Guia de trabajo ZeroC ICE

Implementar una calculadora distribuida

Objetivo del ejercicio: Implementar un servicio de cálculo matemático utilizando ICE, que permita a los clientes realizar operaciones matemáticas básicas como suma, resta, multiplicación y división.

1. Inicializar un proyecto usando gradle.
2. Recuerde que debe editar los archivos**: build.gradle** y **settings.gradle** siguiendo las instrucciones de la clase anterior
3. Crear el archivo **cal.ice** que contiene las definiciones importantes de la interfaz que implementaremos en java:

```
module MathCalc {
    interface Calculator {
        double add(double x, double y);
        double subtract(double x, double y);
        double multiply(double x, double y);
        double divide(double x, double y);
    }
}
```

   Al compilar se genera el paquete MathCalc que contiene las deinfiniciones que podremos usar en el `Server, Client y CalculatorI.` Para ello deberá importar en cada clase el paquete: `MathCalc.*`
4. Editar el archivo **build.gradle** para indicar la ubicación del archivo **cal.ice**
5. Crear las carpetas para cliente y servidor:

```
mkdir server/src/main/java/
mkdir client/src/main/java/
```

6. **Implementar el servidor:** Las siguientes clases deben estar en `server/src/main/java/`
   Crear una clase llamada `CalculatorI` que implementa la interfaz `Calculator` definida en el archivo **cal.ice**, e implementar todos los métodos de la interfaz para realizar las operaciones matemáticas.
   Tener en cuenta que en la implementación todos los métodos tienen un argumento adicional:
   `nombreMetodo(argumentos necesarios, Current current)`
   Para lo cual debe incluir la siguiente línea en el encabezado de la clase:
   `import com.zeroc.Ice.Current;`
   Crear la clase llamada `Server` que sigue la misma estructura del ejemplo de la clase anterior.
   En este caso debe invocar la clase `CalculatorI` para crear el `servant.` El objeto creado se debe agregar al **adapter** con un nombre que sirve como identificador.
7. **Implementar el cliente:**
   Crear una clase llamada `Client` para el cliente
   En el método **main**, inicializar el comunicador.
   Convertir el proxy de string a un proxy de tipo **CalculatorPrx**.
   Verificar si el proxy es nulo y manejar el caso si no se puede conectar al servidor.
   Realizar llamadas a los métodos remotos del servidor para realizar operaciones matemáticas.
   Imprimir los resultados de las operaciones en la consola.

8. Deben garantizar que el cliente pueda repetir el proceso si quiere realizar otra operación.

9. Realizar las modificaciones necesarias para ejecutar el cliente en un host y el servidor en otro host. Ejecutar varios clientes. Tener en cuenta las siguientes recomendaciones.

En el servidor: para crear el adapter usar la siguiente línea. Modificar el localhost por la ip del servidor

```
communicator.createObjectAdapterWithEndpoints("CalculatorAdapter",
"tcp -h localhost -p 10000");
```

En el cliente: usar la siguiente línea para crear el objeto base. Modificar el localhost por la ip del servidor:

```
communicator.stringToProxy("Calculator:tcp -h localhost -p 10000");
```