**Practical no:1**

**Remote Process Communication**

**Aim: Develop a program for a multi-client chat server.(1A)**

**Concept:** Develop a multi-client chat server application where clients chat with each other concurrently. The messages sent by clients are communicated to the server and then the server, on behalf of the source client,communicates the messages to the appropriate destination client.

**Source Code:**

**ChatClient.java** //1st file

package com.mycompany.chatclient;

import java.io.\*;

import java.net.\*;

public class ChatClient {

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public ChatClient()

{

try{

soc=new Socket(InetAddress.getLocalHost(),9999);

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

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

System.out.println("Chat client started.");

while(true){

str=br.readLine();

out.println(str);

new ChatServer();

} }

catch(Exception e){

}

}

class ChatServer extends Thread

{

String str1;

ChatServer()

{

try{

br1=new BufferedReader(new InputStreamReader(soc.getInputStream()));

start();

}

catch(Exception e)

{}

}

public void run(){

try{

str1=br1.readLine();

System.out.println("Server says:"+str1);

}

catch(Exception e){}

}

}

public static void main(String[] args)

{

new ChatClient();

}}

**ChatServer.java** //2nd file

package com.mycompany.chatclient;

import java.io.\*;

import java.net.\*;

public class ChatServer extends Thread{

ServerSocket ss;

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public ChatServer()

{

try{

ss=new ServerSocket(9999);

soc=ss.accept();

br=new BufferedReader(new InputStreamReader(soc.getInputStream()));

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

System.out.println("Chat server started.");

start();

new ChatServer1();

}

catch(Exception e)

{} }

public void run(){

try{

while(true)

{

str=br.readLine();

System.out.println("Client says:"+str);

} }

catch(Exception e)

{} }

class ChatServer1{

String str1;

ChatServer1()

{

try{

br1=new BufferedReader(new InputStreamReader(System.in));

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

while(true)

{

str1=br1.readLine();

out.println(str1);

} }

catch(Exception e)

{}}}

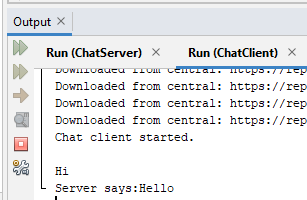
public static void main(String[] args)

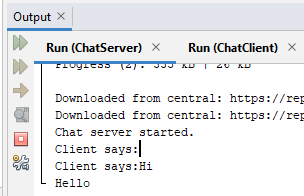
{

new ChatServer();

}}

**Output:**





**Practical No:1-B**

**Aim: Implement a server to find whether an entered string a palindrome using socket**

Source Code:

**PalinClient.java** //1st File

package com.mycompany.palinclient;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.InetAddress;

import java.net.Socket;

public class PalinClient {

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public PalinClient()

{

try{

soc=new Socket("127.0.0.1",8765);

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

br1=new BufferedReader(new InputStreamReader(soc.getInputStream()));

out=new PrintWriter(soc.getOutputStream());

while(true){

System.out.println("Enter the message: ");

str=br.readLine();

out.println(str);

out.flush();

System.out.println("Message from server: ");

str=br1.readLine();

System.out.println(str);

if(str.equals("q"))

break; }

soc.close();

}

catch(Exception e){

}

}

public static void main(String[] args)

{

new PalinClient();

}

}

**PalinServer.java** //2nd File

package com.mycompany.palinclient;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import java.net.ServerSocket;

import java.net.Socket;

public class PalinServer {

ServerSocket ss;

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public PalinServer()

{

try{

ss=new ServerSocket(8765);

System.out.println("Server is listening to port 8765");

soc=ss.accept();

System.out.println("Connection Established");

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

br1=new BufferedReader(new InputStreamReader(soc.getInputStream()));

out=new PrintWriter(soc.getOutputStream());

while(true)

{

System.out.println("Message from client");

str=br1.readLine();

int k=str.length();

System.out.println(str);

int left=0, right=k-1, flag=1;

while(left<=right)

{

if(str.charAt(left)!=(str.charAt(right)))

{

flag=0;

break;

}

else

{

left++; right--;

}

}

if(flag==0)

System.out.println("Not a palindrome string");

else

System.out.println("Is a palindrome string");

out.println(str);

out.flush();

if(str.equals("q"))

break;

}

}

catch(Exception e)

{}

}

public static void main(String[] args)

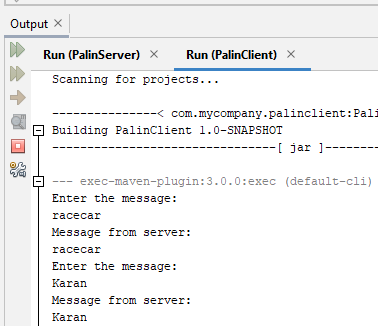
{

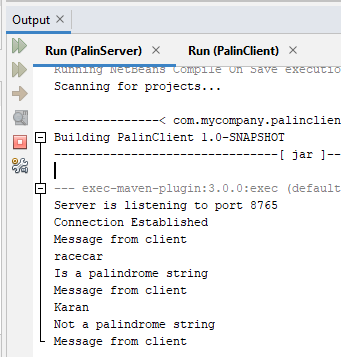
new PalinServer();

}

}

**Output:**

****

****

**Practical No:1-C**

**Aim: Implement a program to return the reverse of the string using socket**

**Source Code:**

**ReverseClient1.java**

package com.mycompany.reverseclient1;

import java.net.\*;

import java.io.\*;

public class ReverseClient1 {

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public ReverseClient1(){

try{

soc=new Socket("127.0.0.1",8765);

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

br1=new BufferedReader(new InputStreamReader(soc.getInputStream()));

out=new PrintWriter(soc.getOutputStream());

while(true){

System.out.println("Enter the message: ");

str=br.readLine();

out.println(str);

out.flush();

System.out.println("Message from server: ");

str=br1.readLine();

System.out.println(str);

if(str.equals("q"))

break;}

soc.close();}

catch (Exception e){}}

public static void main(String[] args) {

new ReverseClient1();}}

**ReverseServer1.java**

package com.mycompany.reverseclient1;

import java.net.\*;

import java.io.\*;

public class ReverseServer1 {

ServerSocket ss;

Socket soc;

BufferedReader br,br1;

PrintWriter out;

String str;

public ReverseServer1(){

try{

ss=new ServerSocket(8765);

System.out.println("Server is listening to port 8765");

soc=ss.accept();

System.out.println("Connection established!!");

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

br1=new BufferedReader(new InputStreamReader(soc.getInputStream()));

out=new PrintWriter(soc.getOutputStream());

while(true){

System.out.println("Message from client");

str=br1.readLine();

int k=str.length();

System.out.println(str);

String reverse="";

for(int i=k-1;i>=0;i--)

{ reverse=reverse+str.charAt(i);}

System.out.println("Reverse of the string is: "+reverse);

out.println(reverse);

out.flush();

if(str.equals("q"))

break;}

soc.close();}

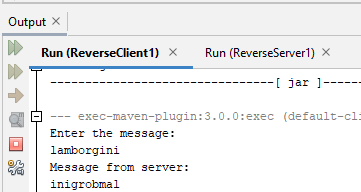
catch (Exception e){}}

public static void main(String[] args) {

new ReverseServer1();

}}

**Output**:



**Practical No:2**

**Remote Procedure Call**

**Concept:**A remote procedure call is an inter process communication technique that is used for client-server-based applications. A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.

**A)Aim:Implement a Server calculator Add(), Mul(), Sub(), Div() using datagram**

**Source Code:**

**RPCCalClient.java**

package com.mycompany.rpccalclient;

import java.net.\*;

import java.io.\*;

public class RPCCalClient {

RPCCalClient(){

try{

InetAddress ia=InetAddress.getLocalHost();

DatagramSocket ds=new DatagramSocket();

DatagramSocket ds1=new DatagramSocket(1300);

System.out.println("\nRPC Client.\n");

System.out.println("Enter method name and parameter like add 3 4:");

while(true){

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

String str=br.readLine();

byte b[]=str.getBytes();

DatagramPacket dp=new DatagramPacket(b,b.length,ia,1200);

ds.send(dp);

dp=new DatagramPacket(b,b.length);

ds1.receive(dp);

String s=new String(dp.getData(),0,dp.getLength());

System.out.println("\nResult="+s+"\n"); } }

catch(Exception e){}}

public static void main(String[] args) {

new RPCCalClient();}}

**RPCCalServer.java**

package com.mycompany.rpccalserver;

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class RPCCalServer {

DatagramPacket dp;

DatagramSocket ds;

String str,methodName,result;

int val1,val2;

RPCCalServer(){

try{

ds=new DatagramSocket(1200);

byte b[]=new byte[4096];

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

while(true){

dp=new DatagramPacket(b,b.length);

ds.receive(dp);

str=new String(dp.getData(),0,dp.getLength());

if(str.equalsIgnoreCase("q")){

System.exit(1);}

else{

StringTokenizer st=new StringTokenizer(str," ");

int i=0;

while(st.hasMoreElements()){

String token=st.nextToken();

methodName=token;

val1=Integer.parseInt(st.nextToken());

val2=Integer.parseInt(st.nextToken());

} }

System.out.println(str);

if(methodName.equalsIgnoreCase("add")){

result=""+add(val1,val2);}

else if(methodName.equalsIgnoreCase("sub")){

result=""+sub(val1,val2);}

else if(methodName.equalsIgnoreCase("mul")){

result=""+mul(val1,val2);}

else if(methodName.equalsIgnoreCase("div")){

result=""+div(val1,val2);}

else{

System.out.println("Enter a valid operation");}

byte b1[]=result.getBytes();

DatagramSocket ds1=new DatagramSocket();

DatagramPacket dp1=new DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1300);

System.out.println("Result: "+result+"\n");

ds1.send(dp1);} }

catch(Exception e){} }

public int add(int val1,int val2){

return val1+val2;}

public int sub(int val1,int val2){

return val1-val2;}

public int mul(int val1,int val2){

return val1\*val2;}

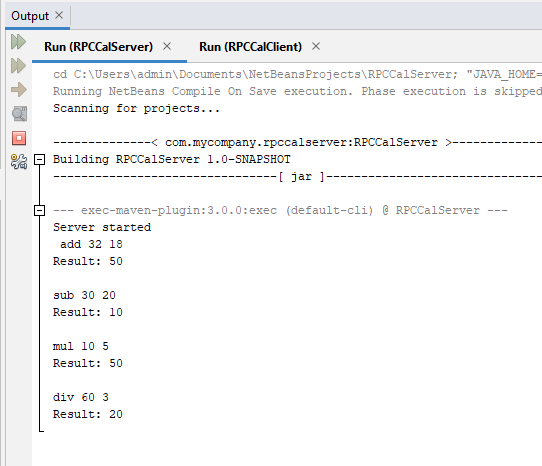
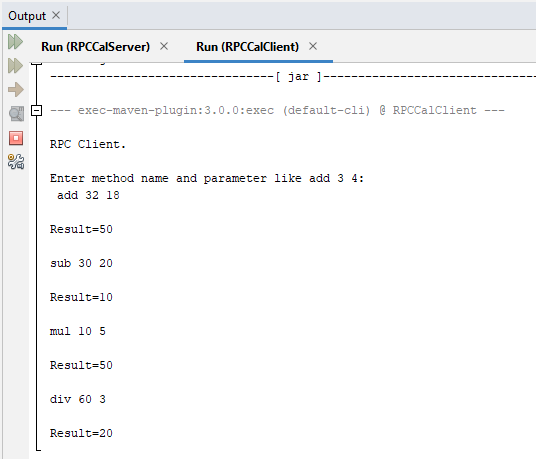
public int div(int val1,int val2){

return val1/val2;}

public static void main(String[] args) {

new RPCCalServer(); }}

**Output:**

****

**B)Aim:Implement a Date Time Server containing date() and time() using datagram**

**Source Code:**

**DateTimeClient.java**

package com.mycompany.datetimeclient;

import java.net.\*;

import java.io.\*;

public class DateTimeClient {

DateTimeClient(){

try{

InetAddress ia=InetAddress.getLocalHost();

DatagramSocket ds=new DatagramSocket();

DatagramSocket ds1=new DatagramSocket(1300);

System.out.println("\nDate Time Client\n");

byte b1[]=new byte[1000];

while(true)

{

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

String str=br.readLine();

byte b[]=str.getBytes();

DatagramPacket dp=new DatagramPacket(b,b.length,ia,1200);

ds.send(dp);

dp=new DatagramPacket(b1,b1.length);

ds1.receive(dp);

String s=new String(dp.getData(),0,dp.getLength());

System.out.println("\nResult="+s+"\n");

}}

catch(Exception e){}}

public static void main(String[] args) {

new DateTimeClient();}}

**DateTimeServer.java**

package com.mycompany.datetimeserver;

import java.net.\*;

import java.io.\*;

import java.util.\*;

import java.text.SimpleDateFormat;

public class DateTimeServer {

DatagramPacket dp;

DatagramSocket ds;

String str,methodName,result;

DateTimeServer() {

try{

ds=new DatagramSocket(1200);

byte b[]=new byte[4096];

System.out.println("\n Date Time Server \n");

while(true)

{

dp=new DatagramPacket(b,b.length);

ds.receive(dp);

str=new String(dp.getData(),0,dp.getLength());

if(str.equalsIgnoreCase("q"))

System.exit(1);

else

{

StringTokenizer st=new StringTokenizer(str," ");

int i=0;

while(st.hasMoreTokens())

{

String token=st.nextToken();

methodName=token;}}

Calendar c=Calendar.getInstance();

SimpleDateFormat dateFormat=new SimpleDateFormat("MM/dd/yyyy");

Date d=c.getTime();

InetAddress ia=InetAddress.getLocalHost();

if(methodName.equalsIgnoreCase("date"))

result=""+dateFormat.format(d);

else if(methodName.equalsIgnoreCase("time"))

{

result=""+d.getHours()+":"+d.getMinutes()+":"+d.getSeconds();

}

byte b1[]=result.getBytes();

DatagramSocket ds1=new DatagramSocket();

DatagramPacket dp1=new DatagramPacket(b1,b1.length,ia,1300);

System.out.println("result:"+result+"\n");

ds1.send(dp1);

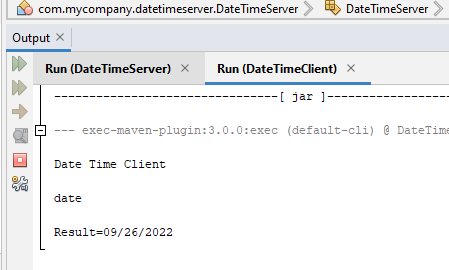
}}

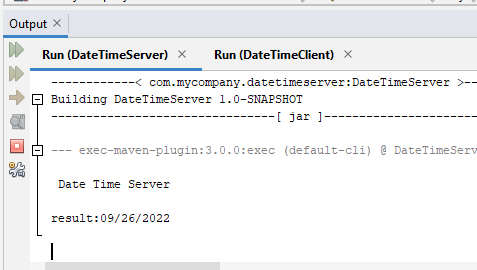
catch(Exception e){}}

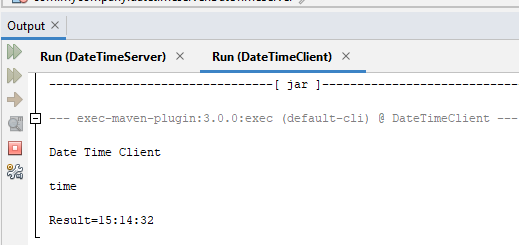
public static void main(String[] args) {

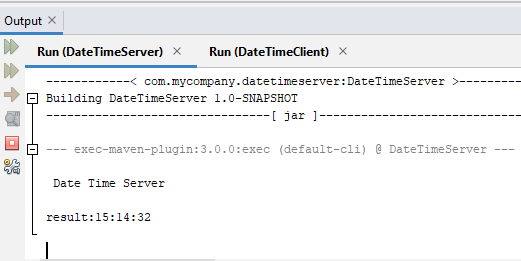
new DateTimeServer();}}

**Output:**

****

****

****

****

**Practical No:3**

**Remote Method Interface**

**Concept: The Remote Method Invocation is an API that provides a mechanism to create distributed application in java. The client invokes methods via an interface. These methods are implemented on the server side.**

**A)Aim: Implement a Server calculator containing ADD(), MUL(), SUB(),DIV() using RMI.**

**Source Code:**

**Calciclient.java**

package com.mycompany.rmical;

import java.rmi.Naming;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;

import java.rmi.RemoteException;

import java.util.Scanner;

public class Calciclient {

public static void main(String[] args) throws NotBoundException,MalformedURLException,RemoteException{

Scanner sc=new Scanner(System.in);

try{

CalciInterface c=(CalciInterface)Naming.lookup("rmi://localhost:1099/CalciInterface");

System.out.println("Client is connected to server.");

System.out.println("Please enter your choice: \n1. add\n2. sub\n3. mul\n4. div");

int choice=sc.nextInt();

int x,y;

switch(choice){

case 1:{

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.add(x,y));

break;

}

case 2:{

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.sub(x,y));

break;

}

case 3:{

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.mul(x,y));

break;

}

case 4:{

System.out.println("Enter x and y: ");

x=sc.nextInt();

y=sc.nextInt();

System.out.println(c.div(x,y));

break;

} }

}catch(Exception e){}

}}

**Calciserver.java**

package com.mycompany.rmical;

import java.rmi.registry.Registry;

import java.rmi.registry.LocateRegistry;

import java.rmi.RemoteException;

import java.rmi.NotBoundException;

public class Calciserver {

public static void main(String[] args) throws RemoteException, NotBoundException{

Registry r=java.rmi.registry.LocateRegistry.createRegistry(1099);

r.rebind("CalciInterface", new CalciRMI());

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

**CalRMI.java**

package com.mycompany.rmical;

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalciRMI extends UnicastRemoteObject implements CalciInterface{

public CalciRMI() throws RemoteException{

int a,b;}

public int add(int a, int b) throws RemoteException{return a+b;}

public int sub(int a, int b) throws RemoteException{return a-b;}

public int mul(int a, int b) throws RemoteException{return a\*b;}

public int div(int a, int b) throws RemoteException{return a/b;}

public static void main(String[] args){}}

**CalciInterface.java(Interface File)**

package com.mycompany.rmical;

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface CalciInterface extends Remote {

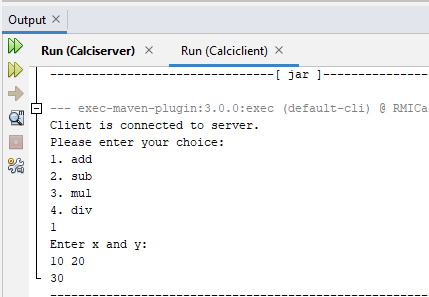
public int add(int x,int y) throws RemoteException;

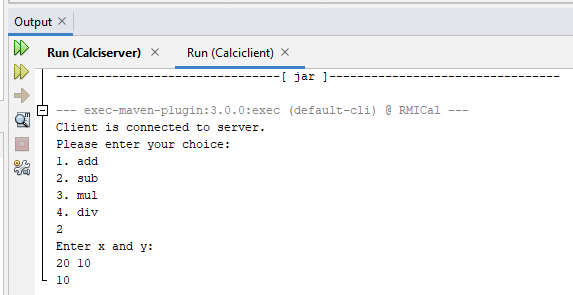
public int sub(int x,int y) throws RemoteException;

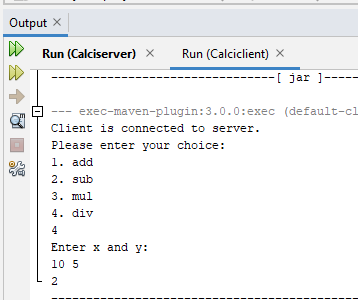
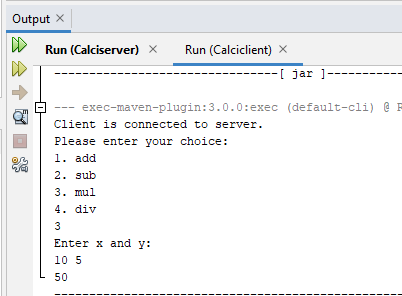
public int mul(int x,int y) throws RemoteException;

public int div(int x,int y) throws RemoteException;}

**Output:**

****

****

****