

**INSTITUTO TECNOLOGICO DE CHILPANCINGO**

**Cómputo En La Nube Y Grandes Datos**

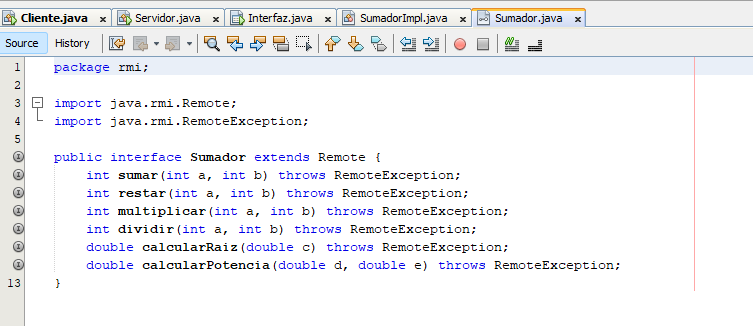
Practica RMI Calculadora Sin Capa

**Alumno:**

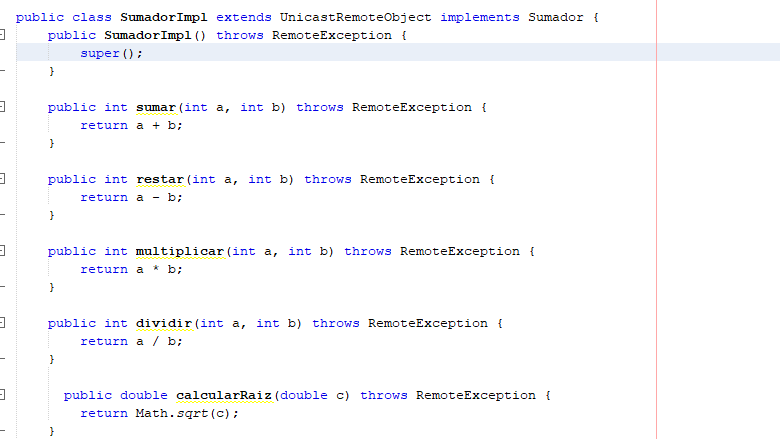
Fierro Monroy Isaac Antonio

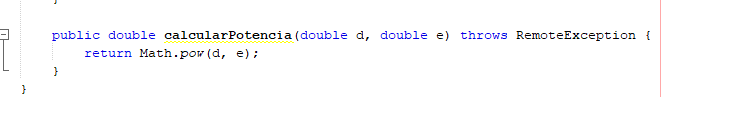
Lunes 13 de mayo de 2024

# RMI SIN CAPA

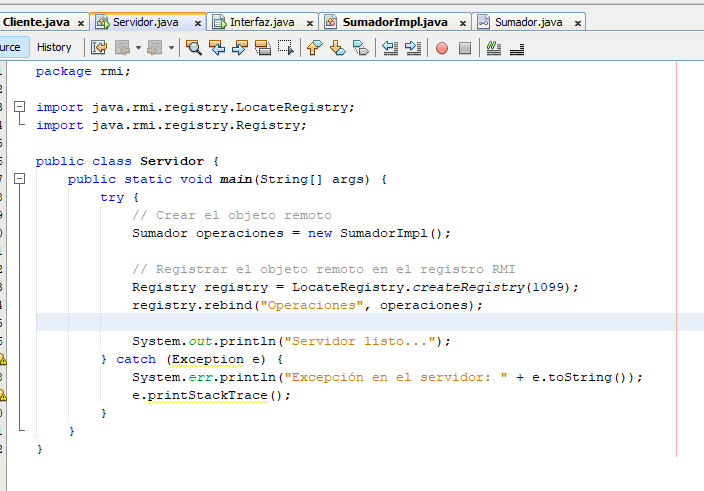


Este código define una interfaz llamada Sumador en el paquete rmi (Remote Method Invocation), que extiende la interfaz Remote. La interfaz Sumador contiene varios métodos que pueden ser invocados de forma remota





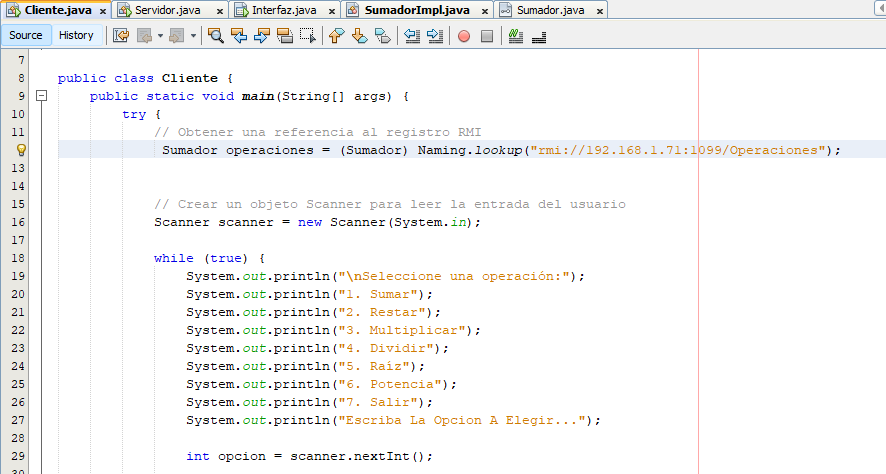
Este código implementa la interfaz Sumador en una clase llamada SumadorImpl en el paquete rmi. La clase SumadorImpl extiende UnicastRemoteObject, lo que la hace disponible para la invocación remota.

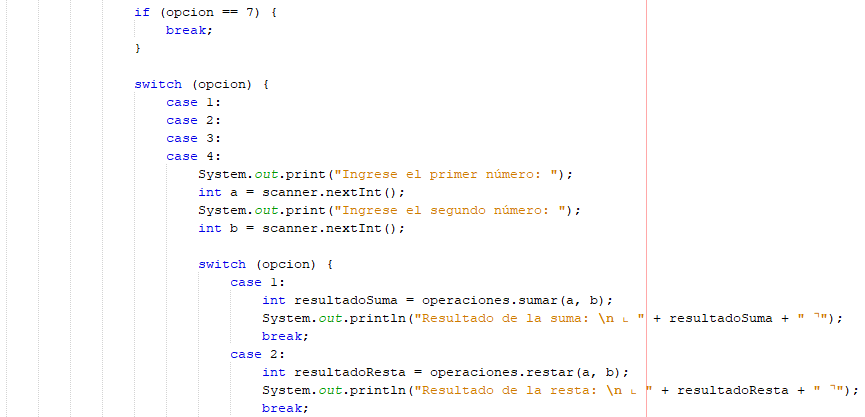


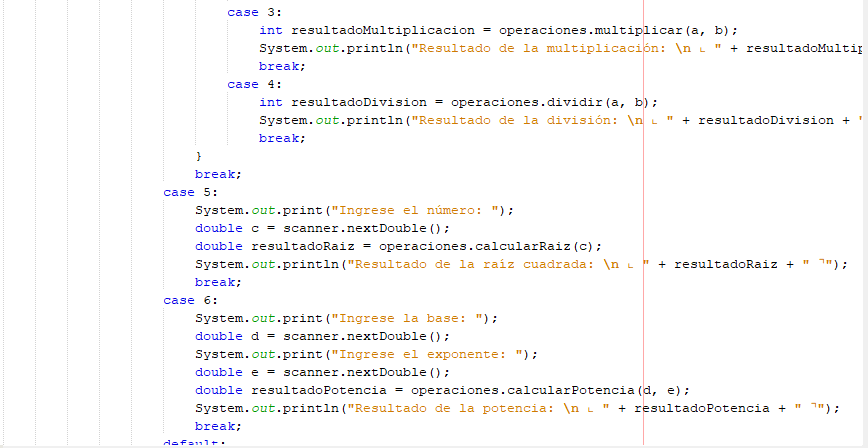
Este código implementa una clase llamada Servidor en el paquete rmi. La clase Servidor contiene un método main que se encarga de iniciar el servidor RMI.

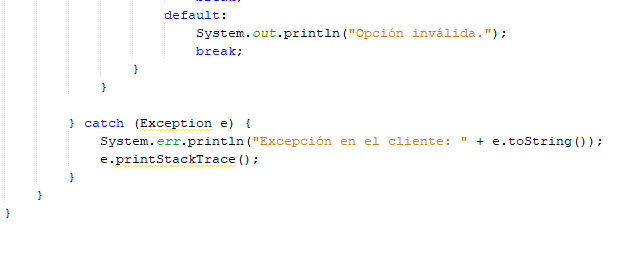
Dentro del método main, se realiza lo siguiente:

1. Se instancia un objeto de la clase SumadorImpl, que implementa la interfaz Sumador. Este objeto representa las operaciones que el servidor RMI ofrecerá a los clientes.
2. Se crea un registro RMI en el puerto 1099 utilizando LocateRegistry.createRegistry(1099). Este registro servirá para que los clientes puedan encontrar y comunicarse con el servidor.
3. Se vincula el objeto remoto (SumadorImpl) al registro RMI utilizando registry.rebind("Operaciones", operaciones). El nombre "Operaciones" es el nombre bajo el cual el objeto remoto será registrado en el registro RMI.
4. Se muestra un mensaje indicando que el servidor está listo para aceptar conexiones.





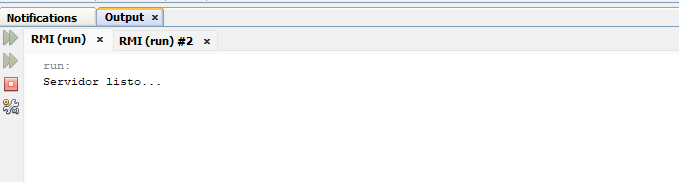


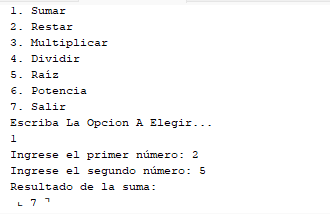


Este código implementa una clase Cliente en el paquete rmi. La clase Cliente contiene un método main que se encarga de interactuar con el servidor RMI para realizar operaciones matemáticas.

Dentro del método main, se realiza lo siguiente:

1. Se utiliza Naming.lookup("rmi://192.168.1.71:1099/Operaciones") para obtener una referencia al objeto remoto registrado en el registro RMI bajo el nombre "Operaciones" en el host con la dirección IP "192.168.1.71" y el puerto "1099".
2. Se crea un objeto Scanner para leer la entrada del usuario desde la consola.
3. Se inicia un bucle while para permitir que el usuario realice múltiples operaciones hasta que decida salir.
4. Se muestra un menú de opciones que incluye las operaciones disponibles: suma, resta, multiplicación, división, cálculo de raíz cuadrada, cálculo de potencia y salir.
5. Se lee la opción elegida por el usuario y se ejecuta el código correspondiente. Dependiendo de la opción seleccionada, se solicitan los operandos necesarios al usuario y se realiza la operación correspondiente utilizando los métodos remotos proporcionados por el objeto operaciones.
6. Se muestra el resultado de la operación en la consola.
7. Si el usuario elige la opción "Salir", se rompe el bucle y el programa termina.





# CONCLUSION

RMI en Java proporciona una forma poderosa y flexible de comunicación entre procesos distribuidos, permitiendo que los objetos Java se invoquen de forma remota. Al implementar interfaces remotas y registrando objetos remotos en un registro RMI, los clientes pueden acceder a los métodos de estos objetos como si estuvieran disponibles localmente.

Sin embargo, al no tener una capa de datos en este contexto, se deben considerar algunos aspectos:

* Seguridad: Es esencial implementar mecanismos adecuados de autenticación y autorización para garantizar la seguridad de las comunicaciones remotas y proteger los datos sensibles.
* Manejo de excepciones: Dado que las llamadas remotas pueden implicar la comunicación a través de la red, es fundamental manejar adecuadamente las excepciones que pueden ocurrir durante estas llamadas, como las excepciones de red o las excepciones específicas de RMI.
* Escalabilidad y rendimiento: Sin una capa de datos, es importante considerar cómo se gestionan y almacenan los datos en el servidor. Se debe tener en cuenta la escalabilidad y el rendimiento del sistema al diseñar la arquitectura de la aplicación y al decidir qué datos deben ser accesibles de forma remota.
* Diseño de la interfaz remota: Es crucial diseñar cuidadosamente las interfaces remotas para minimizar la cantidad de datos transferidos entre el cliente y el servidor, optimizando así el rendimiento y la eficiencia de la comunicación remota.