

SISTEMAS DISTRIBUIDOS

Practica 2

Introducción a la tecnología RMI

Grado en ingeniería informática

Francisco Joaquín Murcia Gómez 48734281H

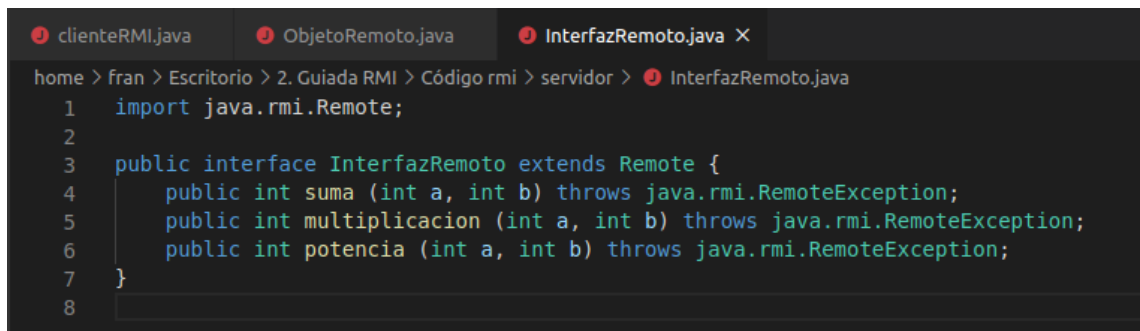
Grupo 1

Introducción

El Programa en cuestión es una estructura de cliente servidor implementada con RMI, el programa que es una calculadora que o bien te suma o bien te multiplica, en mi caso he añadido una función que te hace la potencia

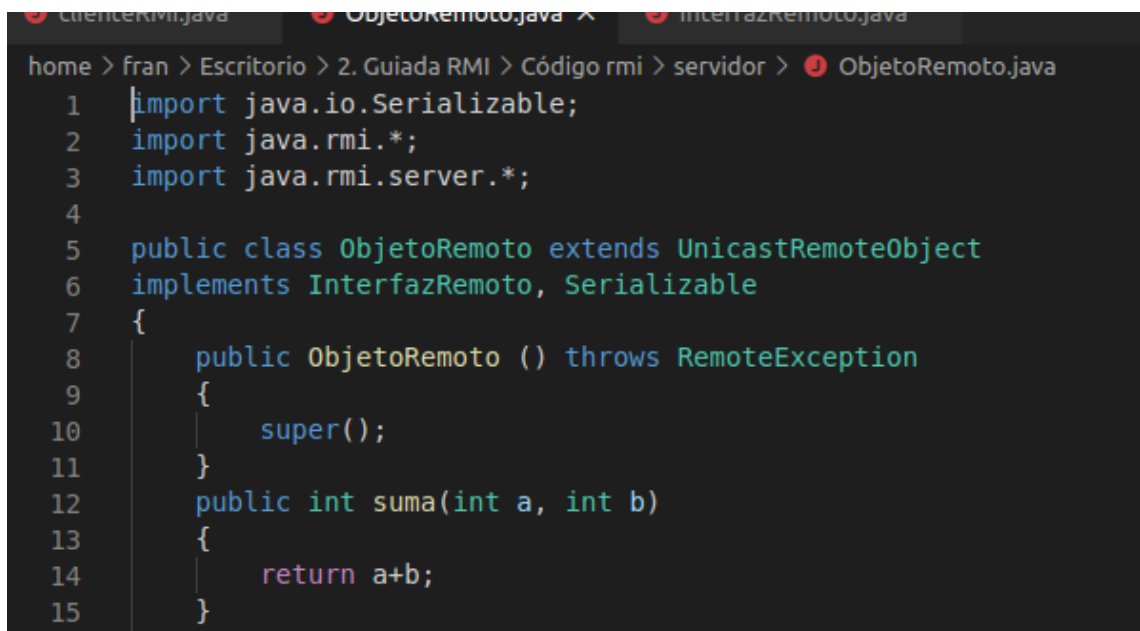
Código RMI

En primer lugar, hemos de tener una interfaz remota que será amiga de cliente y de servidor, en este está la declaración de los métodos suma multiplicación y potencia.



```
clienteRMI.java  ObjetoRemoto.java  InterfazRemoto.java X
home > fran > Escritorio > 2. Guiada RMI > Código rmi > servidor > InterfazRemoto.java
1  import java.rmi.Remote;
2
3  public interface InterfazRemoto extends Remote {
4      public int suma (int a, int b) throws java.rmi.RemoteException;
5      public int multiplicacion (int a, int b) throws java.rmi.RemoteException;
6      public int potencia (int a, int b) throws java.rmi.RemoteException;
7  }
8
```

En objetoRemoto se implementa los métodos de la interfaz.



```
clienteRMI.java  ObjetoRemoto.java X  InterfazRemoto.java
home > fran > Escritorio > 2. Guiada RMI > Código rmi > servidor > ObjetoRemoto.java
1  import java.io.Serializable;
2  import java.rmi.*;
3  import java.rmi.server.*;
4
5  public class ObjetoRemoto extends UnicastRemoteObject
6  implements InterfazRemoto, Serializable
7  {
8      public ObjetoRemoto () throws RemoteException
9      {
10         super();
11     }
12     public int suma(int a, int b)
13     {
14         return a+b;
15     }

```

```
public int multiplicacion(int a,int b)
{
    return a*b;
}
////////////////////////////////
public int potencia(int a, int b) { //a^b

    int resultado = 1;
    for (int i = 1; i <= b; i++) {

        resultado *= a;

    }
    return resultado;
}
////////////////////////////////
}
```

En segundo lugar, esta Cliente, en esta se implementa el menú, he hecho una ampliación de este, ya que he añadido una opción de hacer la potencia como se puede ver a continuación:

Menú

```
operacion = 0;
while (operacion !=1 && operacion !=3)
{
    System.out.println("[1] Sumar");
    System.out.println("[2] Multiplicar");
    System.out.println("[3] Potencia a^b"); //////////////////////////////////////////////////
    System.out.println("Indica la operacion a realizar: ");
    try
    {
        operacion = new Integer(buf.readLine()).intValue();
    }
    catch(Exception e)
    {
        operacion = 0;
    }
}
```

Selección de operación

```
if (operacion == 1)
{
    resultado = objetoRemoto.suma(op1,op2);
}
else
{
    if (operacion == 2)
    {
        resultado = objetoRemoto.multiplicacion(op1,op2);
    }else if(operacion == 3){////////////////////
        resultado = objetoRemoto.potencia(op1,op2);
    }
}
```

Compilación y ejecución

En la carpeta servidor abrimos una terminal.

En primer lugar, compilaremos la interfaz y la clase objeto:

```
x: ~/Escritorio/2. Guía RMI/Código rmi/servidor
~/Código rmi/servidor$ export PATH=$PATH:ruta_j2sdk/bin
~/Código rmi/servidor$ javac InterfazRemoto.java
~/Código rmi/servidor$ export CLASSPATH=$CLASSPATH:ruta_interfaz
~/Código rmi/servidor$ javac ObjetoRemoto.java
~/Código rmi/servidor$
```

Una vez hecho esto generaremos los stubs, esta clase es la que ve el cliente.

```
fran@fran-VirtualBox:~/Escritorio/2. Guía RMI/Código rmi/servidor$ rmic ObjetoRemoto
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
fran@fran-VirtualBox:~/Escritorio/2. Guía RMI/Código rmi/servidor$ jar cvf cliente.jar InterfazRemoto.class ObjetoRemoto_stub.class
manifiesto agregado
agregando: InterfazRemoto.class(entrada = 272) (salida = 184)(desinflado 32%)
agregando: ObjetoRemoto_stub.class(entrada = 2260) (salida = 1060)(desinflado 53%)
fran@fran-VirtualBox:~/Escritorio/2. Guía RMI/Código rmi/servidor$
```

Más tarde generaremos el registro que es donde se instancia el objeto.

```
fran@fran-VirtualBox:~/Escritorio/2. Guía RMI/Código rmi/servidor$ javac Registro.java
Note: Registro.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
fran@fran-VirtualBox:~/Escritorio/2. Guía RMI/Código rmi/servidor$
```

A continuación, abrimos una terminal en cliente

```
cliente$ export CLASSPATH=$CLASSPATH:cliente.jar
cliente$ echo $CLASSPATH

cliente$
```

De esta forma exportamos el archivo .jar generado en servidor.

Ahora registramos el registro en la terminal de servidor:

```
Note: Recompile with -Xlint:deprecation for details.
fran@fran-VirtualBox: ~/Escritorio/2. Guia RMI/Código rmi/servidor$ rmiregistry -Djava.security.policy=registrar.policy

```

En una nueva terminal abierta desde el directorio del servidor ejecutamos el servidor

```
fran@fran-VirtualBox: ~/Escritorio/2. Guia RMI/Código rmi/se...
fran@fran-VirtualBox:~/Escritorio/2. Guia RMI/Código rmi/servidor$ java -Djava.security.policy=registrar.policy Registro
Servidor de objeto preparado.
```

Finalmente, en cliente compilamos y ejecutamos

```
fran@fran-VirtualBox:~/Escritorio/2. Guia RMI/Código rmi/cliente$ javac clienteRMI.java
Note: clienteRMI.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
fran@fran-VirtualBox:~/Escritorio/2. Guia RMI/Código rmi/cliente$ java -Djava.security.policy=registrar.policy clienteRMI localhost 1099
[1] Realizar operacion
[2] Salir
Indique la opcion a realizar:
1
[1] Sumar
[2] Multiplicar
[3] Potencia a^b
Indica la operacion a realizar:

```

Una ejecución del código sería la siguiente:

```
[1] Sumar
[2] Multiplicar
[3] Potencia a^b
Indica la operacion a realizar:
1

Introduzca el primer operando [0-9]:
4

Introduzca el segundo operando [0-9]:
4
El resultado es: 8
Desea realizar otra operacion? [s,n]:
s
[1] Sumar
[2] Multiplicar
[3] Potencia a^b
Indica la operacion a realizar:
3

Introduzca el primer operando [0-9]:
2

Introduzca el segundo operando [0-9]:
3
El resultado es: 8
Desea realizar otra operacion? [s,n]:
n
[1] Realizar operacion
[2] Salir
Indique la opcion a realizar:
2
fran@fran-VirtualBox:~/Escritorio/2. Guiada RMI/Código rmi/cliente$
```

En esta, le indicamos que vamos a sumar sumamos $4 + 4$ a lo que devuelve 8 después le decimos que queremos seguir y decimos que queremos hacer 2^3 a lo que evidentemente el resultado también es 8