**Server:**

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

public class Server {

public static void main(String[] args) {

try {

LocateRegistry.createRegistry(5000); // Start RMI registry

CalculatorImpl calc = new CalculatorImpl();

Naming.rebind("CalculatorService", calc);

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

} catch (Exception e) {

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

}

}

}

**Client:**

import java.rmi.Naming;

import java.util.Scanner;

public class Client {

public static void main(String[] args) {

try {

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

Scanner sc = new Scanner(System.in);

System.out.println("1. Add");

System.out.println("2. Subtract");

System.out.println("3. Multiply");

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

System.out.println("5. Square");

System.out.println("6. Square Root");

System.out.println("7. Cube");

System.out.println("8. Cube Root");

System.out.println("9. Factorial");

System.out.println("10. Fibonacci Series");

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

int choice = sc.nextInt();

double a, b;

int n;

switch (choice) {

case 1:

System.out.print("Enter two numbers: ");

a = sc.nextDouble();

b = sc.nextDouble();

System.out.println("Result: " + calc.add(a, b));

break;

case 2:

System.out.print("Enter two numbers: ");

a = sc.nextDouble();

b = sc.nextDouble();

System.out.println("Result: " + calc.subtract(a, b));

break;

case 3:

System.out.print("Enter two numbers: ");

a = sc.nextDouble();

b = sc.nextDouble();

System.out.println("Result: " + calc.multiply(a, b));

break;

case 4:

System.out.print("Enter two numbers: ");

a = sc.nextDouble();

b = sc.nextDouble();

System.out.println("Result: " + calc.divide(a, b));

break;

case 5:

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

a = sc.nextDouble();

System.out.println("Result: " + calc.square(a));

break;

case 6:

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

a = sc.nextDouble();

System.out.println("Result: " + calc.squareRoot(a));

break;

case 7:

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

a = sc.nextDouble();

System.out.println("Result: " + calc.cube(a));

break;

case 8:

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

a = sc.nextDouble();

System.out.println("Result: " + calc.cubeRoot(a));

break;

case 9:

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

n = sc.nextInt();

System.out.println("Factorial: " + calc.factorial(n));

break;

case 10:

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

n = sc.nextInt();

System.out.println("Fibonacci Series: " + calc.fibonacci(n));

break;

default:

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

}

sc.close();

} catch (Exception e) {

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

}

}

}

**CalculatorImpl**

import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class CalculatorImpl extends UnicastRemoteObject implements Calculator {

protected CalculatorImpl() throws RemoteException {

super();

}

@Override

public double add(double a, double b) {

return a + b;

}

@Override

public double subtract(double a, double b) {

return a - b;

}

@Override

public double multiply(double a, double b) {

return a \* b;

}

@Override

public double divide(double a, double b) {

if (b == 0) throw new ArithmeticException("Cannot divide by zero.");

return a / b;

}

@Override

public double square(double a) {

return a \* a;

}

@Override

public double squareRoot(double a) {

return Math.sqrt(a);

}

@Override

public double cube(double a) {

return a \* a \* a;

}

@Override

public double cubeRoot(double a) {

return Math.cbrt(a);

}

@Override

public long factorial(int n) {

if (n < 0) return -1;

long fact = 1;

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

return fact;

}

@Override

public String fibonacci(int n) {

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

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

long 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();

}

}

**Calculator**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calculator extends Remote {

double add(double a, double b) throws RemoteException;

double subtract(double a, double b) throws RemoteException;

double multiply(double a, double b) throws RemoteException;

double divide(double a, double b) throws RemoteException;

double square(double a) throws RemoteException;

double squareRoot(double a) throws RemoteException;

double cube(double a) throws RemoteException;

double cubeRoot(double a) throws RemoteException;

long factorial(int n) throws RemoteException;

String fibonacci(int n) throws RemoteException;

}

**Running Steps:**

**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**