LPV Practical Code and Outputs

Assignment No 1

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

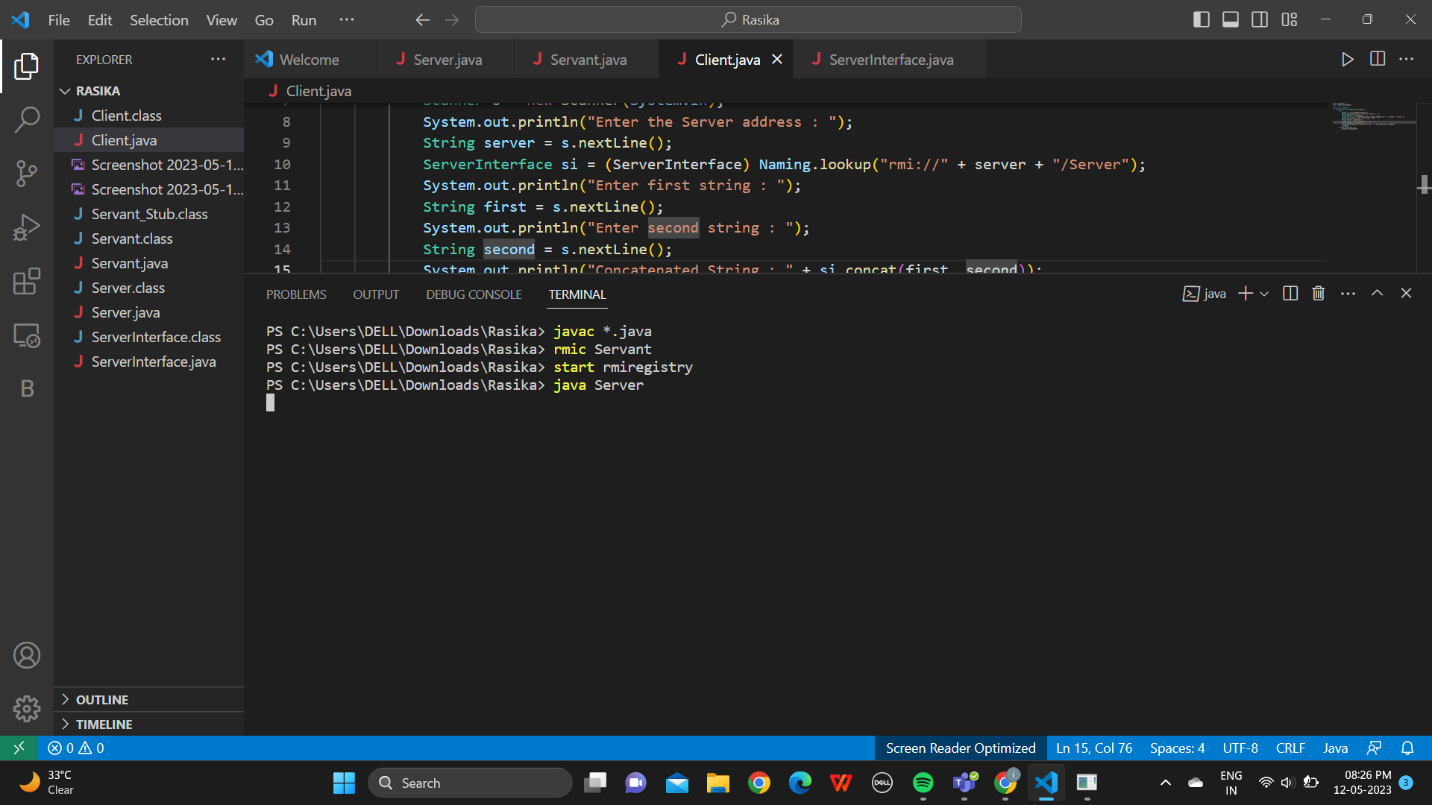
import java.rmi.\*;  
  
public interface ServerInterface extends Remote {  
 String concat(String a, String b) throws RemoteException;  
}

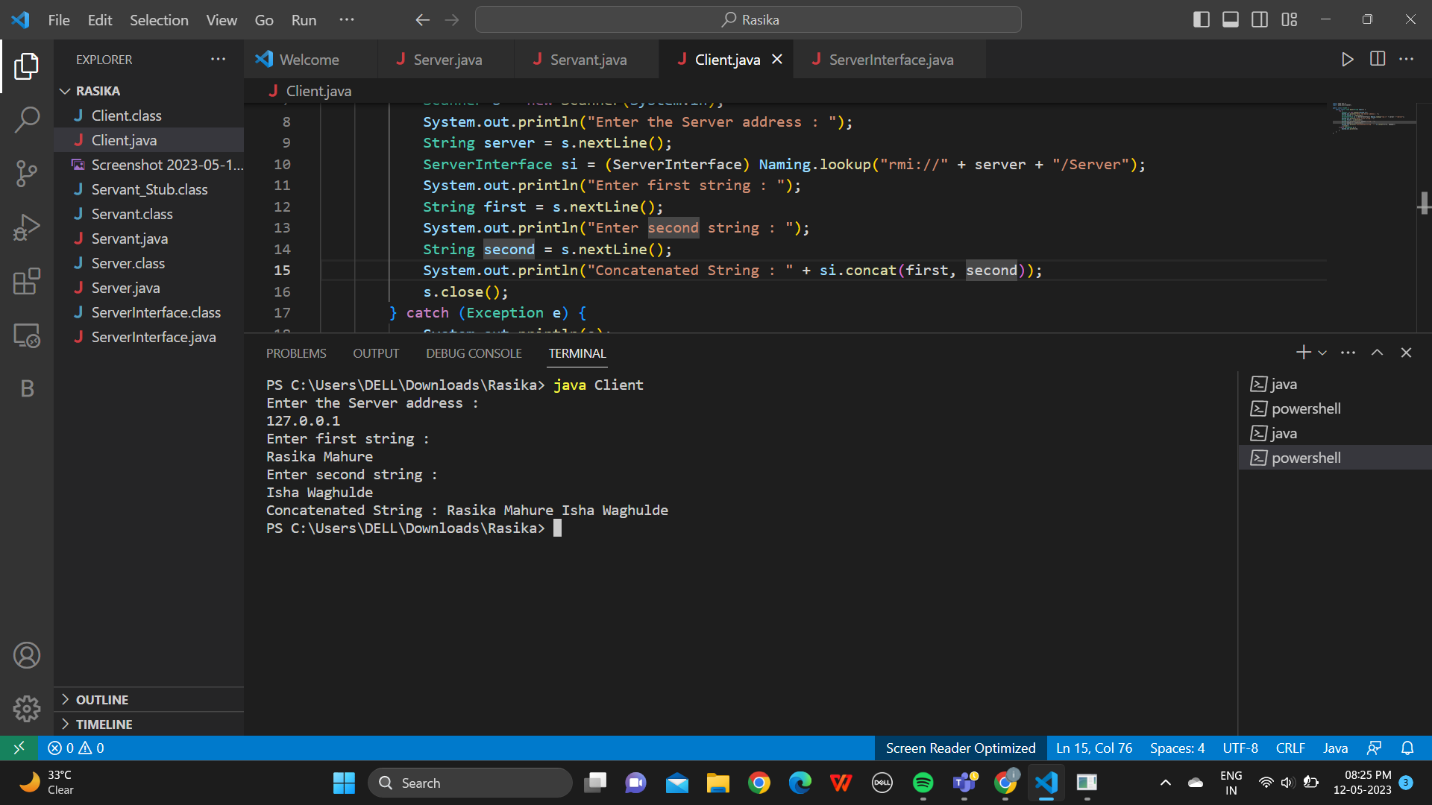
Server.java

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);  
 }  
 }  
}

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);  
 }  
 }  
}





Assignment No 2

1. For example Calc.idl

Include the following code in the idl file

*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);*

*};*

*};*

1. 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("Hello From the server");

            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());

    }

}

1. CalcServer.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);

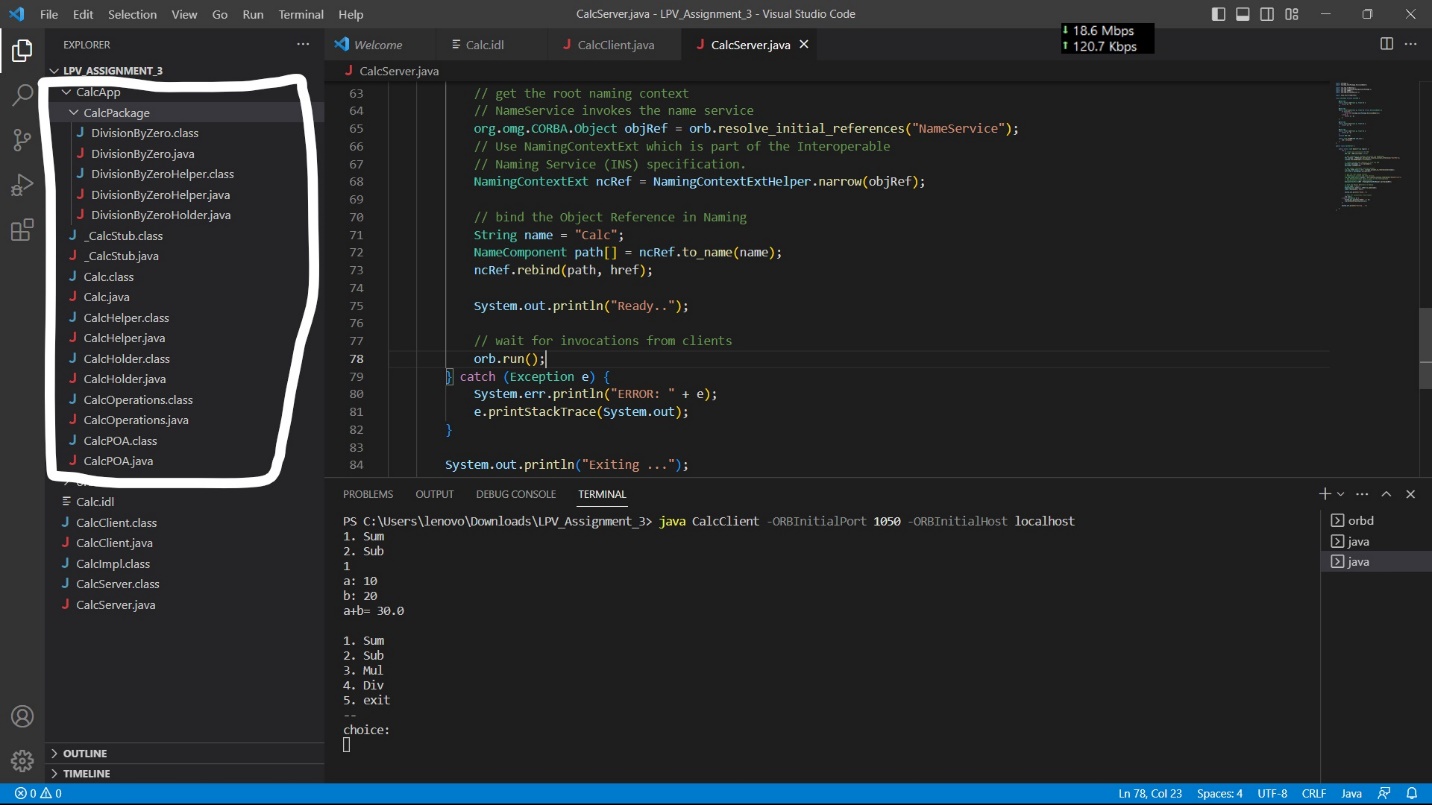
            e.printStackTrace(System.out);

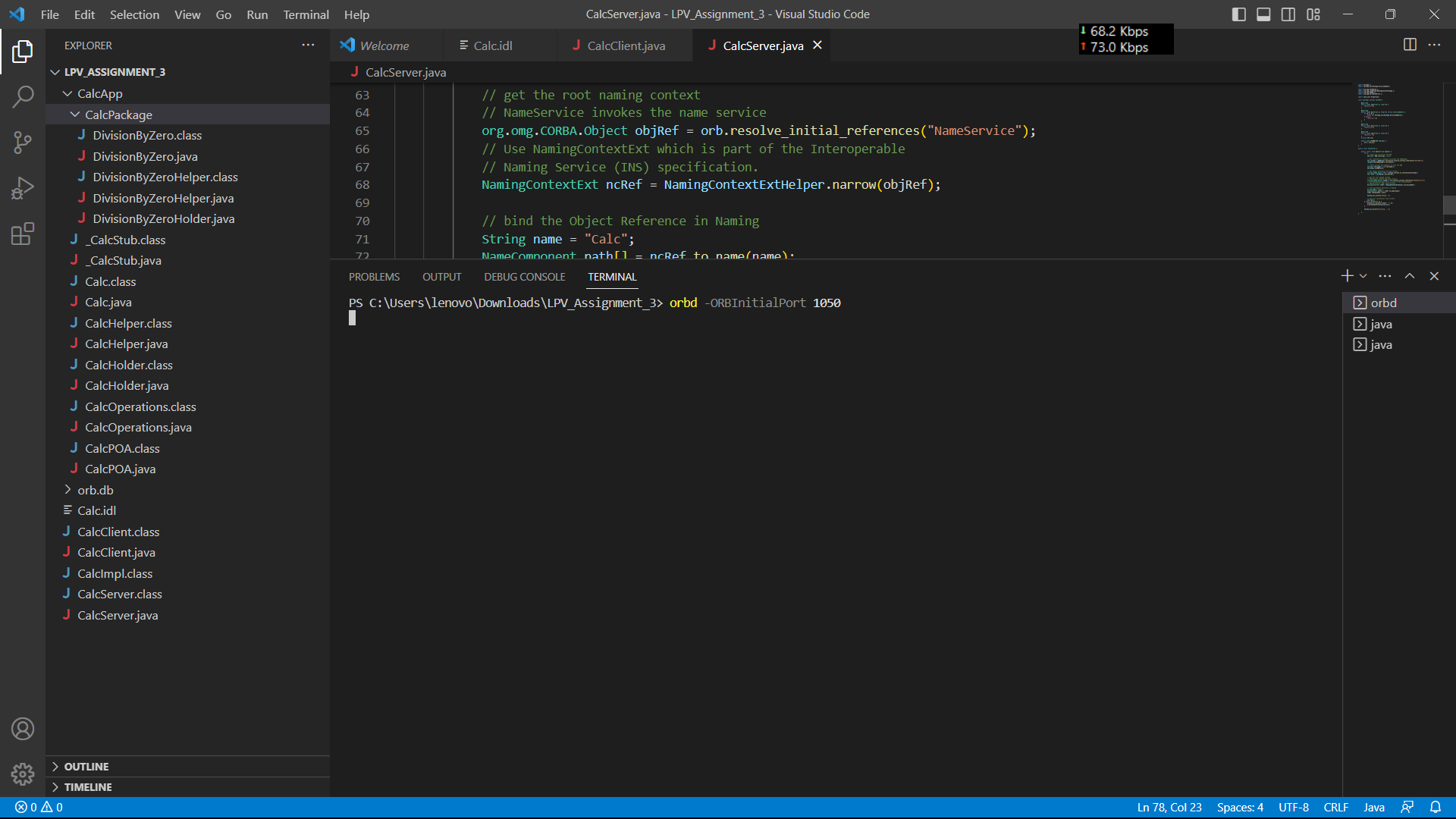
        }

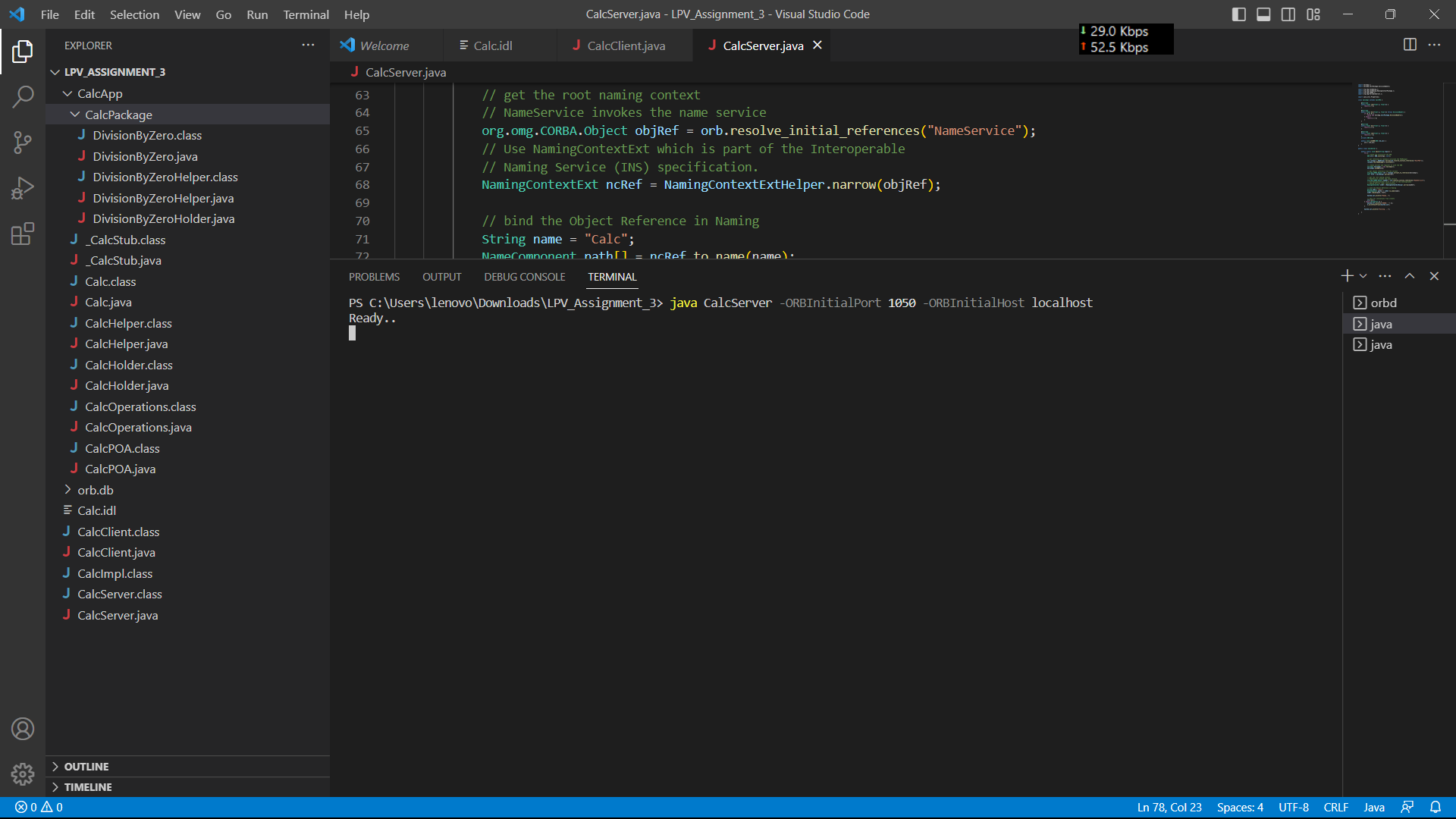
        System.out.println("Exiting ...");

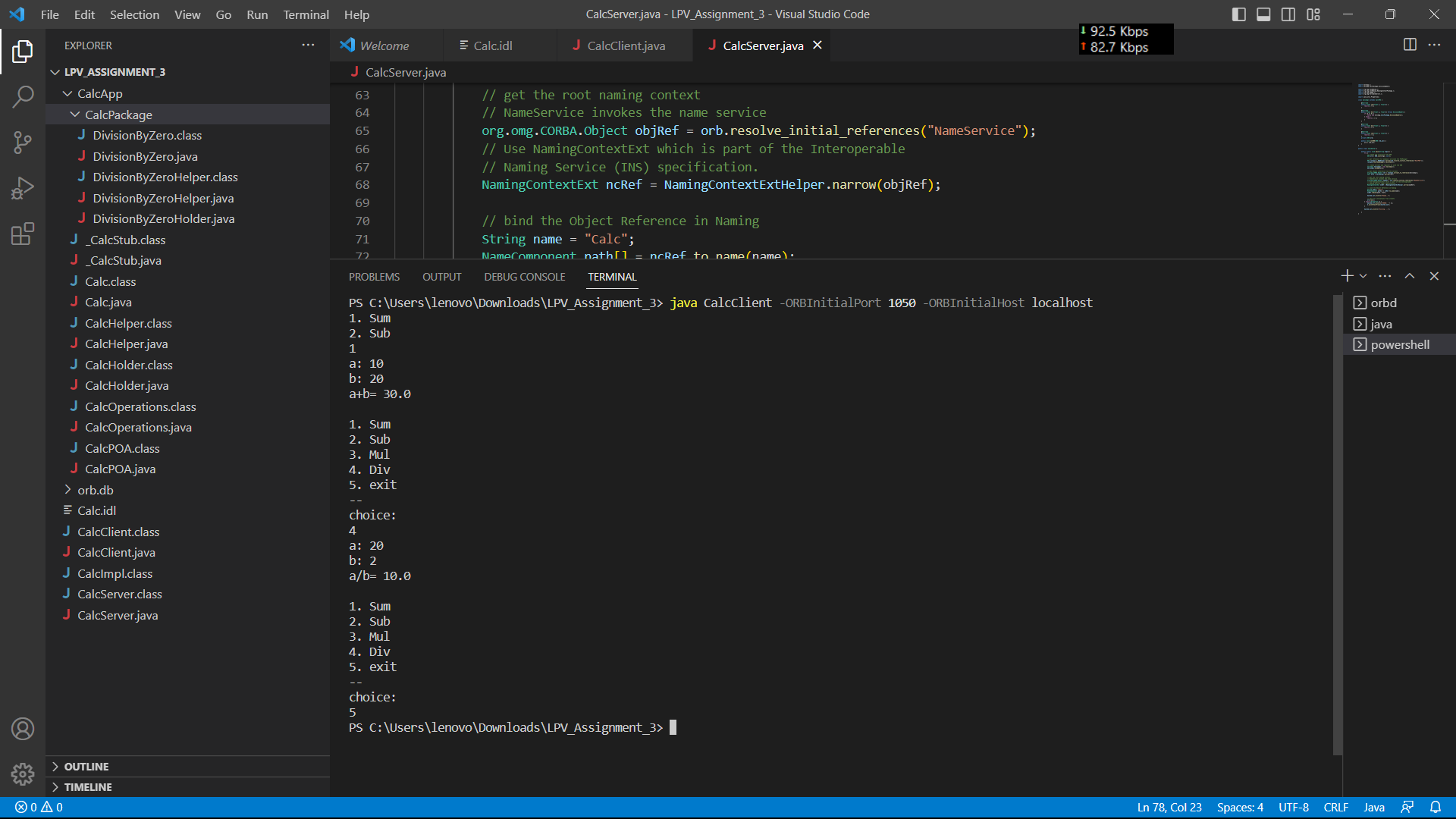
    }

}









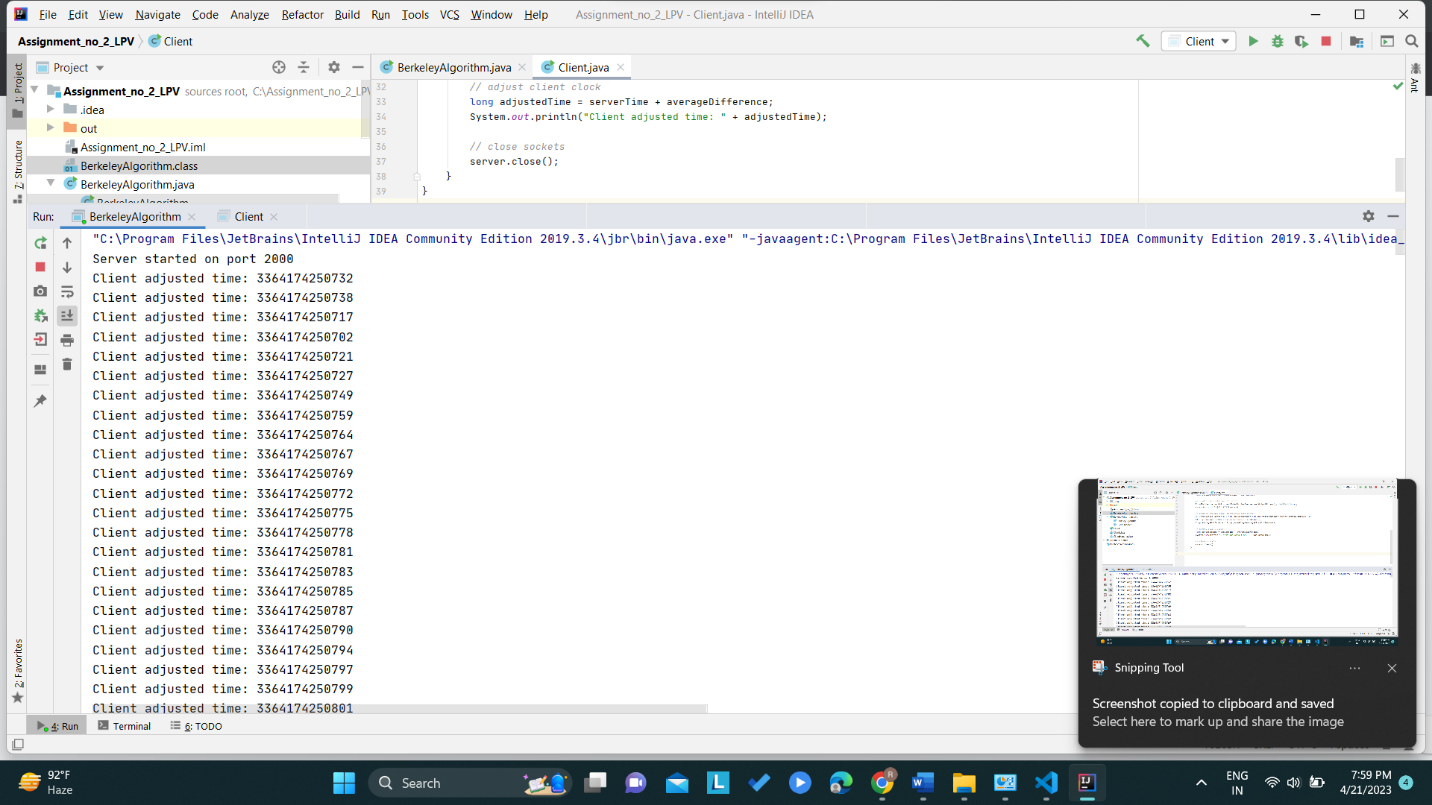
Assignment No 4

Code

import java.net.\*;  
import java.io.\*;  
  
public class BerkeleyAlgorithm {  
 public static void main(String[] args) throws Exception {  
 int port = 2000; *// port number* ServerSocket server = new ServerSocket(port);  
 System.*out*.println("Server started on port " + port);  
  
 while (true) {  
 Socket client = server.accept();  
 new Thread(new ClientHandler(client)).start();  
 }  
 }  
}  
  
class ClientHandler implements Runnable {  
 private Socket client;  
  
 public ClientHandler(Socket client) {  
 this.client = client;  
 }  
  
 public void run() {  
 try {  
 *// receive time request from client* BufferedReader in = new BufferedReader(new InputStreamReader(client.getInputStream()));  
 String request = in.readLine();  
 long requestTime = Long.*parseLong*(request);  
  
 *// send current time to client* long currentTime = System.*currentTimeMillis*();  
 PrintWriter out = new PrintWriter(client.getOutputStream(), true);  
 out.println(currentTime);  
  
 *// calculate clock difference* long clockDifference = currentTime - requestTime;  
  
 *// send clock difference to server* Socket server = new Socket("localhost", 2000);  
 PrintWriter serverOut = new PrintWriter(server.getOutputStream(), true);  
 serverOut.println(clockDifference);  
  
 *// receive average clock difference from server* BufferedReader serverIn = new BufferedReader(new InputStreamReader(server.getInputStream()));  
 String averageClockDifference = serverIn.readLine();  
 long averageDifference = Long.*parseLong*(averageClockDifference);  
  
 *// adjust client clock* long adjustedTime = currentTime + averageDifference;  
 System.*out*.println("Client adjusted time: " + adjustedTime);  
  
 *// close sockets* server.close();  
 client.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

Client

import java.net.\*;  
import java.io.\*;  
  
public class Client {  
 public static void main(String[] args) throws Exception {  
 Socket server = new Socket("localhost", 2000);  
  
 *// get current time* long currentTime = System.*currentTimeMillis*();  
  
 *// send time request to server* PrintWriter out = new PrintWriter(server.getOutputStream(), true);  
 out.println(currentTime);  
  
 *// receive current time from server* BufferedReader in = new BufferedReader(new InputStreamReader(server.getInputStream()));  
 String response = in.readLine();  
 long serverTime = Long.*parseLong*(response);  
  
 *// calculate clock difference* long clockDifference = serverTime - currentTime;  
  
 *// send clock difference to server* PrintWriter serverOut = new PrintWriter(server.getOutputStream(), true);  
 serverOut.println(clockDifference);  
  
 *// receive average clock difference from server* BufferedReader serverIn = new BufferedReader(new InputStreamReader(server.getInputStream()));  
 String averageClockDifference = serverIn.readLine();  
 long averageDifference = Long.*parseLong*(averageClockDifference);  
  
 *// adjust client clock* long adjustedTime = serverTime + averageDifference;  
 System.*out*.println("Client adjusted time: " + adjustedTime);  
  
 *// close sockets* server.close();  
 }  
}

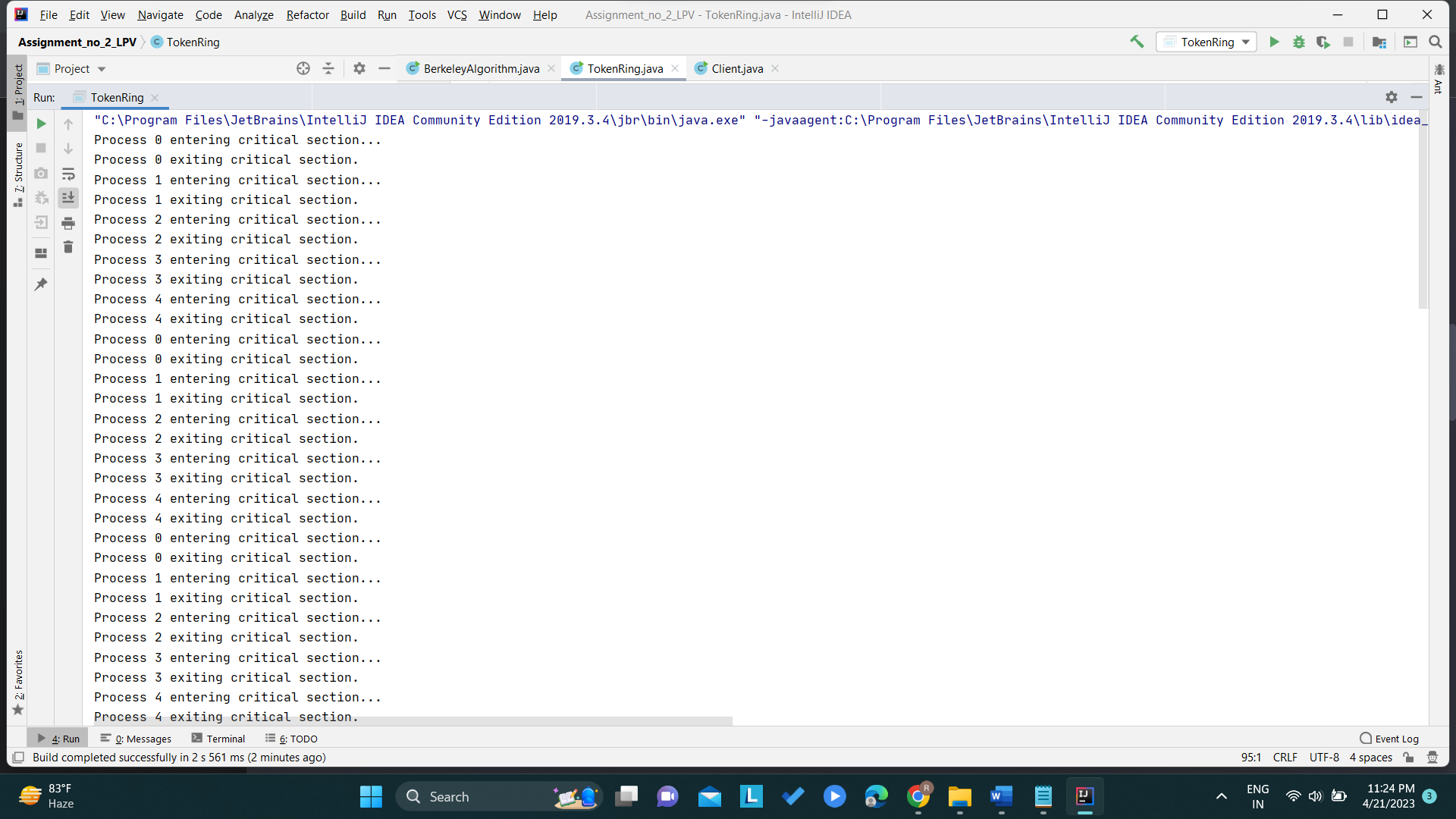


Assignment No 5

Code

import java.util.\*;  
  
public class TokenRing {  
 private static final int *N* = 5; *// Number of processes* private static final int *TOKEN* = -1; *// Token value* private static final int *CS\_TIME* = 1000; *// Critical section time* private static boolean[] *hasToken* = new boolean[*N*]; *// Whether process i has the token* private static boolean[] *inCS* = new boolean[*N*]; *// Whether process i is in the critical section* private static int *tokenHolder* = -1; *// Current token holder* private static void process(int id) throws InterruptedException {  
 while (true) {  
 if (*hasToken*[id]) {  
 *// Enter critical section  
 inCS*[id] = true;  
 System.*out*.println("Process " + id + " entering critical section...");  
 Thread.*sleep*(*CS\_TIME*);  
 System.*out*.println("Process " + id + " exiting critical section.");  
  
 *// Release token  
 hasToken*[id] = false;  
 int nextId = (id + 1) % *N*;  
 *hasToken*[nextId] = true;  
 *tokenHolder* = nextId;  
 } else {  
 *// Wait for token* Thread.*sleep*(100);  
 }  
 }  
 }  
  
 public static void main(String[] args) throws InterruptedException {  
 *// Initialize token holder  
 hasToken*[0] = true;  
 *tokenHolder* = 0;  
  
 *// Start processes* List<Thread> threads = new ArrayList<>();  
 for (int i = 0; i < *N*; i++) {  
 int id = i;  
 Thread thread = new Thread(() -> {  
 try {  
 *process*(id);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 });  
 threads.add(thread);  
 thread.start();  
 }  
  
 *// Wait for processes to finish* for (Thread thread : threads) {  
 thread.join();  
 }  
 }  
}

Output

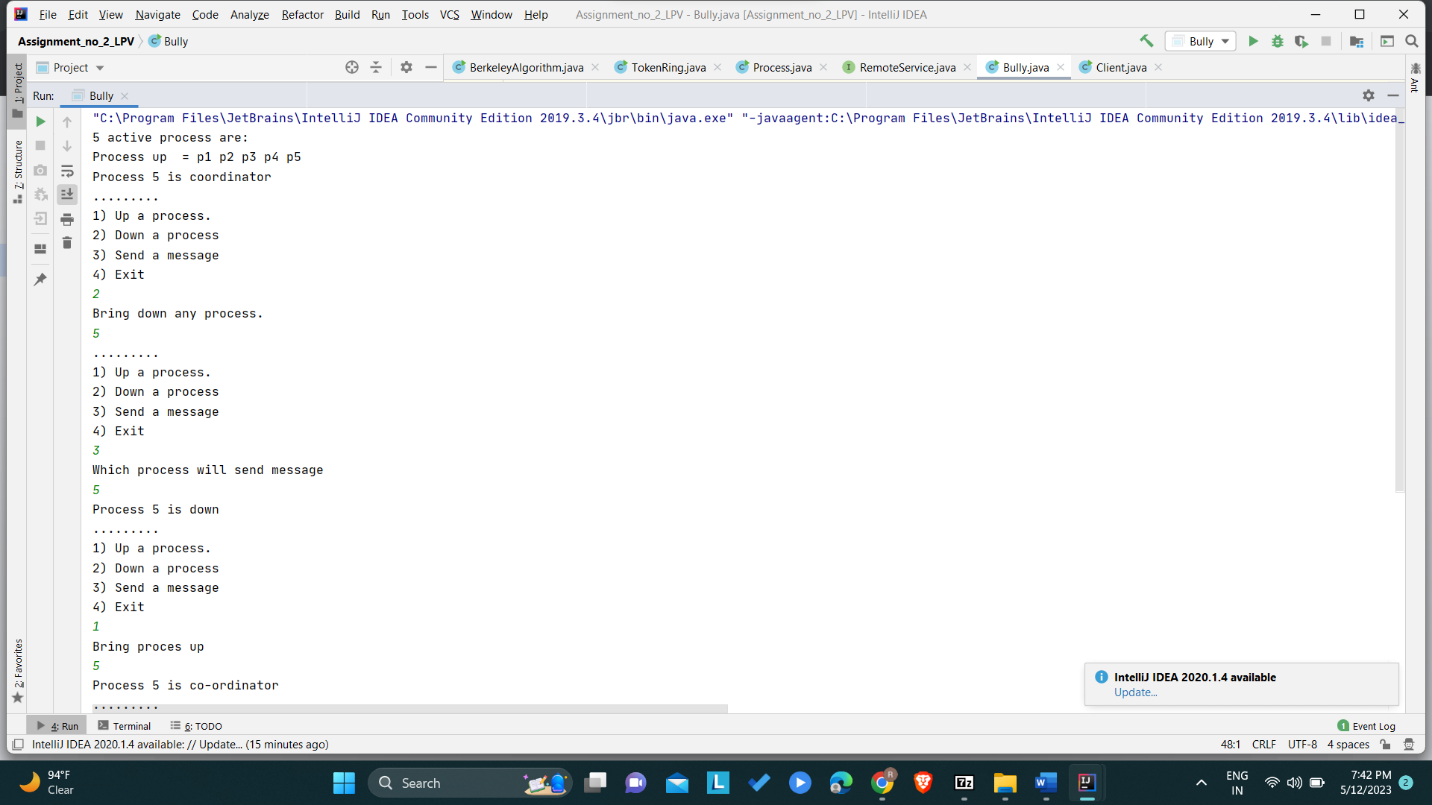


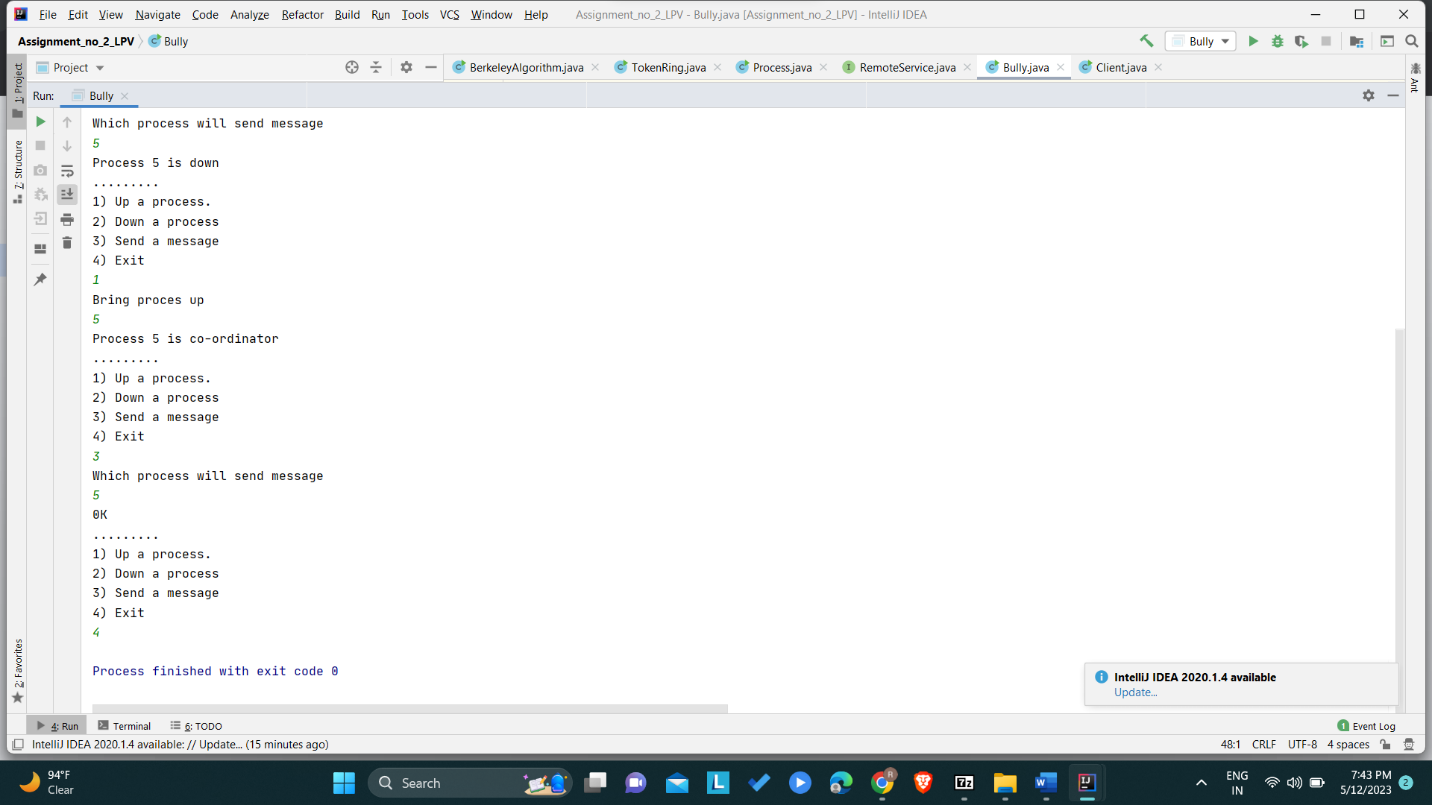
Assignment No 6

Code

1. Bully Algorithm

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();  
  
 }  
}





1. Ring Algorithm
2. 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*}

