**Factorial.java – Remote Interface**

java

CopyEdit

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Factorial extends Remote {

long computeFactorial(int num) throws RemoteException;

}

**2️⃣ FactorialImpl.java – Server Implementation**

java

CopyEdit

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class FactorialImpl extends UnicastRemoteObject implements Factorial {

public FactorialImpl() throws RemoteException {

super();

}

public long computeFactorial(int num) throws RemoteException {

long fact = 1;

for (int i = 2; i <= num; i++) {

fact \*= i;

}

return fact;

}

}

**3️⃣ FactorialServer.java – RMI Server**

java

CopyEdit

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class FactorialServer {

public static void main(String[] args) {

try {

FactorialImpl obj = new FactorialImpl();

Registry registry = LocateRegistry.createRegistry(1099);

registry.rebind("FactorialService", obj);

System.out.println("Factorial RMI Server is running...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

**4️⃣ FactorialClient.java – Multi-threaded Client**

java

CopyEdit

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class FactorialClient implements Runnable {

private final int number;

public FactorialClient(int number) {

this.number = number;

}

public void run() {

try {

// Replace "localhost" with server IP if needed

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

Factorial stub = (Factorial) registry.lookup("FactorialService");

long result = stub.computeFactorial(number);

System.out.println(Thread.currentThread().getName() +

": Factorial of " + number + " is " + result);

} catch (Exception e) {

e.printStackTrace();

}

}

public static void main(String[] args) {

int[] numbers = {5, 7, 10, 12};

for (int n : numbers) {

new Thread(new FactorialClient(n)).start();

}

}

}