

Prof. Rogério Santos Pozza

**Universidade Tecnológica Federal do Paraná
Campus Cornélio Procopio**

Java Aplicada em Redes de Computadores

Especialização em Tecnologia Java

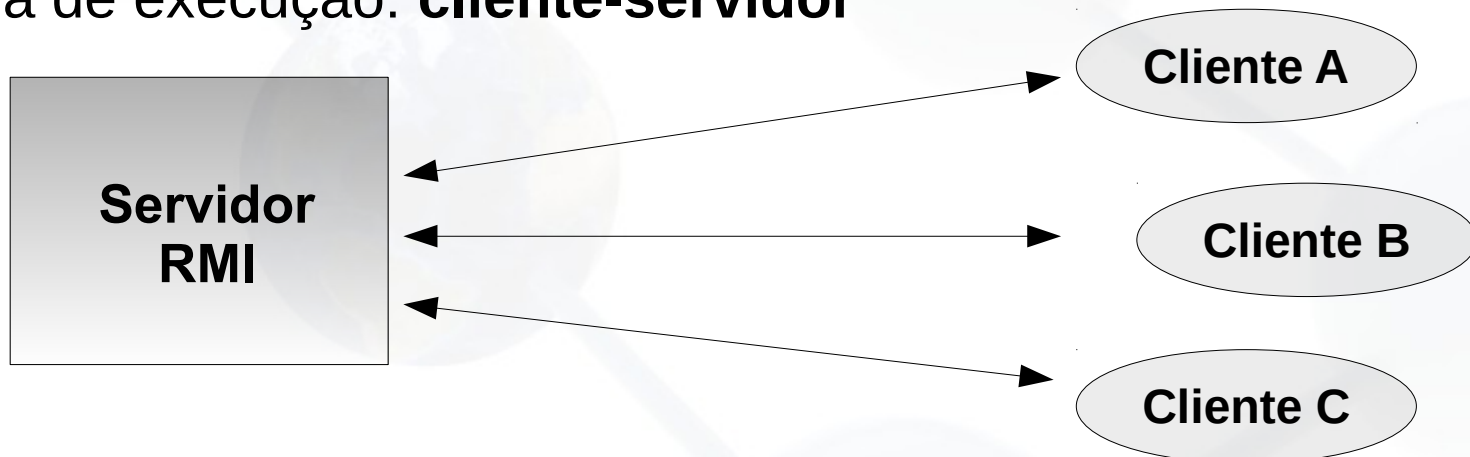
26 de abril de 2018

Na aula de hoje veremos...

- **Invocação de métodos remotos (RMI)**

RMI – Remote Method Invocation

- O Java RMI permite criar aplicações Java distribuídas de modo que os métodos de objetos de uma máquina virtual, possam ser invocados por outras máquinas Java em diferentes máquinas
- Permitir que os programadores possam desenvolver aplicações distribuídas Java com a **mesma sintaxe de programas não distribuídos**
- Classes Java são mapeadas em uma máquina virtual para que outras JVM possam invocar os seus métodos remotamente
- Arquitetura de execução: **cliente-servidor**



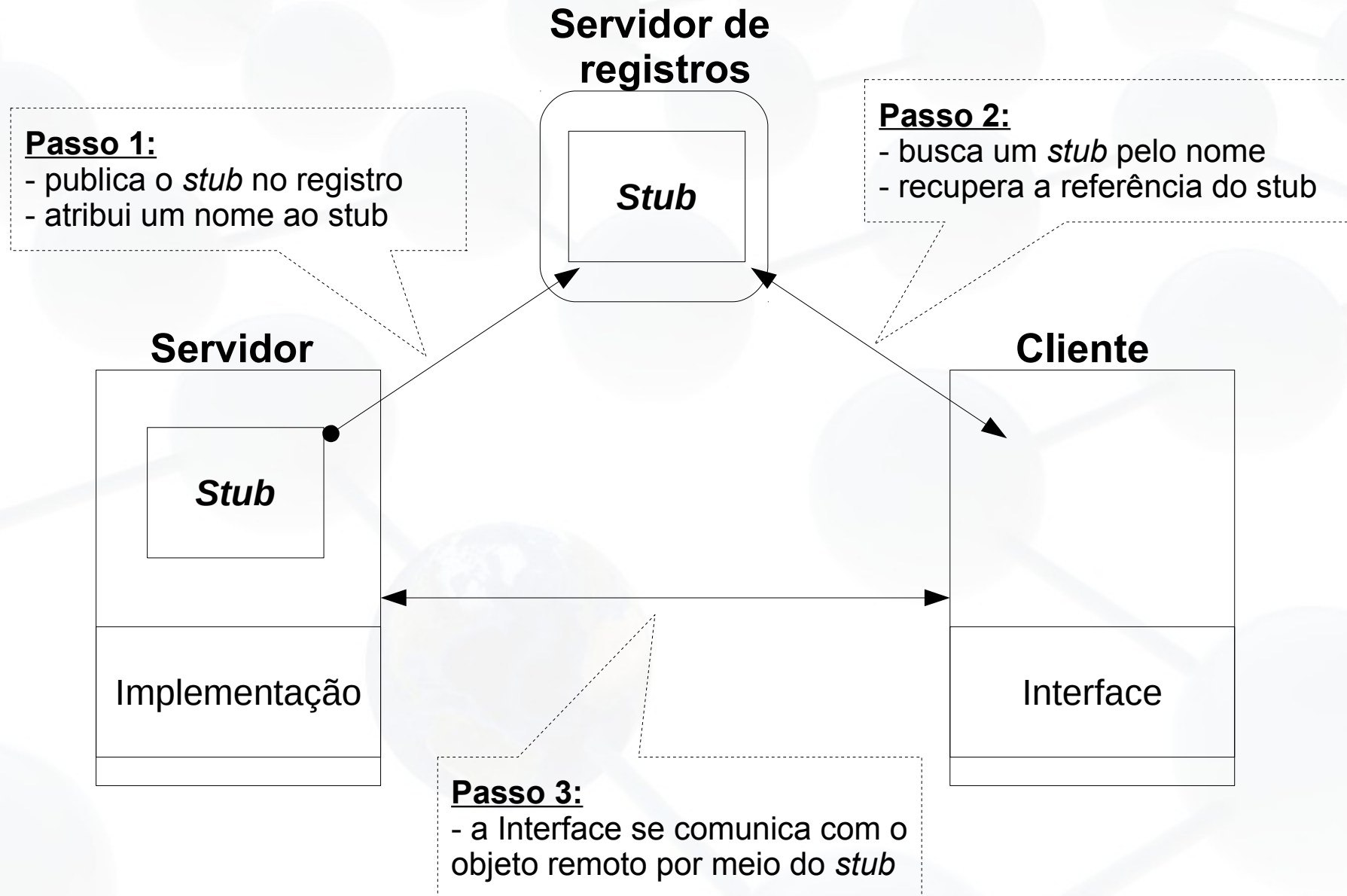
RMI – Remote Method Invocation

- Embora a invocação de métodos remotos seja executada da mesma forma que métodos locais, a invocação de métodos remotos pode falhar por alguns motivos:
 - Máquina servidora desligada;
 - Endereço ou nome do método incorreto;
 - Falhas no meio de transmissão de dados
 - ...
- As invocações de métodos remotos devem ser tratadas obrigatoriamente através de exceções

RMI – Como funciona?

- No RMI alguns conceitos da programação convencional são modificados
- A criação de um objeto local é feita utilizando-se o operador **new**. No RMI apenas o servidor pode criá-lo.
- O acesso a objetos remotos é realizado via um objeto de referência (*stub*)
- Uma referência remota é um ponteiro para um objeto proxy (*stub*)
- O *stub* possui informações que permitem a sua conexão ao objeto remoto de forma transparente, o qual contém a implementação dos métodos.

RMI – Como funciona?



RMI – Como funciona?

- **Localizar objetos remotos** – uma aplicação pode usar dois mecanismos para obter referências de objetos remotos:
 - ***rmiregistry***: registrar o objeto remoto com a ferramenta de nomes do RMI, chamada
 - Passar e retornar referência aos objetos remotos como parte de sua operação normal
- **Comunicação com objetos remotos** – os detalhes de comunicação são abstraídos pelo RMI
 - O programador identifica uma chamada remota de forma idêntica a uma chamada local
- **Carregar “bytecodes” de objetos remotos** – como o RMI permite que objetos remotos sejam passados como parâmetros num método, ele fornece mecanismos necessários para carregar o código dos objetos remotos (serialização/desserialização implícitas)

RMI – *Stub*

- Quando um objeto local invoca um método de um objeto remoto, o *stub* fica responsável por enviar a chamada ao método para o objeto remoto
- Passos do *stub* quando é invocado:
 - Iniciar conexão com a máquina virtual que contém o objeto remoto (servidor)
 - Escrever e transmitir os parâmetros para a máquina virtual remota
 - Esperar pelos resultados da invocação do método
 - Ler os resultados retornar
 - Retornar os valores ao objeto que executou a chamada

RMI – *Stub*

- O stub esconde a serialização dos parâmetros e toda a comunicação em nível de rede com o objetivo de simplificar o mecanismo de realização da invocação do método.

Um *stub* aumenta a **TRANSPARÊNCIA** da comunicação



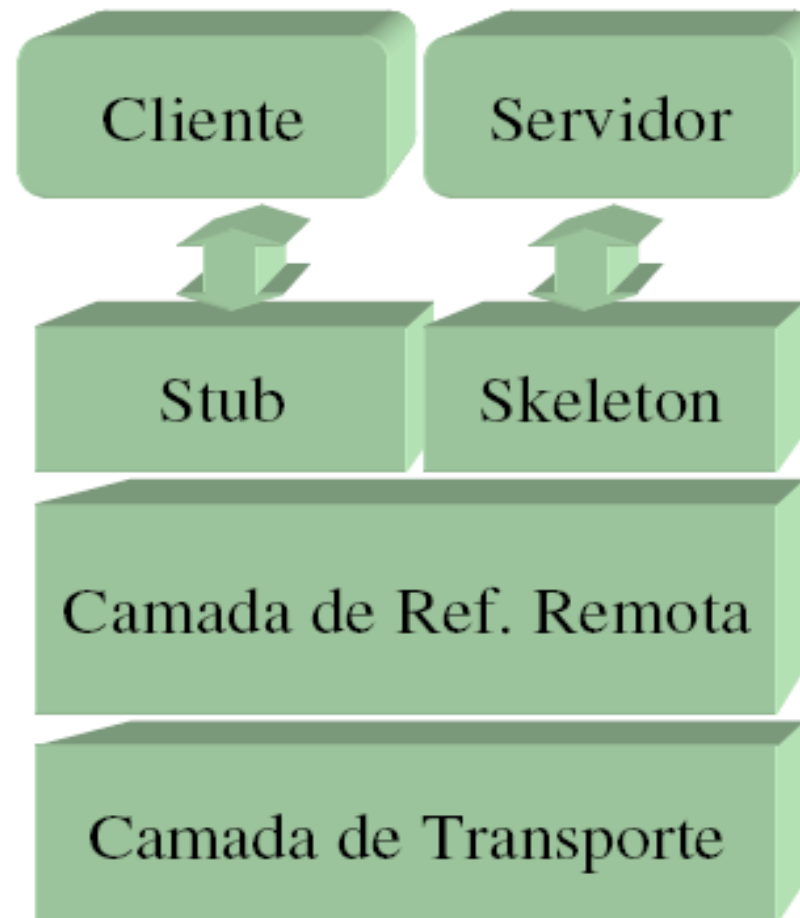
Você se lembra do trabalho de fazer isso com Sockets?

RMI – *Skeleton*

- Para o servidor receber a informação que está no *stub* do cliente, ele cria um objeto *skeleton*.
- O *skeleton* recebe invocações diretamente do *stub* e as repassa para o objeto servidor
- O *skeleton* executa as seguintes ações:
 - Decodifica os parâmetros;
 - Chama localmente o método remoto desejado;
 - Captura o valor de retorno ou exceção e codifica-o;
 - Retornar o valor codificado para o cliente.
- Em RMI a comunicação de rede é realizada entre o *stub* de um objeto com o *skeleton* de outro objeto remoto

RMI – *Skeleton*

Stub e Skeleton



RMI – Implementação

- A implementação de um programa simples em RMI consiste nos seguintes passos:
 1. Criar uma interface
 2. Implementação da interface
 3. Gerar os stubs e skeletons
 4. Implementar o servidor
 5. Implementar o cliente

RMI – Implementação – Passo 1

- Crie o arquivo “Calculadora.java”

```
public interface Calculadora extends java.rmi.Remote {  
    public int add(int a, int b) throws java.rmi.RemoteException;  
}
```

- **Salve e compile:**

```
$ javac Calculadora.java
```

RMI – Implementação – Passo 2

- Crie o arquivo “CalculadoraImpl.java”

```
public class CalculadoraImpl extends  
java.rmi.server.UnicastRemoteObject implements Calculadora {  
  
    public CalculadoraImpl() throws java.rmi.RemoteException {  
        super();  
    }  
  
    public int add(int a, int b) throws java.rmi.RemoteException {  
        return a+b;  
    }  
}
```

- Salve e compile:

```
$ javac CalculadoraImpl.java
```


RMI – Implementação – Passo 3

- **Gere os stubs e skeletons:**

```
$ rmic CalculadoraImpl
```

- ***Verifique se o arquivo “CalculadoraImpl_Stub.class” foi criado no diretório***

RMI – Implementação – Passo 4

- É preciso criar um serviço RMI em um host. Pode-se criar a classe **CalculadoraServer.java**:

```
import java.rmi.Naming;

public class CalculadoraServer {

    public CalculadoraServer() { }

    public static void main(String args[]) {
        try {

            Calculadora calc = new CalculadoraImpl();
            Naming.rebind("rmi://localhost:1099/CalcServer", calc);

        } catch (Exception e) {
            System.out.println("Erro RMI: " + e.toString());
        }
    }
}
```

RMI – Implementação – Passo 5

- A classe cliente pode consumir um método remoto, implementando a classe CalculadoraCliente.java

```
import java.rmi.Naming;

public class CalculadoraCliente {

    public static void main(String args[]) {
        try {

            Calculadora calc =
                (Calculadora)Naming.lookup("rmi://localhost/CalcServer");

            System.out.println("A soma de 1 + 5 é: " + calc.add(1, 5));

        } catch (Exception e) {
            System.out.println("Erro RMI: " + e.toString());
        }
    }
}
```

RMI – Implementação – Executando o RMI

- Para publicar um objeto RMI, deve-se inicializar o servidor de registros por meio do comando:
\$ rmiregistry
- Em seguida, deve-se executar o CalculatorServer para que o serviço seja publicado
\$ java CalculadoraServer
- Por fim, o programa cliente que irá usufruir do método remoto, pode ser executado
\$ java CalculadoraCliente

RMI – Implementação

- Para passar um objeto por valor para um método remoto, deve-se serializá-lo (**java.io.Serializable**)

```
public class MinhaClasse implements Serializable {  
  
}
```

- Um objeto para ser passado por referência deve estender a sua interface **java.rmi.Remote**

```
public interface ClasseInterface extends Remote {  
  
}
```

RMI – Implementação

- Os métodos podem ser invocados em um objeto remoto e devem:
 - Obrigatoriamente públicos
 - Estar declarados na interface remota que esse objeto implementa
 - Tratar obrigatoriamente a exceção `java.rmi.RemoteException`

Exemplo:

```
public int Testar() throws java.rmi.RemoteException
```

- A exceção também deve ser colocada na interface que implementa o método

RMI – Ativando o servidor de registros

- Antes de utilizar o serviço é necessário ativar o servidor de registros:
- **Opção 1:**
Comando: `rmiregistry <porta>`
- **Opção 2:**
`Registry registro = LocateRegistry.createRegistry(porta);`
permite criar o rmiregistry na inicialização do programa, não sendo necessário a execução antes da inicialização do programa.
- **Opção 3:**
`Registry registro = LocateRegistry.getRegistry();`
recupera a referência a um servidor de registro

Exercícios

1. Crie um servidor Java RMI capaz de validar um CPF passado pelo cliente
2. Uma cidade está realizando uma eleição. Você é o desenvolvedor que irá projetar uma aplicação capaz de receber os votos de cada urna espalhada na cidade. Assim, pede-se:
 - O servidor RMI deverá estar apto a realizar duas funções:
 - Contar todos os votos;
 - enviarVotos (contendo os nomes dos candidatos e os números de votos);
 - Cada urna (cliente RMI) deverá enviar os nomes e o número de votos de cada candidato para o servidor.
 - O servidor deverá exibir a apuração de votos atualizada a cada 5 segundos.