**Assignment 1**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Factorial.java**

import java.math.BigInteger;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Factorial extends Remote {

    BigInteger fact(int num) throws RemoteException;

}

**FactorialImpl.java**

import java.math.BigInteger;

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class FactorialImpl extends UnicastRemoteObject implements Factorial {

    public FactorialImpl() throws RemoteException {

        super();

    }

    public BigInteger fact(int num) throws RemoteException {

        BigInteger factorial = BigInteger.ONE;

        for (int i = 1; i <= num; i++) {

            factorial = factorial.multiply(BigInteger.valueOf(i));

        }

        return factorial;

    }

}

**FactorialClient.java**

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.rmi.NotBoundException;

import java.net.MalformedURLException;

public class FactorialClient {

    public static void main(String[] args) {

        try {

            Factorial f = (Factorial) Naming.lookup("rmi://localhost/FactorialService");

            int num = 30; // You can take input from args if needed

            System.out.println("Factorial of " + num + " is: " + f.fact(num));

        } catch (MalformedURLException e) {

            System.out.println("MalformedURLException: " + e);

        } catch (RemoteException e) {

            System.out.println("RemoteException: " + e);

        } catch (NotBoundException e) {

            System.out.println("NotBoundException: " + e);

        } catch (ArithmeticException e) {

            System.out.println("ArithmeticException: " + e);

        }

    }

}

**FactorialServer.java**

import java.rmi.Naming;

public class FactorialServer {

    public FactorialServer() {

        try {

            Factorial f = new FactorialImpl();

            Naming.rebind("rmi://localhost/FactorialService", f);

            System.out.println("FactorialService is ready.");

        } catch (Exception e) {

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

        }

    }

    public static void main(String[] args) {

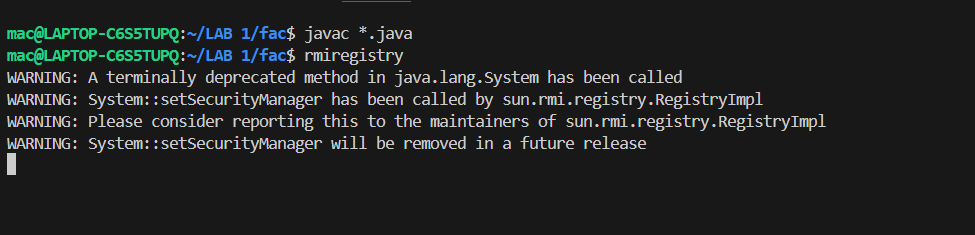
        new FactorialServer();

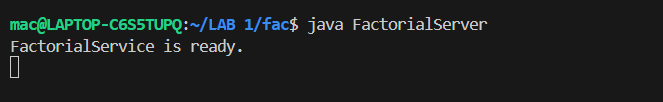
    }

}

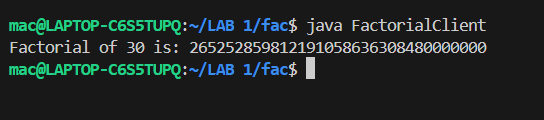
**Output-**

**Terminal-1**

****

**Terminal-2**

**Terminal-3**

****

**Assignment 2**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Reverse.idl**

module ReverseModule {

interface Reverse {

string reverse\_string(in string str);

};

};

**ReverseImpl.java**

// ReverseImpl.java

import ReverseModule.ReversePOA;

class ReverseImpl extends ReversePOA {

    ReverseImpl() {

        super();

        System.out.println("Reverse Object Created");

    }

    public String reverse\_string(String name) {

        StringBuffer str = new StringBuffer(name);

        str.reverse();

        return "Server Send " + str;

    }

}

**ReverseClient.java**

// ReverseClient.java

import ReverseModule.\*;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import java.io.\*;

class ReverseClient {

    public static void main(String args[]) {

        try {

            ORB orb = ORB.init(args, null);

            org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            Reverse ReverseImpl = ReverseHelper.narrow(ncRef.resolve\_str("Reverse"));

            System.out.println("Enter String=");

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

            String str = br.readLine();

            String tempStr = ReverseImpl.reverse\_string(str);

            System.out.println(tempStr);

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

}

**ReverseServer.java**

// ReverseServer.java

import ReverseModule.Reverse;

import ReverseModule.ReverseHelper;

import org.omg.CosNaming.\*;

import org.omg.CosNaming.NamingContextPackage.\*;

import org.omg.CORBA.\*;

import org.omg.PortableServer.\*;

class ReverseServer {

    public static void main(String[] args) {

        try {

            ORB orb = ORB.init(args, null);

            POA rootPOA = POAHelper.narrow(orb.resolve\_initial\_references("RootPOA"));

            rootPOA.the\_POAManager().activate();

            ReverseImpl rvr = new ReverseImpl();

            org.omg.CORBA.Object ref = rootPOA.servant\_to\_reference(rvr);

            Reverse h\_ref = ReverseHelper.narrow(ref);

            org.omg.CORBA.Object objRef = orb.resolve\_initial\_references("NameService");

            NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

            String name = "Reverse";

            NameComponent path[] = ncRef.to\_name(name);

            ncRef.rebind(path, h\_ref);

            System.out.println("Reverse Server reading and waiting....");

            orb.run();

        } catch (Exception e) {

            e.printStackTrace();

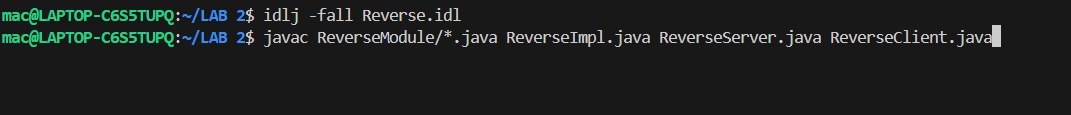
        }

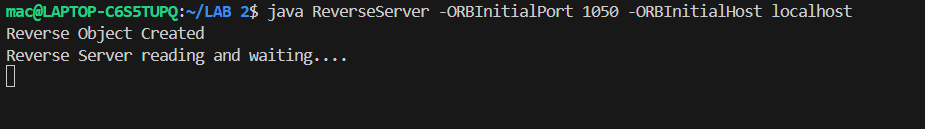
    }

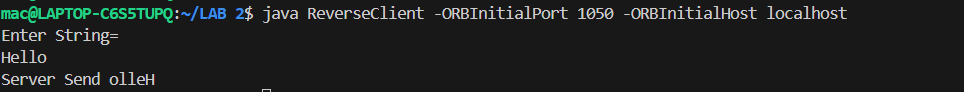
}

**Output-**

**Terminal-1**

****

**Terminal-2**

**Terminal-3**

**Assignment 3**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Sum\_mpi.c**

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

int main(int argc, char\* argv[]) {

    int rank, size, N;

    int \*data = NULL, \*sub\_data;

    int local\_sum = 0, total\_sum = 0;

    MPI\_Init(&argc, &argv);

    MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

    MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

    if (rank == 0) {

        printf("Enter number of elements (N): ");

        scanf("%d", &N);

        if (N % size != 0) {

            printf("N must be divisible by number of processes.\n");

            MPI\_Abort(MPI\_COMM\_WORLD, 1);

        }

        data = (int\*)malloc(N \* sizeof(int));

        printf("Enter %d elements:\n", N);

        for (int i = 0; i < N; i++)

            scanf("%d", &data[i]);

    }

    MPI\_Bcast(&N, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

    int chunk\_size = N / size;

    sub\_data = (int\*)malloc(chunk\_size \* sizeof(int));

    MPI\_Scatter(data, chunk\_size, MPI\_INT, sub\_data, chunk\_size, MPI\_INT, 0, MPI\_COMM\_WORLD);

    for (int i = 0; i < chunk\_size; i++)

        local\_sum += sub\_data[i];

    printf("Local sum at rank %d = %d\n", rank, local\_sum);

    MPI\_Reduce(&local\_sum, &total\_sum, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

    if (rank == 0)

        printf("Final sum = %d\n", total\_sum);

    if (data) free(data);

    free(sub\_data);

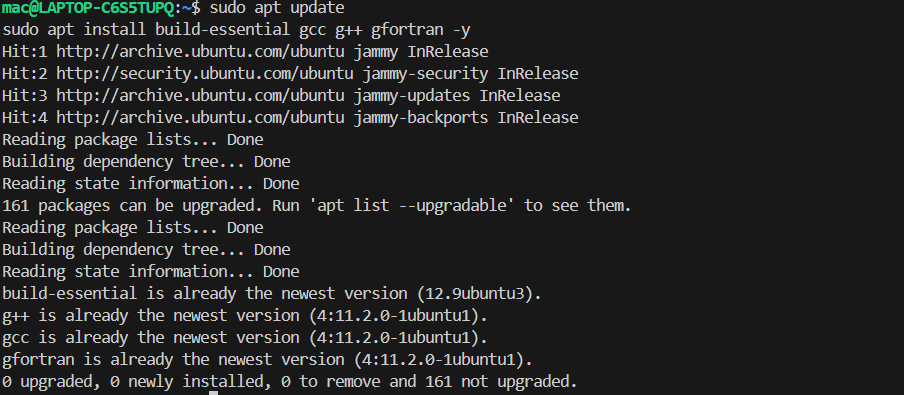
    MPI\_Finalize();

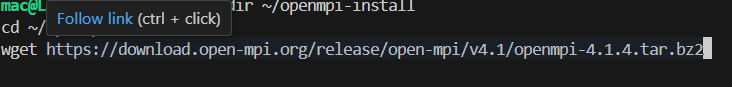
    return 0;

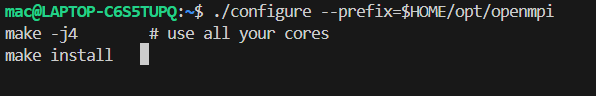
}

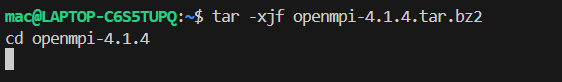
**Output-**

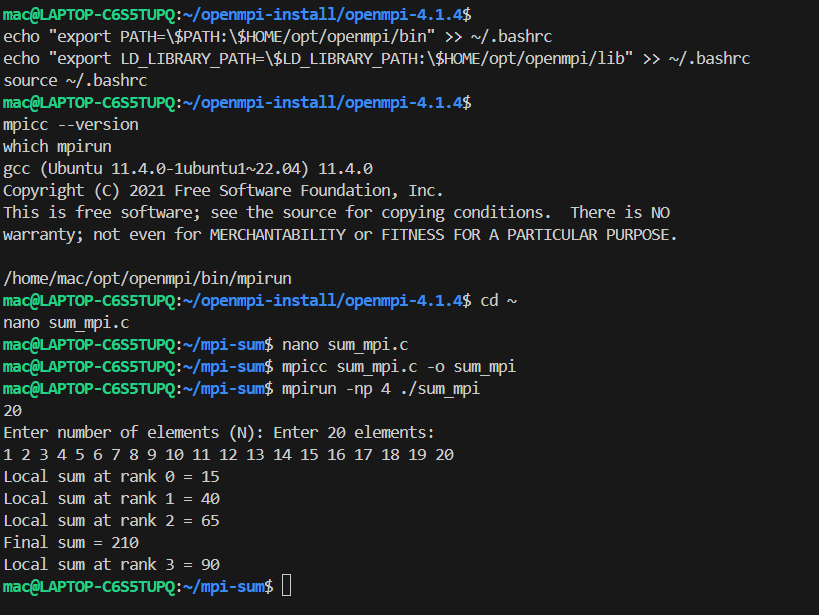
**Terminal-1**

****

****

****

****

****

**Assignment 4**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**master.py**

from functools import reduce

from dateutil import parser

import threading

import datetime

import socket

import time

client\_data = {}

# Function to receive clock time from a 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(f"Client Data updated with: {address}\n")

        time.sleep(5)

# Function to accept client connections

def startConnecting(master\_server):

    while True:

        # Accept a new client

        master\_slave\_connector, addr = master\_server.accept()

        slave\_address = f"{addr[0]}:{addr[1]}"

        print(f"{slave\_address} got connected successfully")

        current\_thread = threading.Thread(

            target = startReceivingClockTime,

            args = (master\_slave\_connector, slave\_address, )

        )

        current\_thread.start()

# Function to get the average clock difference

def getAverageClockDiff():

    current\_client\_data = client\_data.copy()

    time\_difference\_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

# Function to synchronize all clocks

def synchronizeAllClocks():

    while True:

        print("New synchronization cycle started.")

        print(f"Number of clients to be synchronized: {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(f"Something went wrong while sending synchronized time through {client\_addr}")

        else:

            print("No client data. Synchronization not applicable.")

        print("\n\n")

        time.sleep(5)

# Function 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))

    master\_server.listen(10)

    print("Clock server started...\n")

    print("Starting to make connections...\n")

    # Start connection handling in a new thread

    master\_thread = threading.Thread(target=startConnecting, args=(master\_server,))

    master\_thread.start()

    # Start synchronization in another thread

    print("Starting synchronization parallelly...\n")

    sync\_thread = threading.Thread(target=synchronizeAllClocks)

    sync\_thread.start()

# Driver function

if \_\_name\_\_ == '\_\_main\_\_':

    initiateClockServer(port=8080)

**server.py**

from timeit import default\_timer as timer

from dateutil import parser

import threading

import datetime

import socket

import time

# Function to send current client time to server

def startSendingTime(slave\_client):

    while True:

        slave\_client.send(str(datetime.datetime.now()).encode())

        print("Recent time sent successfully\n")

        time.sleep(5)

# Function to receive synchronized time from server

def startReceivingTime(slave\_client):

    while True:

        data = slave\_client.recv(1024).decode()

        synchronized\_time = parser.parse(data)

        print("Synchronized time at the client is: " + str(synchronized\_time) + "\n")

        time.sleep(5)

# Main function to run the client

def initiateSlaveClient(port=8080):

    slave\_client = socket.socket()

    slave\_client.connect(('127.0.0.1', port))

    print("Connected to Clock Server\n")

    # Start sending time to server

    send\_time\_thread = threading.Thread(target=startSendingTime, args=(slave\_client,))

    send\_time\_thread.start()

    # Start receiving synchronized time

    receive\_time\_thread = threading.Thread(target=startReceivingTime, args=(slave\_client,))

    receive\_time\_thread.start()

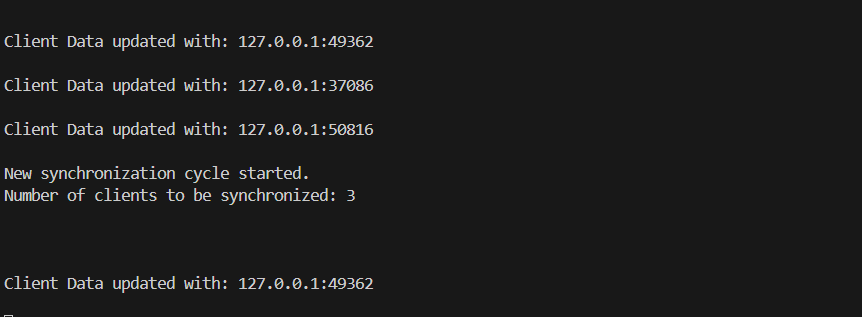
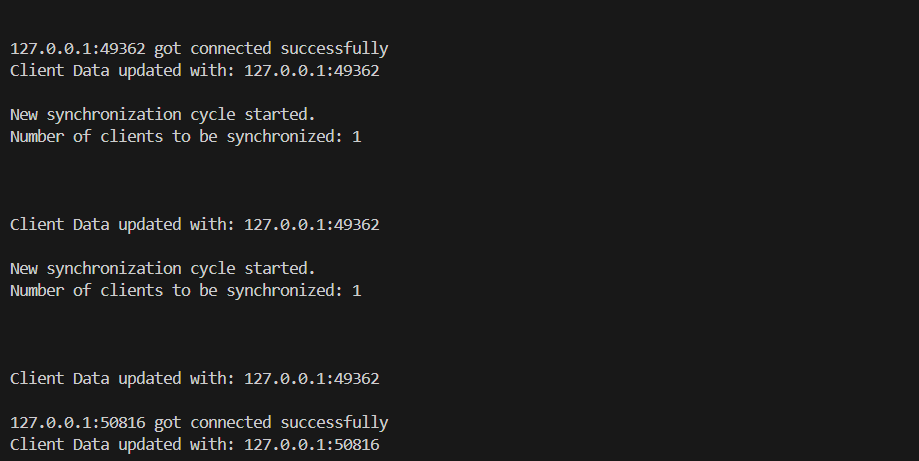
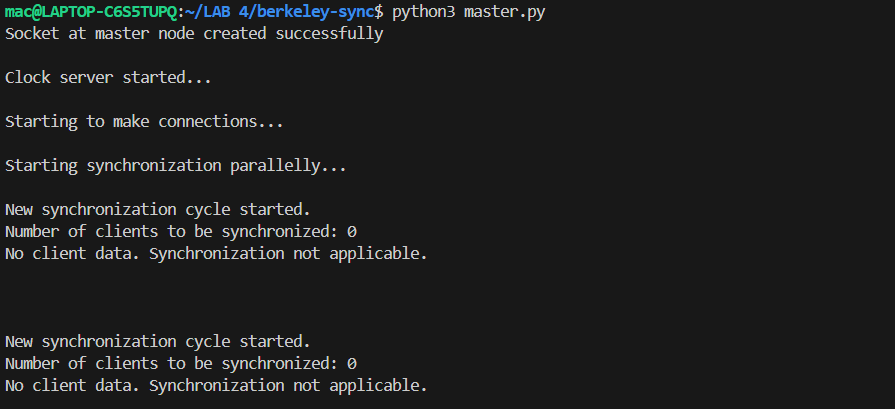
# Run the client

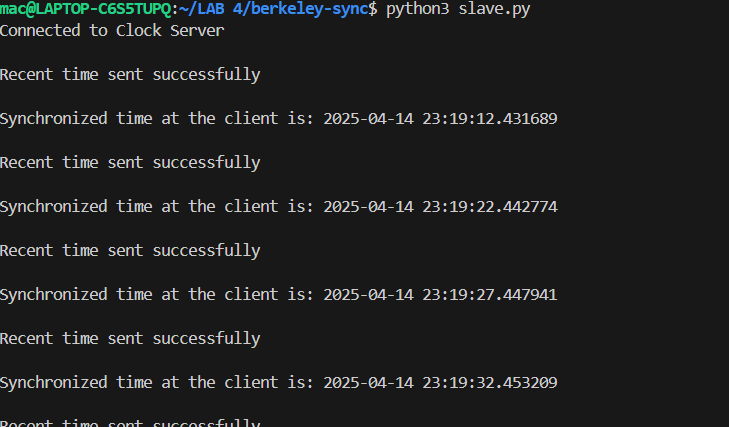
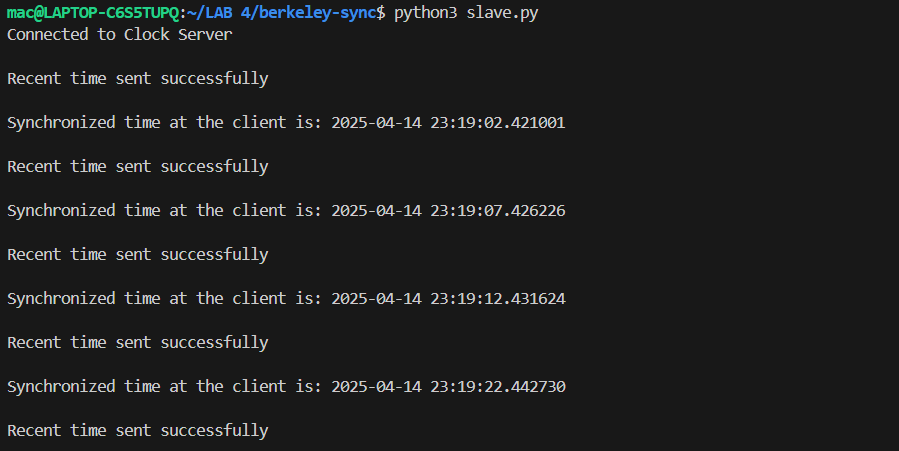
if \_\_name\_\_ == '\_\_main\_\_':

    initiateSlaveClient(port=8080)

**Output-**

**Terminal-1**

****

**Terminal-2**

**Or**

**Input-**

**master1.py**

# master.py

import socket

import time

def get\_slave\_time(host, port):

    s = socket.socket()

    s.connect((host, port))

    s.send("TIME\_REQUEST".encode())

    slave\_time = float(s.recv(1024).decode())

    s.close()

    return slave\_time

def send\_adjustment(host, port, adjustment):

    s = socket.socket()

    s.connect((host, port))

    s.send("TIME\_REQUEST".encode())

    s.recv(1024)  # receive time

    s.send(str(adjustment).encode())

    s.close()

def main():

    slave\_ports = [8001, 8002, 8003]

    times = []

    host = 'localhost'

    master\_time = time.time()

    print(f"[MASTER] Local time: {master\_time}")

    for port in slave\_ports:

        t = get\_slave\_time(host, port)

        times.append(t)

        print(f"[MASTER] Received from slave {port}: {t}")

    times.append(master\_time)

    avg = sum(times) / len(times)

    # Ignore outliers (difference > 2 sec)

    filtered = [t for t in times if abs(t - avg) <= 2]

    avg\_filtered = sum(filtered) / len(filtered)

    print(f"[MASTER] Average time (ignoring outliers): {avg\_filtered}")

    adjustments = [avg\_filtered - t for t in times]

    # Send adjustments

    for i, port in enumerate(slave\_ports):

        send\_adjustment(host, port, adjustments[i])

    print("[MASTER] Synchronization complete.")

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**slave1.py**

import socket

import time

import threading

def handle\_time\_request(port, offset):

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

        s.bind(('localhost', port))

        s.listen(1)

        print(f"[SLAVE {port}] Waiting for master's time request...")

        conn, \_ = s.accept()

        with conn:

            conn.recv(1024)  # Just to simulate "time\_request"

            current\_time = time.time() + offset

            print(f"[SLAVE {port}] Sending local time: {current\_time}")

            conn.sendall(str(current\_time).encode())

def handle\_adjustment(port, offset):

    with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:

        s.bind(('localhost', port))

        s.listen(1)

        print(f"[SLAVE {port}] Waiting for time adjustment from master...")

        conn, \_ = s.accept()

        with conn:

            adjustment = float(conn.recv(1024).decode())

            print(f"[SLAVE {port}] Received adjustment: {adjustment}")

            print(f"[SLAVE {port}] New time would be: {time.time() + offset + adjustment}")

if \_\_name\_\_ == "\_\_main\_\_":

    port = int(input("Enter slave port (e.g. 8001): "))

    offset = float(input("Enter time offset (e.g. 3 or -2): "))

    # Use threads to keep both sockets alive

    t1 = threading.Thread(target=handle\_time\_request, args=(port, offset))

    t2 = threading.Thread(target=handle\_adjustment, args=(port, offset))

    t1.start()

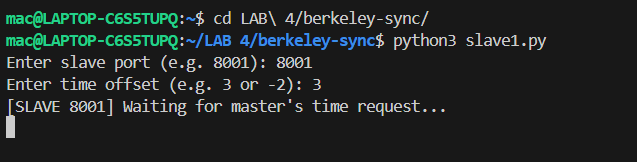
    t1.join()

    time.sleep(1)  # Give master time to compute adjustments

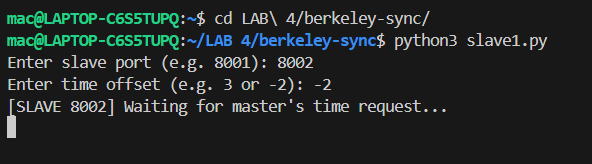
    t2.start()

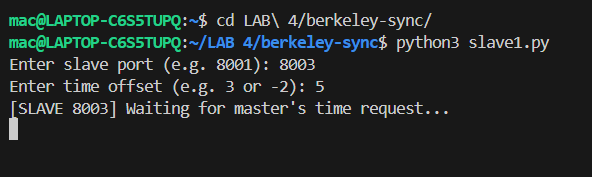
    t2.join()

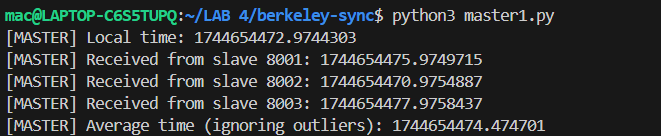
**Output-**

**Terminal-1**

**Terminal-2**

****

**Terminal-3**

**Terminal-4**

**Assignment 5**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Bully.java**

import java.util.\*;

public class Bully {

    int coordinator;

    int max\_processes;

    boolean processes[];

    public Bully(int max) {

        max\_processes = max;

        processes = new boolean[max\_processes];

        coordinator = max;

        System.out.println("Creating processes..");

        for(int i = 0; i < max; i++) {

            processes[i] = true;

            System.out.println("P"+ (i+1) + " created");

        }

        System.out.println("Process P" + coordinator + " is the coordinator");

    }

    void displayProcesses() {

        for(int i = 0; i < max\_processes; i++) {

            if(processes[i]) {

                System.out.println("P" + (i+1) + " is up");

            } else {

                System.out.println("P" + (i+1) + " is down");

            }

        }

        System.out.println("Process P" + coordinator + " is the coordinator");

    }

    void upProcess(int process\_id) {

        if(!processes[process\_id - 1]) {

            processes[process\_id - 1] = true;

            System.out.println("Process " + process\_id + " is now up.");

        } else {

            System.out.println("Process " + process\_id + " is already up.");

        }

    }

    void downProcess(int process\_id) {

        if(!processes[process\_id - 1]) {

            System.out.println("Process " + process\_id + " is already down.");

        } else {

            processes[process\_id - 1] = false;

            System.out.println("Process " + process\_id + " is down.");

        }

    }

    void runElection(int process\_id) {

        coordinator = process\_id;

        boolean keepGoing = true;

        for(int i = process\_id; i < max\_processes && keepGoing; i++) {

            System.out.println("Election message sent from process " + process\_id + " to process " + (i+1));

            if(processes[i]) {

                keepGoing = false;

                runElection(i + 1);

            }

        }

    }

    public static void main(String args[]) {

        Bully bully = null;

        int max\_processes = 0, process\_id = 0;

        int choice = 0;

        Scanner sc = new Scanner(System.in);

        while(true) {

            System.out.println("Bully Algorithm");

            System.out.println("1. Create processes");

            System.out.println("2. Display processes");

            System.out.println("3. Up a process");

            System.out.println("4. Down a process");

            System.out.println("5. Run election algorithm");

            System.out.println("6. Exit Program");

            System.out.print("Enter your choice:- ");

            choice = sc.nextInt();

            switch(choice) {

                case 1:

                    System.out.print("Enter the number of processes:- ");

                    max\_processes = sc.nextInt();

                    bully = new Bully(max\_processes);

                    break;

                case 2:

                    bully.displayProcesses();

                    break;

                case 3:

                    System.out.print("Enter the process number to up:- ");

                    process\_id = sc.nextInt();

                    bully.upProcess(process\_id);

                    break;

                case 4:

                    System.out.print("Enter the process number to down:- ");

                    process\_id = sc.nextInt();

                    bully.downProcess(process\_id);

                    break;

                case 5:

                    System.out.print("Enter the process number which will perform election:- ");

                    process\_id = sc.nextInt();

                    bully.runElection(process\_id);

                    bully.displayProcesses();

                    break;

                case 6:

                    System.exit(0);

                    break;

                default:

                    System.out.println("Error in choice. Please try again.");

                    break;

            }

        }

    }

}

**Ring.java**

import java.util.\*;

public class Ring {

    int max\_processes;

    int coordinator;

    boolean processes[];

    ArrayList<Integer> pid;

    public Ring(int max) {

        coordinator = max;

        max\_processes = max;

        pid = new ArrayList<Integer>();

        processes = new boolean[max];

        for(int i = 0; i < max; i++) {

            processes[i] = true;

            System.out.println("P" + (i+1) + " created.");

        }

        System.out.println("P" + (coordinator) + " is the coordinator");

    }

    void displayProcesses() {

        for(int i = 0; i < max\_processes; i++) {

            if(processes[i])

                System.out.println("P" + (i+1) + " is up.");

            else

                System.out.println("P" + (i+1) + " is down.");

        }

        System.out.println("P" + (coordinator) + " is the coordinator");

    }

    void upProcess(int process\_id) {

        if(!processes[process\_id-1]) {

            processes[process\_id-1] = true;

            System.out.println("Process P" + (process\_id) + " is up.");

        } else {

            System.out.println("Process P" + (process\_id) + " is already up.");

        }

    }

    void downProcess(int process\_id) {

        if(!processes[process\_id-1]) {

            System.out.println("Process P" + (process\_id) + " is already down.");

        } else {

            processes[process\_id-1] = false;

            System.out.println("Process P" + (process\_id) + " is down.");

        }

    }

    void displayArrayList(ArrayList<Integer> pid) {

        System.out.print("[ ");

        for(Integer x : pid) {

            System.out.print(x + " ");

        }

        System.out.print(" ]\n");

    }

    void initElection(int process\_id) {

        if(processes[process\_id-1]) {

            pid.add(process\_id);

            int temp = process\_id;

            System.out.print("Process P" + process\_id + " sending the following list:- ");

            displayArrayList(pid);

            while(temp != process\_id - 1) {

                if(processes[temp]) {

                    pid.add(temp+1);

                    System.out.print("Process P" + (temp + 1) + " sending the following list:- ");

                    displayArrayList(pid);

                }

                temp = (temp + 1) % max\_processes;

            }

            coordinator = Collections.max(pid);

            System.out.println("Process P" + process\_id + " has declared P" + coordinator + " as the coordinator");

            pid.clear();

        }

    }

    public static void main(String args[]) {

        Ring ring = null;

        int max\_processes = 0, process\_id = 0;

        int choice = 0;

        Scanner sc = new Scanner(System.in);

        while(true) {

            System.out.println("Ring Algorithm");

            System.out.println("1. Create processes");

            System.out.println("2. Display processes");

            System.out.println("3. Up a process");

            System.out.println("4. Down a process");

            System.out.println("5. Run election algorithm");

            System.out.println("6. Exit Program");

            System.out.print("Enter your choice:- ");

            choice = sc.nextInt();

            switch(choice) {

                case 1:

                    System.out.print("Enter the total number of processes:- ");

                    max\_processes = sc.nextInt();

                    ring = new Ring(max\_processes);

                    break;

                case 2:

                    ring.displayProcesses();

                    break;

                case 3:

                    System.out.print("Enter the process to up:- ");

                    process\_id = sc.nextInt();

                    ring.upProcess(process\_id);

                    break;

                case 4:

                    System.out.print("Enter the process to down:- ");

                    process\_id = sc.nextInt();

                    ring.downProcess(process\_id);

                    break;

                case 5:

                    System.out.print("Enter the process which will initiate election:- ");

                    process\_id = sc.nextInt();

                    ring.initElection(process\_id);

                    break;

                case 6:

                    System.exit(0);

                    break;

                default:

                    System.out.println("Error in choice. Please try again.");

                    break;

            }

        }

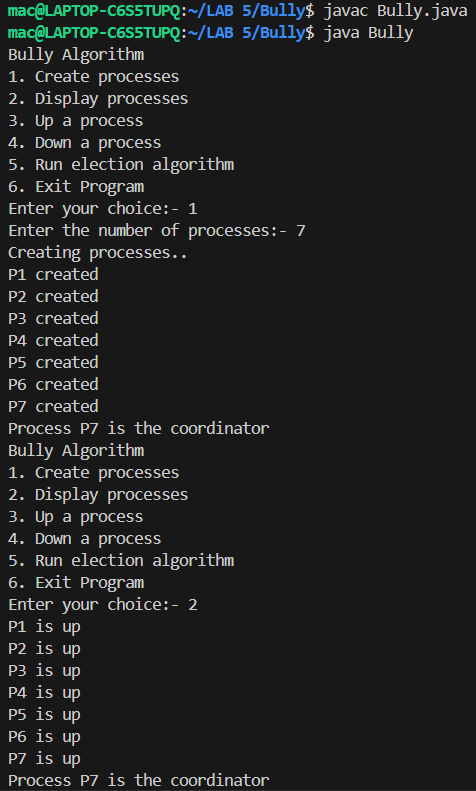
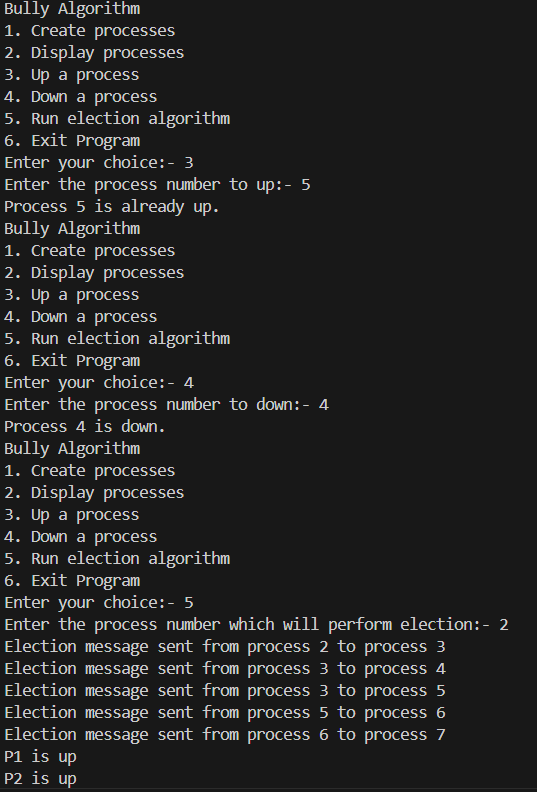
    }

}

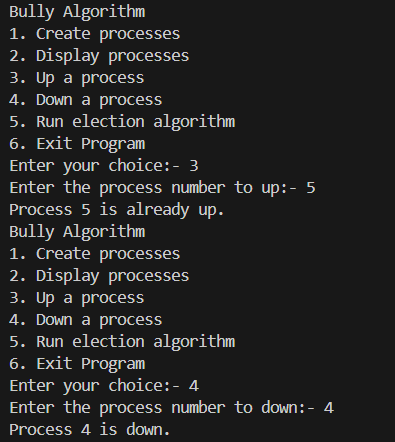
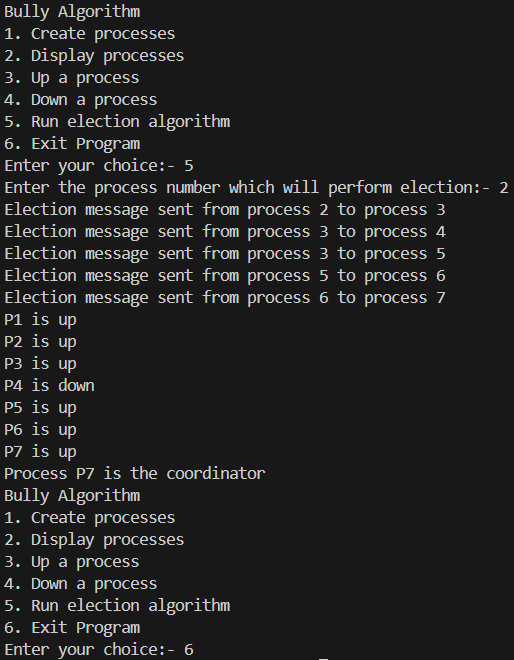
**Output-**

**Terminal-1**

1. **2.**

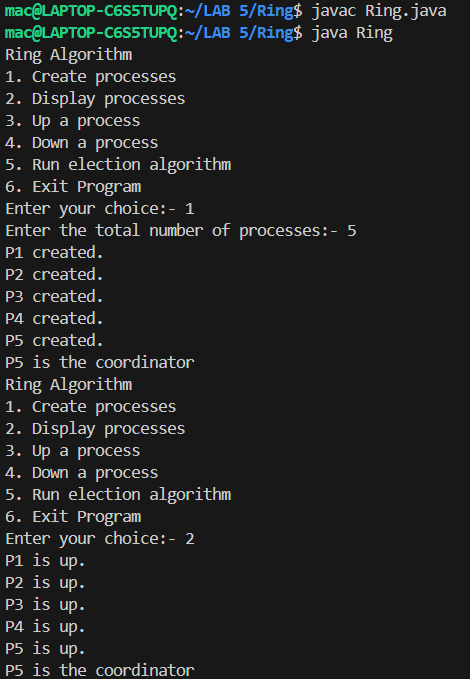
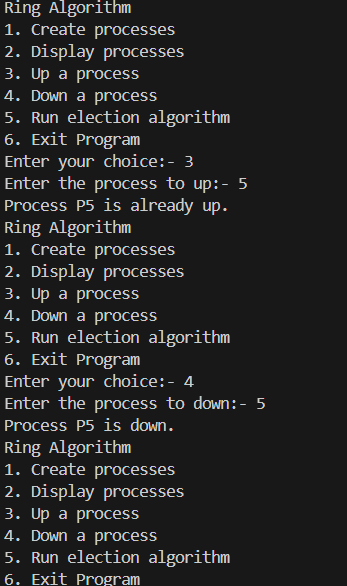
** **

**3. 4.**

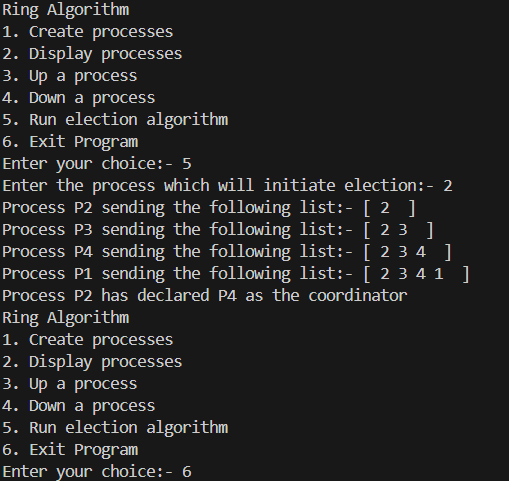
** **

**Terminal-2**

1. **2.**

** **

**3.**

****

**Assignment 6**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Tring.java**

import java.util.\*;

public class TRing{

    public static void main(String[] args){

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Number Of Nodes You Want In The Ring : ");

        int n = sc.nextInt();

        System.out.println("Ring Formed Is As Below: ");

        for(int i=0; i<n; i++){

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

        }

        System.out.println("0");

        int choice = 0;

        do{

            System.out.print("Enter Sender : ");

            int sender = sc.nextInt();

            System.out.print("Enter Receiver : ");

            int receiver = sc.nextInt();

            System.out.print("Enter Data To Send : ");

            int data = sc.nextInt();

            int token = 0;

            System.out.print("Token Passing : ");

            for(int i=token; i<sender; i++){

                System.out.print(" " + i + "->");

            }

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

            System.out.println("Sender:" + sender + " Sending Data: " + data);

            for(int i=sender; i!=receiver; i = (i+1)%n){

                System.out.println("Data: " + data + " Forwarded By: " + i);

            }

            System.out.println("Receiver: " + receiver + " Received The Data: " + data);

            token = sender;

            System.out.print("Do You Want To Send Data Again? If YES Enter 1, If NO Enter 0: ");

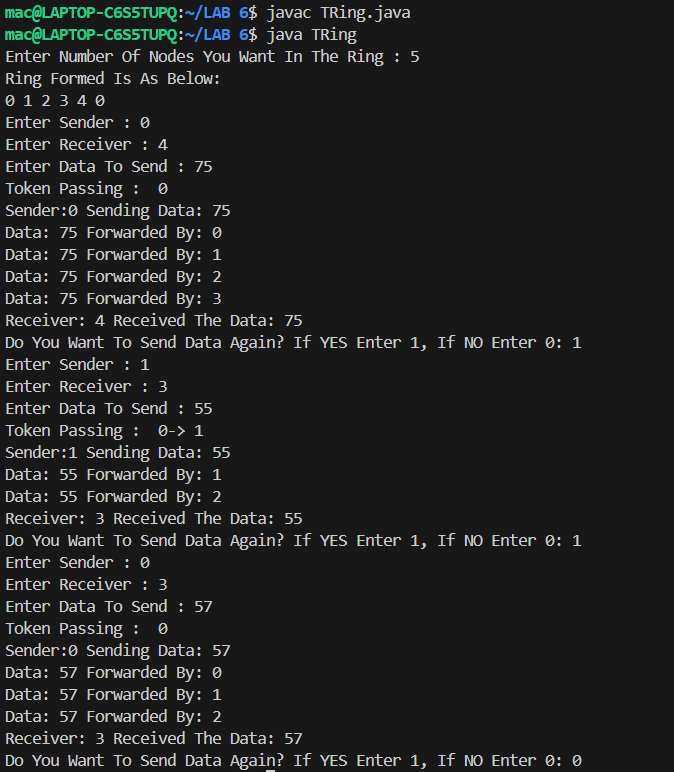
            choice = sc.nextInt();

        }while(choice == 1);

    }

}

**Output-**

****

**Assignment 7**

**Name- Mayank A. Jaiswal**

**Class- BE-A**

**Roll No.- B511042**

**Input-**

**Server**

**MyCalculatorWebService.java**

/\*

 \* To change this license header, choose License Headers in Project Properties.

 \* To change this template file, choose Tools | Templates

 \* and open the template in the editor.

 \*/

package com.myservice;

import javax.jws.WebService;

import javax.jws.WebMethod;

import javax.jws.WebParam;

/\*\*

 \*

 \* @author YOGESH

 \*/

@WebService(serviceName = "MyCalculatorWebService")

public class MyCalculatorWebService {

    /\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "addition")

    public double addition(@WebParam(name = "num1") double num1, @WebParam(name = "num2") double num2) {

        //TODO write your implementation code here:

        return num1 + num2;

    }

    /\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "subtraction")

    public double subtraction(@WebParam(name = "num1") double num1, @WebParam(name = "num2") double num2) {

        //TODO write your implementation code here:

        return num1 - num2;

    }

    /\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "multiplication")

    public double multiplication(@WebParam(name = "num1") double num1, @WebParam(name = "num2") double num2) {

        //TODO write your implementation code here:

        return num1 \* num2;

    }

    /\*\*

     \* Web service operation

     \*/

    @WebMethod(operationName = "division")

    public double division(@WebParam(name = "num1") double num1, @WebParam(name = "num2") double num2) {

        //TODO write your implementation code here:

        return num1 / num2;

    }

}

**Client**

<!DOCTYPE html>

<!--

To change this license header, choose License Headers in Project Properties.

To change this template file, choose Tools | Templates

and open the template in the editor.

-->

<html>

    <head>

        <title>Calculator Web Service Client</title>

        <meta charset="UTF-8">

        <meta name="viewport" content="width=device-width, initial-scale=1.0">

       <style>

           input{

               width: 200px;

               height: 30px;

               padding: 5px;

               border: 1px solid black;

               border-radius: 5px;

               color: green;

               font-size: 25px;

           }

           #submitbtn{

               width: 100px;

               height: 30px;

               padding: 5px;

               border: 1px solid black;

               border-radius: 5px;

               color: red;

               font-size: 20px;

           }

        </style>

    </head>

    <body>

        <form action="CalculatorServlet">

            Enter Number-1: <input type="text" name="number1" value=""/> <br>

            Enter Number-2: <input type="text" name="number2" value=""/> <br>

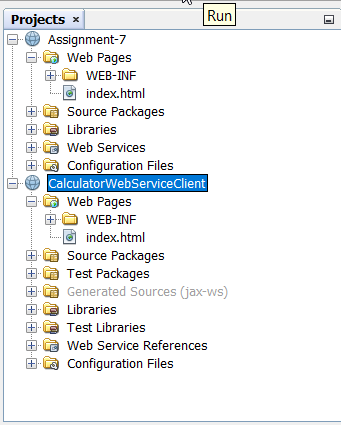
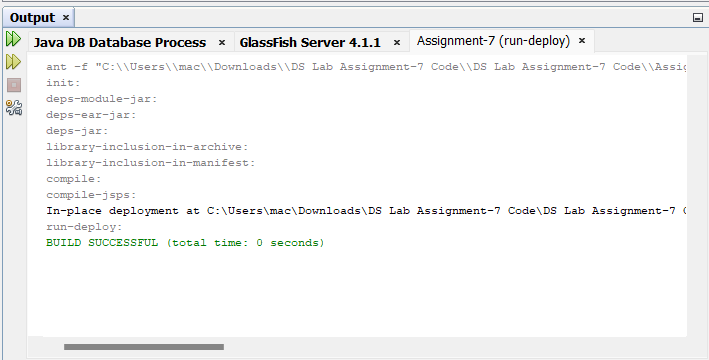
            <input type="submit" value="Submit" id="submitbtn" />

        </form>

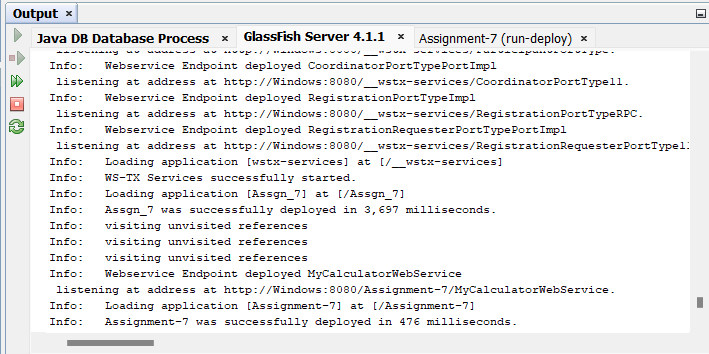
    </body>

</html>

**Outputs-**

 ****

**Fig. 7.1 Folder Structure Fig. 7.2 Server Build Successfully**

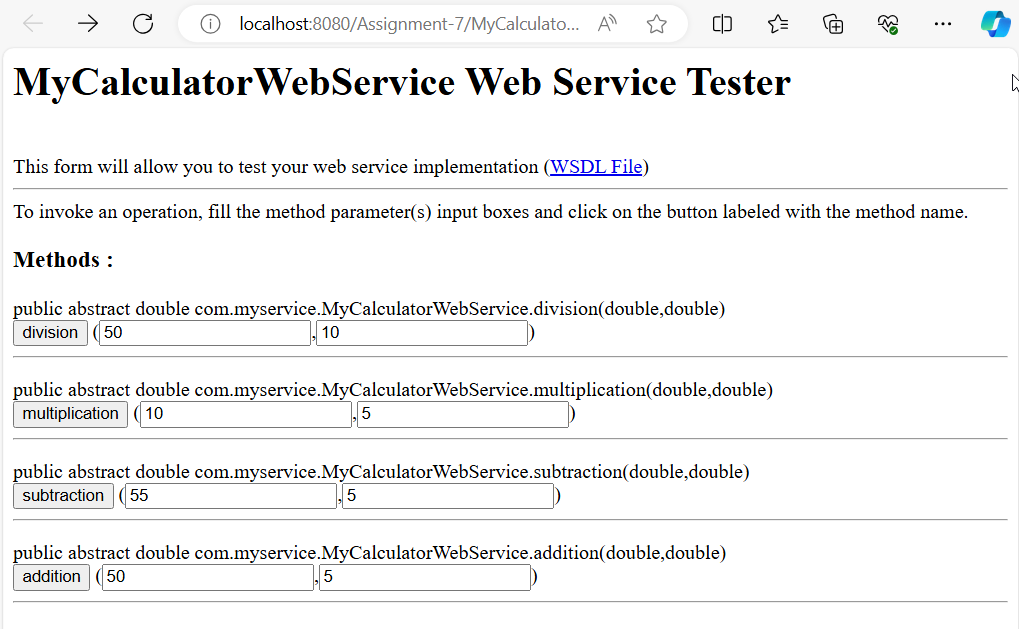
****

**Fig. 7.3 Server Deployed Successfully**

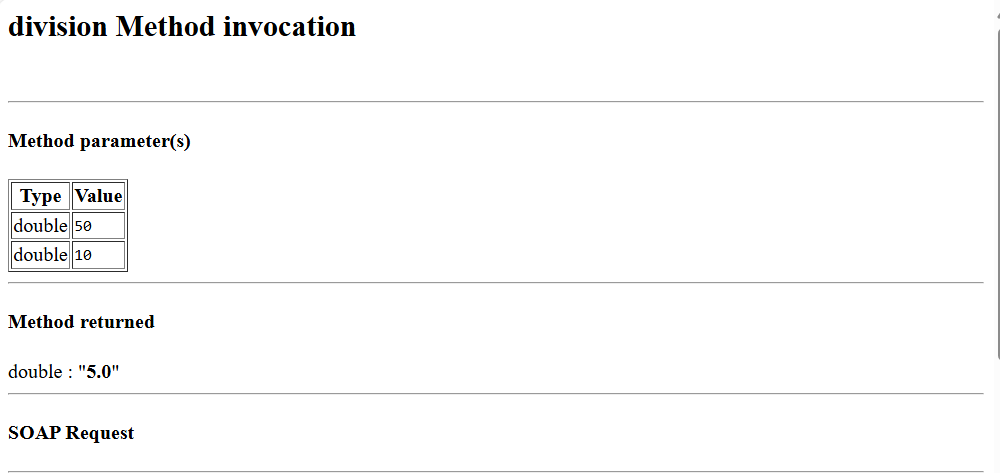
****

**Fig. 7.4 Run MyCalculatorWebService.java**

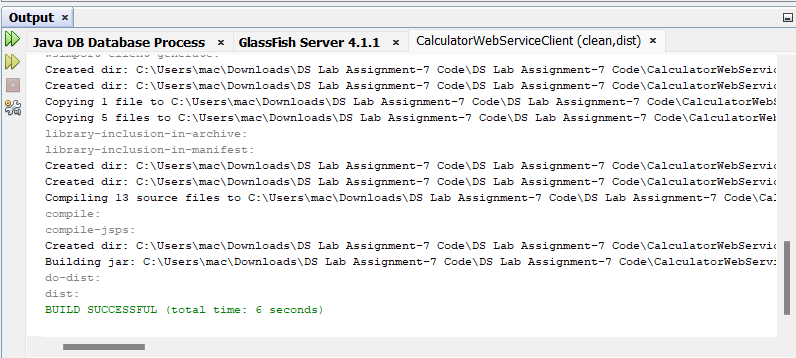
**Code was compiled**

****

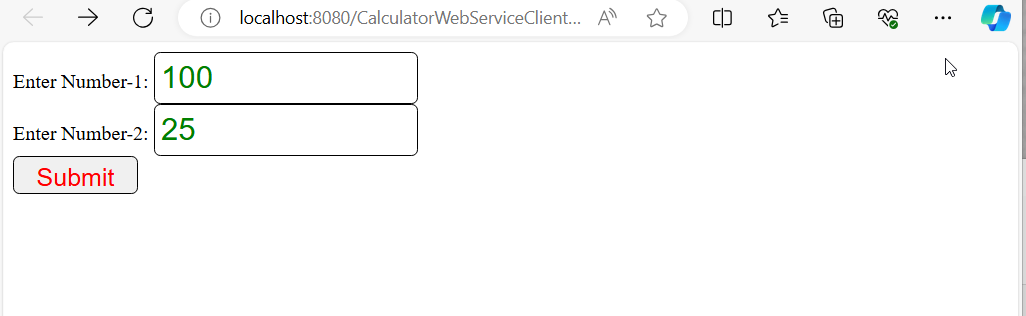
**Fig. 7.5 Calculator Service Deployed Successfully**

****

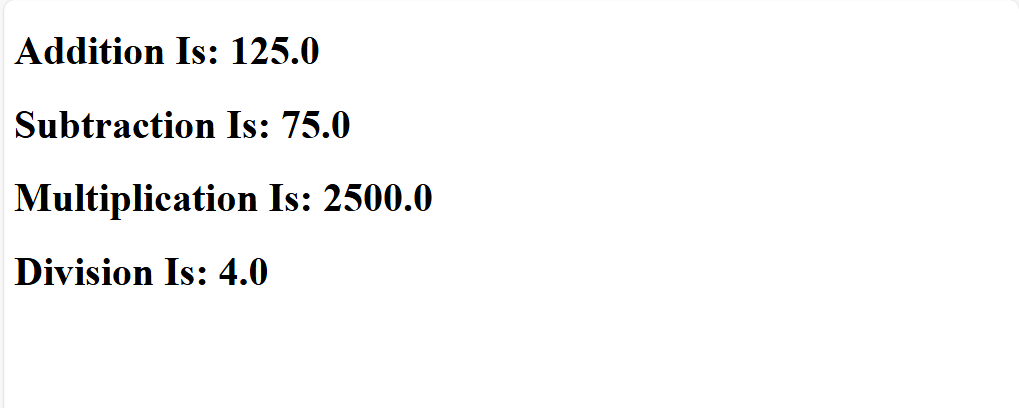
**Fig. 7.6 Calculator Service Results**

****

**Fig. 7.7 Client Web Sevice Bulid Successfully**

****

**Fig. 7.8 Client Deployed on Local Host**

****

**Fig. 7.9 Results**