Practical 9(Factorial rmi)

**FactorialServer.java -**

import java.rmi.\*;

import java.rmi.Naming.\*;

import java.rmi.server.\*;

import java.rmi.registry.\*;

interface FactorialInterface extends Remote {

public long factorial(int n) throws RemoteException;

}

public class FactorialServer extends UnicastRemoteObject implements FactorialInterface {

public FactorialServer() throws RemoteException {

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

}

public long factorial(int n) throws RemoteException {

if (n == 0)

return 1;

else

return (n \* factorial(n - 1));

}

public static void main(String args[]) {

try {

FactorialServer server = new FactorialServer();

Naming.rebind("rmi://localhost/FactorialServer", server);

System.out.println("Factorial Server is ready...");

} catch (Exception e) {

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

}

}

}

**FactorialClient.java -**

import java.rmi.registry.\*;

import java.rmi.Naming;

import java.util.Scanner;

public class FactorialClient {

public static void main(String args[]) {

try {

FactorialInterface factorial = (FactorialInterface) Naming.lookup("rmi://localhost/FactorialServer");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to calculate factorial: ");

int num = scanner.nextInt();

long result = factorial.factorial(num);

System.out.println("Factorial of " + num + " is: " + result);

scanner.close();

} catch (Exception e) {

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

}

}

}