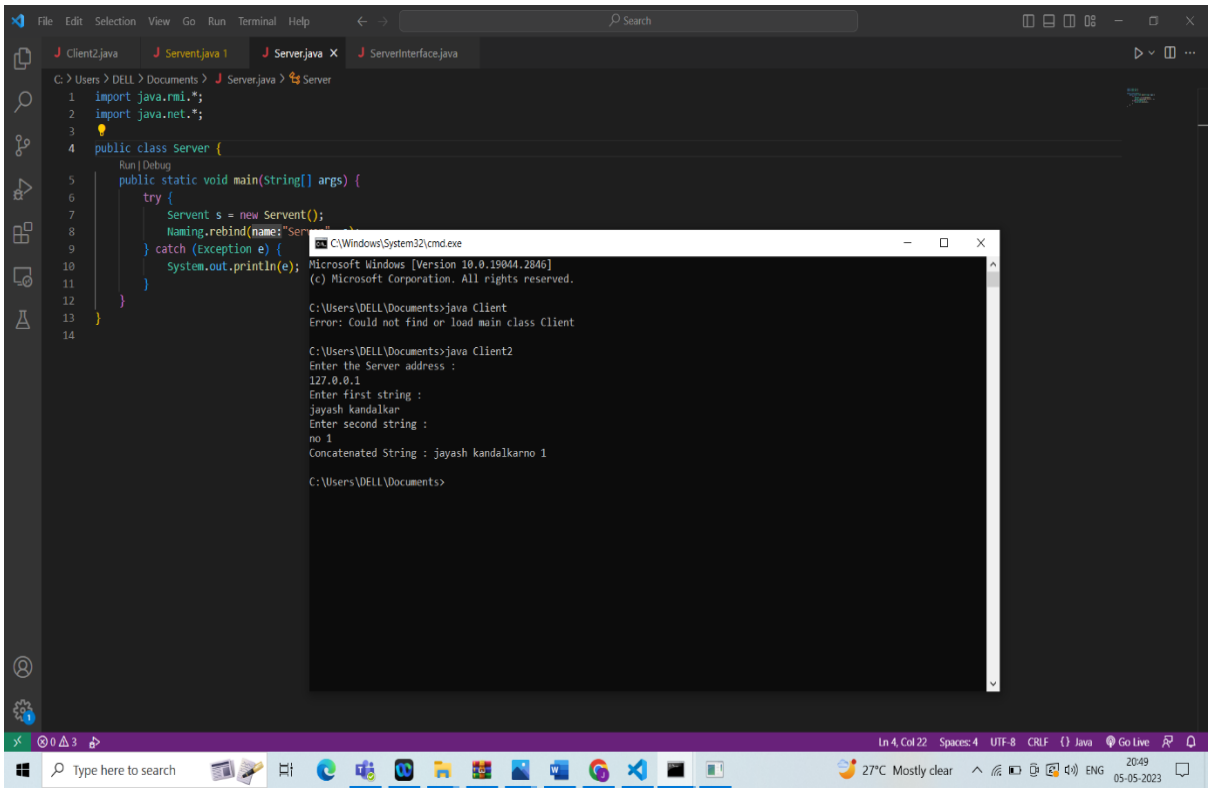
```
            String first = s.nextLine();
            System.out.println("Enter second string : ");
            String second = s.nextLine();
            System.out.println("Concatenated String : " + si.concat(first, second));
            s.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

OUTPUT:

Java Server is Running:



String Concanate using Multithreaded client server model:



Assignment 3

Title: Common Object Request Broker Architecture (CORBA)

Code:

StartServer.java

```
import Calculator.Calc;
import Calculator.CalcHelper;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import org.omg.PortableServer.POA;

public class StartServer {

   public static void main(String args[]) {
      try{
         // create and initialize the ORB
         ORB orb = ORB.init(args, null);

         // get reference to rootpoa & activate the POAManager
         POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
         rootpoa.the_POAManager().activate();

         // create servant and register it with the ORB
         CalcObject calcObj = new CalcObject();
         calcObj.setORB(orb);
```

```
        // get object reference from the servant
        org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcObj);
        Calc href = CalcHelper.narrow(ref);

        // get the root naming context
        // NameService invokes the name service
        org.omg.CORBA.Object nsRef =  orb.resolve_initial_references("NameService");

        // Use NamingContextExt which is part of the Interoperable
        // Naming Service (INS) specification.
        NamingContextExt ncRef = NamingContextExtHelper.narrow(nsRef);

        // bind the Object Reference in Naming
        NameComponent path[] = ncRef.to_name("Calculator");
        ncRef.rebind(path, href);

        System.out.println("CalculatorServer is listening...");

        // wait for invocations from clients
        orb.run();
        System.out.println("I am out");
      }
      catch (Exception e) {
        System.err.println("Server Error: " + e.getMessage());
        e.printStackTrace(System.out);
      }
    }
}
```

StartClient.java

```
import Calculator.*;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import java.util.*;

public class StartClient {
   private static Calc calcObj;

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      try {
         // create and initialize the ORB
      ORB orb = ORB.init(args, null);

         // get the root naming context
      org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");

         // Use NamingContextExt instead of NamingContext. This is
         // part of the Interoperable naming Service.
      NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

         // resolve the Object Reference in Naming
      calcObj = (Calc) CalcHelper.narrow(ncRef.resolve_str("Calculator"));

         while(true) {
            // asking for input and read it
            System.out.println("----------------------------------------");
```

```
            System.out.println("Enter the parameters in this format
[operator][sp][operand1][sp][operand2]."
                    + "\nFor example: + 1 2");
            Scanner c=new Scanner(System.in);
        String input = c.nextLine();


            // if the command is exit, request the server to shutdown
            if (input.toLowerCase().equals("exit")) {
               calcObj.exit();
               break;
            }


            // test the input
            String[] inputParams = input.split(" ");
            if (inputParams.length != 3) {
               System.out.println("Client Exception: Wrong number of parameters. Try again...");
               continue;
            }
            int operatorCode;
            int operand1;
            int operand2;


            // set calculation type
            if (inputParams[0].equals("+")) {
               operatorCode = 1;
            }
            else if (inputParams[0].equals("-")) {
               operatorCode = 2;
            }
            else if (inputParams[0].equals("*")) {
               operatorCode = 3;
```

```
        }
        else if (inputParams[0].equals("/")) {
            operatorCode = 4;
        }
        else {
            System.out.println("Client Exception: Un-recognized operation code. Try again...");
            continue;
        }

        // test input operands are integers
        try {
            operand1 = Integer.parseInt(inputParams[1]);
            operand2 = Integer.parseInt(inputParams[2]);
        }
        catch (NumberFormatException e) {
            System.out.println("Client Exception: Wrong number format. Try again...");
            continue;
        }

        // check if it is divided by zero
        if (operatorCode == 4 && operand2 == 0) {
            System.out.println("Client Exception: Can't be divided by zero. Try again...");
            continue;
        }

        // do the calculation and return result
    int result = calcObj.calculate(operatorCode, operand1, operand2);
        String resultDisplay = "";
        if (result == Integer.MAX_VALUE) {
            resultDisplay = "There might be an Integer Overflow. Please try again...";
        }
```

```
            else if (result == Integer.MIN_VALUE) {
               resultDisplay = "There might be an Integer Underflow. Please try again...";
            }
            else {
               resultDisplay = String.valueOf(result);
            }
      System.out.println("The result is: " + resultDisplay);
         }
      }
      catch (Exception e) {
         System.out.println("Client exception: " + e.getMessage());
         e.printStackTrace();
      }
   }
}
```

Calculator.idl

```
module Calculator {
   interface Calc {
      long calculate (in long opcode, in long op1, in long op2);
      oneway void exit();
   };
};
```

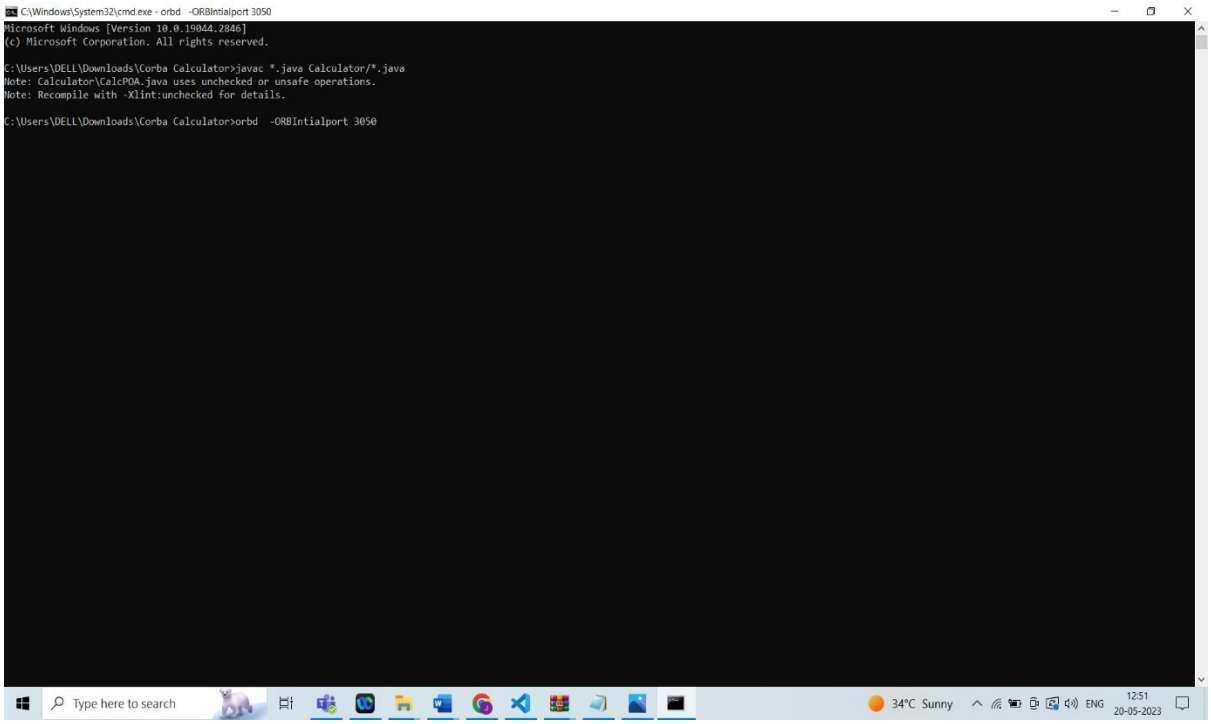
Calcobject.java

```
public class CalcObject extends CalcPOA{
   private ORB orb;

   public void setORB(ORB orb) {
      this.orb = orb;
```

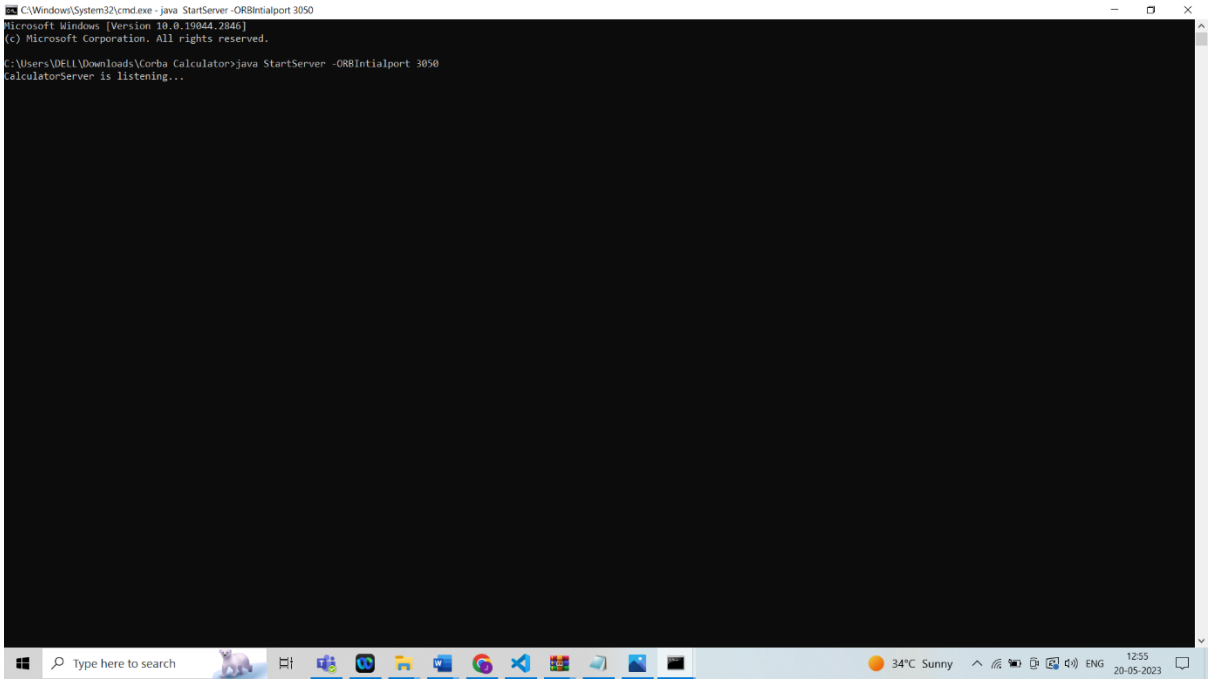
```
}

/** Calculate
   * @param type the type of the operation, 1 -> +, 2 -> -, 3 -> *, 4 -> /
* @param a first number
* @param b second number
* @return calculation result
*/
@Override
public int calculate(int type, int a, int b) {
   long result;

   if (type == 1) {
      result = (long) a + b;
   }
   else if (type == 2) {
      result = (long) a - b;
   }
   else if (type == 3) {
      result = (long) a * b;
   }
   else{
      result = (long) a / b;
   }

   if (result >= Integer.MAX_VALUE) {
      return Integer.MAX_VALUE;
   }
   else if (result <= Integer.MIN_VALUE) {
      return Integer.MIN_VALUE;
   }
```

```
        else {
            return (int) result;
        }
    }

    @Override
    public void exit() {
        orb.shutdown(false);
    }
}
```

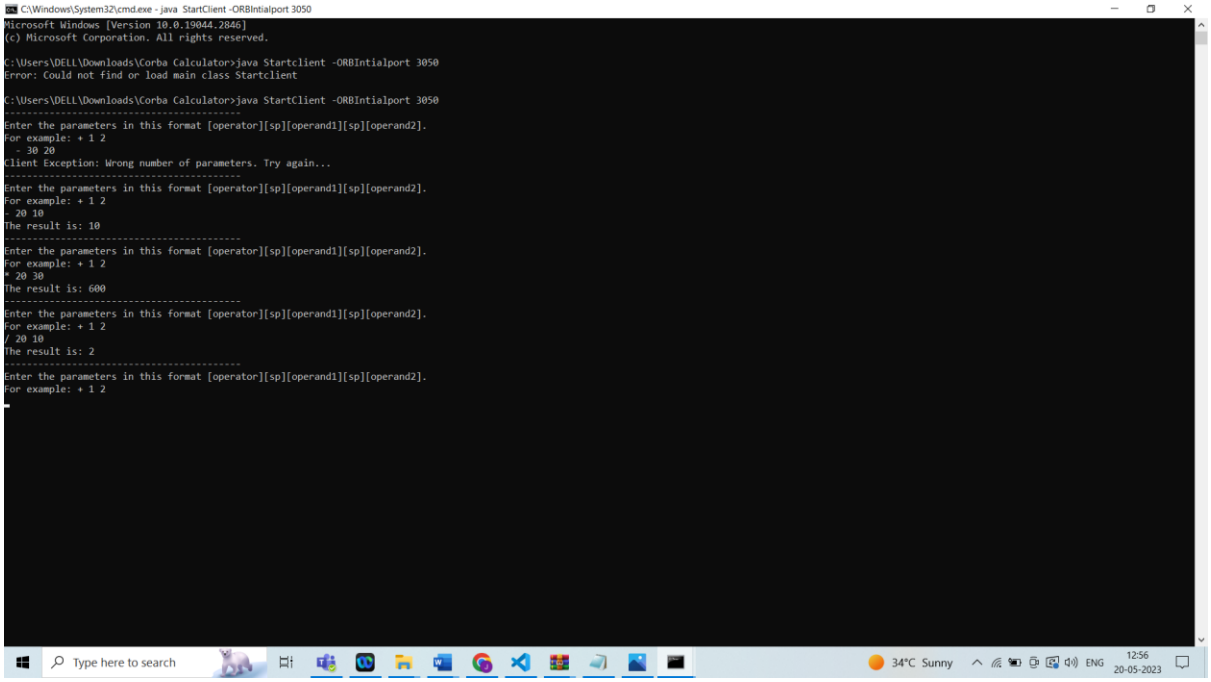
Output:



Server is Running:



Calculator output:



Assignment 4

Title: Message Passing Interface (MPI)

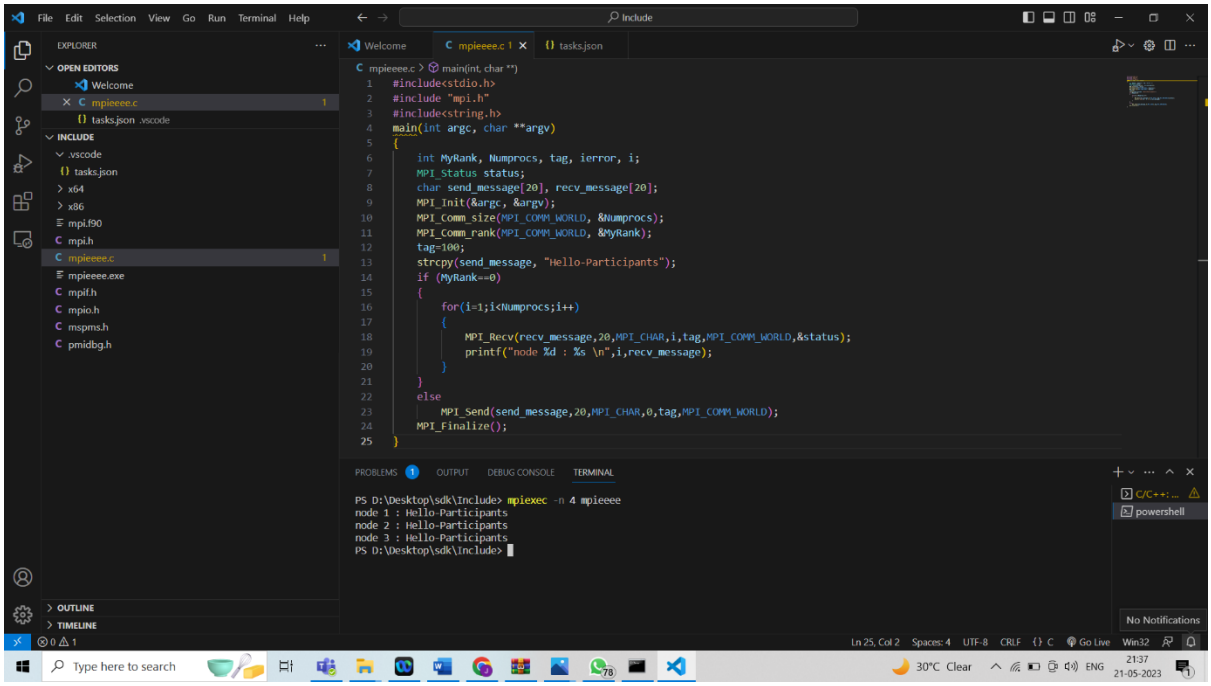
Code:

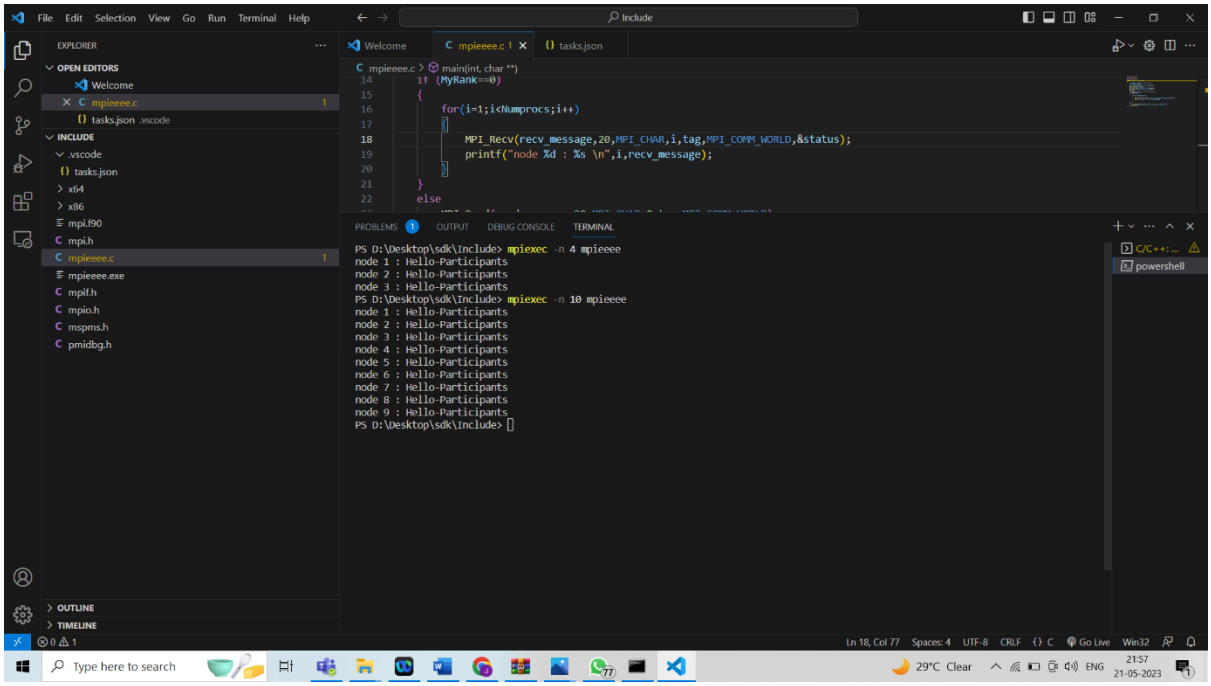
Hello world program

```
#include<stdio.h>
#include "mpi.h"
#include<string.h>
main(int argc, char **argv)
{
   int MyRank, Numprocs, tag, ierror, i;
   MPI_Status status;
   char send_message[20], recv_message[20];
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &Numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD, &MyRank);
   tag=100;
   strcpy(send_message, "Hello-Participants");
   if (MyRank==0)
   {
      for(i=1;i<Numprocs;i++)
      {
         MPI_Recv(recv_message,20,MPI_CHAR,i,tag,MPI_COMM_WORLD,&status);
         printf("node %d : %s \n",i,recv_message);
      }
   }
   else
      MPI_Send(send_message,20,MPI_CHAR,0,tag,MPI_COMM_WORLD);
   MPI_Finalize();
```

}

Output:


mpieeee.c
#include<stdio.h>
#include "mpi.h"
#include<string.h>
main(int argc, char **argv)
{
int MyRank, Numprocs, tag, ierror, i;
MPI_Status status;
char send_message[20], recv_message[20];
MPI_Init(&argc, &argv);
MPI_Comm_size(MPI_COMM_WORLD, &Numprocs);
MPI_Comm_rank(MPI_COMM_WORLD, &MyRank);
tag=100;
strcpy(send_message, "Hello-Participants");
if (MyRank==0)
{
for(i=1;i<Numprocs;i++)
{
MPI_Recv(recv_message,20,MPI_CHAR,i,tag,MPI_COMM_WORLD,&status);
printf("node %d : %s \n",i,recv_message);
}
}
else
MPI_Send(send_message,20,MPI_CHAR,0,tag,MPI_COMM_WORLD);
MPI_Finalize();
}
PS D:\Desktop\sdk\Include> mpiexec -n 4 mpieeee
node 1 : Hello-Participants
node 2 : Hello-Participants
node 3 : Hello-Participants
PS D:\Desktop\sdk\Include>


PS D:\Desktop\sdk\Include> mpiexec -n 4 mpieeee
node 1 : Hello-Participants
node 2 : Hello-Participants
node 3 : Hello-Participants
PS D:\Desktop\sdk\Include> mpiexec -n 10 mpieeee
node 1 : Hello-Participants
node 2 : Hello-Participants
node 3 : Hello-Participants
node 4 : Hello-Participants
node 5 : Hello-Participants
node 6 : Hello-Participants
node 7 : Hello-Participants
node 8 : Hello-Participants
node 9 : Hello-Participants
PS D:\Desktop\sdk\Include>

ASSIGNMENT NO. 5

Title: Clock Synchronization

Code:

Server.java

```
# Python3 program imitating a clock server

from functools import reduce
from dateutil import parser
import threading
import datetime
import socket
import time

# datastructure used to store client address and clock data
client_data = {}

''' nested thread function used to receive
    clock time from a connected client '''
def startReceivingClockTime(connector, address):

    while True:
        # receive clock time
        clock_time_string = connector.recv(1024).decode()
        clock_time = parser.parse(clock_time_string)
        clock_time_diff = datetime.datetime.now() - \
                                    clock_time

        client_data[address] = {
```

```
                "clock_time"     : clock_time,
                "time_difference" : clock_time_diff,
                "connector"  : connector
                }

        print("Client Data updated with: "+ str(address),
                                    end = "\n\n")
        time.sleep(5)

''' master thread function used to open portal for
    accepting clients over given port '''
def startConnecting(master_server):

    # fetch clock time at slaves / clients
    while True:
        # accepting a client / slave clock client
        master_slave_connector, addr = master_server.accept()
        slave_address = str(addr[0]) + ":" + str(addr[1])

        print(slave_address + " got connected successfully")

        current_thread = threading.Thread(
                    target = startReceivingClockTime,
                    args = (master_slave_connector,
                                slave_address, ))
        current_thread.start()

# subroutine function used to fetch average clock difference
def getAverageClockDiff():

    current_client_data = client_data.copy()
```

```
    time_difference_list = list(client['time_difference']
                                for client_addr, client
                                   in client_data.items())


    sum_of_clock_difference = sum(time_difference_list, \
                                datetime.timedelta(0, 0))

    average_clock_difference = sum_of_clock_difference \
                                    / len(client_data)

    return average_clock_difference

''' master sync thread function used to generate
    cycles of clock synchronization in the network '''
def synchronizeAllClocks():

    while True:

        print("New synchronization cycle started.")
        print("Number of clients to be synchronized: " + \
                                str(len(client_data)))

        if len(client_data) > 0:

            average_clock_difference = getAverageClockDiff()

            for client_addr, client in client_data.items():
                try:
                    synchronized_time = \
```

```
                datetime.datetime.now() + \
                        average_clock_difference

            client['connector'].send(str(
                synchronized_time).encode())

        except Exception as e:
            print("Something went wrong while " + \
                "sending synchronized time " + \
                "through " + str(client_addr))

    else :
        print("No client data." + \
                " Synchronization not applicable.")

    print("\n\n")

    time.sleep(5)

# function used to initiate the Clock Server / Master Node
def initiateClockServer(port = 8080):

    master_server = socket.socket()
    master_server.setsockopt(socket.SOL_SOCKET,
                        socket.SO_REUSEADDR, 1)

    print("Socket at master node created successfully\n")

    master_server.bind(('', port))

    # Start listening to requests
```

```
    master_server.listen(10)
    print("Clock server started...\n")

    # start making connections
    print("Starting to make connections...\n")
    master_thread = threading.Thread(
                    target = startConnecting,
                    args = (master_server, ))
    master_thread.start()

    # start synchronization
    print("Starting synchronization parallelly...\n")
    sync_thread = threading.Thread(
                    target = synchronizeAllClocks,
                    args = ())
    sync_thread.start()


# Driver function
if __name__ == '__main__':

    # Trigger the Clock Server
    initiateClockServer(port = 2050)
```

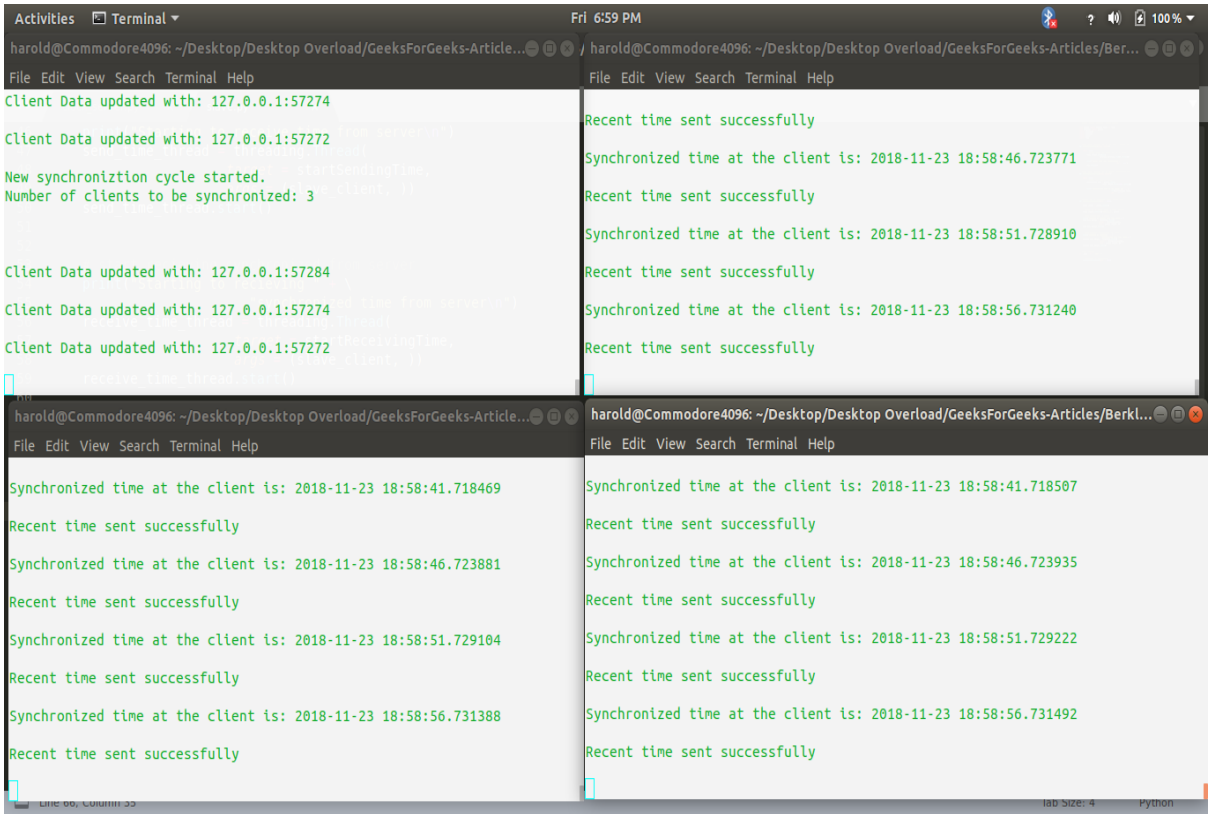
Client.py

```
# Python3 program imitating a client process

from timeit import default_timer as timer
from dateutil import parser
import threading
import datetime
import socket
import time

# client thread function used to send time at client side
def startSendingTime(slave_client):

    while True:
        # provide server with clock time at the client
        slave_client.send(str(
                datetime.datetime.now()).encode())

        print("Recent time sent successfully",
                                end = "\n\n")
        time.sleep(5)

# client thread function used to receive synchronized time
def startReceivingTime(slave_client):

    while True:
        # receive data from the server
        Synchronized_time = parser.parse(
                    slave_client.recv(1024).decode())
```

```
        print("Synchronized time at the client is: " + \
                              str(Synchronized_time),
                              end = "\n\n")

# function used to Synchronize client process time
def initiateSlaveClient(port = 8080):

    slave_client = socket.socket()

    # connect to the clock server on local computer
    slave_client.connect(('127.0.0.1', port))

    # start sending time to server
    print("Starting to receive time from server\n")
    send_time_thread = threading.Thread(
                target = startSendingTime,
                args = (slave_client, ))
    send_time_thread.start()

    # start receiving synchronized from server
    print("Starting to receiving " + \
                    "synchronized time from server\n")
    receive_time_thread = threading.Thread(
                target = startReceivingTime,
                args = (slave_client, ))
    receive_time_thread.start()

# Driver function
if __name__ == '__main__':

    # initialize the Slave / Client
```

initiateSlaveClient(port = 2050)

output:



ASSIGNMENT NO. 6

Title: Mutual Exclusion

Code:

TokenServer1.java

```
import java.io.*;
import java.net.*;
public class TokenServer1
{
   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
        {
```

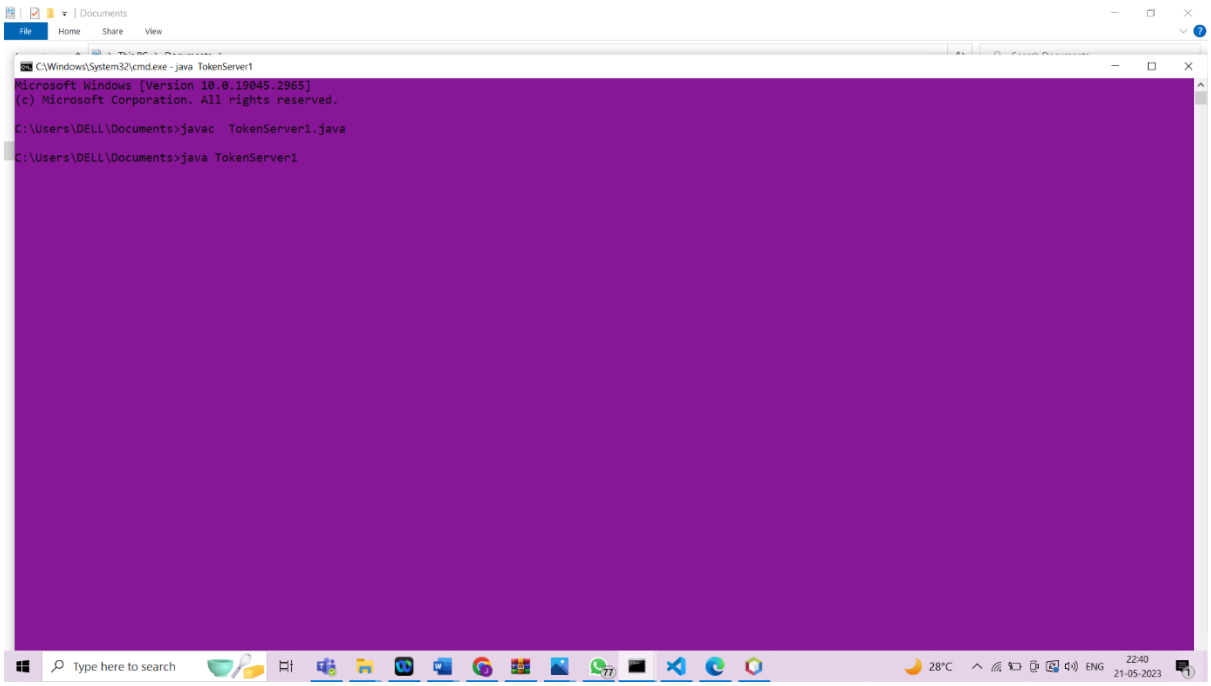
```
InetAddress lclhost;
BufferedReader br;
String str1;
TokenClient21 tkcl;
TokenClient21 ser;
while(true)
{
lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient21(lclhost);
tkcl.setRecPort(8004);
tkcl.setSendPort(9002);
lclhost=InetAddress.getLocalHost();
ser = new TokenClient21(lclhost);
ser.setSendPort(9000);
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;
```

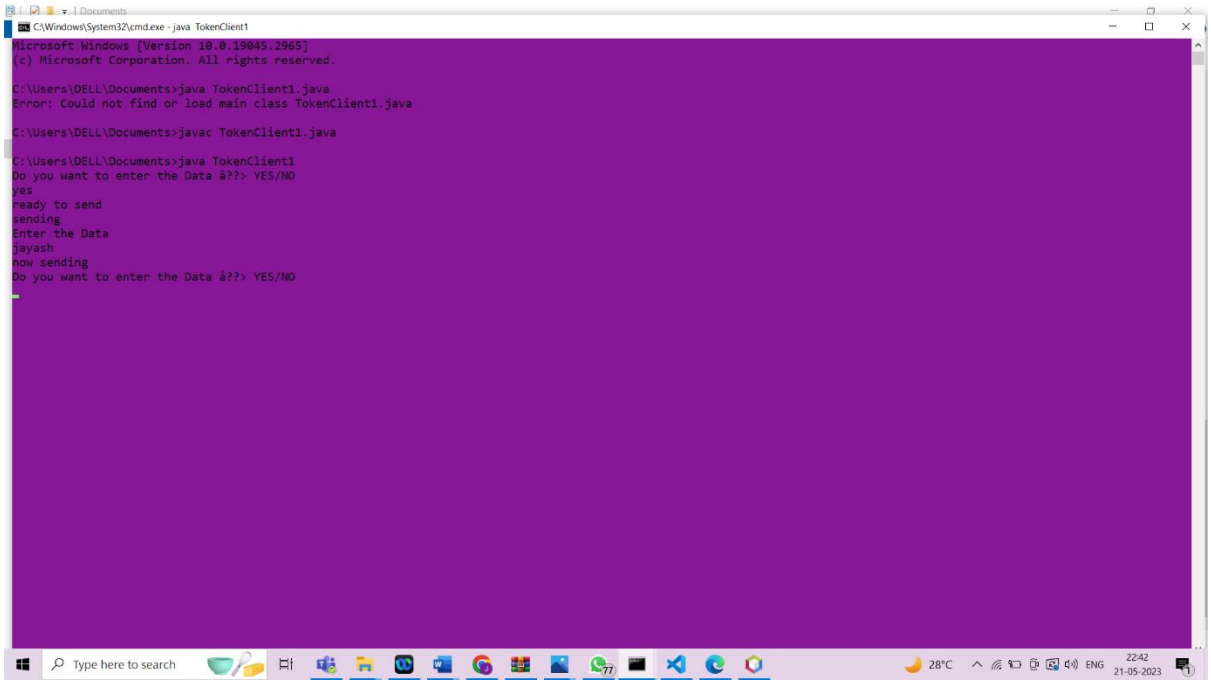
```
            }
        }
        else
        {
        System.out.println("entering recieving mode");
            tkcl.recData();
            hasToken=true;
        }
      }
   }
}
class TokenClient21
{
   InetAddress lclhost;
   int sendport,recport;
   boolean setSendData = false;
   boolean hasToken = false;
   TokenClient21 tkcl;
   TokenClient21 ser;

   TokenClient21(InetAddress lclhost)
   {

      this.lclhost = lclhost;
   }

   void setSendPort(int sendport)
   {
      this.sendport = sendport;
   }
   void setRecPort(int recport)
```

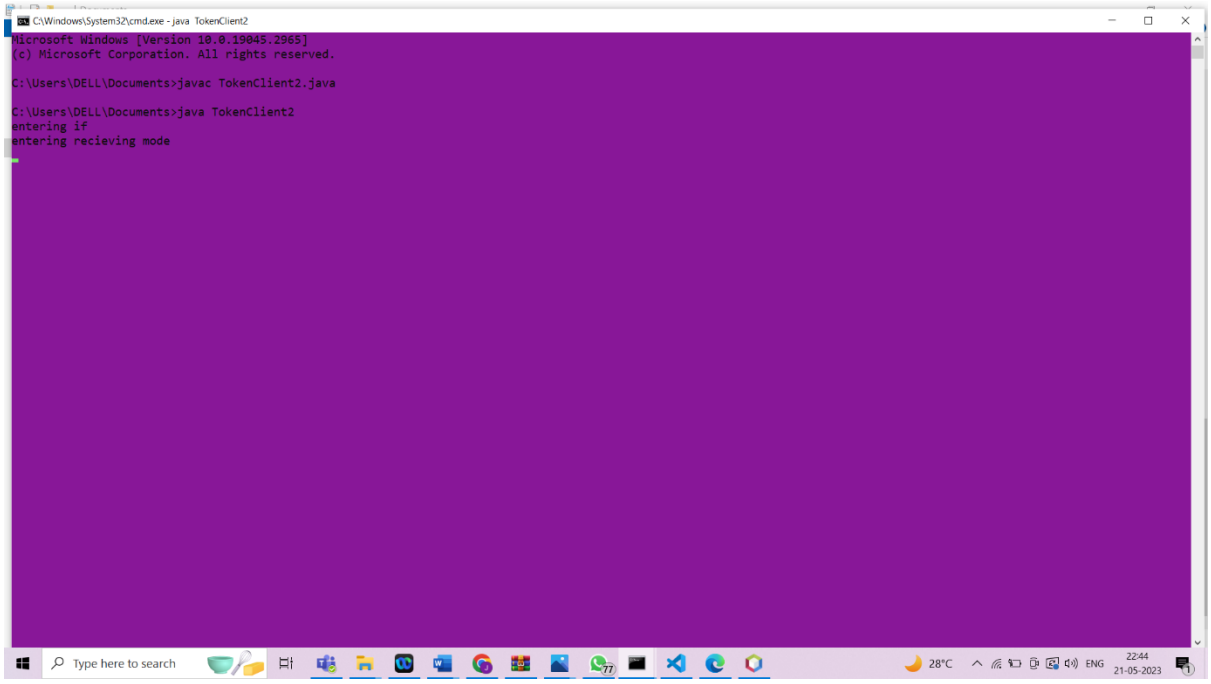
```
  {
     this.recport = recport;
  }
  void sendData() throws Exception
  {
     System.out.println("case");
     BufferedReader br;
     String str="Token";
     DatagramSocket ds;
     DatagramPacket dp;

     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.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:




C:\Windows\System32\cmd.exe - java TokenClient1
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.
C:\Users\DELL\Documents>java TokenClient1.java
Error: Could not find or load main class TokenClient1.java
C:\Users\DELL\Documents>javac TokenClient1.java
C:\Users\DELL\Documents>java TokenClient1
Do you want to enter the Data â??> YES/NO
yes
ready to send
sending
Enter the Data
jayash
now sending
Do you want to enter the Data â??> YES/NO
Type here to search
28°C
ENG
22:42
21-05-2023


C:\Windows\System32\cmd.exe - java TokenClient2
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.
C:\Users\DELL\Documents>javac TokenClient2.java
C:\Users\DELL\Documents>java TokenClient2
entering if
entering recieving mode
Type here to search
28°C
ENG
22:44
21-05-2023

ASSIGNMENT NO. 7

Title: Election Algorithms

Bully.java

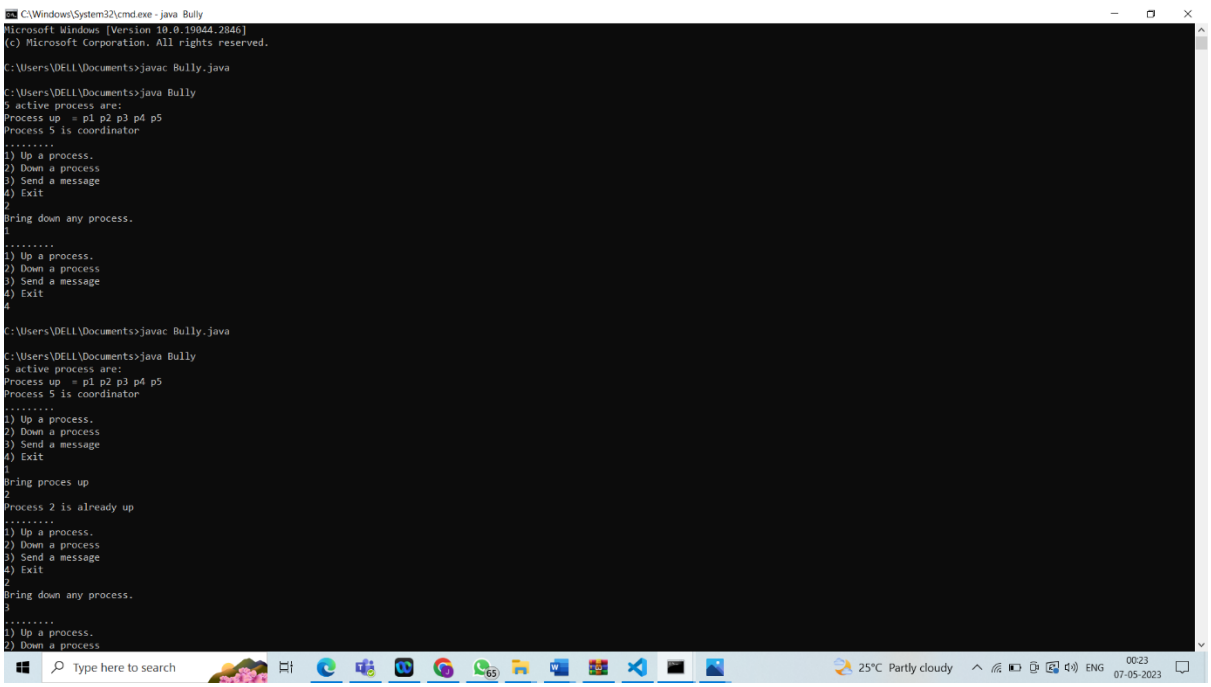
```
import java.io.InputStream;
import java.io.PrintStream;
import java.util.Scanner;

public class Bully {
   static boolean[] state = new boolean[5];
   int coordinator;

   public static void up(int up) {
      if (state[up - 1]) {
         System.out.println("Process " + up + " is already up");
      } else {
         int i;
         Bully.state[up - 1] = true;
         System.out.println("Process " + up + " held election");
         for (i = up; i < 5; ++i) {
            System.out.println("Election message sent from process " + up + " to process " + (i + 1));
         }
         for (i = up + 1; i <= 5; ++i) {
            if (!state[i - 1]) continue;
            System.out.println("Alive message send from process " + i + " to process " + up);
            break;
         }
      }
   }
```

```
public static void down(int down) {
   if (!state[down - 1]) {
      System.out.println("Process " + down + " is already dowm.");
   } else {
      Bully.state[down - 1] = false;
   }
}

public static void mess(int mess) {
   if (state[mess - 1]) {
      if (state[4]) {
         System.out.println("0K");
      } else if (!state[4]) {
         int i;
         System.out.println("Process " + mess + " election");
         for (i = mess; i < 5; ++i) {
            System.out.println("Election send from process " + mess + " to process " + (i + 1));
         }
         for (i = 5; i >= mess; --i) {
            if (!state[i - 1]) continue;
            System.out.println("Coordinator message send from process " + i + " to all");
            break;
         }
      }
   } else {
      System.out.println("Process " + mess + " is down");
   }
}

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

```
int choice;
Scanner sc = new Scanner(System.in);
for (int i = 0; i < 5; ++i) {
   Bully.state[i] = true;
}
System.out.println("5 active process are:");
System.out.println("Process up  = p1 p2 p3 p4 p5");
System.out.println("Process 5 is coordinator");
do {
   System.out.println(".....");
   System.out.println("1) Up a process.");
   System.out.println("2) Down a process");
   System.out.println("3) Send a message");
   System.out.println("4) Exit");
   choice = sc.nextInt();
   switch (choice) {
      case 1: {
         System.out.println("Bring proces up");
         int up = sc.nextInt();
         if (up == 5) {
            System.out.println("Process 5 is co-ordinator");
            Bully.state[4] = true;
            break;
         }
         Bully.up(up);
         break;
      }
      case 2: {
         System.out.println("Bring down any process.");
         int down = sc.nextInt();
         Bully.down(down);
```

```
                break;
            }
            case 3: {
                System.out.println("Which process will send message");
                int mess = sc.nextInt();
                Bully.mess(mess);
            }
          }
      } while (choice != 4);
      sc.close();

   }
}
```

Output:



Ring.java

Code:

```
import java.util.Scanner;


public class Ring1 {


   public static void main(String[] args) {


      // TODO Auto-generated method stub


      int temp, i, j;
      char str[] = new char[10];
      Rr proc[] = new Rr[10];


// object initialisation
      for (i = 0; i < proc.length; i++)
         proc[i] = new Rr();


// scanner used for getting input from console
      Scanner in = new Scanner(System.in);
      System.out.println("Enter the number of process : ");
      int num = in.nextInt();


// getting input from users
      for (i = 0; i < num; i++) {
         proc[i].index = i;
         System.out.println("Enter the id of process : ");
         proc[i].id = in.nextInt();
         proc[i].state = "active";
         proc[i].f = 0;
      }
```

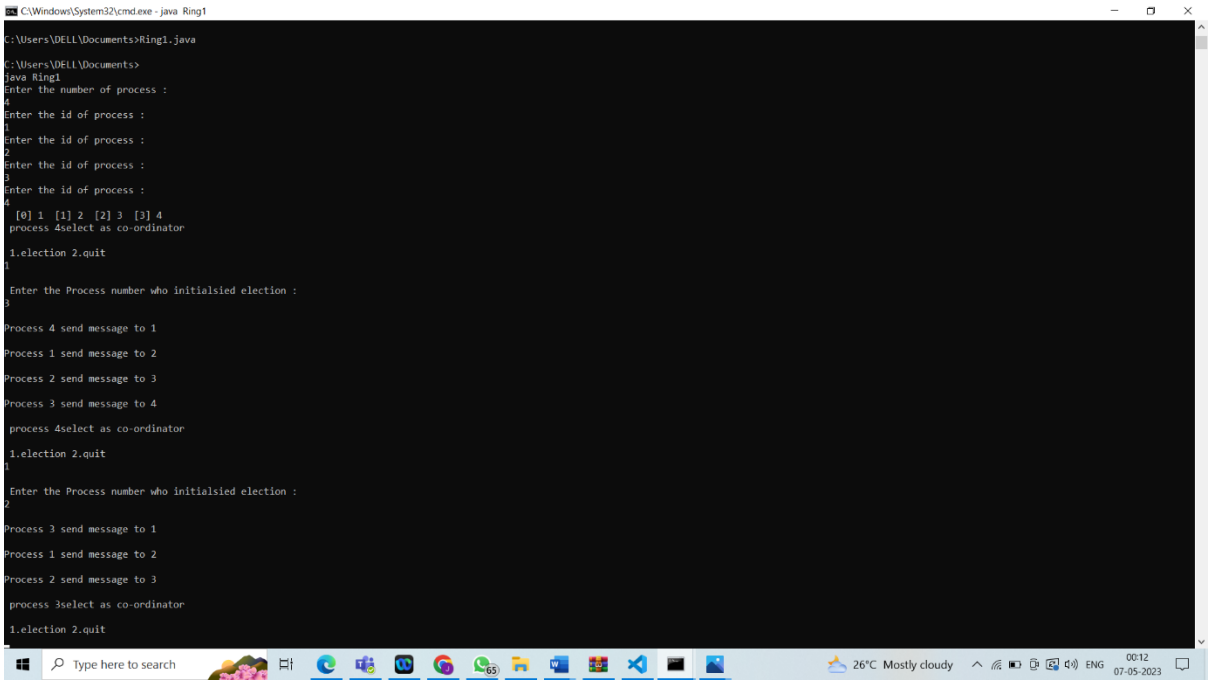
```
// sorting the processes from on the basis of id
      for (i = 0; i < num - 1; i++) {
         for (j = 0; j < num - 1; j++) {
            if (proc[j].id > proc[j + 1].id) {
               temp = proc[j].id;
               proc[j].id = proc[j + 1].id;
               proc[j + 1].id = temp;
            }
         }
      }

      for (i = 0; i < num; i++) {
         System.out.print("  [" + i + "]" + " " + proc[i].id);
      }



      int init;
      int ch;
      int temp1;
      int temp2;
      int ch1;
      int arr[] = new int[10];

      proc[num - 1].state = "inactive";

      System.out.println("\n process " + proc[num - 1].id + "select as co-ordinator");

      while (true) {
         System.out.println("\n 1.election 2.quit ");
         ch = in.nextInt();
```

```
for (i = 0; i < num; i++) {
   proc[i].f = 0;
}

switch (ch) {
case 1:
   System.out.println("\n Enter the Process number who initialsied election : ");
   init = in.nextInt();
   temp2 = init;
   temp1 = init + 1;

   i = 0;

   while (temp2 != temp1) {
      if ("active".equals(proc[temp1].state) && proc[temp1].f == 0) {

         System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);
         proc[temp1].f = 1;
         init = temp1;
         arr[i] = proc[temp1].id;
         i++;
      }
      if (temp1 == num) {
         temp1 = 0;
      } else {
         temp1++;
      }
   }

   System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);
```

```
            arr[i] = proc[temp1].id;
            i++;
            int max = -1;

// finding maximum for co-ordinator selection
            for (j = 0; j < i; j++) {
               if (max < arr[j]) {
                  max = arr[j];
               }
            }

// co-ordinator is found then printing on console
            System.out.println("\n process " + max + "select as co-ordinator");

            for (i = 0; i < num; i++) {

               if (proc[i].id == max) {
                  proc[i].state = "inactive";
               }
            }
            break;
         case 2:
         System.out.println("Program terminated ...");
         return ;
         default:
            System.out.println("\n invalid response \n");
            break;
         }

      }
   }
```

```
}

class Rr {

    public int index;   // to store the index of process
    public int id;      // to store id/name of process
    public int f;
    String state;       // indiactes whether active or inactive state of node

}
```

Output:



ASSIGNMENT NO. 8

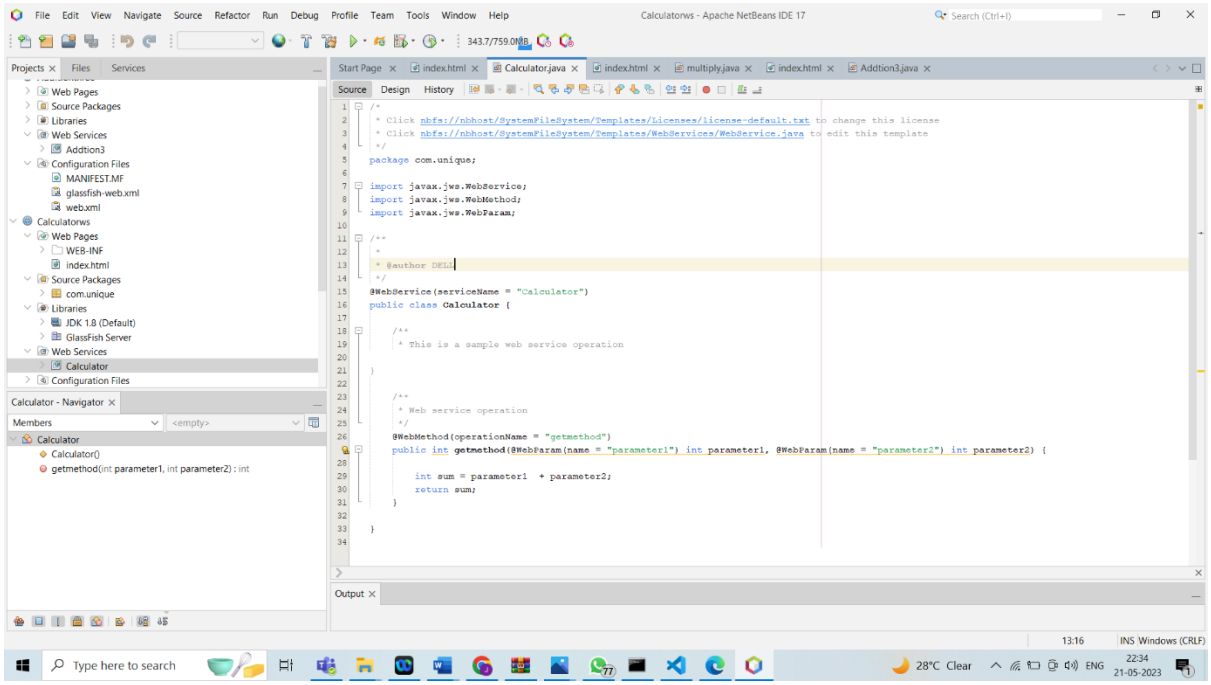
Title: Web Services

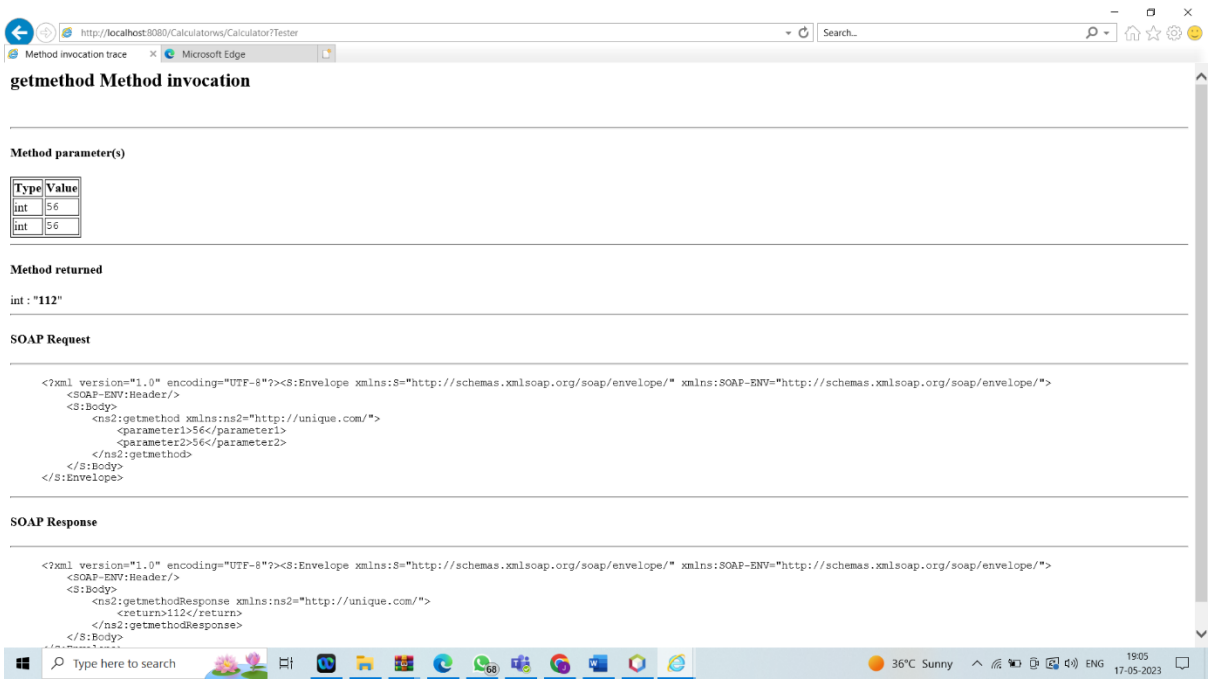
Calculator.java

package com.unique;

```
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author DELL
 */
@WebService(serviceName = "Calculator")
public class Calculator {

    /**
     * This is a sample web service operation

}

    /**
     * Web service operation
     */
    @WebMethod(operationName = "getmethod")
    public int getmethod(@WebParam(name = "parameter1") int parameter1, @WebParam(name =
"parameter2") int parameter2) {

        int sum = parameter1  + parameter2;
        return sum;
    }
    }
```

Output:


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Calculatorws - Apache NetBeans IDE 17
Projects
Files
Services
Web Pages
Source Packages
Libraries
Web Services
Addtion3
Configuration Files
MANIFEST.MF
glassfish-web.xml
web.xml
Calculatorws
Web Pages
WEB-INF
index.html
Source Packages
com.unique
Libraries
JDK 1.8 (Default)
GlassFish Server
Web Services
Calculator
Configuration Files
Calculator - Navigator
Members
Calculator()
getmethod(int parameter1, int parameter2) : int
Start Page
index.html
Calculator.java
index.html
multiply.java
index.html
Addtion3.java
Source
Design
History
package com.unique;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService(serviceName = "Calculator")
public class Calculator {
/**
* This is a sample web service operation
*/
@WebMethod(operationName = "getmethod")
public int getmethod(@WebParam(name = "parameter1") int parameter1, @WebParam(name = "parameter2") int parameter2) {
int sum = parameter1 + parameter2;
return sum;
}
}
Output
13:16
INS Windows (CRLF)
Type here to search
28°C Clear
ENG
22:34
21-05-2023


http://localhost:8080/Calculatorws/Calculator?Tester
Search...
Method invocation trace
Microsoft Edge
getmethod Method invocation
Method parameter(s)
Type
Value
int
56
int
56
Method returned
int : "112"
SOAP Request
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:getmethod xmlns:ns2="http://unique.com/">
<parameter1>56</parameter1>
<parameter2>56</parameter2>
</ns2:getmethod>
</S:Body>
</S:Envelope>
SOAP Response
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Header/>
<S:Body>
<ns2:getmethodResponse xmlns:ns2="http://unique.com/">
<return>112</return>
</ns2:getmethodResponse>
</S:Body>
Type here to search
36°C Sunny
ENG
19:05
17-05-2023

••