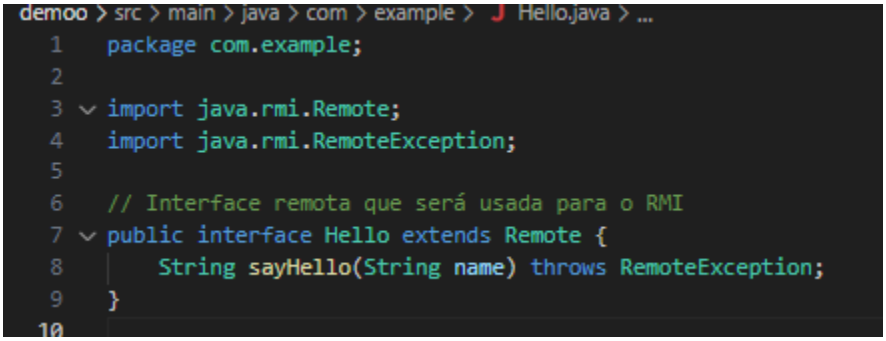
O Java RMI (Remote Method Invocation) é uma tecnologia que permite a comunicação entre objetos distribuídos em diferentes computadores dentro de uma rede. Ele possibilita a execução de métodos remotos, como se estivessem sendo chamados localmente, tornando-se uma solução eficiente para sistemas distribuídos.

Por exemplo, imagine um cenário onde há um servidor que oferece um serviço e um cliente que deseja utilizar esse serviço. O cliente pode chamar um método remotamente como se ele estivesse no mesmo computador, mas, na realidade, o código está sendo executado no servidor, que processa a solicitação e envia o resultado de volta ao cliente.

O código a seguir implementa esse conceito utilizando Java RMI:

1. **Interface Remota (Hello.java)**
    - Define um contrato para o serviço. Diz quais métodos podem ser chamados remotamente.
    - **Interface Hello extends remote:** indica que essa interface pode ser usada remotamente.
    - **String sayHello(String name) throws RemoteException:** Método que o cliente poderá chamar;
    - **Throws RemoteException:** caso ocorra algum erro na comunicação ocorra, ele irá tratá-los.



2. **Implementação do serviço(HelloImpl.java)**
    - Implementa a lógica do serviço;
    - **Extends unicastRemoteObject:** permite que a classe seja usada remotamente;
    - **Implements Hello:** implementa a interface Hello, ou seja, define o que o método **say hello** faz;
    - **super():** registra o objeto no RMI para que ele possa ser acessado;
    - **sayHello(String name):** Quando o cliente chamar esse método, ele vai receber a resposta "Olá, nome! Este é um exemplo de RMI.".

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

// Implementação da interface remota
public class HelloImpl extends UnicastRemoteObject implements Hello {
    protected HelloImpl() throws RemoteException {
        super();
    }

    @Override
    public String sayHello(String name) throws RemoteException {
        return "Olá, " + name + "! Este é um exemplo de RMI.";
    }
}
```

3. Servidor(Server.java)

- Inicia o serviço e disponibiliza para os clientes;
- **LocateRegistry.createRegistry(1099):** cria o RMI Registry, que é onde os serviços ficam registrados para os clientes encontrá-los;
- **HelloImpl hello = new HelloImpl():** Cria uma instância do serviço;
- **Naming.rebind("HelloService", hello):** Dá um nome para o servico (HelloService) para que o cliente possa encontrá-lo;
- **System.out.println("Servidor RMI pronto!"):** Exibe no console que o servidor está funcionando.

```java
package com.example;

import java.rmi.Naming;

public class Client {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Procura pelo serviço remoto no registro
            Hello hello = (Hello) Naming.lookup("rmi://localhost:1099/HelloService");
            // Invoca o método remoto
            String response = hello.sayHello(name:"Usuário");
            System.out.println("Resposta do servidor: " + response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Cliente(Cliente.java)

- Representa o cliente, que se conecta ao servidor e chama o metodo sayHello remotamente;
- **Naming.lookup("rmi://localhost:1099/HelloService"):** O cliente procura o servico no RMI Registry;
- **hello.sayHello("Usuário"):** Chama o método remoto, que será executado no servidor;
- **System.out.println("Resposta do servidor: " + response):** Exibe no console a resposta do servidor.

```java
package com.example;

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

public class Server {
    Run | Debug
    public static void main(String[] args) {
        try {
            // Cria e exporta o registro RMI na porta 1099
            LocateRegistry.createRegistry(1099);
            // Cria uma instância do serviço e o vincula ao registro
            HelloImpl hello = new HelloImpl();
            Naming.rebind("HelloService", hello);
            System.out.println("Servidor RMI pronto!");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Servidor iniciado

```
caiocezar@DESKTOP-0TV6PJ1:~/JavaSemRMI$  cd /home/caiocezar/JavaSemRMI ; /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java -XX:+ShowCodeDetail
sInExceptionMessages -cp /home/caiocezar/JavaSemRMI/demo/target/classes com.example.Server
Servidor pronto na porta 4000...
```

Depois Cliente executado

```
caiocezar@DESKTOP-0TV6PJ1:~/javaComRMI$  cd /home/caiocezar/javaComRMI ; /usr/bin/env /usr/lib/jvm/java-21-openjdk-amd64/bin/java @/tmp/cp_8ryz4daf7q
ar0plyi5f2o9u9k.argfile com.example.Client
Resposta do servidor: Olá, Usuário! Este é um exemplo de RMI.
```