H. J. Thim Trust's

THEEM COLLEGE OF ENGINEERING

Village Betegaon, Boisar Chilhar Road, Boiser (E),
Tal. & Dist. Palghar - 401 501.

Success... through quality education !

Index

Experiment in the Subject of Distributed Computing

Sr. No.	Name of the Experiment	Date	Page No	Remarks
1.	To implement client/server using RPC/RMI	16/01/23	1	
2.	To implement Inter process communication.	23/01/23	11	
3.	To perform group communication	30/01/23	16	
4.	To implement clock synchronization algorithm using Java/Python.	06/02/23	23	
5.	Case study :- CORBA	13/02/23	31	
6.	To implement Election algorithm using bully algorithm.	20/02/23	36	
7.	To implement deadlock management in D.S.	27/02/23	44	
8.	To Implement load balancing algorithm.	27/03/23	51	
9.	Case study on Distributed file system (DFS)	03/04/23	59	
10.	Case study on Android Stock.	03/04/23	65	
	Assignment No :- 01.			

Ameya

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Experiment No :- 01

Aim :- To implement client/server using RPC/RMI

Theory :- RMI stands for Remote method invocation. It is a mechnaism that allows an object residing in one machine (JVM) to access/invoke an object running on another JVM. RMI are used to build distributed applications. It provides remote communication between Java programms. It provides in the package java.rmi.

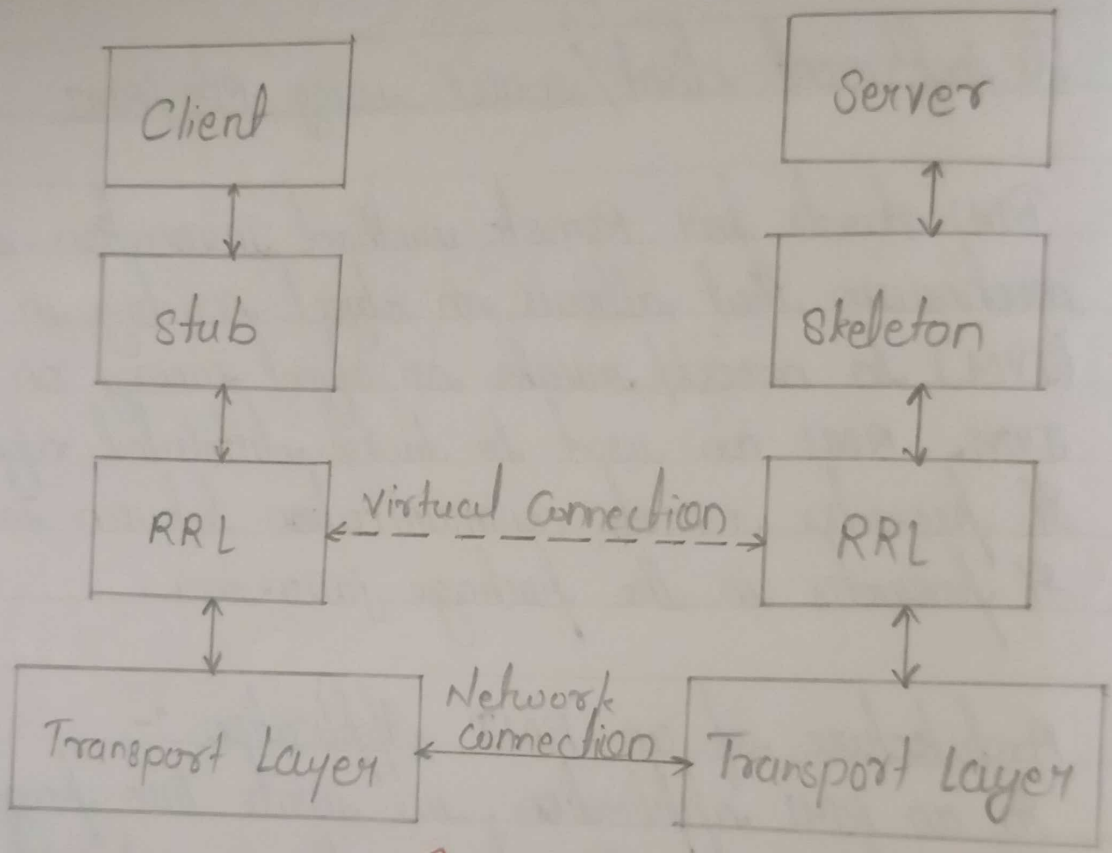
Architecture of an RMI Application :-
In an RMI application, we write two programm, a server programm (resides on the server) and a client programm (resides on the client).

- Inside the server programm, a remote object is created and reference of that object is made available for the client (using the registry).
- The client programm request the remote objects on the server and tries to invoke its method.

The components in this architecture are as follows :-

- Transport layer - This layer connects the client and the server. it manages the existing connection and sets up new connections.

* Diagram :-



(Architecture of RMI Application)

Date : _____

- Stub :- A stub is a representation of remote object and the client. It resides in the client system, it act as gateway for the client programm.

- Skeleton :- This is the object that resides on the server stub. communicates with this skeleton to pass request to the remote object.

- RRL (Remote reference layer) - it is the layer which manages the reference made by the client to the remote object.

RMI uses stub and skeleton object for communication with the remote object.
A remote object is an object for communication whose method can be invoked from another jvm. Let understand the stub and skeleton objects :
stub

The stub is an object, act as gateway for the client side. All the outgoing system request routed through it. It resides at the client side and represent the remote object. When the caller invokes method on the stub object, it does the following tasks.

1. It initiate the connection with remote virtual Machine. (JVM).

2. It writes and transmit (marshals) the parameters to the remote virtual Machine (JVM).
3. It waits for the result.
4. It reads (unmarshals) the return value or exception
5. It finally, returns the value to the caller.

Skeleton

The skeleton is an object, act as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request it does the following tasks:

1. It reads the parameters for remote method.
2. It invokes the method on the actual remote object.
3. It writes and transmits (marshals) the results to the caller.

Whenever a client invokes a method that accepts parameters on a remote object, the parameters are bundles into a message before using being sent on network. These parameter may be of primitive type or objects. In case of primitive type, the parameter are put together and a header is attached to it. In case the parameter are objects then they are serialized. The process is known as marshaling

Date : _____

At the server side, the packed parameters are unbundled and then the required method invoked. This process is known as unmarshalling.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date : _____

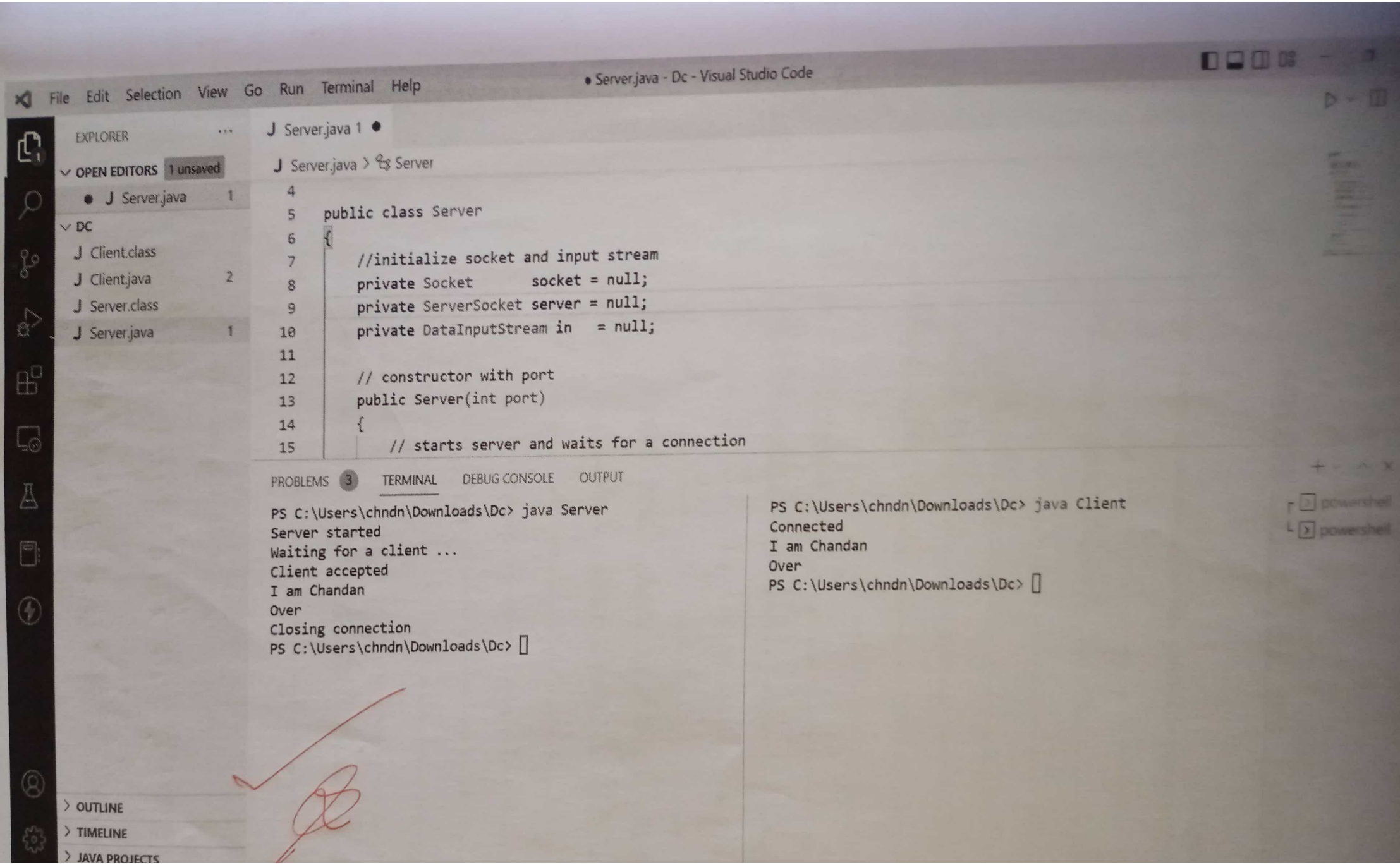
Code :-

```
// A Java programm for a server
import java.net.*;
import java.io.*;

public class Server
{
    private socket socket = null;
    private ServerSocket server = null;
    private DataInputStream in = null;

public Server (int port) {
    try {
        server = new ServerSocket (port);
        System.out.println ("Server started");
        System.out.println ("Waiting for client ");
        Socket = server.accept ();
        System.out.println ("Client accepted");

    in = new DataInputStream (
        new BufferInputStream (socket.getInputStream()));
String line = "";
while (!line.equals ("over")) {
    try {
        line = in.readUTF();
```

```
        System.out.println(line);
    }
    catch (IOException) {
        System.out.println(i); } }
    socket.close();
    in.close();
}
catch (IOException) {
    // System.out.println("closing connection");
    System.out.println(i) } }

public static void main (String[] args) {
    Server server = new Server(5000);
}
}
```

Date : _____

```
// Java programm for a client
import java.net.*;
import java.io.*;
public class client {
    private Socket socket = null;
    private DataInputStream input = null;
    private DataOutputStream out = null;
public client (String address, int port)
    {
      try {
          socket = new Socket(address, port);
      System.out.println("connected")
      input = new DataInputStream (System.in);
      out = new DataOutputStream (Socket.getOutputStream());
}
catch (UnknownHostException) {
      System.out.println(u); }
catch (IOException i) {
      System.out.println(i); }
String line = "";
while (!line.equals("over")) {
    try {
      line = input.readLine();
      out.writeUTF(line); }
catch (IOException i) {
```


File Edit Selection View Go Run Terminal Help
• Server.java - Dc - Visual Studio Code
EXPLORER
OPEN EDITORS 1 unsaved
• Server.java 1
DC
Client.class
Client.java 2
Server.class
Server.java 1
Server.java 1 •
Server.java > Server
4
5 public class Server
6 {
7 //initialize socket and input stream
8 private Socket socket = null;
9 private ServerSocket server = null;
10 private DataInputStream in = null;
11
12 // constructor with port
13 public Server(int port)
14 {
15 // starts server and waits for a connection
PROBLEMS 3 TERMINAL DEBUG CONSOLE OUTPUT
PS C:\Users\chndn\Downloads\Dc> java Server
Server started
Waiting for a client ...
Client accepted
I am Chandan
Over
Closing connection
PS C:\Users\chndn\Downloads\Dc>
PS C:\Users\chndn\Downloads\Dc> java Client
Connected
I am Chandan
Over
PS C:\Users\chndn\Downloads\Dc>
powershell
powershell
OUTLINE
TIMELINE
JAVA PROJECTS

```
        System.out.println(i); } }
try {
    input.close();
    out.close();
    socket.close();
}
catch (IOException i) {
    System.out.println(i); }
}

public static void main (String[] args) {
    Client client = new Client("127.0.0.1", 5000);
    }
}
```

Conc:- We have successfully implemented client/server using RPC/RMI

06/02/23

09

Date :

Experiment No :- 02

Aim:- To implement Inter process communication.

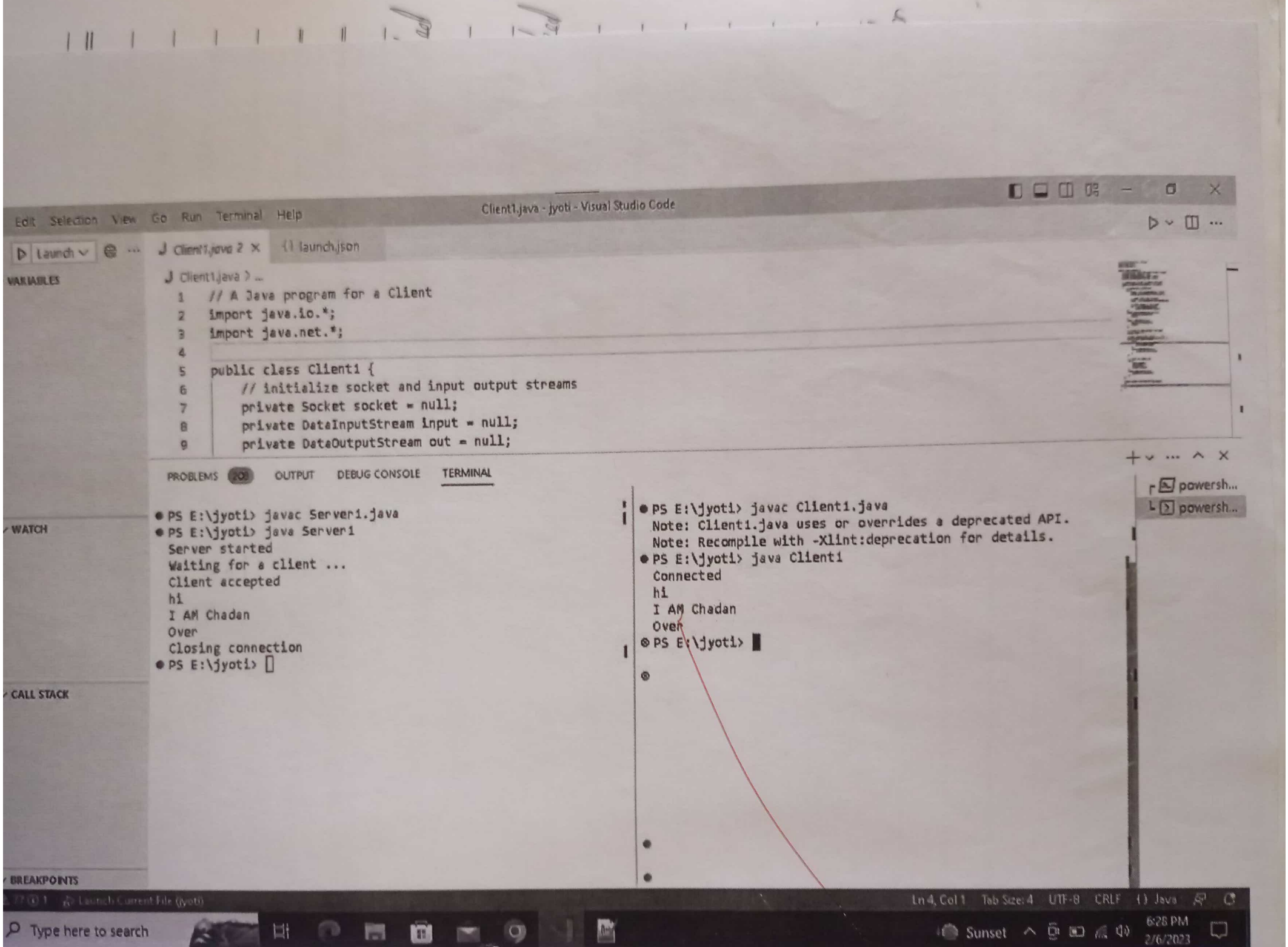
Theory:- Java Socket programming is used for communication between the application running on different JRE.

Java socket programming can be connection-oriented or connection-less. Socket and server socket classes are used for connection-oriented socket programming and Datagram Sockets and Datagram packet classes are used for connection less socket programming.

The client in socket programming must know two information.

- IP Address of server.
- Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here two classes are being used : Socket and server socket. The socket class is used to communicate client and server. Through this class, we can read and write message. The server socket class is used at server-side. The accept() method of server socket class blocks the console until the client is connected. After the successfull connection of client, it returns the instance of socket at server side.


Edit Selection View Go Run Terminal Help
Client1.java - jyoti - Visual Studio Code
Launch
Client1.java 2
launch.json
VARIABLES
Client1.java > ...
1 // A Java program for a Client
2 import java.io.*;
3 import java.net.*;
4
5 public class Client1 {
6 // initialize socket and input output streams
7 private Socket socket = null;
8 private DataInputStream input = null;
9 private DataOutputStream out = null;
PROBLEMS 208
OUTPUT
DEBUG CONSOLE
TERMINAL
PS E:\jyoti> javac Server1.java
PS E:\jyoti> java Server1
Server started
Waiting for a client ...
Client accepted
hi
I AM Chadan
Over
Closing connection
PS E:\jyoti>
PS E:\jyoti> javac Client1.java
Note: Client1.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
PS E:\jyoti> java Client1
Connected
hi
I AM Chadan
Over
PS E:\jyoti>
powersh...
powersh...
WATCH
CALL STACK
BREAKPOINTS
Launch Current File (jyoti)
Ln 4, Col 1
Tab Size: 4
UTF-8
CRLF
Java
Type here to search
Sunset
6:28 PM
2/6/2023

Date :

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods	Method Description
Public InputStream getInputStream().	returns the inputstream attached with the socket.
public outputstream getoutputstream()	returns the outputstream attached with this socket.
public synchronized void close()	closes this socket
Server Socket class	The server socket class can be used to create a server socket. This object is used to establish communicator with the client.

Conc :- we have successfully implemented inter process communication.

```
Import java.io.*;
Import java.net.*;

Public class Server {
 Public static void main(String[] args) throws IOException {
ServerSocketserverSocket = new ServerSocket(9999);
  Socket clientSocket = serverSocket.accept();
PrintWriter out =
     New PrintWriter(clientSocket.getOutputStream(), true);
Out.println("Hello, client!");
Out.close();
clientSocket.close();
serverSocket.close();
 }
}

 Import java.io.*;
 Import java.net.*;

 Public class Client {
  Public static void main(String[] args) throws IOException {
   Socket socket = new Socket("localhost", 9999);
  BufferedReader in = new BufferedReader(
      New InputStreamReader(socket.getInputStream()));
  System.out.println(in.readLine());
  In.close();
  Socket.close();
   }
  }
```

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Experiment No :- 03

Aim :- To perform Group Communication.

Theory :- Java socket programming :-
It is used for communication between the application running on different JRE.
Java socket programming can be connection-oriented or connection less.
Socket and serversocket classes are used for connection-oriented socket programming and Datagram Socket and Datagram Packet classes are used for connection-less socket programming.

- The client in socket programming must know two information.
  1. IP Address of server and
  2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here two classes are being used : Socket and server-socket. The socket class is used to communicate client and server. Through this class we can read and write message.

Date :

The server-socket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successfull connection of client, it returns the instance of socket at server-side.

Socket class :-
A socket is simply an endpoint for communication between the machines. The Socket class can be used to create a socket.

Server Socket class :-
The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Creating Server :-
To create a server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept method waits for the client. If client connects with the given port number, it return an instance of Socket.

Client1.java | Client2.java | Client3.java | Server.java

```java
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client3 {
    public static void main(String[] args) throws Exception {
        Scanner sc = new Scanner(System.in);
        Socket socket = new Socket("localhost", 9001);
        BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
        System.out.print("Enter your name: ");
        String name = sc.nextLine();
        while (true) {
            String line = in.readLine();
            if (line.startsWith("SUBMITNAME")) out.println(name);
            else if (line.startsWith("MESSAGE"))
                System.out.println(line.substring(8));
            if(name.startsWith("master")){
                System.out.print("Enter a message: ");
                out.println(sc.nextLine());
            }
        }
    }
}
```

Problems | Javadoc | Declaration | Console

Server (3) [Java Application] [pid: 3688]

```
The server is running at port 9001.
chandan_1 joined
chandan_2 joined
chandan_3 joined
```

Date :

1. ServerSocket ss = new ServerSocket(6666);
2. Socket s = ss.accept(); //establish connection and waits for the client.

Creating clients :-

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the server and a port number. Here, we are using local host because our server is running on same system.

Conclusion :- Thus, we have successfully implemented group communication.

20/02/09

Program

Client1

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
public class Client1 {
    public static void main(String[] args) throws
Exception {
        Scanner sc = new Scanner(System.in);
        Socket socket = new Socket("localhost", 9001);
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
        System.out.print("Enter your name: ");
        String name = sc.nextLine();
        while (true) {
          String line = in.readLine();
          if (line.startsWith("SUBMITNAME"))
out.println(name);
          else if (line.startsWith("MESSAGE"))
            System.out.println(line.substring(8));
          if(name.startsWith("master")){
            System.out.print("Enter a message: ");
            out.println(sc.nextLine());
          }
        }
    }
}
```

Client2

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
public class Client2 {
    public static void main(String[] args) throws
Exception {
        Scanner sc = new Scanner(System.in);
        Socket socket = new Socket("localhost", 9001);
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
        System.out.print("Enter your name: ");
        String name = sc.nextLine();
        while (true) {
          String line = in.readLine();
          if (line.startsWith("SUBMITNAME"))
out.println(name);
          else if (line.startsWith("MESSAGE"))
            System.out.println(line.substring(8));
          if(name.startsWith("master")){
            System.out.print("Enter a message: ");
            out.println(sc.nextLine());
          }
        }
    }
}
```

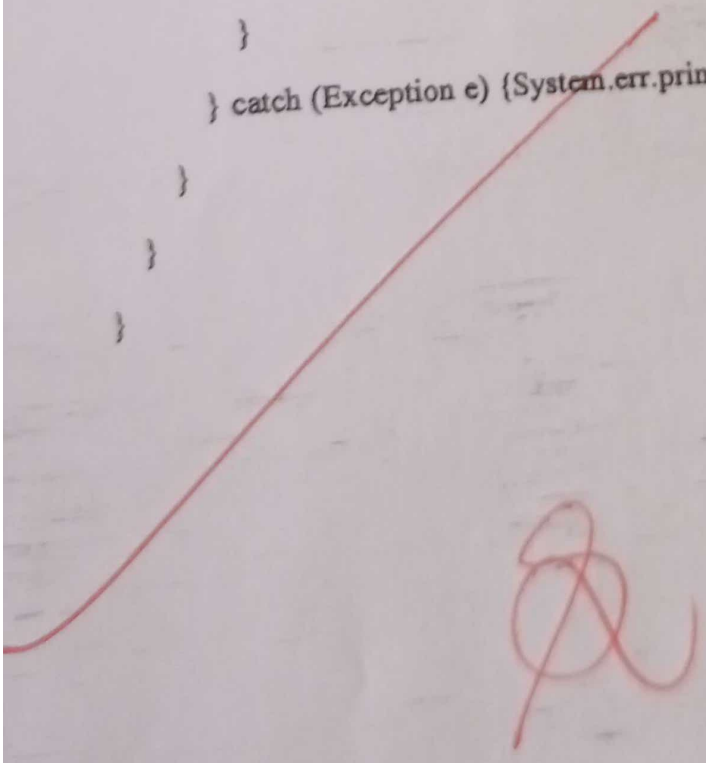
Client3

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;
public class Client3 {
    public static void main(String[] args) throws
Exception {
        Scanner sc = new Scanner(System.in);
        Socket socket = new Socket("localhost", 9001);
        BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
        PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
        System.out.print("Enter your name: ");
        String name = sc.nextLine();
        while (true) {
          String line = in.readLine();
          if (line.startsWith("SUBMITNAME"))
out.println(name);
          else if (line.startsWith("MESSAGE"))
            System.out.println(line.substring(8));
          if(name.startsWith("master")){
            System.out.print("Enter a message: ");
            out.println(sc.nextLine());
          }
        }
    }
}
```

Server

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Vector;
public class Server {
    private static Vector<PrintWriter> writers = new
Vector<PrintWriter>();
    public static void main(String[] args) throws
Exception {
        ServerSocket listener = new ServerSocket(9001);
        System.out.println("The server is running at port
9001.");
        while (true)
            new Handler(listener.accept()).start();
    }
    private static class Handler extends Thread {
        private Socket socket;
        public Handler(Socket socket) {
            this.socket = socket;
        }
        public void run() {
```

```
        try {
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out = new
PrintWriter(socket.getOutputStream(), true);
            out.println("SUBMITNAME");
            String name = in.readLine();
            System.out.println(name+" joined");
            writers.add(out);
            while (true) {
                String input = in.readLine();
                for (PrintWriter writer : writers)
                    writer.println("MESSAGE " + name + ": "
+ input);
            }
        } catch (Exception e) {System.err.println(e);}
    }
  }
}
```



Date : _____

Experiment No :- 04

Aim :- To implement clock synchronization algorithm using Java.

Theory :- Digital system is a collection of computers connected via the high speed communication network. In the distributed system, the hardware and software components communicate and coordinate their action by message passing. Each node in distributed system can share their resource with other nodes so there is need of proper allocation of resource the reserve the state of resource and help coordinate between the several process. To resolve such conflicts synchronization is used. Synchronization in distributed system is achieved via clocks.

The physical clock are used to adjust the time of nodes each nodes in the system can share its local time with other nodes in the system. The time is set based on UTC. UTC is used as a reference time clock for the nodes in the system.

The clock synchronization can be achieved by 2 ways:

1) External clock synchronization is the one in which an external reference clock is present. It is used as

Date :

rel reference and the nodes in the system can set and adjust their time accordingly.

2) Internal clock synchronization is the one in which each node shares its time with other nodes and all the nodes set and adjust their time accordingly.

Berkeley's Algorithm

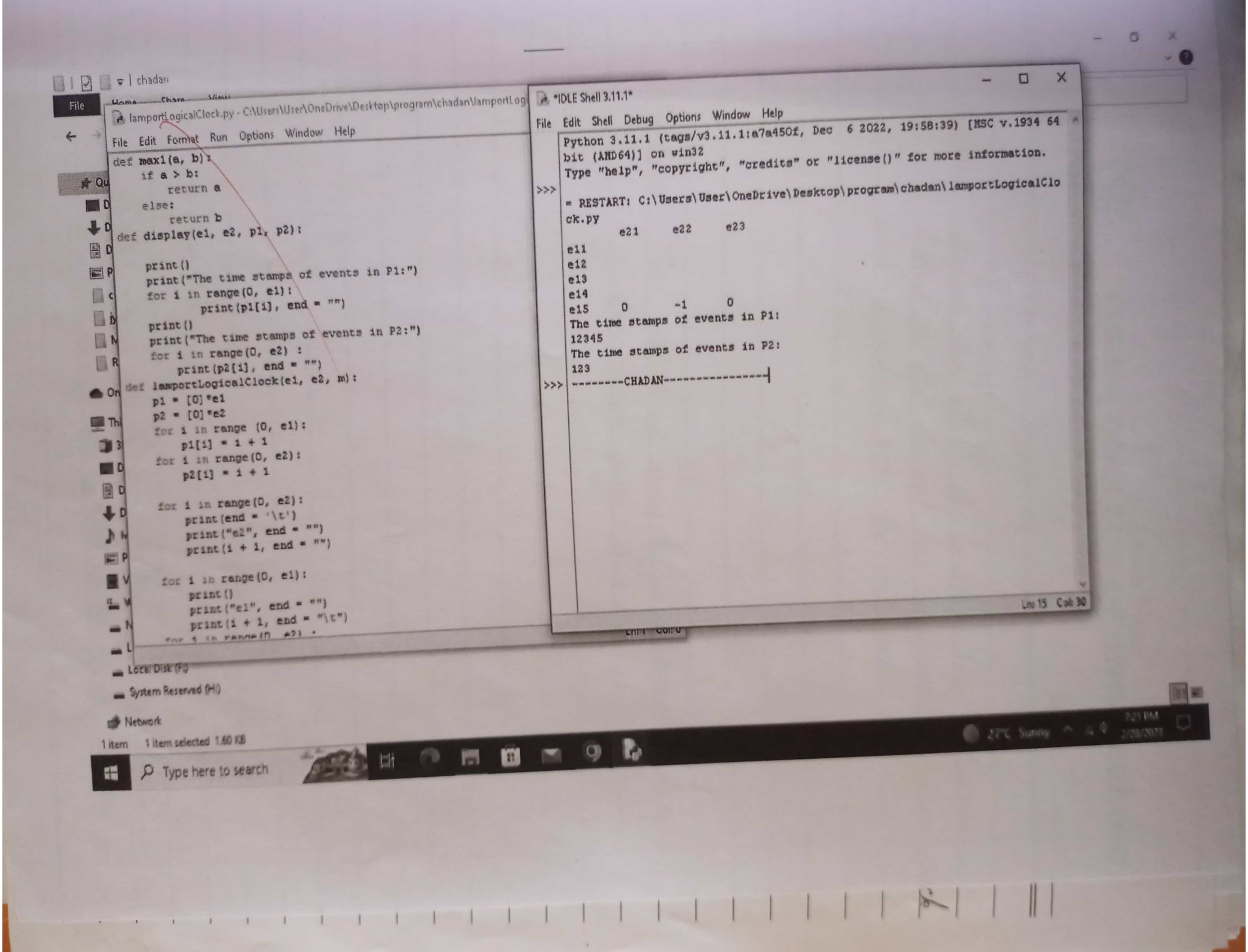
Berkeley's algorithm is a clock synchronization technique used in distributed systems. The algorithm assumes that node in the network either doesn't have an accurate time source or doesn't posses an UTC server.

Algorithm :-

1) An individual node is choosen as a master node in the network. This node is the main node in the network which acts as a master and rest of the nodes acts as a slaves. Master node is choosen using a election / Feeder process election algorithm.

2) Master node periodically pings slaves nodes and fetch clock time at them using cristian's algorithm.

3) Master node calculate the average time difference between all the clock times retrieved and the clock time given by master's system clock itself. This average time difference is added to the current time at master's


lamportLogicalClock.py - C:\Users\User\OneDrive\Desktop\program\chadan\lamportLogi
File Edit Format Run Options Window Help
def max1(a, b):
if a > b:
return a
else:
return b
def display(e1, e2, p1, p2):
print()
print("The time stamps of events in P1:")
for i in range(0, e1):
print(p1[i], end = "")
print()
print("The time stamps of events in P2:")
for i in range(0, e2) :
print(p2[i], end = "")
def lamportLogicalClock(e1, e2, m):
p1 = [0]*e1
p2 = [0]*e2
for i in range (0, e1):
p1[i] = i + 1
for i in range(0, e2):
p2[i] = i + 1
for i in range(0, e2):
print(end = '\t')
print("e2", end = "")
print(i + 1, end = "")
for i in range(0, e1):
print()
print("e1", end = "")
print(i + 1, end = "\t")
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec 6 2022, 19:58:39) [MSC v.1934 64
bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\User\OneDrive\Desktop\program\chadan\lamportLogicalClo
ck.py
e21 e22 e23
e11
e12
e13
e14
e15 0 -1 0
The time stamps of events in P1:
12345
The time stamps of events in P2:
123
>>> --------CHADAN----------
Ln: 15 Col: 30
chadan
File
Local Disk (F:)
System Reserved (H:)
Network
1 item 1 item selected 1.60 KB
Type here to search
27°C Sunny

Date :

system system clock itself. This average time difference is added to the current time at master's system clock and broadcast over the network.

Conclusion:- Thus, we have successfully implemented clock synchronization using Java.

Code:

```
// Java program to illustrate the Lamport's
// Logical Clock
import java.util.*;
public class GFG {
// Function to find the maximum timestamp
// between 2 events
static int max1(int a, int b)
{
        // Return the greatest of the two
        if (a > b)
        return a;
        else
        return b;
 }
 // Function to display the logical timestamp
 static void display(int e1, int e2, int p1[], int p2[])
 {      int i;
        System.out.print(
        "\nThe time stamps of events in P1:\n");

        for (i = 0; i < e1; i++) {
        System.out.print(p1[i] + " ");
        }
        System.out.println(
        "\nThe time stamps of events in P2:");
        // Print the array p2[]
        for (i = 0; i < e2; i++)
        System.out.print(p2[i] + " ");
   }
```

```
// Function to find the timestamp of events
static void lamportLogicalClock(int e1, int e2,int m[][])
{
        int i, j, k;
        int p1[] = new int[e1];
        int p2[] = new int[e2];
        // Initialize p1[] and p2[]
        for (i = 0; i < e1; i++)
        p1[i] = i + 1;
        for (i = 0; i < e2; i++)
        p2[i] = i + 1;
        for (i = 0; i < e2; i++)
        System.out.print("\te2" + (i + 1));
        for (i = 0; i < e1; i++) {
        System.out.print("\n e1" + (i + 1) + "\t");
        for (j = 0; j < e2; j++)
                System.out.print(m[i][j] + "\t");
        }
        for (i = 0; i < e1; i++) {
        for (j = 0; j < e2; j++) {
                if (m[i][j] == 1) {
                p2[j] = max1(p2[j], p1[i] + 1);
                for (k = j + 1; k < e2; k++)
                        p2[k] = p2[k - 1] + 1;
                }
                if (m[i][j] == -1) {
                p1[i] = max1(p1[i], p2[j] + 1);
                for (k = i + 1; k < e1; k++)
                        p1[k] = p1[k - 1] + 1;
                }
        }
```

```
        }
        display(e1, e2, p1, p2);
}
public static void main(String args[])
{
        int e1 = 5, e2 = 3;
        int m[][] = new int[5][3];
        /*dep[i][j] = 1, if message is sent
                                        from ei to ej
                dep[i][j] = -1, if message is received
                                                by ei from ej
                dep[i][j] = 0, otherwise*/
            m[0][0] = 0;
            m[0][1] = 0;
            m[0][2] = 0;
            m[1][0] = 0;
            m[1][1] = 0;
            m[1][2] = 1;
            m[2][0] = 0;
            m[2][1] = 0;
            m[2][2] = 0;
            m[3][0] = 0;
            m[3][1] = 0;
            m[3][2] = 0;
            m[4][0] = 0;
            m[4][1] = -1;
            m[4][2] = 0;
            lamportLogicalClock(e1, e2, m);
    }
    }
```

Case Study:1

Aim: Case Study on CORBA

Theory:

Introduction:

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG)that enables softwarecomponents written in multiple computer languages and running on multiple computers to work together (i.e., it supports multiple platforms). CORBA provides a platform-independent, language-independent way to write applications that can invoke objectsthat live across the room or across the planet. CORBA enables separate pieces of softwarewritten in different languages and running on different computers to workwith each other like a single application or set of services. More specifically,CORBA is a mechanism in softwarefor normalizing the method-call semantics between application objects residing either in the same address space (application) or remote address space (same host, or remote host on a network). Common ObjectRequest Broker Architecture enables separate pieces of softwarewritten in different languages to interact with each other. It uses an InterfaceDefinition language (IDL) to specify the interfaces whichobjects present to the outer world.CORBAspecification tells there should be an ORB (ObjectRequest Blockers) through whichan application would interact with other objects. In practice, the application simply initialize the ORB and access an interact objectAdapter which maintains things like reference counting, object installation policies and objectlifetime policy. CORBA uses an interface definition language (IDL) to specify the interfaces which objects present to the outer world. CORBA then specifies a mapping fromIDL to a specific implementation language like C++ or Java.Standard mappings exist for Ada, C, C++, Lisp, Ruby, Smalltalk, Java,COBOL, PL/Iand Python.There are also non-standard mappings for Perl, Visual Basic, Erlang, and Tcl implemented by object request brokers (ORBs) written for those languages.

Features/ Characteristics:

CORBA achieves its flexibility in several ways:

- It specifies an interface description language (IDL), that allows you to specify the interfaces to objects. IDL object interfaces describe, among other things:
- The data that the object makes public.
- The operations that the object can respond to, including the complete signature of the operation. CORBA operations are mapped to Java methods, and the IDL operation parameter types map to Java datatypes.
- Exceptions that the object can throw. IDL exceptions are also mapped to Java exceptions, and the mapping is very direct.
- CORBA provides bindings for many languages, including both non-object languages such as COBOL and C and object-oriented languages such as Smalltalk and Java.
- All CORBA implementations provide an object request broker (ORB), that handles the routing of object requests in a way that is largely transparent to the application developer.

For example, requests (method invocations) on remote objects that appear in the client code look just like local method invocations. The remote call functionality, including marshalling of parameter and return data, is taken care of for the programmer by the ORB.

- CORBA specifies a network protocol, the Internet Inter-ORB Protocol (IIOP), that provides for transmission of ORB requests and data over a widely-available transport protocol: TCP/IP, the Internet standard.

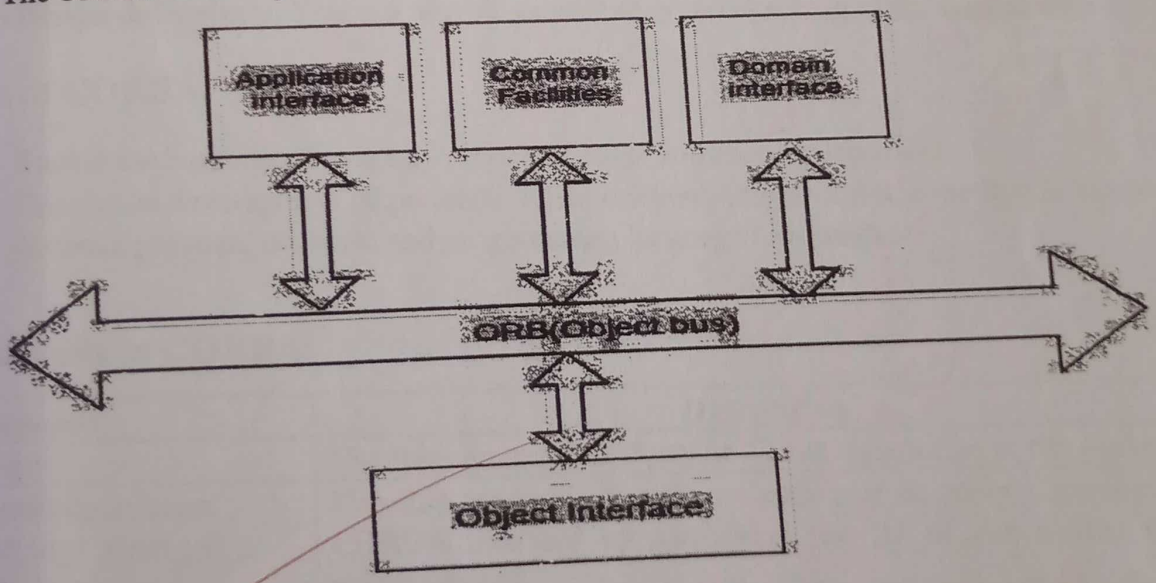
Architecture of CORBA:

Common Object Request Broker Architecture (CORBA) could be a specification of a regular design for middleware. It is a client-server software development model.

Using a CORBA implementation, a shopper will transparently invoke a way on a server object, which may air a similar machine or across a network. The middleware takes the decision, associated is to blame for finding an object which will implement the request, passing it the parameters, invoking its methodology, and returning the results of the invocation. The shopper doesn't need to remember of wherever the item is found, its programming language, its software package or the other aspects that don't seem to be a part of the associated object's interface.

CORBA Reference Model:

The CORBA reference model known as Object Management design (OMA) is shown below figure. The OMA is itself a specification (actually, a group of connected specifications) that defines



Object Management Architecture(OMA)

a broad vary of services for building distributed client-server applications. several services one may expect to search out in a very middleware product like CORBA (e.g., naming, dealings, and asynchronous event management services) are literally fixed as services within the OMA Different parts communicate victimization ORB. ORB is additionally referred to as the item bus. An

associate example of the application interface is a distributed document facility. In a very domain interface, it will have domain dependent services, for instance, producing domain.

Object interface has some domain freelance services:

1. **Naming-Service:**
   Naming service is additionally known as a white page service. victimization naming service server-name will be searched and it's location or address pointed-out.

2. **Trading-Service:**
   Commercialism service is additionally known as a yellow page service. victimization commercialism service a selected service will be searched. this is often corresponding to looking out a service like an automobile store in a very yellow page directory.

The Broker pattern is particularly useful in helping you follow these design principles:

- **Divide and conquer.** The remote objects can be independently designed.
- **Increase reusability.** It is often possible to design the remote objects so that other systems can use them too.
- **Increase reuse.** You may be able to reuse remote objects that others have created.
- **Design for flexibility.** The broker objects can be updated as required, or you can redirect the proxy to communicate with a different remote object.
- **Design for portability.** You can write clients for new platforms while still accessing brokers and remote objects on other platforms.
- **Design defensively.** You can provide careful assertion checking in the remote objects.

Goals of CORBA:

- Enable the building of plug and play component software environment
- Enable the development of portable, object oriented, interoperable code that is hardware, operating system, network, and programming language independent

Commands in CORBA:

Commands	Description
Exit/quit	Use this command to close the Telnet connection to the probe.
acknowledgeAlarms alarm_id_1 alarm_id_2 … alarm_id_n	Use this command to acknowledge one or more alarms in the CORBA interface by specifying the ID of the alarms being acknowledged.
cleanupSubscription subscription_id	Use this command to detach the specified subscription from the SAM server.
clearAlarms alarm_id_1 alarm_id_2 … alarm_id_n	Use this command to clear one or more alarms in the CORBA interface by specifying the ID of the alarms being cleared.

commentAlarms comment_text alarm_id_1 alarm_id_2 … alarm_id_n	Use this command to add comments to one or more alarms in the CORBA interface by specifying the text to be added, followed by the ID of the alarms being commented. If the comment that you want to add to the alarm contains whitespace, you must put the comment portion of the command inside double quotation marks.
getEventQueueSize	Use this command to display the current size of the internal event queue.
getSubscriptionStatus subscription_id	Use this command to display the status of the specified subscription.
help	Use this command to display online help about the CLI
resync	Use this command to perform a resynchronization with the CORBA interface. This command reuses the filter specified by the ResyncFilter property.
shutdown	Use this command to shut down the probe.

Types of CORBA:

- **Atomic data types**
  - **Boolean:-** The term Boolean means **a result that can only have one of two possible values: true or false**.
  - **Char:-** The CHAR data type stores character data in a fixed-length field.
  - **Double:-** The double data type is a double-precision 64-bit IEEE 754 floating point.
  - **Float:-** The FLOAT data type stores double-precision floating-point numbers with up to 17 significant digits.
  - **Long:-** Long (long integer) variables are stored as signed 32-bit (4-byte) numbers ranging in value from -2,147,483,648 to 2,147,483,647.
  - **Octet (hexadecimal) :-** Octets can be represented using number systems of varying bases such as the hexadecimal, decimal, or octal number systems.
  - **Short:-** The short data type is a 16-bit signed two's complement integer.
  - **ULong (unsigned long):-** The ULong data type widens to Decimal , Single , and Double .
  - **UShort (unsigned short) :-** The UShort data type widens to Integer , UInteger , Long , ULong , Decimal , Single , and Double .
- **Enum (enumerations) :-** An enumeration is a data type that consists of a set of named values that represent integral constants, known as enumeration constants.
- **LongLong (long long):-** LongLong (LongLong integer) variables are stored as signed 64-bit (8-byte) numbers ranging in value from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.
- **Struct :-** Structures (also called structs) are a way to group several related variables into one place.
- **ULongLong (unsigned long long):-** A ULONGLONG is a 64-bit unsigned integer (range: 0 through 18446744073709551615 decimal).
- **WChar (wide character):-** The wchar_t type is an implementation-defined wide character type.

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Experiment No :- 06

Aim :- To implement election algorithm using bully algorithm.

Theory :- Bully algorithm :

Bully algorithm is a method used to elect a leader in distributed system where multiple processes are competing to become the leader. It assumes that each process in the system has a unique identifier.

Working of bully algorithm :-

1) When a process notices that the leader has failed it sends an election message to all process with higher ID.

2) The processes that receive the election message respond with an ok message including that they are still alive.

3) If the process that sent the election message receives an ok message from all higher processes.

4) If the message processer that sent the election messages does not receive an ok message from all higher process.

5) If a process receives an election message from another process with a lower ID.

Here is step by step description on how the election algorithm using Bully algorithm works.

1) When a process detect that the leader has failed

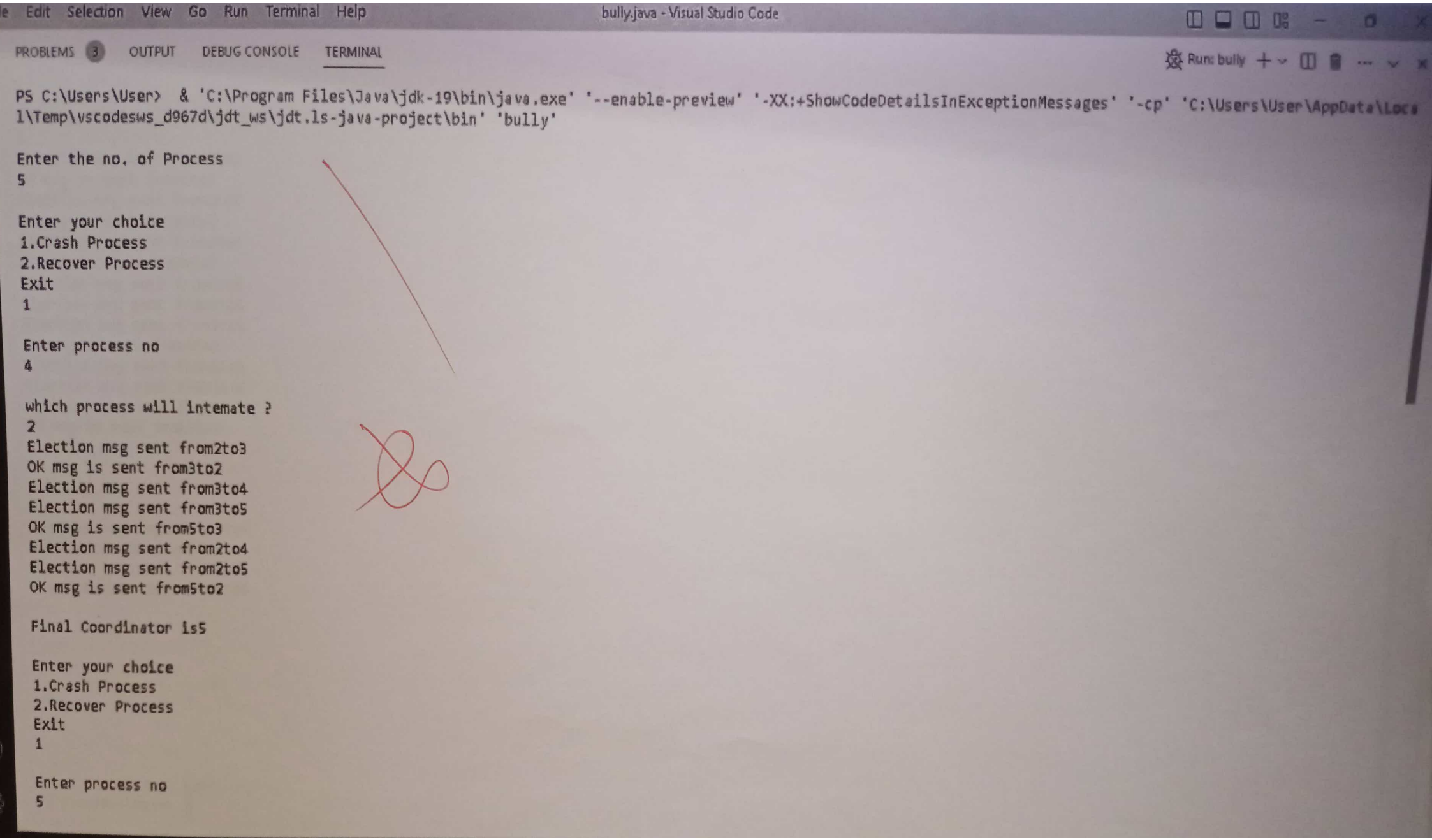
CODE:-

Bully.java

```
import java.io.*;
import java.util.*;

public class bully {
   static int n;
   static int pro[] = new int[100];
   static int sta[] = new int[100];
   static int co;
   public static void main(String args[]) {
      System.out.println("");
      System.out.println("Enter the no. of Process");
      Scanner sc = new Scanner(System.in);
      n = sc.nextInt();
      int i, j, c, c1 = 1;
      for (i = 0; i < n; i++) {
         sta[i] = 1;
         pro[i] = i;
      }
      boolean choice = true;
      int ch;
      do {
         System.out.println("");
         System.out.println("Enter your choice");
         System.out.println("1.Crash Process");
         System.out.println("2.Recover Process");
         System.out.println("Exit");
         ch = sc.nextInt();
         switch (ch) {
```

```
            case 1:
                System.out.println("");
                System.out.println("Enter process no");
                c = sc.nextInt();
                sta[c - 1] = 0;
                c1 = 1;
                break;
            case 2:
                System.out.println("");
                System.out.println("Enter process no");
                c = sc.nextInt();
                sta[c - 1] = 1;
                c1 = 1;
                break;
            case 3:
                choice = false;
                c1 = 0;
                break;
        }
        if (c1 == 1) {
            System.out.println("");
            System.out.println("which process will intemate ?");
            int ele = sc.nextInt();
            elect(ele);
        }
        System.out.println("");
        System.out.println("Final Coordinator is" + co);
    } while (choice);
}
```

```
static void elect(int ele) {
    ele = ele - 1;
    co = ele + 1;
    for (int i = 0; i < n; i++) {
        if (pro[ele] < pro[i]) {
            System.out.println("Election msg sent from" + (ele + 1) + "to" + (i + 1));
            if (sta[i] == 1) {
                System.out.println("OK msg is sent from" + (i + 1) + "to" + (ele + 1));
                if (sta[i] == 1)
                    elect(i + 1);
            }
        }
    }

}
}
```


Edit Selection View Go Run Terminal Help
bully.java - Visual Studio Code
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
Run: bully
PS C:\Users\User> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\User\AppData\Local\Temp\vscodesws_d967d\jdt_ws\jdt.ls-java-project\bin' 'bully'
Enter the no. of Process
5
Enter your choice
1.Crash Process
2.Recover Process
Exit
1
Enter process no
4
which process will intemate ?
2
Election msg sent from2to3
OK msg is sent from3to2
Election msg sent from3to4
Election msg sent from3to5
OK msg is sent from5to3
Election msg sent from2to4
Election msg sent from2to5
OK msg is sent from5to2
Final Coordinator is5
Enter your choice
1.Crash Process
2.Recover Process
Exit
1
Enter process no
5

```
which process will intemate ?
1
Election msg sent from1to2
OK msg is sent from2to1
Election msg sent from2to3
OK msg is sent from3to2
Election msg sent from3to4
OK msg is sent from4to3
Election msg sent from4to5
Election msg sent from3to5
Election msg sent from2to4
OK msg is sent from4to2
Election msg sent from4to5
Election msg sent from2to5
Election msg sent from1to3
OK msg is sent from3to1
Election msg sent from3to4
OK msg is sent from4to3
Election msg sent from4to5
Election msg sent from3to5
Election msg sent from1to4
OK msg is sent from4to1
Election msg sent from4to5
Election msg sent from1to5

Final Coordinator is4

Enter your choice
1.Crash Process
2.Recover Process
Exit
3

Final Coordinator is4
PS C:\Users\User>
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

```
Enter process no
5

which process will intemate ?
2
Election msg sent from2to3
OK msg is sent from3to2
Election msg sent from3to4
Election msg sent from3to5
Election msg sent from2to4
Election msg sent from2to5

Final Coordinator is3

Enter your choice
1.Crash Process
2.Recover Process
Exit
2

Enter process no
4

which process will intemate ?
1
Election msg sent from1to2
OK msg is sent from2to1
Election msg sent from2to3
OK msg is sent from3to2
Election msg sent from3to4
OK msg is sent from4to3
Election msg sent from4to5
Election msg sent from3to5
Election msg sent from2to4
OK msg is sent from4to2
Election msg sent from4to5
```

Date :

it sends an election message to all process with higher ID.

2) If a process receives an election message, it responds with an ok message and starts its own election.

3) If a process receive an okey message from all higher processes, it becomes the leader.

4) If is a process receives a victory message, it recognize the new leader and updates its internal state to reflect the change in leadership.

The bully algorithm is simple and efficient, but it has a drawback. it assumes all processes that they have unique IDs and that the process with highest ID is the most qualified to be the leader. If two or more processes have the same ID.

Conclusion :-

Thus we have successfully implemented election algorithm using bully algorithm.

08104

H. J. Thim Trust's
THEEM COLLEGE OF ENGINEERING

Date :

Experiment No :- 07

Aim :- To implement deadlock management in distributed system.

Theory Deadlock :-

It can occur in a distributed system when multiple processes are competing for shared resources and become blocked. In distributed system, you can a combination of prevention and detection.

1) Prevention :-

The best way to manage deadlock is to prevent them from occuring in the first place. You can use several techniques to prevent deadlocks in a distributed system, including resources allocation algorithms, such as banker's algorithm.

2) Detection :-

Even with prevention techniques in place, deadlocks can still occur. To detect deadlock in a distributed system you can use variety of techniques, including the banker's algorithm.

3) Recovery :-

Once a deadlock has been detected, you can use several techniques to recover from it. One approach is to roll back one or more of the transitions that caused the deadlock.

CODE:-

LoadBalancer.java:-

```
import java.util.Scanner;

public class LoadBalancer {
   static void printLoad(int servers, int Processes) {
      int each = Processes / servers;
      int extra = Processes % servers;
      int total = 0;
      for (int i = 0; i < servers; i++) {
         if (extra-- > 0)
            total = each + 1;
         else
            total = each;
         System.out.println("Server" + (char) ('A' + i) + "has" + total + "Processes");
      }
   }

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Enter the number of servers and Processes: ");
      int servers = sc.nextInt();
      int Processes = sc.nextInt();
      while (true) {
         printLoad(servers, Processes);
         System.out.print("1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes
5.Exit: ");
         switch (sc.nextInt()) {
            case 1:
               System.out.print("How many more servers?: ");
```

```
            servers += sc.nextInt();
            break;
        case 2:
            System.out.print("How many servers to remove?: ");
            servers -= sc.nextInt();
            break;
        case 3:
            System.out.print("How many more Processes?: ");
            Processes += sc.nextInt();
            break;
        case 4:
            System.out.print("How many Processes to remove?: ");
            Processes -= sc.nextInt();
            break;
        case 5:
            return;
    }

  }

 }
}
```

File Edit Selection View Go Run Terminal Help — LoadBalancer.java - Visual Studio Code

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

Run LoadBalancer

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\User>  & 'C:\Program Files\Java\jdk-19\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\
l\Temp\vscodesws_62c2c\jdt_ws\jdt.ls-java-project\bin' 'LoadBalancer'
Enter the number of servers and Processes: 5
16
ServerAhas4Processes
ServerBhas3Processes
How many more servers?: 2
ServerAhas3Processes
ServerBhas3Processes
ServerChas2Processes
ServerDhas2Processes
ServerEhas2Processes
ServerFhas2Processes
ServerGhas2Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 3
How many more Processes?: 150
ServerAhas24Processes
ServerBhas24Processes
ServerChas24Processes
ServerDhas24Processes
ServerEhas24Processes
ServerFhas23Processes
ServerGhas23Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 2
How many servers to remove?: 2
ServerAhas34Processes
ServerBhas33Processes
ServerChas33Processes
ServerDhas33Processes
ServerEhas33Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 4
How many Processes to remove?: 100
```

main ⊗0 ⚠2

Ln 30, Col 24 Spaces: 4 UTF-8 CRLF Java

Type here to search

File Edit Selection View Go Run Terminal Help

LoadBalancer.java - Visual Studio Code

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

```
How many more servers?: 2
ServerAhas3Processes
ServerBhas3Processes
ServerChas2Processes
ServerDhas2Processes
ServerEhas2Processes
ServerFhas2Processes
ServerGhas2Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 3
How many more Processes?: 150
ServerAhas24Processes
ServerBhas24Processes
ServerChas24Processes
ServerDhas24Processes
ServerEhas24Processes
ServerFhas23Processes
ServerGhas23Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 2
How many servers to remove?: 2
ServerAhas34Processes
ServerBhas33Processes
ServerChas33Processes
ServerDhas33Processes
ServerEhas33Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 4
How many Processes to remove?: 100
ServerAhas14Processes
ServerBhas13Processes
ServerChas13Processes
ServerDhas13Processes
ServerEhas13Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 5
PS C:\Users\User>
```

Date :

another approach is to release resources that are being held by one or more of the processes involved in the deadlock, either by forcing those processes to release the resources or by preempting them and taking resources away.

Conclusion :- Thus we have successfully to implemented deadlock managment in distributed system.

06/04.

Date :

Experiment No :- 08

Aim :- To implement load balancing algorithm

Theory :- Load balancing is the process of distributed workload across multiple computing resource system. to optimize resource utilization and improve performance. There are several load balancing algorithms that you can use to distribute workloads effectively across resources Here's an overview of how to implement a load balancing algorithm.

1) Identify the resource :-

The first step in implementing load balancing algorithm is to identify available resource that can be used to distribute the workloads. This can be done by discovering the available servers.

2) Collect the workload information :-

once you have identified the resources, you need to collect information about the workloads that needs to be distributed.

3) Determine the selection criteria :-

The next step is to determine the criteria for selecting the resources to which a workload will be assigned. The selection criteria can be include factors such as available capacity of each resource the processing power of each resources.

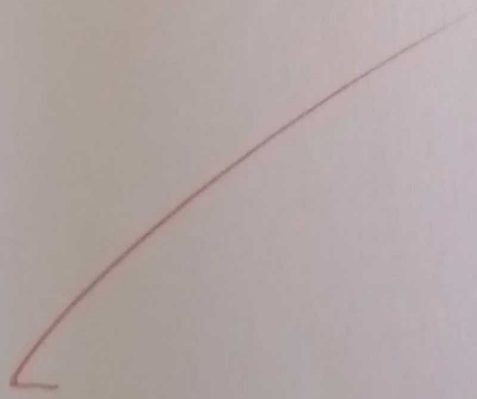
CODE:-

Bankers.java:

```
import java.util.Scanner;

public class Bankers {
  private int need[][], allocate[][], max[][], avail[][], np, nr;
  private void input() {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter no. of processes and resources: ");
    np = sc.nextInt(); // no. of process
    nr = sc.nextInt(); // no. of resources
    need = new int[np][nr]; // initializing arrays
    max = new int[np][nr];
    allocate = new int[np][nr];
    avail = new int[1][nr];
    System.out.println("Enter allocation matrix -->");
    for (int i = 0; i < np; i++)
      for (int j = 0; j < nr; j++)
        allocate[i][j] = sc.nextInt(); // allocation matrix
    System.out.println("Enter max matrix -->");
    for (int i = 0; i < np; i++)
      for (int j = 0; j < nr; j++)
        max[i][j] = sc.nextInt(); // max matrix
    System.out.println("Enter available matrix -->");
    for (int j = 0; j < nr; j++)
      avail[0][j] = sc.nextInt(); // available matrix
    sc.close();
  }

  private int[][] calc_need() {
```

```
    for (int i = 0; i < np; i++)
      for (int j = 0; j < nr; j++) // calculating need matrix
        need[i][j] = max[i][j] - allocate[i][j];
    return need;
  }

  private boolean check(int i) {
    // checking if all resources for ith process can be allocated
    for (int j = 0; j < nr; j++)
      if (avail[0][j] < need[i][j])
        return false;
    return true;
  }

  public void isSafe() {
    input();
    calc_need();
    boolean done[] = new boolean[np];
    int j = 0;
    while (j < np) { // until all process allocated
      boolean allocated = false;
      for (int i = 0; i < np; i++)
        if (!done[i] && check(i)) { // trying to allocate
          for (int k = 0; k < nr; k++)
            avail[0][k] = avail[0][k] - need[i][k] + max[i][k];
          System.out.println("Allocated process : " + i);
          allocated = done[i] = true;
          j++;
        }
      if (!allocated)
```

```
            break; // if no allocation

        }
        if (j == np) // if all processes are allocated
            System.out.println("\nSafely allocated");
        else
            System.out.println("All processes cannot be allocated safely");
    }

    public static void main(String[] args) {
        new Bankers().isSafe();
    }
}
```



File Edit Selection View Go Run Terminal Help

Bankers.java - Visual Studio Code

Bankers.java 1 ×

C: > Users > User > OneDrive > Desktop > program > EXP 06 to 10 > EXP 7 > Bankers.java > ...

```
1  import java.util.Scanner;
2
3  public class Bankers {
4      private int need[][], allocate[][], max[][], avail[][], np, nr;
5
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

Run: Bankers

```
l\Temp\vscodesws_f73e8\jdt_ws\jdt.ls-java-project\bin' 'Bankers'
Enter no. of processes and resources: 3 4
Enter allocation matrix -->
1 2 4
5 4 9
6 7 5
8 5 4
Enter max matrix -->
5 4 6
7 8 9
5 4 7
5 4 1
Enter available matrix -->
7 5 1 4
Allocated process : 1
Allocated process : 2
Allocated process : 0

Safely allocated
PS C:\Users\User>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\User> & 'C:\Program Files\Java\jdk-19\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\Use
l\Temp\vscodesws_62c2c\jdt_ws\jdt.ls-java-project\bin' 'LoadBalancer'
Enter the number of servers and Processes: 5
16
ServerAhas4Processes
ServerBhas3Processes
How many more servers?: 2
ServerAhas3Processes
ServerBhas3Processes
ServerChas2Processes
ServerDhas2Processes
ServerEhas2Processes
ServerFhas2Processes
ServerGhas2Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 3
How many more Processes?: 150
ServerAhas24Processes
ServerBhas24Processes
ServerChas24Processes
ServerDhas24Processes
ServerEhas24Processes
ServerFhas23Processes
ServerGhas23Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 2
How many servers to remove?: 2
ServerAhas34Processes
ServerBhas33Processes
ServerChas33Processes
ServerDhas33Processes
ServerEhas33Processes
1.Add Servers 2.Remove Servers 3.Add Processes 4.Remove Processes 5.Exit: 4
How many Processes to remove?: 100
```

Date :

4) Choose a load balancing algorithm. There are several load balancing algorithm that you can use include round-robin, least connection and ip hash. Each algorithm has its own advantages and disadvantages and you should choose algorithm that best suits your
5) Implement the load balancing system :-
once you have choosen the load balancing algorithm you need to implement it. This can involves configuring load.
6) Monitor and adjust :-
After implementing the load balancing algorithm you should monitor the system to ensure that is is working as expected. you may need to adjust the selection criteria or algorithm over time to optimize performance based on changing workload and resources requirements.

Conclusion :-
Here we have successfully implemented load balancing algorithm.

68104

Experiment No.: -09

Aim: - Case Study on Distributed File System.

Introduction:

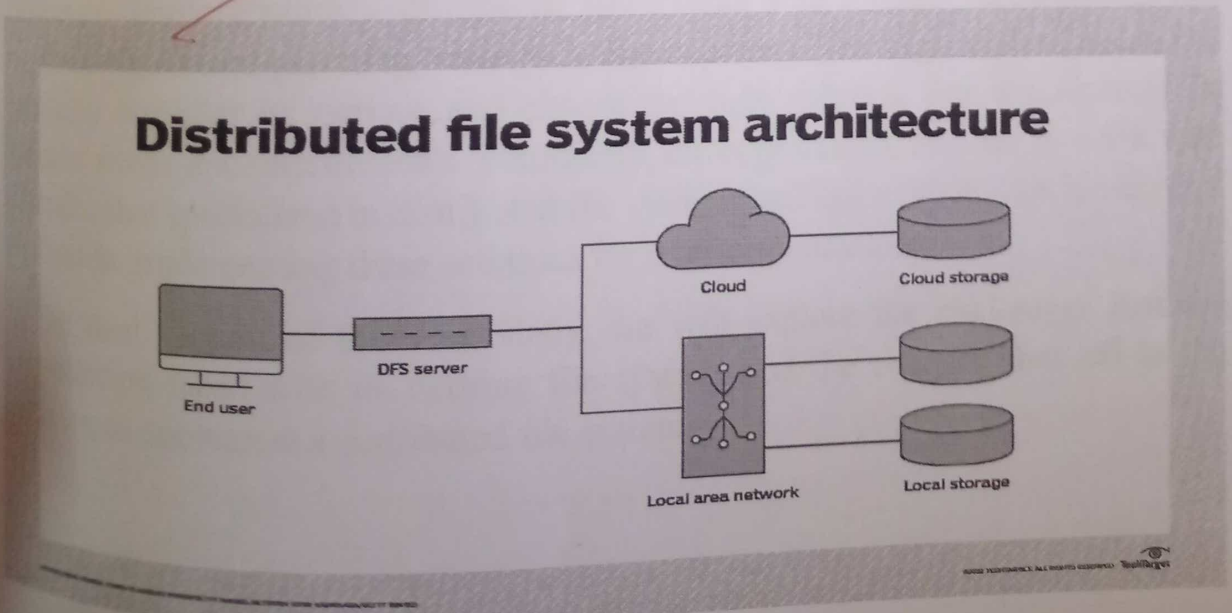
What is a distributed file system (DFS)?

A distributed file system (DFS) is a file system that enables clients to access file storage from multiple hosts through a computer network as if the user was accessing local storage. Files are spread across multiple storage servers and in multiple locations, which enables users to share data and storage resources. A DFS can be designed so geographically distributed users, such as remote workers and distributed teams, can access and share files remotely as if they were stored locally.

How a DFS works

A DFS clusters together multiple storage nodes and logically distributes data sets across multiple nodes that each have their own computing power and storage. The data on a DFS can reside on various types of storage devices, such as solid-state drives and hard disk drives.

Data sets are replicated onto multiple servers, which enables redundancy to keep data highly available. The DFS is located on a collection of servers, mainframes or a cloud environment over a local area network (LAN) so multiple users can access and store unstructured data. If organizations need to scale up their infrastructure, they can add more storage nodes to the DFS.



Clients access data on a DFS using namespaces. Organizations can group shared folders into logical namespaces. A namespace is the shared group of networked storage on a DFS root. These present files to users as one shared folder with multiple subfolders. When a user requests a file, the DFS brings up the first available copy of the file.

Background:

The organization in this case study is a large multinational corporation with a complex IT infrastructure that supports a range of business operations. Over the years, the organization had accumulated a massive amount of data from various sources, including customer data, sales data, and marketing data. This data was critical to the organization's decision-making process, but managing it had become a major challenge.

The organization's existing file system was a traditional centralized file system that had been in place for several years. However, as the amount of data continued to grow, the file system struggled to keep up with the demand for fast and reliable data access. This led to slow access times, frequent crashes, and increased risk of data loss.

The organization realized that it needed a more scalable and reliable solution for managing its data. After considering various options, the organization decided to implement a distributed file system. The organization hoped that this solution would provide greater scalability and redundancy, as well as faster data access times and improved data security.

However, implementing a distributed file system was a major undertaking that required significant planning, resources, and expertise. The organization needed to carefully consider its options and choose the right solution that would meet its specific needs and requirements. Ultimately, the organization decided to work with a vendor that specialized in distributed file systems and had a proven track record of success in implementing these solutions for large organizations.

In the next section of this case study, we will explore the challenges that the organization faced with its existing file system, and the analysis that led to the decision to implement a distributed file system.

Challenges:

The organization faced several challenges with its existing centralized file system. These challenges included:

Scalability: The centralized file system was not designed to handle the massive volume of data that the organization had accumulated over the years. As a result, the system was slow and unresponsive, and data access times were unacceptable.

Reliability: The centralized file system was vulnerable to hardware failures and other issues that could lead to data loss. This was a major concern for the organization, as the data stored in the file system was critical to its business operations.

Security: The centralized file system was not designed to provide the level of security that the organization required. The organization was concerned about the risk of data breaches and other security threats, and needed a solution that could provide better data protection.

Cost: The cost of maintaining and upgrading the centralized file system was becoming increasingly expensive, and the organization needed a more cost-effective solution that could meet its growing data needs.

Given these challenges, the organization realized that it needed to explore alternative solutions for managing its data. After conducting a thorough analysis of various options, the organization decided to implement a distributed file system. This solution was seen as a way to address the scalability, reliability, security, and cost issues that the organization was facing with its existing file system.

Analysis:

To determine the best solution for managing its data, the organization conducted a thorough analysis of various options, including cloud-based solutions, network-attached storage (NAS) devices, and distributed file systems. After evaluating these options, the organization decided that a distributed file system would best meet its needs.

The organization's analysis focused on several key factors, including:

Scalability: The organization needed a solution that could handle its growing data needs and support future growth. A distributed file system was seen as the best option, as it would allow the organization to add more storage capacity and processing power as needed.

Reliability: The organization was concerned about the risk of data loss due to hardware failures or other issues. A distributed file system was seen as a more reliable option, as data could be stored across multiple servers, reducing the risk of data loss.

Security: The organization needed a solution that could provide better data protection and reduce the risk of data breaches. A distributed file system was seen as a more secure option, as data could be encrypted and stored across multiple servers, making it more difficult for hackers to access.

Cost: The organization needed a solution that could provide better value for money and reduce the cost of maintaining and upgrading its file system. A distributed file system was seen as a more cost-effective option, as it would allow the organization to leverage existing hardware resources and reduce the need for costly upgrades.

Based on these factors, the organization decided that a distributed file system was the best solution for its data management needs. The organization then worked with a vendor that specialized in distributed file systems to develop and implement the new system. In the next section of this case study, we will explore the solution that was ultimately implemented and the results that were achieved.

Solution:

After careful analysis and evaluation of various options, the organization decided to implement a distributed file system to manage its growing data needs. The organization worked with a vendor that specialized in distributed file systems to develop and implement the new system.

The new distributed file system was designed to provide the following benefits:

Scalability: The distributed file system was designed to be highly scalable, allowing the organization to add more storage capacity and processing power as needed. The system could be easily expanded by adding more servers to the network.

Reliability: The distributed file system was designed to be highly reliable, with data stored across multiple servers to reduce the risk of data loss due to hardware failures or other issues. The system was also designed to automatically detect and repair any issues that did arise.

Security: The distributed file system was designed to be highly secure, with data encrypted and stored across multiple servers to reduce the risk of data breaches. The

system also included advanced security features such as access controls, user authentication, and auditing.

Cost-effectiveness: The distributed file system was designed to be more cost-effective than the organization's previous centralized file system. The system was designed to leverage existing hardware resources and reduce the need for costly upgrades. In addition, the system was designed to be easy to manage and maintain, reducing the cost of IT operations.

The new distributed file system was implemented over a period of several months, with the vendor working closely with the organization to ensure a smooth transition. The new system was fully operational within six months of the project start date.

In the next section of this case study, we will explore the results that were achieved with the new distributed file system.

Conclusion:

In conclusion, the implementation of a distributed file system has provided significant benefits to the organization, allowing it to manage its growing data needs more effectively and efficiently. The new system has improved scalability, reliability, security, and cost-effectiveness, providing a solid foundation for future growth and expansion.

The organization worked closely with a vendor that specialized in distributed file systems to design and implement the new system. The implementation was completed over a period of several months, with the vendor providing ongoing support and assistance throughout the process.

Overall, the organization has been highly satisfied with the new distributed file system, and has experienced significant improvements in several key areas. The organization is now well-positioned to manage its growing data needs, and is able to focus on other strategic IT initiatives knowing that its file system is secure, reliable, and scalable.

Based on the success of this project, the organization plans to continue working with the vendor to explore additional ways in which it can leverage distributed computing technologies to improve its IT infrastructure and operations.

Experiment No.: -10

Aim: - Case Study on Android stack.

Introduction:

The Android stack is a complex system of software components that provides the foundation for the Android operating system. It is made up of multiple layers of software, each providing different functionality to the overall system. The Android stack is designed to be open and flexible, allowing developers to create applications that can run on a wide range of devices with different hardware configurations.

The Android stack includes several layers, including the Linux kernel, the Android runtime, the application framework, and the applications themselves. The Linux kernel is the core component of the Android stack, providing low-level system services such as memory management, process management, and device drivers. The Android runtime provides a virtual machine that runs Android applications, while the application framework provides a set of APIs that developers can use to create applications that integrate with the overall Android system. The applications themselves are the user-facing components of the Android system, providing functionality such as messaging, browsing, and gaming.

The Android stack presents a range of challenges for developers, including fragmentation, security, and complexity. However, it also provides a powerful and flexible platform for creating engaging and immersive applications that take full advantage of the capabilities of modern mobile devices. In this case study, we will explore the Android stack in detail, examining its various layers, its challenges and opportunities, and its impact on the mobile application development industry.

Background:

The Android stack is composed of several layers, including the following:

Linux kernel layer: The Linux kernel layer provides the basic functionality of the Android operating system, including device drivers, memory management, and process management.

Native libraries layer: The native libraries layer provides a set of libraries that enable developers to write native code for the Android platform using C and C++.

Android runtime layer: The Android runtime layer provides a set of core libraries that enable developers to write Java-based applications for the Android platform. The Android runtime includes the Dalvik virtual machine, which is optimized for mobile devices.

Application framework layer: The application framework layer provides a set of APIs and services that enable developers to build rich applications for the Android platform. The application framework includes a range of components, including activity and service components, content providers, and broadcast receivers.

Applications layer: The applications layer includes the user-facing applications that are built on top of the Android platform. These applications include core system applications such as the phone dialer, browser, and messaging applications, as well as third-party applications developed by independent developers.

Challenges:

The Android stack provides a rich set of functionality and capabilities, but it also presents a range of challenges for developers and users. Some of the key challenges of the Android stack include the following:

Fragmentation: The Android stack is used across a wide range of devices, with different hardware configurations, screen sizes, and other variables. This can lead to fragmentation in the platform, making it difficult for developers to create applications that work seamlessly across all devices.

Security: The Android platform is a popular target for malware and other security threats, due to its popularity and open nature. As a result, the Android stack includes a range of security features, but these features can be complex and difficult to manage.

Complexity: The Android stack is composed of multiple layers and components, which can be complex and difficult to understand for developers and users. This complexity can make it challenging to develop and deploy applications on the Android platform.

Analysis:

Despite these challenges, the Android stack provides a powerful and flexible platform for mobile devices. The stack is designed to provide a rich set of functionality and capabilities, while also enabling developers to build robust applications that run seamlessly across a wide range of devices.

One of the key strengths of the Android stack is its open nature, which enables developers to customize and extend the platform in a variety of ways. This openness has led to a vibrant ecosystem of third-party applications and services, providing users with a rich set of options for customizing and enhancing their devices.

In addition, the Android stack includes a range of advanced features and capabilities, such as support for multitasking, advanced graphics and animation capabilities, and a rich set of multimedia features. These features enable developers to create rich, engaging applications that take full advantage of the capabilities of modern mobile devices.

Solution:

To address some of the challenges of the Android stack, developers can take a range of steps, such as the following:

Developing for multiple screen sizes: To address fragmentation in the Android platform, developers can build applications that are optimized for a range of screen sizes and resolutions. This can help ensure that applications work seamlessly across a wide range of devices.

Utilizing security features: To address security challenges, developers can utilize the range of security features provided by the Android platform.

Conclusion:

In conclusion, the Android stack is a complex and powerful platform for mobile devices, providing users with a rich set of features and capabilities. The stack includes multiple layers, each providing different functionality to the overall system, and is designed to be open and flexible, enabling developers to create robust applications that run seamlessly across a wide range of devices.

While the Android stack presents a range of challenges, such as fragmentation, security, and complexity, developers can take steps to address these challenges and create high-quality applications that meet the needs of users. By leveraging the advanced features and capabilities of the Android platform, developers can create engaging and immersive applications that take full advantage of the capabilities of modern mobile devices. Overall, the Android stack is a powerful and flexible platform that provides a range of opportunities for developers and users alike.

06/04.