Ass7

from flask import Flask, request, jsonify

@app.route('/add', methods=['POST'])

def add():

    data = request.get\_json()

    num1 = data['num1']

    num2 = data['num2']

    num3 = num1 + num2

    return jsonify({"result": num3})

@app.route('/multiply', methods=["POST"])

def multiply():

    data = request.get\_json()

    num1 = data['num1']

    num2 = data['num2']

    num3 = num1\*num2

    return jsonify({"result": num3})

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

    app.run(debug=True)

1.app.py

1. pip install flask

Python app.py

2.client.py

Pip install requests

Python client.py

import requests

url = 'http://127.0.0.1:5000/'

def add\_num(num1, num2):

    endpoint = url + 'add'

    data = {"num1": num1, "num2": num2}

    response = requests.post(endpoint, json=data)

    result = response.json()['result']

    return result

def multiply\_num(num1, num2):

    endpoint = url + 'multiply'

    data = {"num1": num1, "num2": num2}

    response = requests.post(endpoint, json=data)

    result = response.json()["result"]

    return result

state = True

while state:

    try:

        print("Enter the first number:")

        num1 = int(input())

        print("Enter the second number:")

        num2 = int(input())

        print("Do you want to:\n1. Add\n2. Multiply\n3. Exit")

        choice = int(input(""))

        if choice == 1:

            print(add\_num(num1, num2))

            print("Do you wish to continue? (Yes, No)")

            if input().lower() == "no":

                state = False

        elif choice == 2:

            print(multiply\_num(num1, num2))

            print("Do you wish to continue? (Yes, No)")

            if input().lower() == "no":

                state = False

        elif choice == 3:

            print("Thank you for using the service")

            state = False

        else:

            print("Invalid Input")

        if state:

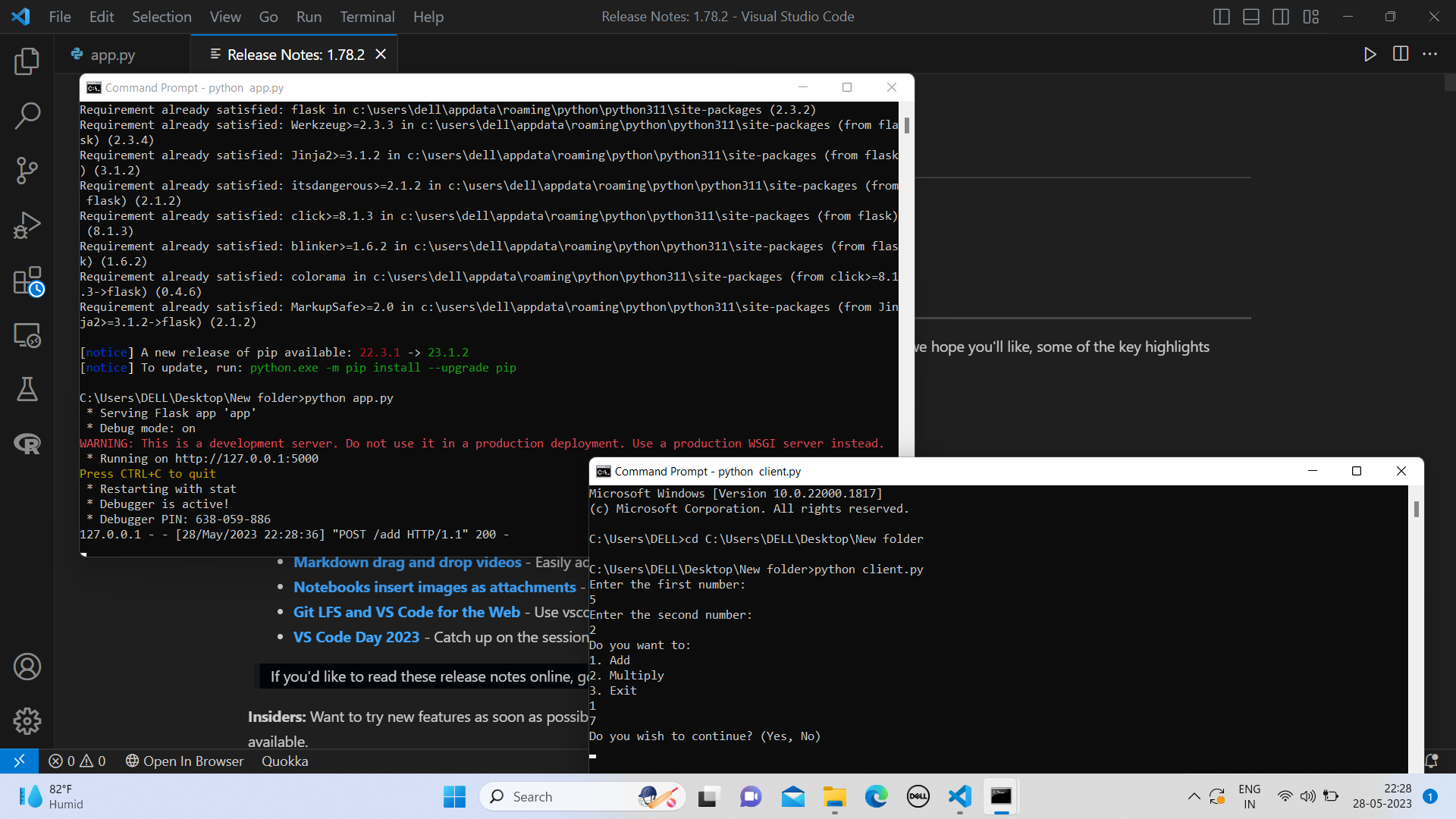
            print("New Calculation")

            print("\_" \* 10, end="\n")

    except Exception as e:

        print("Encountered Error:", str(e))

        print("Restarting interface", end="\n")



Ass 6:1)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("Prccess" + 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);

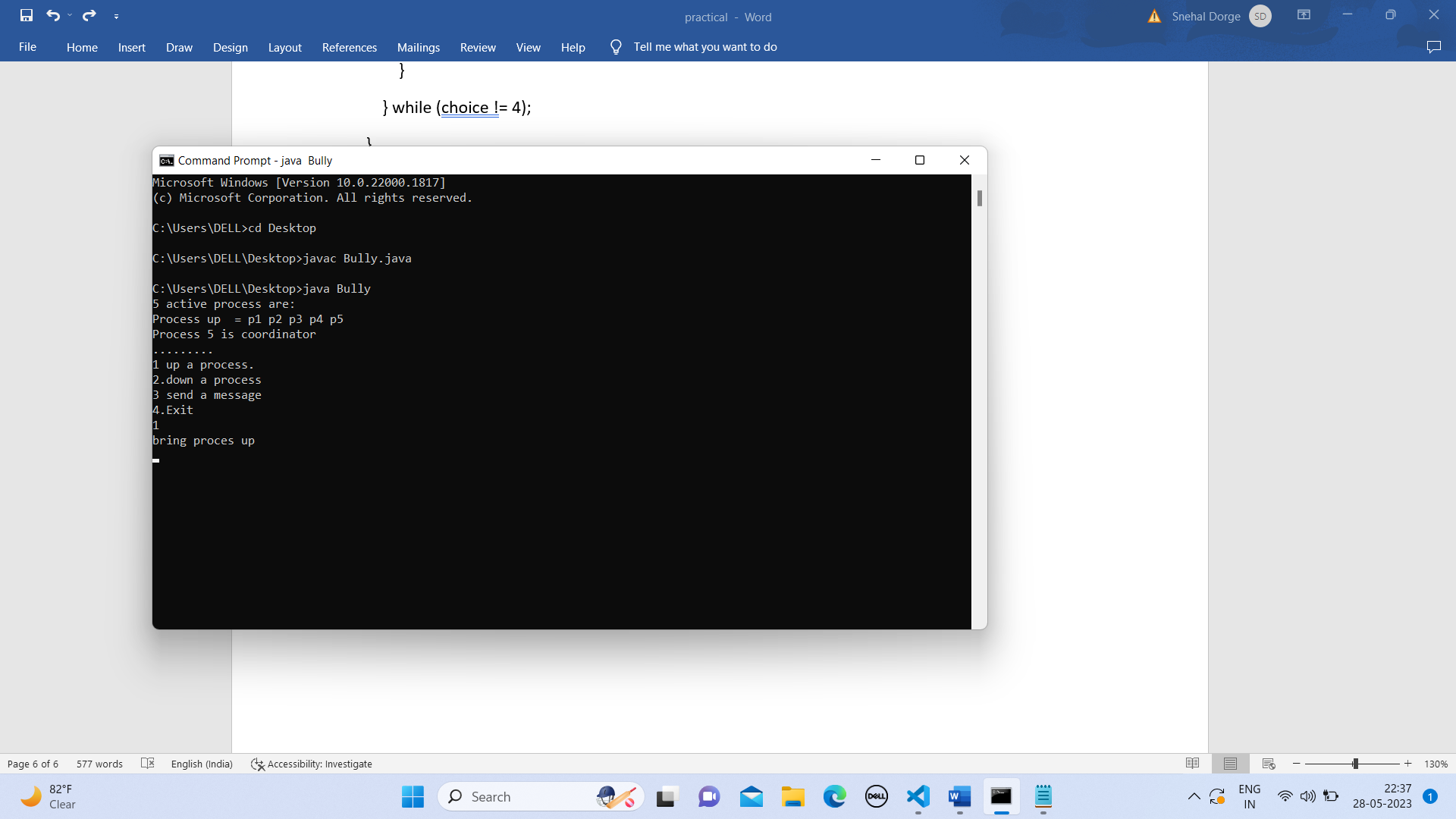
}

}

} while (choice != 4);

}

}



2)Ring.java

import java.util.Scanner;

public class Ring {

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 {

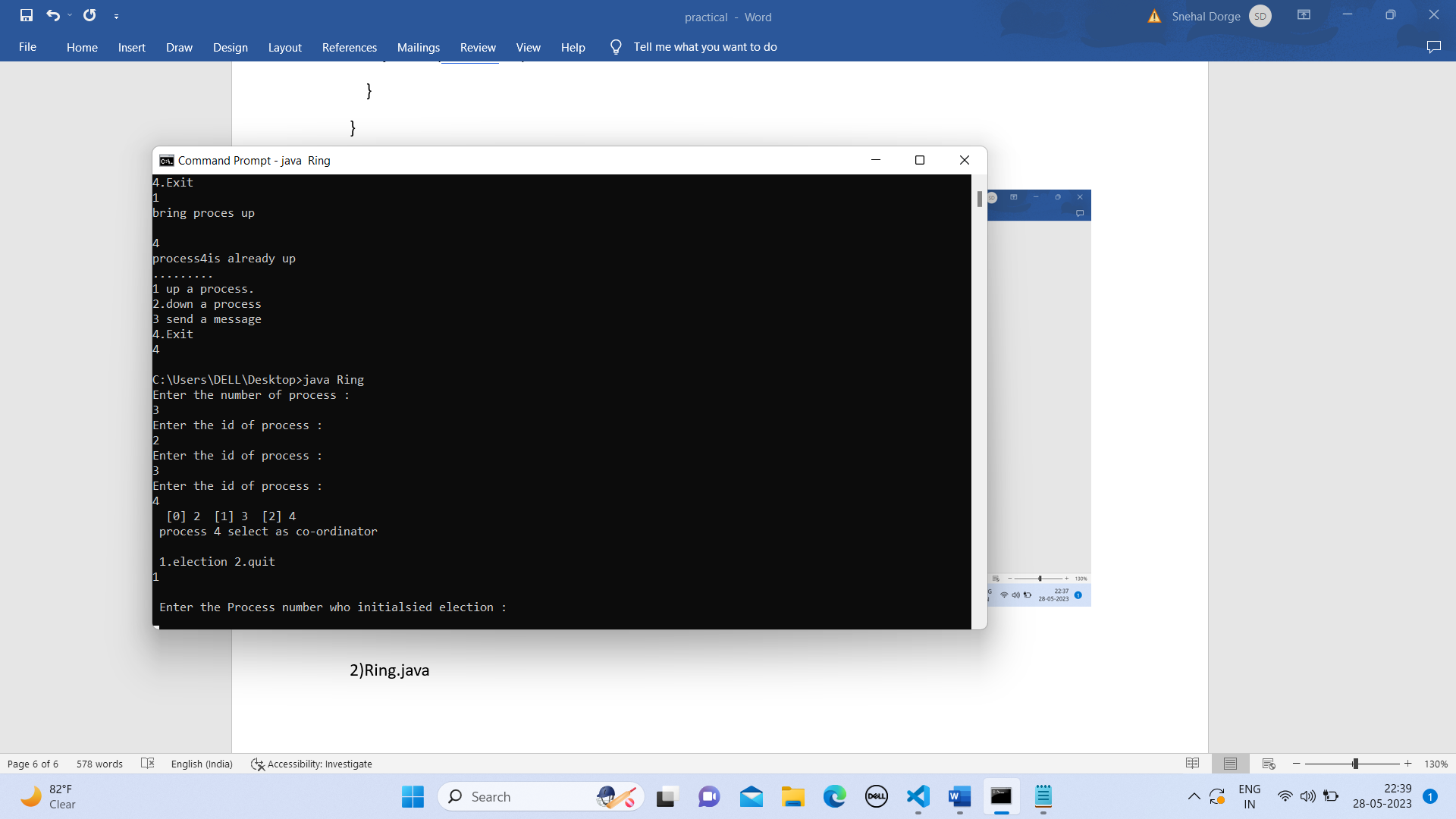
public int index; // to store the index of process

public int id; // to store id/name of process

public int f;

String state; // indiactes whether active or inactive state of node

}



Ass 5

import java.io.\*;

import java.util.\*;

class Tok {

public static void main(String args[]) throws Throwable {

Scanner scan = new Scanner(System.in);

System.out.println("Enter the num of nodes:");

int n = scan.nextInt();

int m = n - 1;

// Decides the number of nodes forming the ring

int token = 0;

int ch = 0, flag = 0;

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

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

}

System.out.println(" " + 0);

do{

System.out.println("Enter sender:");

int s = scan.nextInt();

System.out.println("Enter receiver:");

int r = scan.nextInt();

System.out.println("Enter Data:");

int a;

a = scan.nextInt();

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

for (int i = token, j = token; (i % n) != s; i++, j = (j + 1) % n) {

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

}

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

System.out.println("Sender " + s + " sending data: " + a);

for (int i = s + 1; i != r; i = (i + 1) % n) {

System.out.println("data " + a + " forwarded by " + i);

}

System.out.println("Receiver " + r + " received data: " + a +"\n");

token = s;

do{

try {

if( flag == 1)

System.out.print("Invalid Input!!...");

System.out.print("Do you want to send again?? enter 1 for Yes and 0 for No : ");

ch = scan.nextInt();

if( ch != 1 && ch != 0 )

flag = 1;

else

flag = 0;

} catch (InputMismatchException e){

System.out.println("Invalid Input");

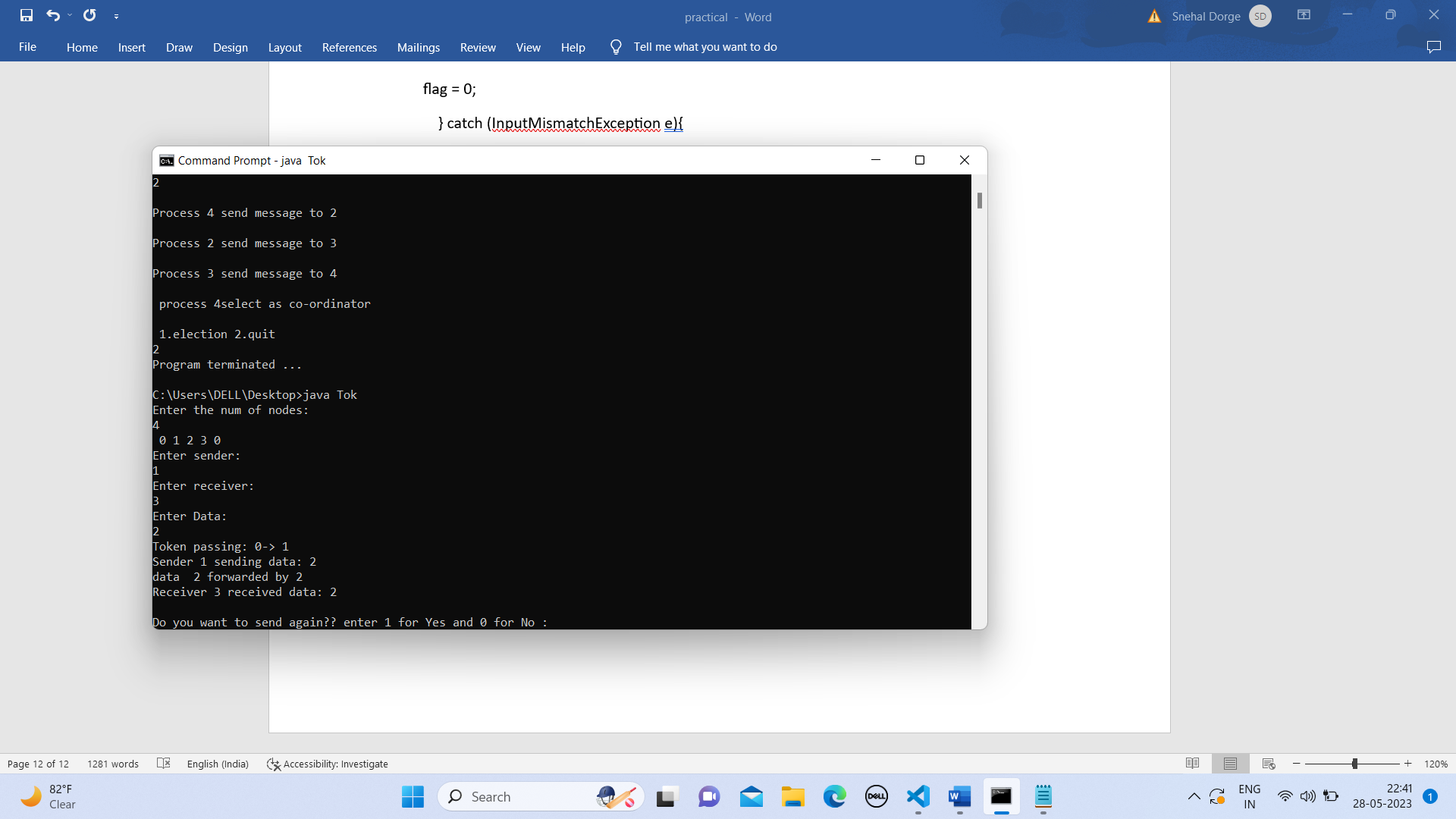
}

}while( ch != 1 && ch != 0 );

}while( ch == 1 );

}

}



Asss 4

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class BerkeleyAlgorithm {

// Define the port number that will be used for communication

private static final int PORT = 1024;

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

// Create a server socket to listen for incoming messages

ServerSocket serverSocket = new ServerSocket(PORT);

// Create a list to store the time differences for each node

List<Long> timeDiffs = new ArrayList<Long>();

// Create a new thread to handle the time requests from nodes

Thread timeServerThread = new Thread(new Runnable() {

public void run() {

while (true) {

try {

// Wait for a node to connect and request the current time

Socket clientSocket = serverSocket.accept();

ObjectInputStream in = new ObjectInputStream(clientSocket.getInputStream());

// Read the current time from the node's request

Date clientTime = (Date) in.readObject();

// Send the current time to the node as a response

ObjectOutputStream out = new ObjectOutputStream(clientSocket.getOutputStream());

out.writeObject(new Date());

// Calculate the time difference between the server and the node

long timeDiff = (new Date().getTime() - clientTime.getTime()) / 2;

timeDiffs.add(timeDiff);

// Close the input/output streams and the socket

in.close();

out.close();

clientSocket.close();

} catch (Exception e) {

e.printStackTrace();

}

}

}

});

timeServerThread.start();

// Create a new thread to periodically send time requests to the server

Thread timeClientThread = new Thread(new Runnable() {

public void run() {

while (true) {

try {

// Connect to the server and send a time request

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

ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());

out.writeObject(new Date());

// Read the current time from the server's response

ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

Date serverTime = (Date) in.readObject();

// Calculate the time difference between the node and the server

long timeDiff = (serverTime.getTime() - new Date().getTime()) / 2;

timeDiffs.add(timeDiff);

// Close the input/output streams and the socket

in.close();

out.close();

socket.close();

// Wait for a short period of time before sending the next time request

Thread.sleep(1000);

} catch (Exception e) {

e.printStackTrace();

}

}

}

});

timeClientThread.start();

// Wait for a sufficient number of time differences to be recorded

Thread.sleep(10000);

// Compute the average time difference and adjust the node's clock

long sumTimeDiff = 0;

for (Long timeDiff : timeDiffs) {

sumTimeDiff += timeDiff;

}

long avgTimeDiff = sumTimeDiff / timeDiffs.size();

System.out.println("Average time difference: " + avgTimeDiff);

// Adjust the node's clock by adding the average time difference

Calendar calendar = Calendar.getInstance();

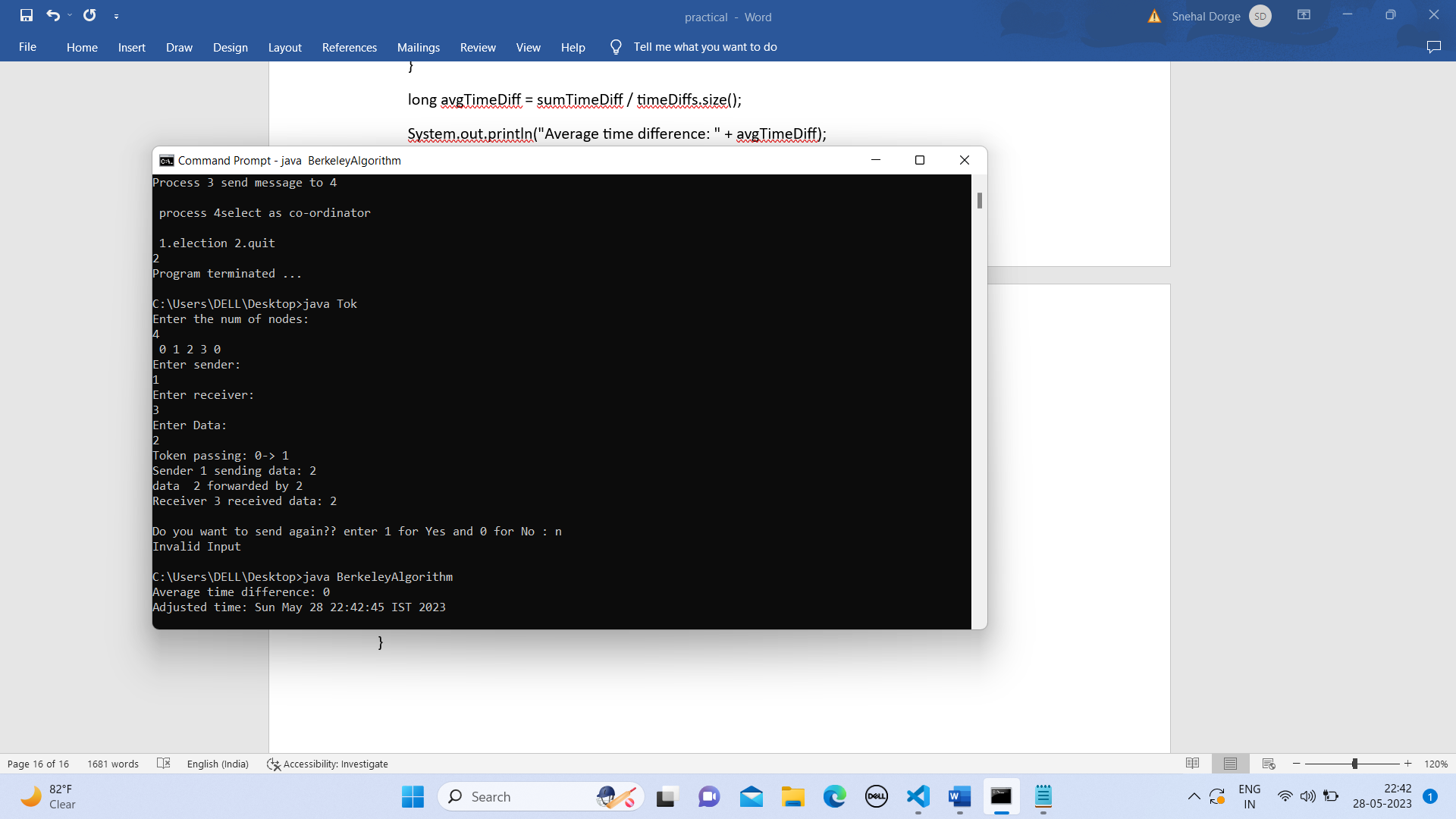
calendar.setTime(new Date());

calendar.add(Calendar.MILLISECOND, (int) avgTimeDiff);

System.out.println("Adjusted time: " + calendar.getTime());

}

}



Asss 3

from mpi4py import MPI

import numpy as np

comm = MPI.COMM\_WORLD

rank = comm.rank

send\_buf = []

if rank == 0:

    arr =np.array([12,21241,5131,1612251,161,6,161,1613,161363,12616,367,8363])

    arr.shape = (3, 4)

    send\_buf = arr

v = comm.scatter(send\_buf, root=0)

print("Local sum at rank{0}: {1}".format(comm.rank, np.sum(v)))

recvbuf = comm.reduce(v, root=0)

if comm.rank == 0:

    global\_sum = np.sum(recvbuf)

    print("Global sum: "+ str(global\_sum))

install mpi4py

set environment

check istall in successful using mpiexec -help

1)pip install mpi4py

2)mpiexec -np 3 python sum.py

Ass 2

Calc.idl

module CalcApp {  
 interface Calc {  
 exception DivisionByZero {};  
 float sum(in float a, in float b);  
 float div(in float a, in float b)raises (DivisionByZero);  
 float mul(in float a, in float b);  
 float sub(in float a, in float b);  
 };  
};

CalcImpl.java

import CalcApp.\*;

 import CalcApp.CalcPackage.DivisionByZero;

  import org.omg.CosNaming.\*;

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

  import org.omg.CORBA.\*;

  import org.omg.PortableServer.\*;

  import java.util.Properties;

 class CalcImpl extends CalcPOA {

 @Override

 public float sum(float a, float b) {

 return a + b;

 }

 @Override

 public float div(float a, float b) throws DivisionByZero {

 if (b == 0) {

 throw new CalcApp.CalcPackage.DivisionByZero();

 } else {

 return a / b;

 }

 }

 @Override

 public float mul(float a, float b) {

 return a \* b;

 }

 @Override

 public float sub(float a, float b) {

 return a - b;

 }

 private ORB orb;

 public void setORB(ORB orb\_val) {

 orb = orb\_val;

 }

 }

 public class CalcServer {

 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

 CalcImpl helloImpl = new CalcImpl();

 helloImpl.setORB(orb);

 // get object reference from the servant

 org.omg.CORBA.Object ref = rootpoa.servant\_to\_reference(helloImpl);

 Calc href = CalcHelper.narrow(ref);

 // get the root naming context

 // NameService invokes the name service

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

 // Use NamingContextExt which is part of the Interoperable

 // Naming Service (INS) specification.

 NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

 // bind the Object Reference in Naming

 String name = "Calc";

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

 ncRef.rebind(path, href);

 System.out.println("Ready..");

 // wait for invocations from clients

 orb.run();

 } catch (Exception e) {

 System.err.println("ERROR: " + e);

 e.printStackTrace(System.out);

 }

 System.out.println("Exiting ...");

 }

 }

CalcClient.java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import CalcApp.\*;

import CalcApp.CalcPackage.DivisionByZero;

import org.omg.CosNaming.\*;

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

import org.omg.CORBA.\*;

import static java.lang.System.out;

public class CalcClient {

    static Calc calcImpl;

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

    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

             String name = "Calc";

             calcImpl = CalcHelper.narrow(ncRef.resolve\_str(name));

             // System.out.println(calcImpl);

             while (true) {

                 out.println("1. Sum");

                 out.println("2. Sub");

                 out.println("3. Mul");

                 out.println("4. Div");

                 out.println("5. exit");

                 out.println("--");

                 out.println("choice: ");

                 try {

                     String opt = br.readLine();

                     if (opt.equals("5")) {

                         break;

                         } else if (opt.equals("1")) {

                         out.println("a+b= " + calcImpl.sum(getFloat("a"), getFloat("b")));

                         } else if (opt.equals("2")) {

                         out.println("a-b= " + calcImpl.sub(getFloat("a"), getFloat("b")));

                         } else if (opt.equals("3")) {

                         out.println("a\*b= " + calcImpl.mul(getFloat("a"), getFloat("b")));

                         } else if (opt.equals("4")) {

                         try {

                             out.println("a/b= " + calcImpl.div(getFloat("a"),

                                    getFloat("b")));

                             } catch (DivisionByZero de) {

                             out.println("Division by zero!!!");

                             }

                         }

                     } catch (Exception e) {

                     out.println("===");

                     out.println("Error with numbers");

                     out.println("===");

                     }

                 out.println("");

                 }

             //calcImpl.shutdown();

             } catch (Exception e) {

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

             e.printStackTrace(System.out);

             }

         }

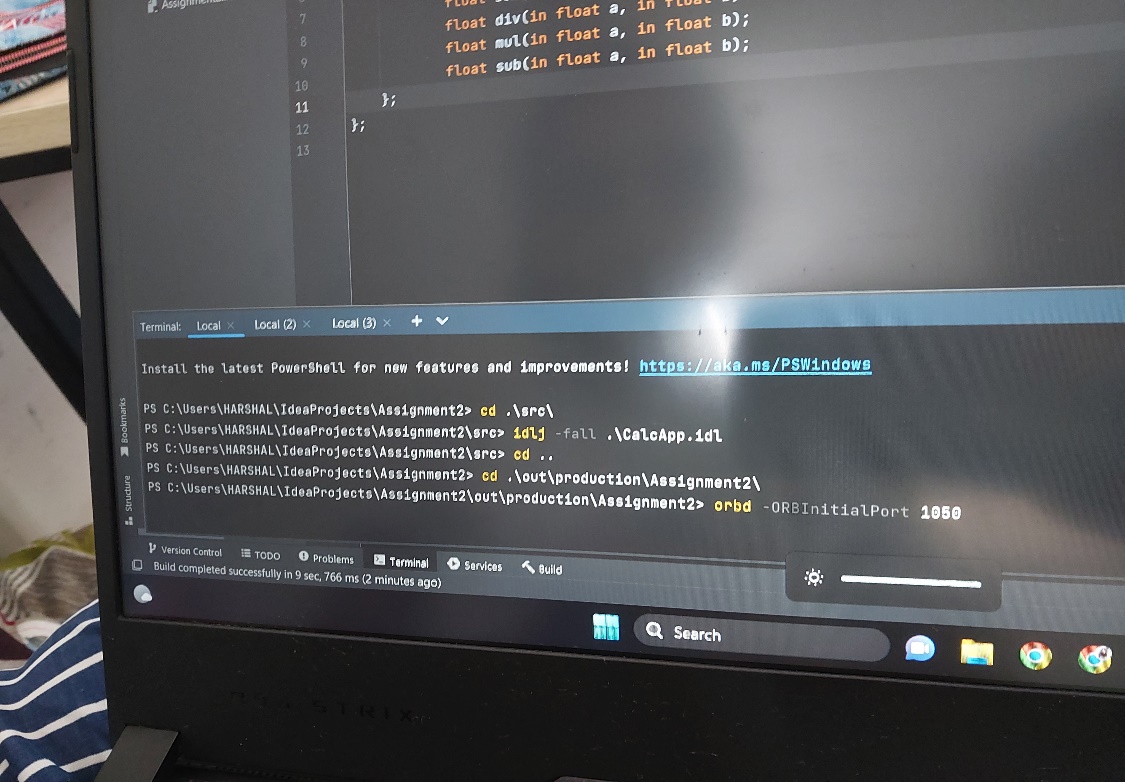
         static float getFloat(String number) throws Exception {

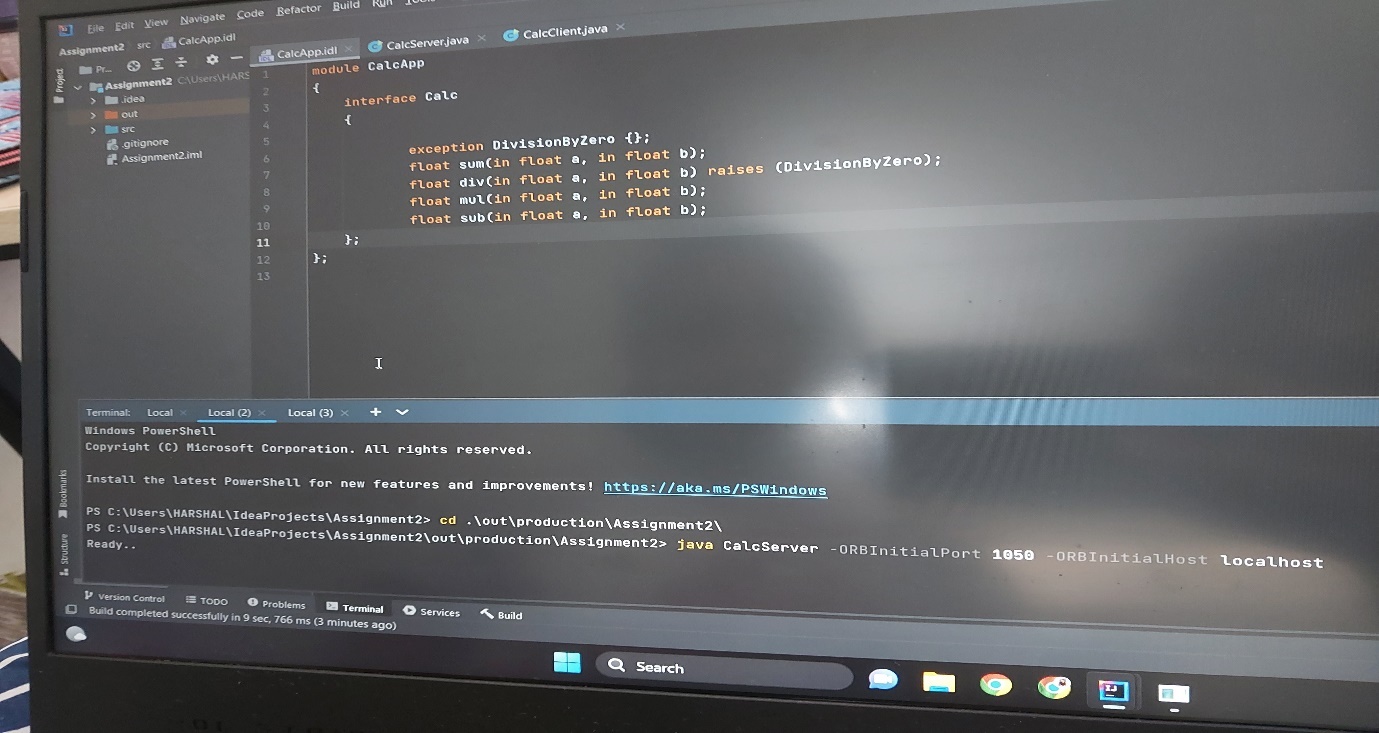
         out.print(number + ": ");

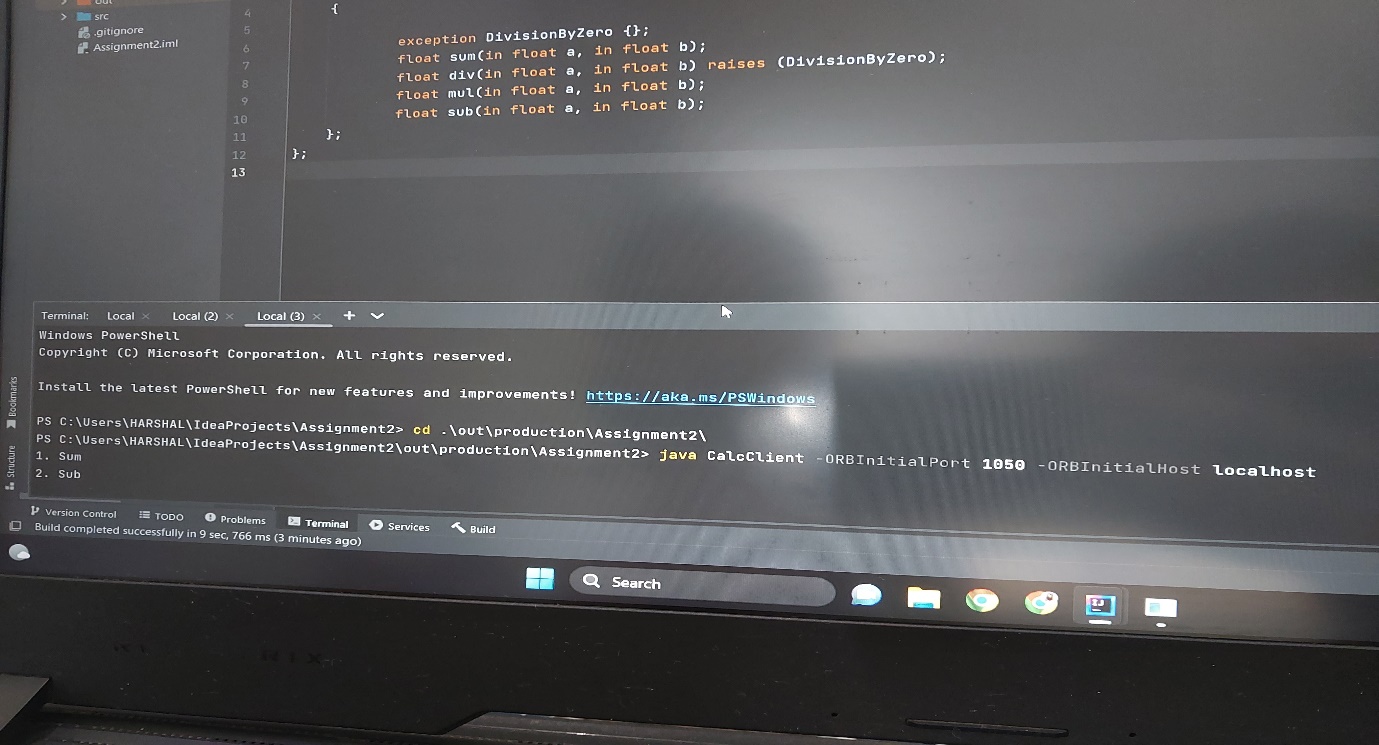
         return Float.parseFloat(br.readLine());

         }

 }







Ass1

RemoteInterface.java

import java.rmi.Remote;

 import java.rmi.RemoteException;

 public interface RemoteInterface extends Remote{

void sendMessage(String message) throws RemoteException;

 String receiveMessage() throws RemoteException;

 }

ServerImpl.java

import java.rmi.Naming;

  import java.rmi.RemoteException;

  import java.rmi.server.UnicastRemoteObject;

  public class ServerImpl extends UnicastRemoteObject implements RemoteInterface{

  String Msg = "Good day, ";

  public ServerImpl() throws RemoteException{

  super();

  }

 @Override

 public void sendMessage(String message) throws RemoteException{

 System.out.println("Message received at server"+message);

 Thread thread = new Thread(() -> {

 try{

 String response = "Response to: " + message;

 Msg+= message;

 System.out.println("Sending response to client "+ response);

 sendResponseToClient(response);

 } catch (RemoteException e){

 System.err.println("Error sending response to client: "+e.getMessage());

 }

 });

 thread.start();

 }

 @Override

 public String receiveMessage() throws RemoteException{

 return this.Msg;

 }

 private void sendResponseToClient(String response) throws RemoteException{

 System.out.println("Response send");

 }

 public static void main(String[] args) {

 try {

 ServerImpl server = new ServerImpl();

 // Bind the server object to the RMI registry

 Naming.rebind("rmi://localhost/Server", server);

 System.out.println("Server running...");

 } catch (Exception e) {

 System.err.println("Server exception: " + e.toString());

 e.printStackTrace();

 }

 }

 }

Client.java

import java.rmi.Naming;

  import java.rmi.RemoteException;

  public class Client {

  public static void main(String[] args){

  try{

  RemoteInterface server = (RemoteInterface)

Naming.lookup("rmi://localhost/Server");

  server.sendMessage("User");

  String response = server.receiveMessage();

  System.out.println("Response from server:" + response);

 }catch(Exception e){

 System.err.println("Client exception: " + e.toString());

 e.printStackTrace();

}

}

}

