[Assignment 1](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)

AddClient.java :

import java.rmi.\*;

public class AddClient {

public static void main(String args[]) {

try {

String addServerURL = "rmi://" + args[0] + "/AddServer";

AddServerIntf addServerIntf =

(AddServerIntf)Naming.lookup(addServerURL);

System.out.println("The first number is: " + args[1]);

double d1 = Double.valueOf(args[1]).doubleValue();

System.out.println("The second number is: " + args[2]);

double d2 = Double.valueOf(args[2]).doubleValue();

System.out.println("The sum is: " + addServerIntf.add(d1, d2));

}

catch(Exception e) {

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

AddServer.java :

import java.net.\*;

import java.rmi.\*;

public class AddServer {

public static void main(String args[]) {

try {

AddServerImpl addServerImpl = new AddServerImpl();

Naming.rebind("AddServer", addServerImpl);

}

catch(Exception e) {

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

AddServerImpl.java :

import java.rmi.\*;

import java.rmi.server.\*;

public class AddServerImpl extends UnicastRemoteObject

implements AddServerIntf {

public AddServerImpl() throws RemoteException {

}

public double add(double d1, double d2) throws RemoteException {

return d1 + d2; } }

AddServerIntf.java :

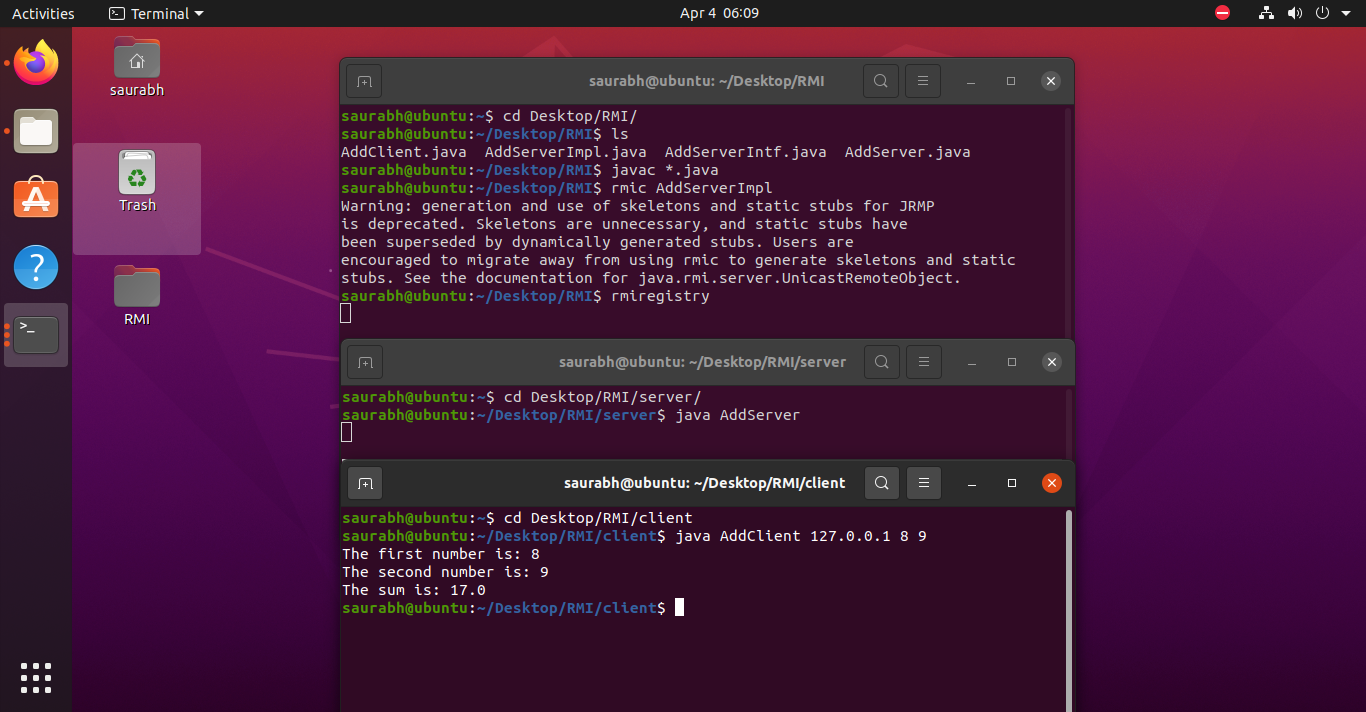
import java.rmi.\*;

public interface AddServerIntf extends Remote {

double add(double d1, double d2) throws RemoteException;

}

Output :



Assignment 2

ReverseClient.java :

// Client

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[])

{

Reverse ReverseImpl=null;

try

{

// initialize the ORB object request broker

org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

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

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

String name = "Reverse";

// narrow converts generic object into string type

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

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();

}

}

}

ReverseImpl.java :

import ReverseModule.ReversePOA;

import java.lang.String;

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));

}

}

ReverseModule.idl :

module ReverseModule

{

interface Reverse

{

string reverse\_string(in string str);

};

};

ReverseServer.java :

import ReverseModule.Reverse;

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

{

// initialize the ORB

org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

// initialize the BOA/POA

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

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

// creating the object

ReverseImpl rvr = new ReverseImpl();

// get the object reference from the servant class

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

System.out.println("Step1");

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

System.out.println("Step2");

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

System.out.println("Step3");

NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);

System.out.println("Step4");

String name = "Reverse";

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

ncRef.rebind(path,h\_ref);

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

orb.run();

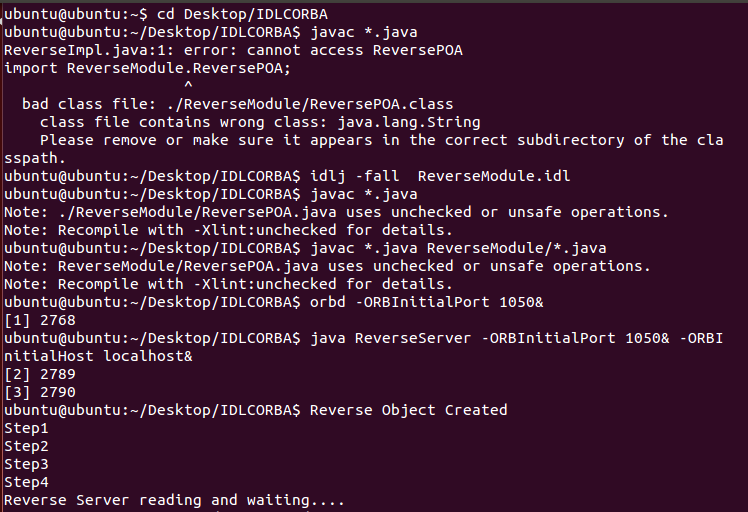
}

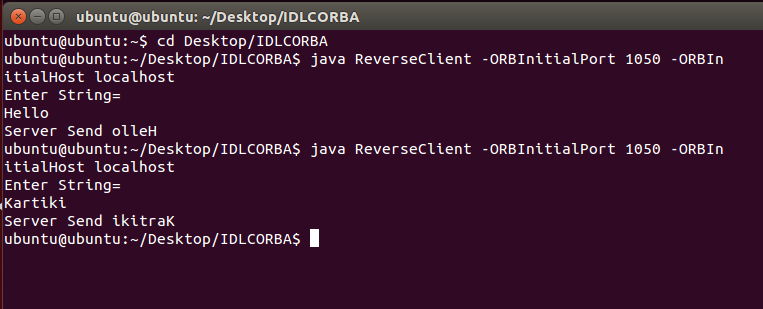
catch(Exception e)

{

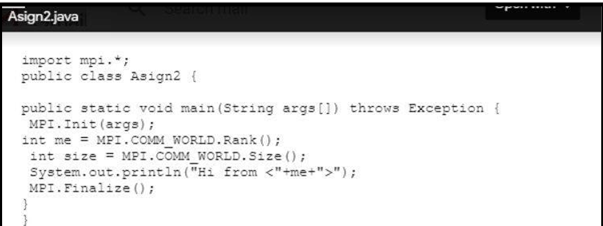
e.printStackTrace(); } } }

Output :





[Assignment 3](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)



[Assignment 4](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)

Berkeley.java :

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Berkeley {

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

private static final int PORT = 9876;

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());

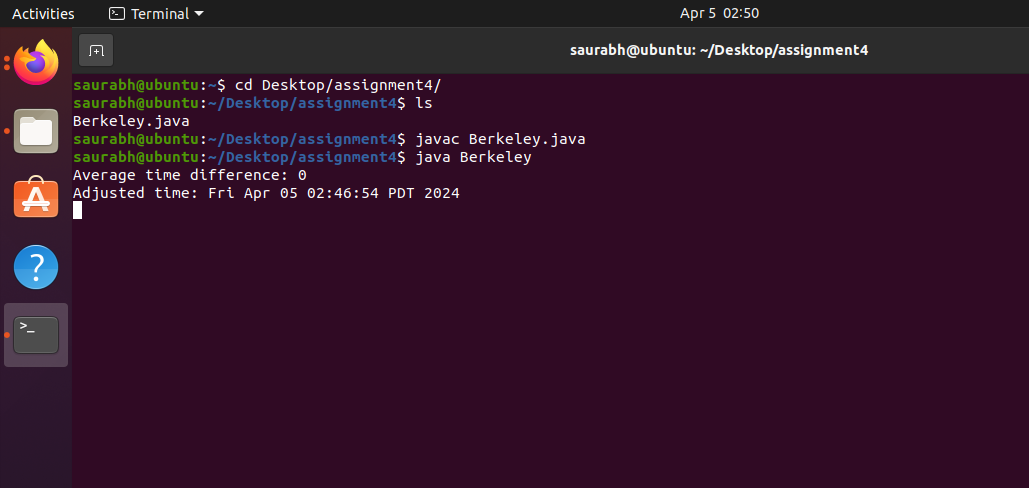
calendar.add(Calendar.MILLISECOND, (int) avgTimeDiff);

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

}

}

Output :



[Assignment 5](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)

Tring.java :

import java.util.Scanner;

class Tring {

public static void main(String args[]) {

Scanner sc = new Scanner(System.in);

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

int n = sc.nextInt();

// Decides the number of nodes forming the ring

int token = 0;

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

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

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

try {

while (true) {

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

int s = sc.nextInt();

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

int r = sc.nextInt();

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

String d = sc.next();

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

//current token not equal to sender, increment i by 1 and j by j+1%n

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: " + d);

// start forwarding from node after sender until it becomes equal to receiver and increment by i+1%n

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

System.out.println("Data " + d + " forwarded by " + i);

}

System.out.println("Receiver " + r + " received data: " + d);

token = s;

}

} catch (Exception e) {

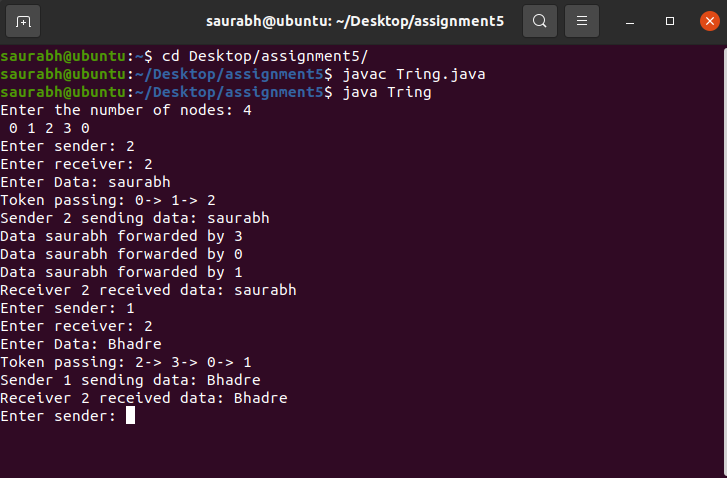
System.out.println("Error occurred: " + e.getMessage());

}

}

}

Output :



[Assignment 6](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)

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;

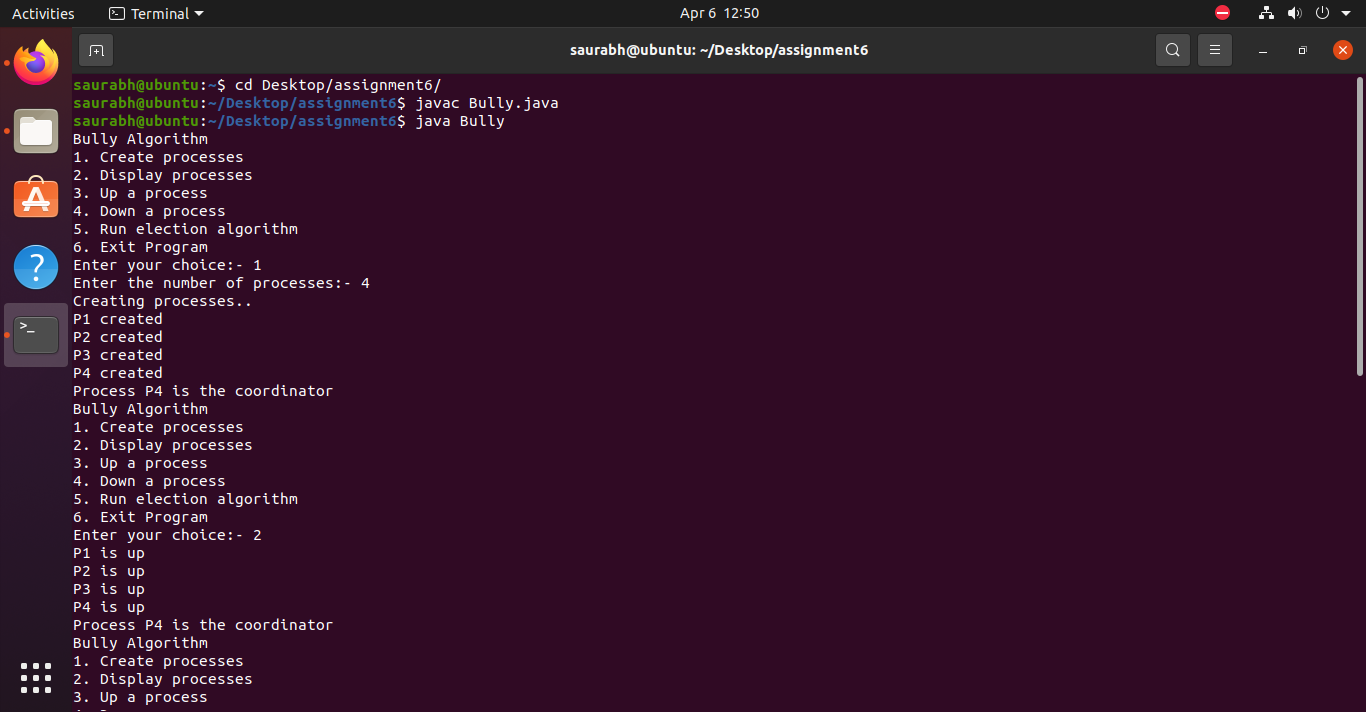
}

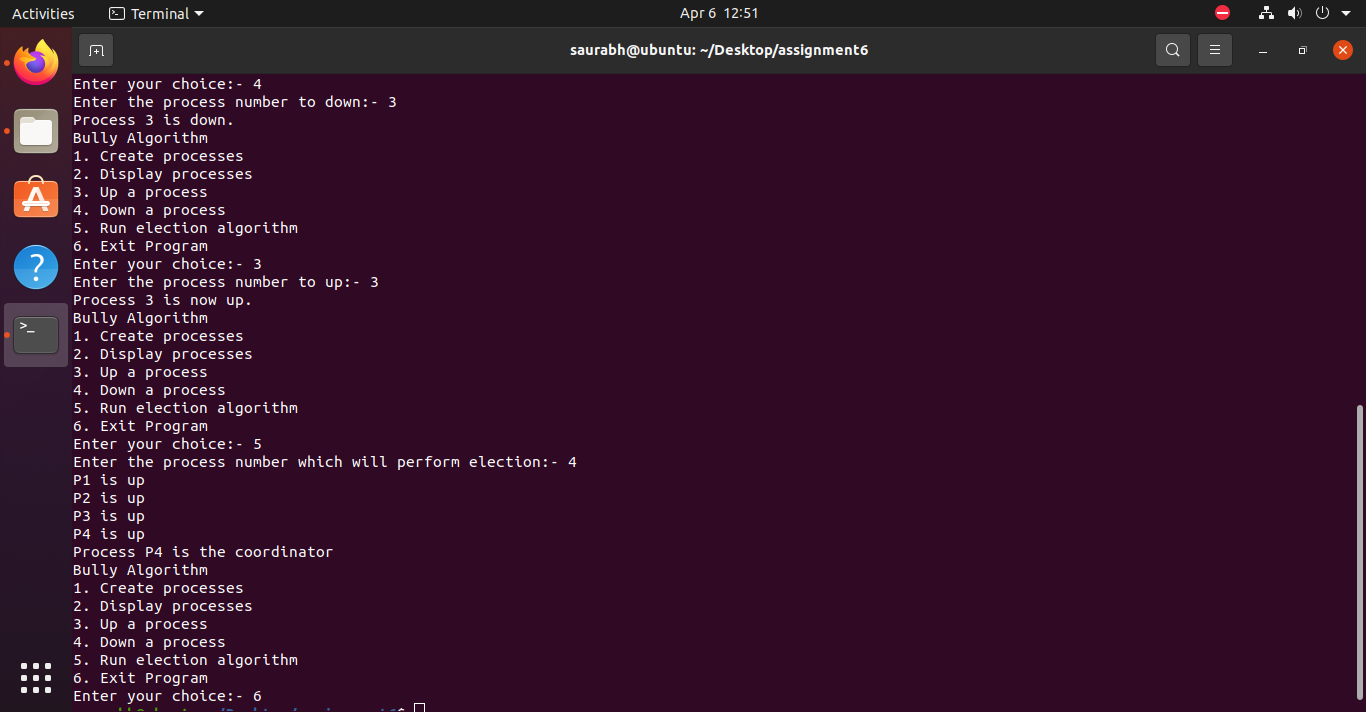
}

}

}

Output :





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);

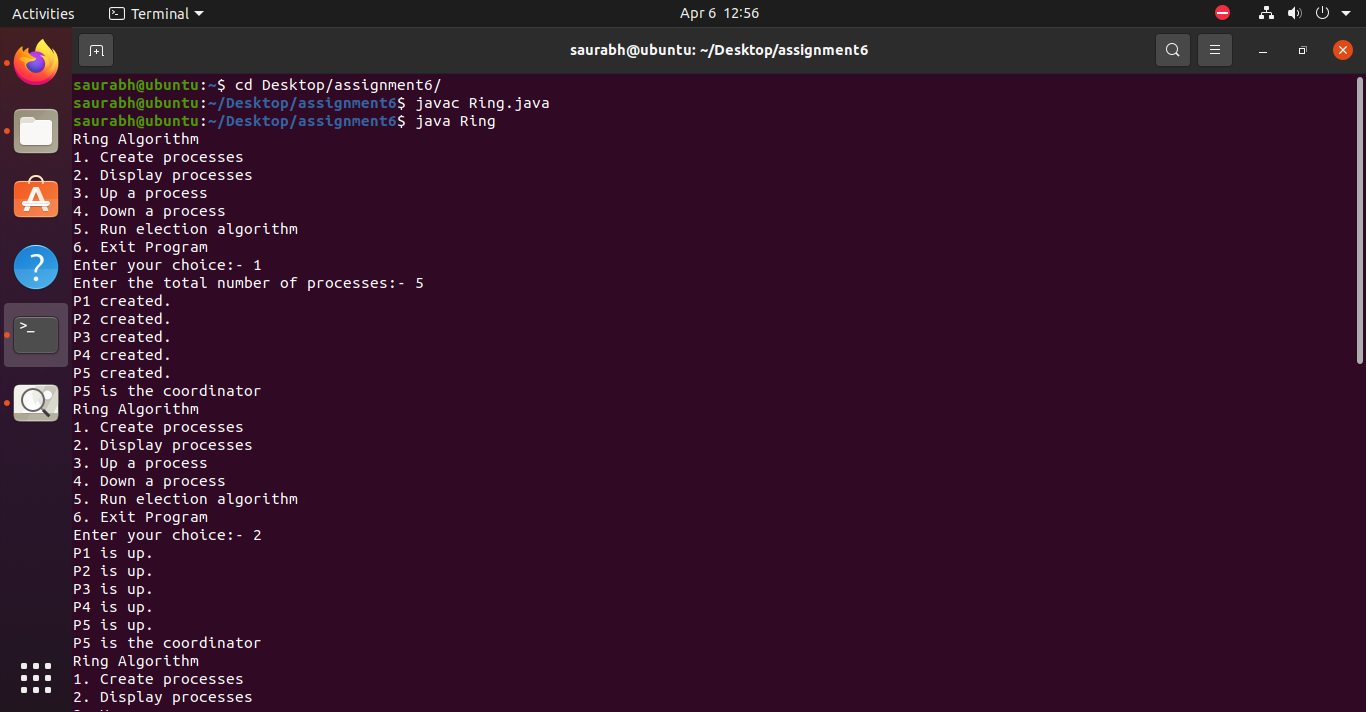
break;

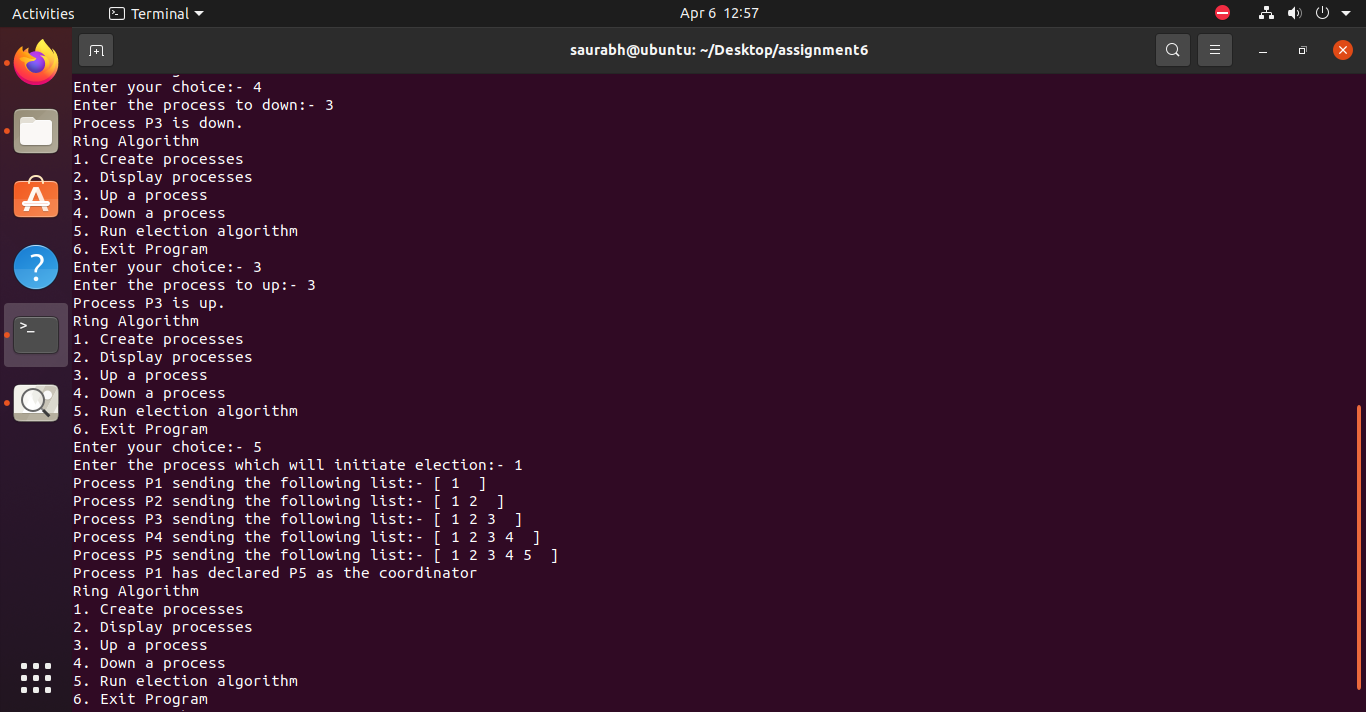
default:

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

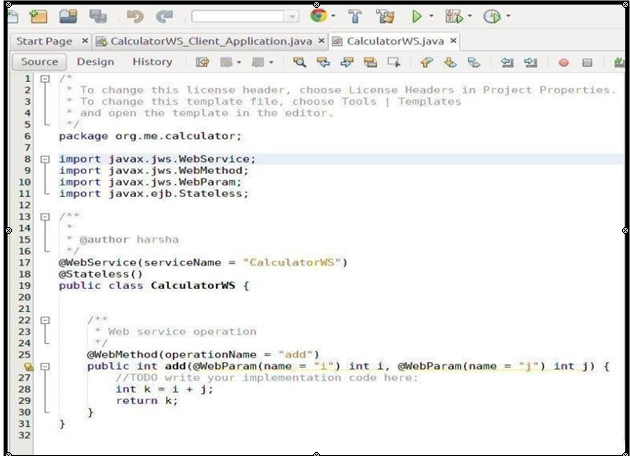
break; } } } }

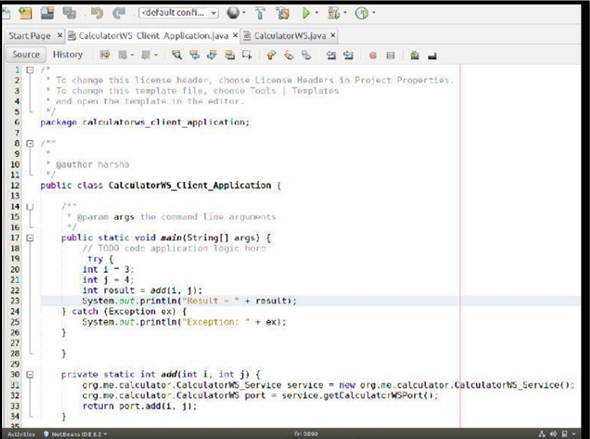
Output :

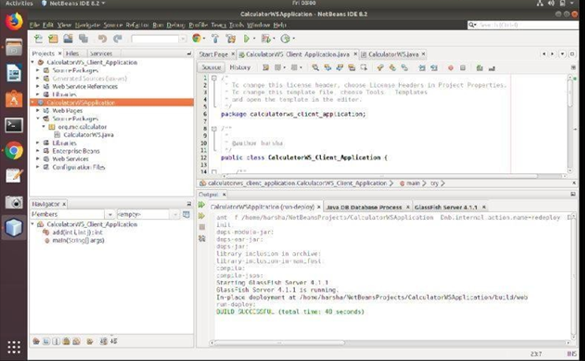


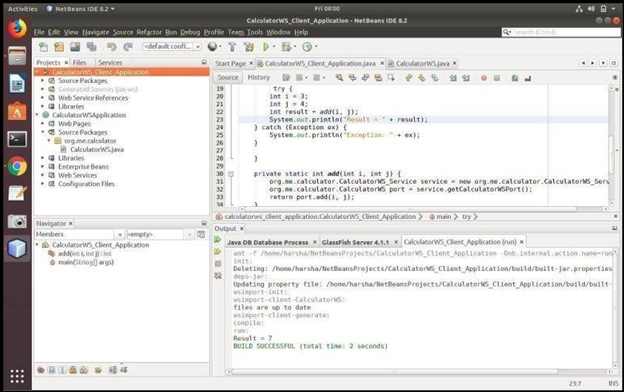


[Assignment 7](https://github.com/meghadandapat/BE-IT-DS/blob/main/Assign1)









**Compiling and executing the solution:**

Right Click on the Project and Choose Run.



