BÁO CÁO THỰC HÀNH

Bài thực hành số 05: RMI và Corba

Môn học: Lập trình ứng dụng mạng

Lớp: NT109.O21.MMCL

THÀNH VIÊN THỰC HIỆN:

STT	Họ và tên	MSSV
1	Lê Hoàng Khánh	21522205

ĐÁNH GIÁ KHÁC:

Tổng thời gian thực hiện	1 tuần
Ý kiến *(nếu có)* + Khó khăn + Đề xuất, kiến nghị	Không có

MỤC LỤC

A. BÁO CÁO CHI TIẾT

1. RMI.

Java RMI (Remote Method Invocation) là một công nghệ lập trình phân tán mạnh mẽ trong Java, cho phép các đối tượng Java trên các máy ảo Java (JVM) khác nhau giao tiếp và tương tác với nhau như thể chúng đang nằm trên cùng một máy. Điều này mở ra khả năng xây dựng các ứng dụng phân tán linh hoạt và hiệu quả.

a. RMIServer.

Triển khai một Math Server sử dụng RMI. Server cung cấp các phép tính cơ bản như cộng, trừ, nhân, chia và giải phương trình bậc hai. Client từ xa có thể kết nối đến máy chủ này và gọi các phương thức tính toán từ xa thông qua giao thức RMI.

```java
package io.github.tuilakhanh.Lab05.RMIServer;

import java.rmi.Remote;
import java.rmi.RemoteException;

public interface IMath extends Remote
{
    int add(int a, int b) throws RemoteException;
    int sub(int a, int b) throws RemoteException;
    int mul(int a, int b) throws RemoteException;
    int div(int a, int b) throws RemoteException;
    float[] PTBac2(float a, float b, float c) throws RemoteException;
}
```

IMath (Interface):

- Đây là một interface định nghĩa các phương thức mà máy chủ sẽ cung cấp.
- Các phương thức này được đánh dấu bằng từ khóa throws RemoteException để chỉ ra rằng chúng có thể ném ra ngoại lệ khi có lỗi trong quá trình giao tiếp từ xa.

```java
package io.github.tuilakhanh.Lab05.RMIServer;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class MathObject extends UnicastRemoteObject implements IMath {
    public MathObject() throws RemoteException {
        super();
    }
    @Override
    public int add(int a, int b) throws RemoteException {
        return a+b;
    }
    @Override
    public int sub(int a, int b) throws RemoteException {
        return a-b;
    }
    @Override
    public int mul(int a, int b) throws RemoteException {
        return a*b;
    }
    @Override
    public int div(int a, int b) throws RemoteException {
        return a/b;
    }
    public float[] PTBac2(float a, float b, float c) throws RemoteException{

        float delta = b*b - 4*a*c;

        if (delta < 0) {
            throw new RemoteException("Phuong trinh khong co nghiem");
        }

        var x1 = (float) ((-b + Math.sqrt(delta)) / (2 * a));
        var x2 = (float) ((-b - Math.sqrt(delta)) / (2 * a));

        return new float[] {x1, x2};
    }
}
```

MathObject (Class):

- Đây là lớp triển khai interface IMath, thực hiện các phép tính toán cụ thể.
- Lớp này kế thừa từ UnicastRemoteObject, lớp cơ sở cho các đối tượng RMI từ xa.
- Các phương thức add, sub, mul, div thực hiện các phép tính cộng, trừ, nhân, chia.
- Phương thức PTBac2 giải phương trình bậc hai và trả về một mảng chứa hai nghiệm (hoặc ném ra ngoại lệ nếu phương trình vô nghiệm).

```java
package io.github.tuilakhanh.Lab05.RMIServer;

import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;

public class Main {
    public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1099);
            Naming.bind("Service", new MathObject());
            System.out.println("Server Started!!!");
        }
        catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

Main (Class):

- Lớp này chứa phương thức main, là điểm bắt đầu của chương trình máy chủ.
- Trong phương thức main, nó tạo một registry (cơ sở dữ liệu đối tượng) trên cổng 1099 và đăng ký đối tượng MathObject với tên "Service".
- Sau đó, máy chủ chờ các yêu cầu từ client.

Cách hoạt động:

- **Bước 1:** Khởi động máy chủ: Chạy lớp Main để khởi động máy chủ RMI.
- **Bước 2:** Client kết nối: Client sẽ sử dụng lớp Naming để tìm kiếm đối tượng "Service" trên registry của máy chủ.
- **Bước 3:** Gọi phương thức từ xa: Sau khi lấy được tham chiếu đến đối tượng MathObject, client có thể gọi các phương thức tính toán (như add, sub, ...) như thể chúng là các phương thức cục bộ. RMI sẽ đảm nhiệm việc truyền các tham số và kết quả giữa client và server một cách minh bạch.

b. RMIClient

Trước tiền cần Import RMIServer.jar như một thư viện để có thể sử dụng MathObject.

Thêm thư viện thông qua dòng lệnh sau vào file gradle.

```
implementation(files("libs/RMIServer-1.0-SNAPSHOT.jar"))
```

```java
package io.github.tuilakhanh.Lab05.RMIClient;

import io.github.tuilakhanh.Lab05.RMIServer.IMath;

import java.rmi.Naming;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        var scanner = new Scanner(System.in);

        System.out.print("Nhap so thu nhat, a = ");
        int a = scanner.nextInt();
        System.out.print("Nhap so thu hai, b = ");
        int b = scanner.nextInt();
        try {
            IMath imath = (IMath) Naming.lookup("rmi://localhost/Service");

            var add = imath.add(a, b);
            var sub = imath.sub(a, b);
            var mul = imath.mul(a, b);
            var div = imath.div(a, b);
            System.out.println("Tong 2 so: " + add);
            System.out.println("Hieu 2 so: " + sub);
            System.out.println("Tich 2 so: " + mul);
            System.out.println("Thuong 2 so: " + div);
        }
        catch (Exception e) {
            System.err.println("Error: " + e.getMessage());
        }
    }
}
```

Import:

- Đoạn code này import interface IMath từ package io.github.tuilakhanh.Lab05.RMIServer. Điều này cho phép client sử dụng các phương thức được định nghĩa trong interface này để giao tiếp với máy chủ.

Main method:

- Phương thức main là điểm bắt đầu của chương trình client.
- Đầu tiên, nó tạo một đối tượng Scanner để đọc dữ liệu nhập vào từ bàn phím.
- Sau đó, nó yêu cầu người dùng nhập hai số nguyên a và b.
- Tiếp theo, nó cố gắng kết nối đến máy chủ RMI tại địa chỉ "rmi://localhost/Service" (localhost ở đây là máy tính đang chạy máy chủ) và lấy tham chiếu đến đối tượng IMath.

- Nếu kết nối thành công, nó sẽ gọi các phương thức add, sub, mul, div của đối tượng IMath để thực hiện các phép tính trên a và b, và in kết quả ra màn hình.
- Nếu có lỗi xảy ra trong quá trình kết nối hoặc gọi phương thức, nó sẽ in thông báo lỗi.

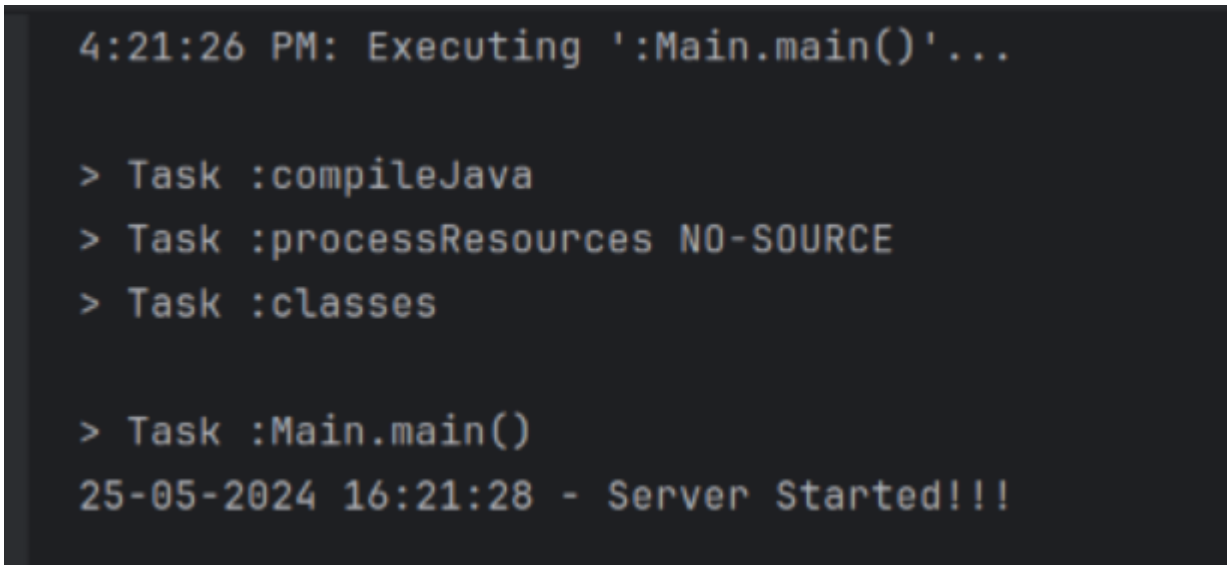
Cách hoạt động:

- **Bước 1:** Khởi động máy chủ: Đảm bảo rằng máy chủ RMI (đoạn code trước) đã được chạy trước.
- **Bước 2:** Chạy client: Chạy lớp Main này để khởi động client.
- **Bước 3:** Nhập dữ liệu: Nhập hai số nguyên khi được yêu cầu.
- **Bước 4:** Kết nối và tính toán: Client sẽ kết nối đến máy chủ, gọi các phương thức tính toán từ xa và nhận kết quả.
- **Bước 5:** Hiển thị kết quả: Kết quả tính toán sẽ được in ra màn hình.

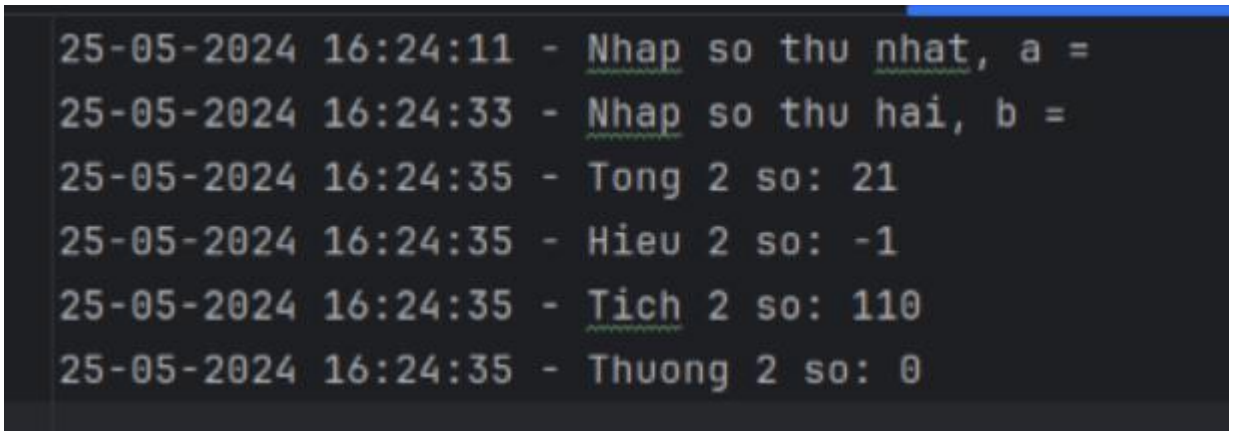
c. Demo.

Chạy Project Server sau đó đến Project Client.

Server:



Client:



d. Thêm giao diện.

Sử dụng giao diện máy tính từ bài Lab02, thay các đoạn code xử lý phép toàn bằng đoạn RMI sử dụng Server để tính toán.

```java
import io.github.tuilakhanh.Lab05.RMIServer.IMath;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.rmi.Naming;
import java.rmi.RemoteException;

// ... (other imports and UI components remain the same)

public class CalculatorUI extends Frame implements ActionListener {
    // ... (UI components)

    IMath imath;

    public CalculatorUI() throws Exception {
        // ... (UI setup)

        // Lookup the remote math object
        imath = (IMath) Naming.lookup("rmi://localhost/Service");
    }

    // ... (actionPerformed method)

    private void calculate() throws RemoteException {
        double num2 = Double.parseDouble(display.getText());
        double result;

        // Use the remote math object for calculations
        switch (operator) {
            case '+':
                result = imath.add((int) num1, (int) num2);
                break;
            case '-':
                result = imath.sub((int) num1, (int) num2);
                break;
            // ... (other cases for *, /)
        }

        // ... (display result)
    }
}
```

Remote Object Lookup:

```
imath = (IMath) Naming.lookup("rmi://localhost/Service");
```

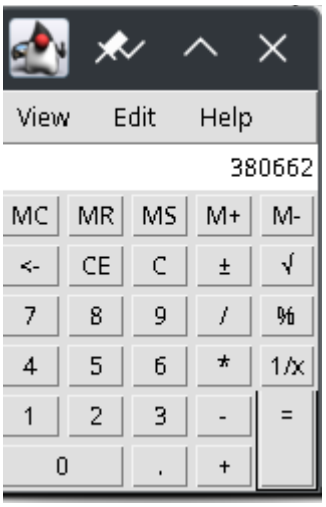
Thể hiện cách client lấy tham chiếu đến đối tượng IMath từ xa đã được đăng ký trên RMI server.

Gọi Phương Thức Từ Xa (Remote Method Invocation): Bên trong phương thức calculate, các dòng như

```
result = imath.add((int) num1, (int) num2);
```

Cho thấy cách client gọi các phương thức trên đối tượng từ xa (imath) như thể chúng là các phương thức cục bộ. Đây là bản chất của RMI.

e. **Demo UI.**



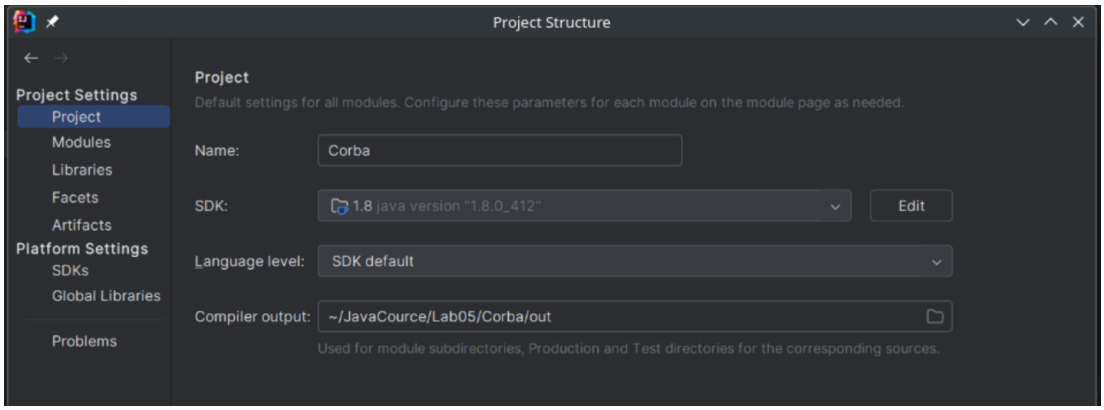

```
> Task :Main.main()
25-05-2024 17:02:10 - 1.0 + 2.0 = 3.0

25-05-2024 17:02:15 - 2.0 + 9.0 = 11.0

25-05-2024 17:02:20 - 111.0 / 3.0 = 37.0

25-05-2024 17:02:26 - 379.0 / 6.0 = 63.0

25-05-2024 17:02:31 - 63444.0 * 6.0 = 380664.0

25-05-2024 17:02:38 - 380664.0 - 2.0 = 380662.0
```

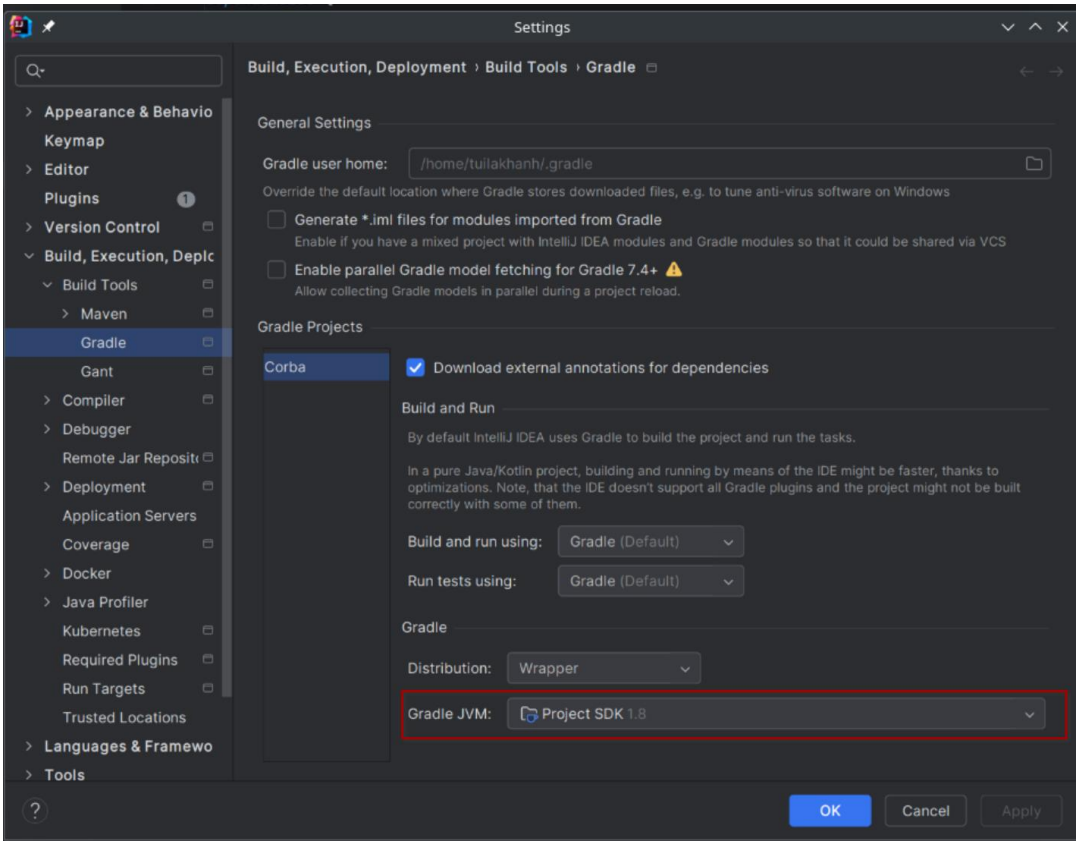
2. Corba

a. Điều chỉnh môi trường.

Vì Corba chỉ tương thích với Java 8 nên cần phải điều chỉnh phiên bản Java của máy về Java 8 hoặc từ IDE.

```
~ via v3.12.3
❯ java -version
openjdk version "1.8.0_412"
OpenJDK Runtime Environment (build 1.8.0_412-b08)
OpenJDK 64-Bit Server VM (build 25.412-b08, mixed mode)
```





b. Điều chỉnh file Gradle để compile file IDL.

```
plugins {
    id("java")
}

group = "io.github.tuilakhanh.Lab05.Corba"
version = "1.0-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    testImplementation(platform("org.junit:junit-bom:5.10.0"))
    testImplementation("org.junit.jupiter:junit-jupiter")
    implementation("org.jacorb:jacorb:3.9")
    implementation("org.jacorb:jacorb-idl-compiler:3.9")
}

tasks.test {
    useJUnitPlatform()
}

tasks.register<JavaExec>("compileIdl")  {
    group = "build"
    description = "Compiles CORBA IDL files to Java"

    val idlDir = file("src/main/idl")
    val javaOutputDir = file("$buildDir/generated/sources/idl/")

    inputs.dir(idlDir)
    outputs.dir(javaOutputDir)

    classpath = sourceSets.main.get().runtimeClasspath
    mainClass.set("org.jacorb.idl.parser")
    args("-d", "$projectDir/src/main/java", "$projectDir/src/main/idl/Calc.idl")
}

sourceSets {
    main {
        java {
            srcDir("$projectDir/src/main/java")
        }
    }
}
```

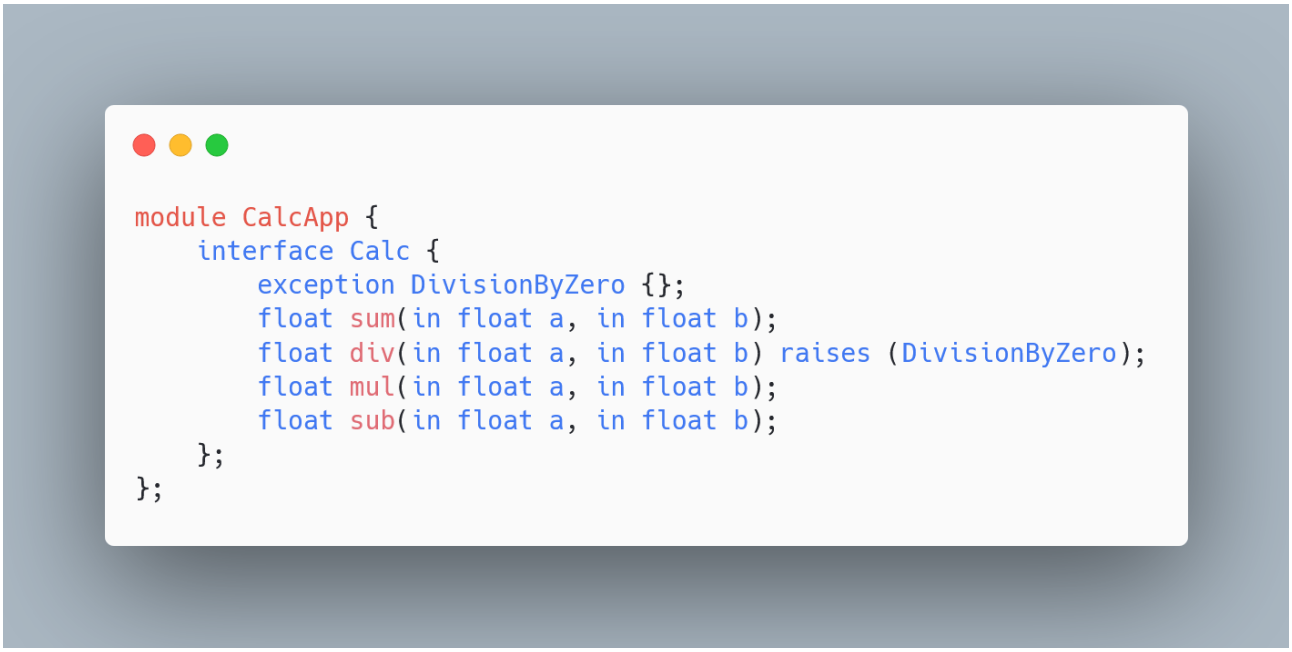
Thêm task Gradle có tên "compileIdl". Task này có mục tiêu biên dịch các tệp IDL (định nghĩa giao diện trong kiến trúc CORBA - Common Object Request Broker Architecture) thành mã nguồn Java.

Các file IDL được đặt tại folder src/main/idl, sau khi compile file IDL thành Java các file Java sẽ được đặt vào folder mặc định chứa source code Java tại /src/main/java.

Để có thể compile file IDL thông qua Corba, cần thêm thư viện giúp làm điều này thông qua.

```
implementation("org.jacorb:jacorb:3.9")
implementation("org.jacorb:jacorb-idl-compiler:3.9")
```

c. **IDL**



```
module CalcApp {
    interface Calc {
        exception DivisionByZero {};
        float sum(in float a, in float b);
        float div(in float a, in float b) raises (DivisionByZero);
        float mul(in float a, in float b);
        float sub(in float a, in float b);
    };
};
```

Tệp Interface Definition Language (IDL) CalcApp.idl định nghĩa một module CORBA có tên CalcApp. Module này cung cấp một interface duy nhất là Calc, đóng vai trò như một máy tính đơn giản. Dưới đây là chi tiết về các thành phần của interface này:

Exception DivisionByZero:

- Là một ngoại lệ (exception) do người dùng định nghĩa.
- Được kích hoạt (raise) khi có một phép chia cho số 0 xảy ra trong phương thức div.

Các phương thức (Operations):

- sum(in float a, in float b): Tính tổng của hai số a và b, trả về kết quả kiểu float.
- div(in float a, in float b) raises (DivisionByZero):
    - Thực hiện phép chia a cho b, trả về kết quả kiểu float.
    - Nếu b bằng 0, ngoại lệ DivisionByZero sẽ được kích hoạt, báo hiệu cho client (ứng dụng sử dụng interface này) về lỗi.
- mul(in float a, in float b): Tính tích của hai số a và b, trả về kết quả kiểu float.
- sub(in float a, in float b): Tính hiệu của hai số a và b, trả về kết quả kiểu float.

d. Server

```java
package io.github.tuilakhanh.Lab05.Corba.Server;

import CalcApp.*;
import CalcApp.CalcPackage.DivisionByZero;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;
import org.omg.PortableServer.*;
import io.github.tuilakhanh.log.Log;

class CalcImpl extends CalcPOA {
    @Override
    public float sum(float a, float b) {
        return a + b;
    }
    @Override
    public float div(float a, float b) throws DivisionByZero {
        if (b == 0) {
            throw new CalcApp.CalcPackage.DivisionByZero();
        } else {
            return a / b;
        }
    }
    @Override
    public float mul(float a, float b) {
        return a * b;
    }
    @Override
    public float sub(float a, float b) {
        return a - b;
    }
}

public class Server {
    public static void main(String[] args) {
        Log log = new Log("configs/config.txt");
        try {
            // Initialize ORB and get references
            ORB orb = ORB.init(args, null);
            POA rootpoa = POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
            rootpoa.the_POAManager().activate();

            // Create servant and register with POA
            CalcImpl calcImpl = new CalcImpl();
            org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcImpl);

            Calc href = CalcHelper.narrow(ref);
            NamingContextExt ncRef =
NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
            ncRef.rebind(ncRef.to_name("Calc"), href);

            log.writeLog("[Server] Calc Server ready and waiting ...");
            orb.run();
        } catch (Exception e) {
            System.err.println("[Server] ERROR: " + e);
            e.printStackTrace(System.out);
        }
        log.writeLog("[Server] Exiting ...");
    }
}
```

Đoạn code này triển khai một server đơn giản sử dụng công nghệ CORBA (Common Object Request Broker Architecture) để cung cấp các phép tính toán số học cơ bản (cộng, trừ, nhân, chia) cho các client từ xa. Máy chủ này được viết bằng Java và sử dụng các thư viện CORBA chuẩn để xử lý việc giao tiếp và đăng ký dịch vụ.

Class CalcImpl:

- Kế thừa từ CalcPOA: Lớp này triển khai các phương thức tính toán (sum, div, mul, sub) được định nghĩa trong interface IDL (Interface Definition Language) Calc.
- Xử lý ngoại lệ DivisionByZero: Ném ra ngoại lệ DivisionByZero khi có phép chia cho 0.

Class Server:

- Chứa phương thức main: Điểm khởi đầu của ứng dụng server.
- Khởi tạo ORB (Object Request Broker):

```
ORB orb = ORB.init(args, null);
```

- Lấy tham chiếu đến POA (Portable Object Adapter):

```
POA rootpoa =
POAHelper.narrow(orb.resolve_initial_references("RootPOA"));
```

- Kích hoạt POA Manager:

```
rootpoa.the_POAManager().activate();
```

- Tạo đối tượng CalcImpl (servant):

```
CalcImpl calcImpl = new CalcImpl();
```

- Đăng ký servant với POA:

```
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(calcImpl);
```

- Chuyển đổi tham chiếu thành kiểu Calc:

```
Calc href = CalcHelper.narrow(ref);
```

- Lấy tham chiếu đến dịch vụ đặt tên (Naming Service):

```
NamingContextExt ncRef =
NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService
"));
```

- Đăng ký đối tượng Calc với tên "Calc":

```
ncRef.rebind(ncRef.to_name("Calc"), href);
```

- Chạy ORB để lắng nghe yêu cầu từ client:

```
orb.run();
```

- Xử lý ngoại lệ: In thông báo lỗi và stack trace.

e. Client.

```java
package io.github.tuilakhanh.Lab05.Corba.Client;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import CalcApp.*;
import CalcApp.CalcPackage.DivisionByZero;
import io.github.tuilakhanh.log.Log;
import org.omg.CosNaming.*;
import org.omg.CORBA.*;

public class Client {
    static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    static Log log = new Log("configs/config.txt");
    public static void main(String[] args) {
        try {
            ORB orb = ORB.init(args, null);
            NamingContextExt ncRef =
NamingContextExtHelper.narrow(orb.resolve_initial_references("NameService"));
            Calc calcImpl = CalcHelper.narrow(ncRef.resolve_str("Calc"));

            label:
            while (true) {
                log.writeLog("[Client] 1. Sum");
                log.writeLog("[Client] 2. Sub");
                log.writeLog("[Client] 3. Mul");
                log.writeLog("[Client] 4. Div");
                log.writeLog("[Client] 5. exit");
                log.writeLog("[Client] --");
                System.out.print("[Client] Choice: ");
                try {
                    String opt = br.readLine();
                    switch (opt) {
                        case "5":
                            break label;
                        case "1":
                            log.writeLog("[Client] a + b = " + calcImpl.sum(getFloat("a"),
                                    getFloat("b")));
                            break;
                        case "2":
                            log.writeLog("[Client] a - b = " + calcImpl.sub(getFloat("a"),
                                    getFloat("b")));
                            break;
                        case "3":
                            log.writeLog("[Client] a * b = " + calcImpl.mul(getFloat("a"),
                                    getFloat("b")));
                            break;
                        case "4":
                            try {
                                log.writeLog("[Client] a / b = " +
                                        calcImpl.div(getFloat("a"), getFloat("b")));
                            } catch (DivisionByZero de) {
                                log.writeLog("[Client] Division by zero!!!");
                            }
                            break;
                    }
                } catch (Exception e) {
                    log.writeLog("[Client] ERROR : " + e);
                    e.printStackTrace(System.out);
                }
            }
            //calcImpl.shutdown();
        } catch (Exception e) {
            log.writeLog("[Client] ERROR : " + e);
            e.printStackTrace(System.out);
        }
    }

    static float getFloat(String number) throws Exception {
        log.writeLog(number + ": ");
        return Float.parseFloat(br.readLine());
    }
}
```

Đoạn code này triển khai client tương tác với máy chủ CORBA đã được triển khai trước đó để thực hiện các phép tính số học. Client cho phép người dùng lựa chọn phép tính từ menu (cộng, trừ, nhân, chia) và nhập các số tương ứng để tính toán. Kết quả được nhận từ máy chủ và hiển thị cho người dùng.

Các biến toàn cục:

- **br:** Đối tượng BufferedReader để đọc dữ liệu từ bàn phím.
- **log:** Đối tượng Log tùy chỉnh để ghi log (tương tự như trong server).

Phương thức main:

- Khởi tạo ORB.
- Lấy tham chiếu đến dịch vụ đặt tên (NamingContextExt).
- Lấy tham chiếu đến đối tượng Calc từ máy chủ bằng tên "Calc".
- Vòng lặp while(true):
    - Hiển thị menu lựa chọn phép tính.
    - Đọc lựa chọn của người dùng.
    - Gọi phương thức tương ứng từ đối tượng calcImpl và hiển thị kết quả.
    - Xử lý ngoại lệ DivisionByZero nếu có.
    - Thoát vòng lặp khi người dùng chọn "5".
- Xử lý ngoại lệ chung.
- Phương thức getFloat:
- Đọc một số thực từ bàn phím.
- Xử lý ngoại lệ nếu người dùng nhập không hợp lệ.

Phương thức getFloat:

- Đọc một số thực từ bàn phím.
- Xử lý ngoại lệ nếu người dùng nhập không hợp lệ.

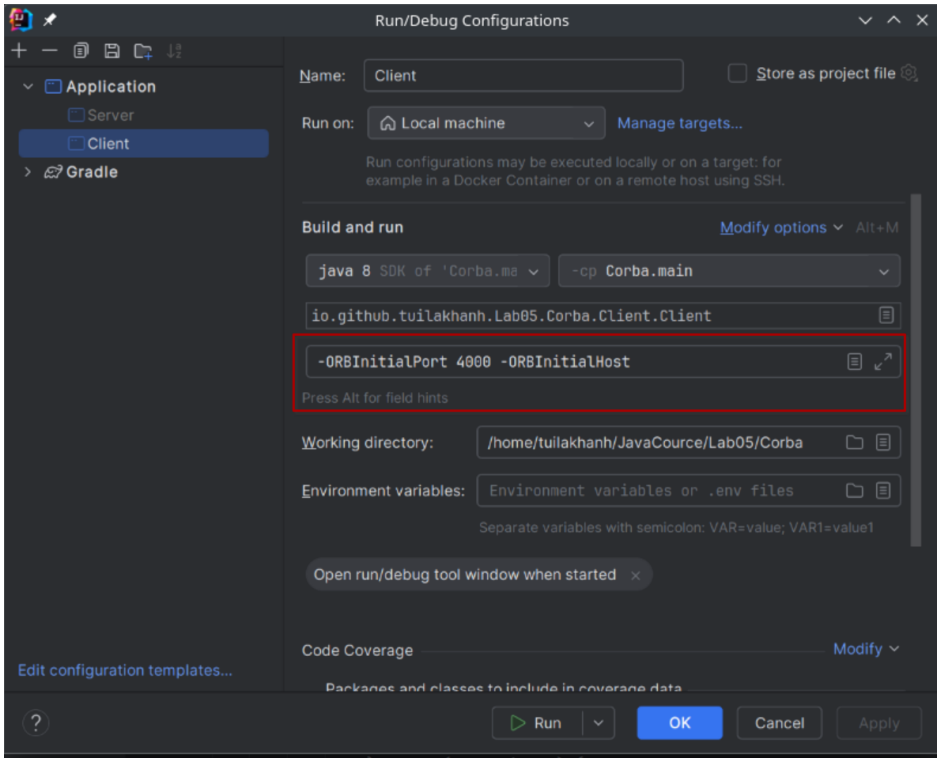
f. Demo

Mở kết nối ORB trên port 4000 thông qua lệnh.

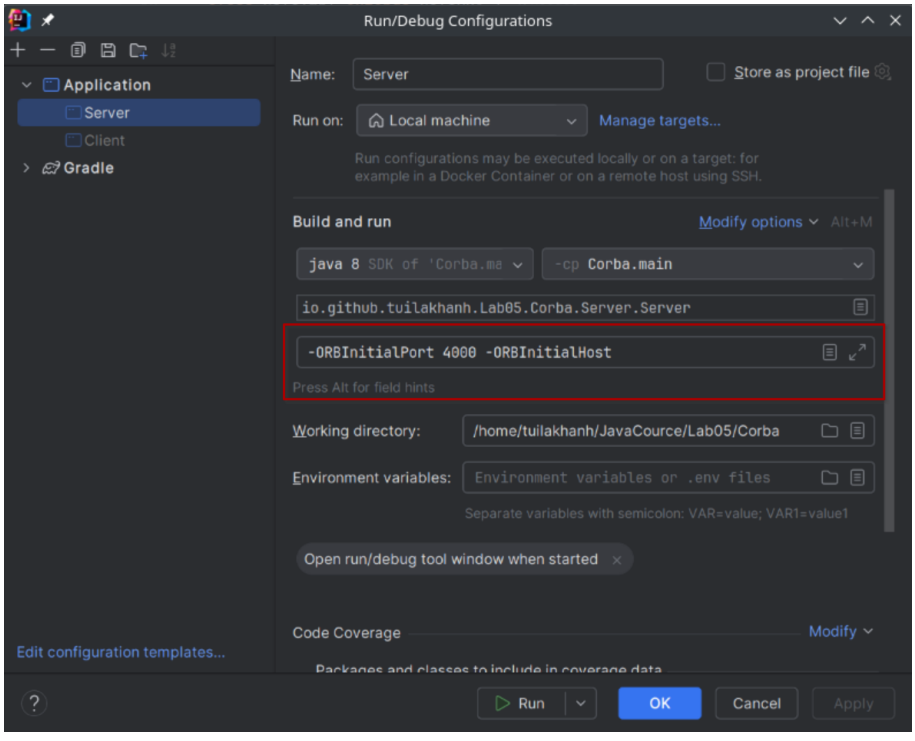
```
orbd -ORBInitialPort 4000
```

Điều chỉnh start **args** Client và Server để kết nối đến Port chỉ định.

Client:



Server:



Server:

```
2:52:41 PM: Executing ':Server.main()'...

> Task :compileJava
> Task :processResources NO-SOURCE
> Task :classes

> Task :Server.main()
25-05-2024 14:52:45 - [Server] Calc Server ready and waiting ...
```

Client:

```
> Task :Client.main()
25-05-2024 15:43:31 - [Client] 1. Sum

25-05-2024 15:43:31 - [Client] 2. Sub

25-05-2024 15:43:31 - [Client] 3. Mul

25-05-2024 15:43:31 - [Client] 4. Div

25-05-2024 15:43:31 - [Client] 5. exit

25-05-2024 15:43:31 - [Client] --

[Client] Choice: 1
25-05-2024 15:43:53 - a:

1
25-05-2024 15:43:56 - b:

2
25-05-2024 15:43:57 - [Client] a + b = 3.0
```

B. TÀI LIỆU THAM KHẢO