**RemoteService.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

import java.util.Date;

public interface RemoteService extends Remote {

String getDateTime() throws RemoteException;

String convertToWords(int number) throws RemoteException;

String fibonacciSeries(int n) throws RemoteException;

long factorial(int n) throws RemoteException;

}

**RemoteServiceImpl.java**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

import java.text.SimpleDateFormat;

import java.util.Date;

public class RemoteServiceImpl extends UnicastRemoteObject implements RemoteService {

protected RemoteServiceImpl() throws RemoteException {

super();

}

@Override

public String getDateTime() throws RemoteException {

SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return "Server Date & Time: " + formatter.format(new Date());

}

@Override

public String convertToWords(int number) throws RemoteException {

String[] words = { "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine" };

String numStr = Integer.toString(number);

StringBuilder result = new StringBuilder();

for (char digit : numStr.toCharArray()) {

result.append(words[digit - '0']).append(" ");

}

return result.toString().trim();

}

@Override

public String fibonacciSeries(int n) throws RemoteException {

if (n <= 0) return "0";

StringBuilder result = new StringBuilder("0 1");

int a = 0, b = 1, c;

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

c = a + b;

result.append(" ").append(c);

a = b;

b = c;

}

return result.toString();

}

@Override

public long factorial(int n) throws RemoteException {

if (n < 0) return -1; // Error for negative numbers

long fact = 1;

for (int i = 1; i <= n; i++) fact \*= i;

return fact;

}

}

**Server.java**

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class Server {

public static void main(String[] args) {

try {

LocateRegistry.createRegistry(1099); // Start RMI registry

RemoteServiceImpl service = new RemoteServiceImpl();

Naming.rebind("RemoteService", service);

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

} catch (Exception e) {

System.err.println("Server error: " + e);

}

}

}

**Client.java**

import java.rmi.Naming;

import java.util.Scanner;

public class Client {

public static void main(String[] args) {

try {

RemoteService service = (RemoteService) Naming.lookup("rmi://localhost/RemoteService");

Scanner sc = new Scanner(System.in);

System.out.println("1. Get Server Date & Time");

System.out.println("2. Convert Digits to Words");

System.out.println("3. Fibonacci Series");

System.out.println("4. Factorial");

System.out.print("Choose an option: ");

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.println(service.getDateTime());

break;

case 2:

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

int number = sc.nextInt();

System.out.println("In words: " + service.convertToWords(number));

break;

case 3:

System.out.print("Enter number of terms: ");

int terms = sc.nextInt();

System.out.println("Fibonacci Series: " + service.fibonacciSeries(terms));

break;

case 4:

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

int num = sc.nextInt();

System.out.println("Factorial: " + service.factorial(num));

break;

default:

System.out.println("Invalid choice!");

}

sc.close();

} catch (Exception e) {

System.err.println("Client error: " + e);

}

}

}

**Run**

cd C:\Users\YourName\Desktop\rpc

javac \*.java

rmiregistry # Keep this open

java Server # Run this in a new terminal

java Client # Run this in another new terminal