**Program No.1**

**//WAP to simulate the functionality of Lamport's Logical clock in C.**

#include<stdio.h>

#include<conio.h>

#include<iostream.h>

#include<stdlib.h>

#include<graphics.h>

#include<string.h>

#include<dos.h>

void main(){

 int s[4][9],n,m=0;

 int i,j,next=0,step=0;

 int msg[10][4]={0},totmsg;

 char op;

 int pi,pj,ei,ej;

 clrscr();

 cout<<"\nProgram for Lamport Logical Clock";

 cout<<"\nEnter Number Of Process ";

 cin>>n;

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

 cout<<"\nEnter number of STATES of process P"<<i<<" ";

 cin>>s[i][8];

 for(j=1;j<=s[i][8];j++){

 s[i][j]=j;

 }

 }

 do{

 cout<<"\nEnter message transit";

 cout<<"\nFROM ->\nEnter Process Number P";

 cin>>msg[m][0];

 cout<<"\nEnter Event Number e";

 cin>>msg[m][1];

 cout<<"\nTO ->\nEnter Process Number P";

 cin>>msg[m][2];

 cout<<"\nEnter Event Number e";

 cin>>msg[m][3];

 cout<<"\n\nPress 'y' to continue";

 op=getch();

 cout<<op;

 m++;

 totmsg=m;

 }while(op=='y');

 m=0;

 for (i=0;i<totmsg;i++){

 pi=msg[i][0];

 ei=msg[i][1];

 pj=msg[i][2];

 ej=msg[i][3];

 if(s[pj][ej]< (s[pi][ei]+1)){

 s[pj][ej]=s[pi][ei]+1;

 for (j=ej+1;j<=s[pj][8];j++){

 s[pj][j]=s[pj][j-1]+1;

 }

 }

 }

 int gd=DETECT,gm;

 initgraph(&gd,&gm,"C:\\TC\\BGI");

 outtextxy(200,15,"Program For Lamport Logical Clock");

 //drawing process and events

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

 char\* p1;

 itoa(i,p1,10);

 outtextxy(5,100+next,"P");

 outtextxy(13,100+next,p1);

 line(100,100+next,600,100+next);

 for(j=1;j<=s[i][8];j++){

 char\* p2;

 itoa(j,p2,10);

 outtextxy(100+step,90+next,"e");

 outtextxy(110+step,90+next,p2);

 //timestamp

 char\* p3;

 itoa(s[i][j]-1,p3,10);

 outtextxy(100+step,110+next,"t");

 outtextxy(110+step,110+next,p3);

 circle(105+step,100+next,5);

 step+=50;

 }

 step=0;

 next+=100;

 }

 delay(2000);

 //drawing message transit

 for(m=0;m<totmsg;m++){

 setlinestyle(SOLID\_LINE,1,3);

 setcolor(m+4);

 line(msg[m][1]\*50+50,msg[m][0]\*100+100,msg[m][3]\*50+50,msg[m][2]\*100+100);

 if (msg[m][2]>msg[m][0]){

 line(msg[m][3]\*50+50,msg[m][2]\*100+100,msg[m][3]\*50+50,msg[m][2]\*100+90);

 line(msg[m][3]\*50+50,msg[m][2]\*100+100,msg[m][3]\*50+40,msg[m][2]\*100+90);

 }

 else{

 line(msg[m][3]\*50+50,msg[m][2]\*100+100,msg[m][3]\*50+50,msg[m][2]\*100+110);

 line(msg[m][3]\*50+50,msg[m][2]\*100+100,msg[m][3]\*50+40,msg[m][2]\*100+110);

 }

 }

 getch();

}

**Output**





**Program No.2**

**//WAP to Implement Vector clock in C.**

#include<stdio.h>

#include<conio.h>

#include<stdio.h>

#include<stdlib.h>

long \*p1(int i,long \*comp);

long \*p2(int i,long \*comp);

long \*p3(int i,long \*comp);

void main()

{

 long start[]={0,0,0},\*vector;

 clrscr();

 while(!kbhit())

 {

 p1(1,&start[0]);

 }

 printf("\n Process Vector\n");

 vector=p1(0,&start[0]);

 printf("p1[%ld%ld%ld]\n",\*vector,\*(vector+1),\*(vector+2));

 vector=p2(0,&start[0]);

 printf("p2[%ld%ld%ld]\n",\*vector,\*(vector+1),\*(vector+2));

 vector=p3(0,&start[0]);

 printf("p3[%ld%ld%ld]\n",\*vector,\*(vector+1),\*(vector+2));

}

 long \*p1(int i,long \*comp)

 {

 static long a[]={0,0,0};

 int next;

 if(i==1)

 {

 a[0]++;

 if(\*(comp+1)>a[1])

 a[1]=\*(comp+1);

 if(\*(comp+2)>a[2])

 a[2]=\*(comp+2);

 next=random(2);

 if(next==0)

 p2(1,&a[0]);

 else if(next==1)

 p3(1,&a[0]);

 return(&a[0]);

 }

 else

 return(&a[0]);

 }

long \*p2(int i,long \*comp)

{

 static long b[]={0,0,0};

 int next;

 if(i==1)

 {

 b[i]++;

 if(\*comp>b[0])

 b[0]=\*(comp);

 if(\*(comp+2)>b[2])

 b[2]=\*(comp+2);

 next=random(2);

 if(next==0)

 p1(1,&b[0]);

 else if(next==1)

 p3(1,&b[0]);

 return &b[0];

 }

 else

 return &b[0];

}

long \*p3(int i,long \*comp)

{

 static long c[]={0,0,0};

 int next;

 if(i==1)

 {

 c[2]++;

 if(\*comp>c[0])

 c[0]=\*(comp);

 if(\*(comp+1)>c[1])

 c[1]=\*(comp+1);

 next=random(2);

 if(next==0)

 p1(1,&c[0]);

 return &c[0];

 }

 else

 return &c[0];

}

**Program No.3**

**\\Simulation of Distributed mutual exclusion in java.**

import Utilities.\*;

import Synchronization.\*;

class Message { public int number, id;

 public Message(int number, int id) { this.number = number; this.id = id;}

}

class Node extends MyObject implements Runnable {

 private static final int MAIN = 0, REQUESTS = 1, REPLIES = 2;

 private int whichOne = 0;

 private int id = -1;

 private int numNodes = -1;

 private int napOutsideCS = 0; // both are in

 private int napInsideCS = 0; // milliseconds

 private MessagePassing[] requestChannel = null;

 private MessagePassing[] replyChannel = null;

 private MessagePassing requestsToMe = null;

 private MessagePassing repliesToMe = null;

 private int number = 0;

 private int highNumber = 0;

 private boolean requesting = false;

 private int replyCount = 0;

 private BinarySemaphore s = new BinarySemaphore(1);

 private BinarySemaphore wakeUp = new BinarySemaphore(0);

 private boolean[] deferred = null;

 public Node(String name, int id, int numNodes,

 int napOutsideCS, int napInsideCS,

 MessagePassing[] requestChannel, MessagePassing replyChannel[],

 MessagePassing requestsToMe, MessagePassing repliesToMe) {

 super(name + " " + id);

 this.id = id;

 this.numNodes = numNodes;

 this.napOutsideCS = napOutsideCS;

 this.napInsideCS = napInsideCS;

 this.requestChannel = requestChannel;

 this.replyChannel = replyChannel;

 this.requestsToMe = requestsToMe;

 this.repliesToMe = repliesToMe;

 deferred = new boolean[numNodes];

 for (int i = 0; i < numNodes; i++) deferred[i] = false;

 System.out.println(getName() + " is alive, napOutsideCS="

 + napOutsideCS + ", napInsideCS=" + napInsideCS);

 new Thread(this).start();

 }

 public void run() { // start three different threads in the same object

 int meDo = whichOne++;

 if (meDo == MAIN) {

 new Thread(this).start();

 main();

 } else if (meDo == REQUESTS) {

 new Thread(this).start();

 handleRequests();

 } else if (meDo == REPLIES) {

 handleReplies();

 }

 }

 private void chooseNumber() {

 P(s);

 requesting = true;

 number = highNumber + 1;

 V(s);

 }

 private void sendRequest() {

 replyCount = 0;

 for (int j = 0; j < numNodes; j++) if (j != id)

 send(requestChannel[j], new Message(number, id));

 }

 private void waitForReply() {

 P(wakeUp);

 }

 private void replyToDeferredNodes() {

 P(s);

 requesting = false;

 V(s);

 for (int j = 0; j < numNodes; j++) {

 if (deferred[j]) {

 deferred[j] = false;

 send(replyChannel[j], id);

 }

 }

 }

 private void outsideCS() {

 int napping;

 napping = ((int) random(napOutsideCS)) + 1;

 System.out.println("age()=" + age() + ", " + getName()

 + " napping outside CS for " + napping + " ms");

 nap(napping);

 }

 private void insideCS() {

 int napping;

 napping = ((int) random(napInsideCS)) + 1;

 System.out.println("age()=" + age() + ", " + getName()

 + " napping inside CS for " + napping + " ms");

 nap(napping);

 }

 private void main() {

 while (true) {

 outsideCS();

 System.out.println("age()=" + age() + ", node " + id

 + " wants to enter its critical section");

 chooseNumber(); // PRE-PROTOCOL

 sendRequest(); // "

 waitForReply(); // "

 insideCS();

 System.out.println("age()=" + age() + ", node " + id

 + " has now left its critical section");

 replyToDeferredNodes(); // POST-PROTOCOL

 }

 }

 private void handleRequests() {

 while (true) {

 Message m = (Message) receive(requestsToMe);

 int receivedNumber = m.number;

 int receivedID = m.id;

 highNumber = Math.max(highNumber, receivedNumber);

 P(s);

 boolean decideToDefer = requesting && (number < receivedNumber

 || (number == receivedNumber && id < receivedID));

 if (decideToDefer) deferred[receivedID] = true;

 else send(replyChannel[receivedID], id);

 V(s);

 }

 }

 private void handleReplies() {

 while (true) {

 int receivedID = receiveInt(repliesToMe);

 replyCount++;

 if (replyCount == numNodes - 1) V(wakeUp);

 }

 }

}

class DistributedMutualExclusion extends MyObject {

 public static void main(String[] args) {

 // parse command line options, if any, to override defaults

 GetOpt go = new GetOpt(args, "Un:R:");

 String usage = "Usage: -n numNodes -R runTime"

 + " napOutsideCS[i] napInsideCS[i] i=0,1,...";

 go.optErr = true;

 int ch = -1;

 int numNodes = 5;

 int runTime = 60; // seconds

 while ((ch = go.getopt()) != go.optEOF) {

 if ((char)ch == 'U') {

 System.out.println(usage); System.exit(0);

 }

 else if ((char)ch == 'n')

 numNodes = go.processArg(go.optArgGet(), numNodes);

 else if ((char)ch == 'R')

 runTime = go.processArg(go.optArgGet(), runTime);

 else {

 System.err.println(usage); System.exit(1);

 }

 }

 System.out.println("DistributedMutualExclusion: numNodes="

 + numNodes + ", runTime=" + runTime);

 // process non-option command line arguments

 int[] napOutsideCS = new int[numNodes];

 int[] napInsideCS = new int[numNodes];

 int argNum = go.optIndexGet();

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

 napOutsideCS[i] = go.tryArg(argNum++, 8);

 napInsideCS[i] = go.tryArg(argNum++, 2);

 }

 // create communication channels

 MessagePassing[] requestChannel = null, replyChannel = null,

 requestChannelS = null, requestChannelR = null,

 replyChannelS = null, replyChannelR = null;

 requestChannel = new MessagePassing[numNodes];

 replyChannel = new MessagePassing[numNodes];

 requestChannelS = new MessagePassing[numNodes];

 replyChannelS = new MessagePassing[numNodes];

 requestChannelR = new MessagePassing[numNodes];

 replyChannelR = new MessagePassing[numNodes];

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

 requestChannel[i] = new AsyncMessagePassing();

 replyChannel[i] = new AsyncMessagePassing();

 requestChannelS[i] = new MessagePassingSendOnly(requestChannel[i]);

 replyChannelS[i] = new MessagePassingSendOnly(replyChannel[i]);

 requestChannelR[i] = new MessagePassingReceiveOnly(requestChannel[i]);

 replyChannelR[i] = new MessagePassingReceiveOnly(replyChannel[i]);

 }

 // create the Nodes (they start their own threads)

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

 new Node("Node", i, numNodes,

 napOutsideCS[i]\*1000, napInsideCS[i]\*1000,

 requestChannelS, replyChannelS,

 requestChannelR[i], replyChannelR[i]);

 System.out.println("All Nodes created");

 // let the Nodes run for a while

 nap(runTime\*1000);

 System.out.println("age()=" + age()

 + ", time to stop the threads and exit");

 System.exit(0);

 }

}

**Output:**

D:\Prakash\Java\RND\Advanced>javac dimu.java

D:\ Prakash\Java\RND\Advanced >java DistributedMutualExclusion -R20

DistributedMutualExclusion: numNodes=5, runTime=20

Node 0 is alive, napOutsideCS=8000, napInsideCS=2000

Node 1 is alive, napOutsideCS=8000, napInsideCS=2000

Node 2 is alive, napOutsideCS=8000, napInsideCS=2000

Node 3 is alive, napOutsideCS=8000, napInsideCS=2000

Node 4 is alive, napOutsideCS=8000, napInsideCS=2000

age()=170, Node 1 napping outside CS for 2719 ms

age()=170, Node 2 napping outside CS for 279 ms

All Nodes created

age()=170, Node 3 napping outside CS for 2355 ms

age()=220, Node 0 napping outside CS for 2393 ms

age()=220, Node 4 napping outside CS for 8 ms

age()=220, node 4 wants to enter its critical section

age()=330, Node 4 napping inside CS for 911 ms

age()=440, node 2 wants to enter its critical section

age()=1260, node 4 has now left its critical section

age()=1260, Node 4 napping outside CS for 4042 ms

age()=1260, Node 2 napping inside CS for 183 ms

age()=1480, node 2 has now left its critical section

age()=1480, Node 2 napping outside CS for 7335 ms

age()=2530, node 3 wants to enter its critical section

age()=2530, Node 3 napping inside CS for 741 ms

age()=2580, node 0 wants to enter its critical section

age()=2860, node 1 wants to enter its critical section

age()=3300, node 3 has now left its critical section

age()=3300, Node 3 napping outside CS for 6849 ms

age()=3300, Node 0 napping inside CS for 1710 ms

age()=5000, node 0 has now left its critical section

age()=5000, Node 0 napping outside CS for 5253 ms

age()=5000, Node 1 napping inside CS for 1694 ms

age()=5330, node 4 wants to enter its critical section

age()=6700, node 1 has now left its critical section

age()=6700, Node 1 napping outside CS for 3063 ms

age()=6700, Node 4 napping inside CS for 397 ms

age()=7140, node 4 has now left its critical section

age()=7140, Node 4 napping outside CS for 3687 ms

age()=8790, node 2 wants to enter its critical section

age()=8790, Node 2 napping inside CS for 102 ms

age()=8900, node 2 has now left its critical section

age()=8900, Node 2 napping outside CS for 1174 ms

age()=9780, node 1 wants to enter its critical section

age()=9780, Node 1 napping inside CS for 1617 ms

age()=10110, node 2 wants to enter its critical section

age()=10160, node 3 wants to enter its critical section

age()=10270, node 0 wants to enter its critical section

age()=10820, node 4 wants to enter its critical section

age()=11430, node 1 has now left its critical section

age()=11430, Node 1 napping outside CS for 5326 ms

age()=11430, Node 2 napping inside CS for 628 ms

age()=12090, node 2 has now left its critical section

age()=12090, Node 2 napping outside CS for 4970 ms

age()=12090, Node 3 napping inside CS for 545 ms

age()=12630, node 3 has now left its critical section

age()=12630, Node 3 napping outside CS for 7989 ms

age()=12630, Node 0 napping inside CS for 904 ms

age()=13510, node 0 has now left its critical section

age()=13510, Node 0 napping outside CS for 4162 ms

age()=13510, Node 4 napping inside CS for 1440 ms

age()=15000, node 4 has now left its critical section

age()=15000, Node 4 napping outside CS for 2578 ms

age()=16750, node 1 wants to enter its critical section

age()=16750, Node 1 napping inside CS for 123 ms

age()=16860, node 1 has now left its critical section

age()=16860, Node 1 napping outside CS for 3709 ms

age()=17030, node 2 wants to enter its critical section

age()=17030, Node 2 napping inside CS for 97 ms

age()=17140, node 2 has now left its critical section

age()=17140, Node 2 napping outside CS for 7901 ms

age()=17580, node 4 wants to enter its critical section

age()=17580, Node 4 napping inside CS for 1695 ms

age()=17690, node 0 wants to enter its critical section

age()=19280, node 4 has now left its critical section

age()=19280, Node 4 napping outside CS for 3751 ms

age()=19280, Node 0 napping inside CS for 869 ms

age()=20160, node 0 has now left its critical section

age()=20160, Node 0 napping outside CS for 6489 ms

age()=20160, time to stop the threads and exit

 ... end of example run(s) \*/

**Program No.4**

**Implement a distributed chat server using TCP sockets in java.**

**1.Server.java**

import java.net.\*;

import java.io.\*;

public class server{

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

 ServerSocket s1=null;

 try{

 s1=new ServerSocket(98);

 }catch(Exception e){

 System.out.println("Port not found");

 e.printStackTrace();

 }

 Socket c=null;

 try{

 c=s1.accept();

 System.out.println("Connection from"+c);

 }catch(Exception e){

 System.out.println("not accepted");

 e.printStackTrace();

 }

 PrintWriter out=new PrintWriter(c.getOutputStream(),true);

 BufferedReaderin=new BufferedReader(new InputStreamReader(c.getInputStream()));

 String l;

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

 System.out.println("I am ready type now");

 while((l=sin.readLine())!=null){

 out.println(l);

 }

 out.close();

 sin.close();

 c.close();

 s1.close();

 }

}

**2.Client.java**

import java.net.\*;

import java.io.\*;

public class client{

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

 Socket s=null;

 BufferedReader b=null;

 try{

 s=new Socket(InetAddress.getLocalHost(),98);

 b=new BufferedReader(new InputStreamReader(s.getInputStream()));

 }catch(Exception e){

 System.out.println("I do not host");

 e.printStackTrace();

 }

 String inp;

 while((inp=b.readLine())!=null){

 System.out.println(inp);

 }

 b.close();

 s.close();

 }

}

**Running the application**

Open two cmd prompt and follow these

1.java Server

2.java client

**Output**

D:\Prakash\RND\Java\NetWorking\ChatServer>java server

Connection fromSocket[addr=/127.0.0.1,port=1120,localport=98]

I am ready type now

Hello how r u? dude…

D:\Prakash\RND\Java\NetWorking\ChatServer>java client

Hello how r u? dude…

**Program No.5**

**Implement‘Java RMI’mechanism for accessing methods of remote systems.**

1.CalculatorImpl.java

public class CalculatorImpl

 extends

 java.rmi.server.UnicastRemoteObject

 implements Calculator {

 public CalculatorImpl()

 throws java.rmi.RemoteException {

 super();

 }

 public long add(long a, long b)

 throws java.rmi.RemoteException {

 return a + b;

 }

 public long sub(long a, long b)

 throws java.rmi.RemoteException {

 return a - b;

 }

 public long mul(long a, long b)

 throws java.rmi.RemoteException {

 return a \* b;

 }

 public long div(long a, long b)

 throws java.rmi.RemoteException {

 return a / b;

 }

}

2.Calculator.java

public interface Calculator

 extends java.rmi.Remote {

 public long add(long a, long b)

 throws java.rmi.RemoteException;

 public long sub(long a, long b)

 throws java.rmi.RemoteException;

 public long mul(long a, long b)

 throws java.rmi.RemoteException;

 public long div(long a, long b)

 throws java.rmi.RemoteException;

}

3.CalculatorServer.java

import java.rmi.Naming;

public class CalculatorServer {

 public CalculatorServer() {

 try {

 Calculator c = new CalculatorImpl();

 Naming.rebind("rmi://localhost:1099/CalculatorService", c);

 } catch (Exception e) {

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

 }

 }

public static void main(String args[]) {

 new CalculatorServer();

 }

}

4.CalculatorClient.java

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;

 public class CalculatorClient {

 public static void main(String[] args) {

 try {

 Calculator c = (Calculator)

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

 System.out.println( c.sub(4, 3) );

 System.out.println( c.add(4, 5) );

 System.out.println( c.mul(3, 6) );

 System.out.println( c.div(9, 3) );

 }

 catch (MalformedURLException murle) {

 System.out.println();

 System.out.println("MalformedURLException");

 System.out.println(murle);

 }

 catch (RemoteException re) {

 System.out.println();

 System.out.println("RemoteException");

 System.out.println(re);

 }

 catch (NotBoundException nbe) {

 System.out.println();

 System.out.println(

 "NotBoundException");

 System.out.println(nbe);

 }

 catch ( java.lang.ArithmeticException ae) {

 System.out.println();

 System.out.println(

 "java.lang.ArithmeticException");

 System.out.println(ae);

 }

 }

}

**Running The Application:**

D:\Prakash\RND\Java\NetWorking\RMI>rmic CalculatorImpl

Now open three cmd prompts and follow these at each.

1. D:\Prakash\RND\Java\NetWorking\RMI>Rmiregistry

2. D:\Prakash\RND\Java\NetWorking\RMI>java CalculatorServer

3. D:\Prakash\RND\Java\NetWorking\RMI>java CalculatorClient

**Output:**

If all goes well you will see the following output:

1

9

18

3

**Program No.6**

**Implementation of CORBA (Common Object Request Broker Architecture) mechanism.**

1.FileInterface.idl

interface FileInterface {

 typedef sequence<octet> Data;

 Data downloadFile(in string fileName);

};

Now, let's compile the FileInterface.idl and generate server-side skeletons. Using the command:

D:\Prakash\RND\Java\CORBA> idlj -fserver FileInterface.idl

2.FileServant.java

import java.io.\*;

public class FileServant extends \_FileInterfaceImplBase {

 public byte[] downloadFile(String fileName){

 File file = new File(fileName);

 byte buffer[] = new byte[(int)file.length()];

 try {

 BufferedInputStream input = new

 BufferedInputStream(new FileInputStream(fileName));

 input.read(buffer,0,buffer.length);

 input.close();

 } catch(Exception e) {

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

 e.printStackTrace();

 }

 return(buffer);

 }

}

3.FileServer.java

import java.io.\*;

import org.omg.CosNaming.\*;

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

import org.omg.CORBA.\*;

public class FileServer {

 public static void main(String args[]) {

 try{

 // create and initialize the ORB

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

 // create the servant and register it with the ORB

 FileServant fileRef = new FileServant();

 orb.connect(fileRef);

 // get the root naming context

 org.omg.CORBA.Object objRef =

 orb.resolve\_initial\_references("NameService");

 NamingContext ncRef = NamingContextHelper.narrow(objRef);

 // Bind the object reference in naming

 NameComponent nc = new NameComponent("FileTransfer", " ");

 NameComponent path[] = {nc};

 ncRef.rebind(path, fileRef);

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

 // Wait for invocations from clients

 java.lang.Object sync = new java.lang.Object();

 synchronized(sync){

 sync.wait();

 }

 } catch(Exception e) {

 System.err.println("ERROR: " + e.getMessage());

 e.printStackTrace(System.out);

 }

 }

}

4.FileClient.java

import java.io.\*;

import java.util.\*;

import org.omg.CosNaming.\*;

import org.omg.CORBA.\*;

public class FileClient {

 public static void main(String argv[]) {

 try {

 // create and initialize the ORB

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

 // get the root naming context

 org.omg.CORBA.Object objRef =

 orb.resolve\_initial\_references("NameService");

 NamingContext ncRef = NamingContextHelper.narrow(objRef);

 NameComponent nc = new NameComponent("FileTransfer", " ");

 // Resolve the object reference in naming

 NameComponent path[] = {nc};

 FileInterfaceOperations fileRef =

 FileInterfaceHelper.narrow(ncRef.resolve(path));

 if(argv.length < 1) {

 System.out.println("Usage: java FileClient filename");

 }

 // save the file

 File file = new File(argv[0]);

 byte data[] = fileRef.downloadFile(argv[0]);

 BufferedOutputStream output = new

 BufferedOutputStream(new FileOutputStream(argv[0]));

 output.write(data, 0, data.length);

 output.flush();

 output.close();

 } catch(Exception e) {

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

 e.printStackTrace();

 }

 }

}

**Running the application**

**1.** D:\Prakash\RND\Java\CORBA>tnameserv

2. D:\Prakash\RND\Java\CORBA>java FileServer

3. D:\Prakash\RND\Java\CORBA>idlj -fclient FileInterface.idl

4. D:\Prakash\RND\Java\CORBA>java FileClient hello.txt

**Output:**

****

**Program No.7**

**Write a java program for implementing sliding window protocol.**

**1. slic.java**

//SLIDING WINDOW PROTOCOL – CLIENT

import java.io.\*;

import java.net.\*;

public class slic{

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

Socket s = new Socket(“local host”,8888);

Buffered Reader from server = new BufferReader(new InputStreamReader(s.getInputStream()));

DataOutputStream toserver = new DataOutputStream(s.getOutputStream());

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

String dout,din;

System.out.println(“\t”+fromserver.readLine());

System.out.println(“enter quit to exit”);

System.out.println(“enter data for server :”);

While(true)

{

Dout = d.readLine();

If(dout.equals(“quit”))

Break;

toserver.writeBytes(dout+’\n’);

din = fromserver.readLine();

System.out.println(“Server :”+din);

System.out.println(“\nEnter for server :”);

}

 }

}

**2. slis.java**

//SLIDING WINDOW PROTOCOL – SERVER

import java.io.\*;

import java.net.\*;

public class slis{

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

ServerSocket ss = new ServerSocket(8888);

System.out.println(“\t waiting for client…”);

Socket client = ss.accept();

BufferedReader fromclient = new BufferedReader(new InputStreamReader(client.getInputStream()));

DataOutputStream toclient = new DataOutputStream(client.getOutputStream());

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

String dout,din;

toclient.writeBytes(“Server ready….”+’\n’);

while(true){

din = fromclient.readLine();

System.out.println(“\n client data:”+din);

System.out.println(“enter for client :”);

dout = d.readLine();

if(dout.equals(“quit”))

break;

toclient.writeBytes(dout+’\n’);

}

}

}