RMI
Remote Method Invocation

Java RMI

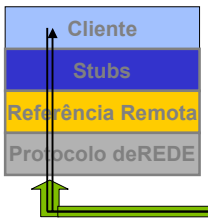
- Java Remote Method Invocation (RMI) é um protocolo Java para comunicação entre processos
- Permite objetos Java invocar transparentemente métodos de outros objetos (que podem estar em máquinas diferentes – objetos remotos)
- Java RMI libera o programador de tratar de detalhes como endereçamento e codificação/ decodificação de mensagens

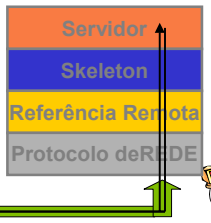
Chamadas Remotas de Procedimentos (RPC)

- RMI consiste em chamada remota de procedimentos em Java (onde as operações são representadas por métodos)
- Chamada Remota de Procedimento (RPC) ou Invocação Remota de Método (RMI) ou Chamada Remota de Função
- Método de transferência de controle de parte de um processo para outra parte
- Modelo de RPC é derivado da Chamada de Procedimentos Convencionais

RPC - Implementação

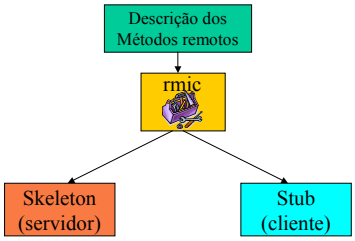
- Stubs do lado do cliente: Stubs
- Stubs do lado do servidor: Skeleton





Compilador rmic

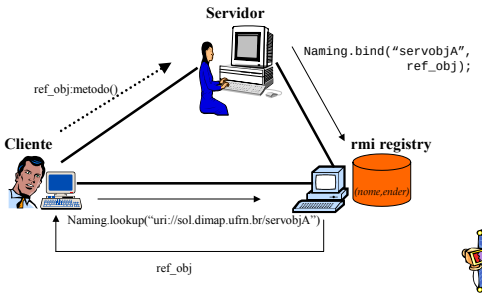
- O compilador *rmic*
  - recebe como entrada a descrição dos métodos remotos
  - gera dois arquivos: Stub e Skeleton.



Localização de Objetos Remotos

- Seria impraticável se para cada invocação de método remoto fosse necessário incluir o par (máquina,porta) para identificar onde se encontra o objeto que contém o método
- RMI oferece um *Serviço de Nomes* (***RMI Registry***) que oferece informações sobre a localização de objetos remotos.
  - o `rmiregistry` executa em um endereço bem conhecido.

RMI Registry



Programação com RMI

- O modelo RMI:
  - O servidor define objetos que o cliente pode usar remotamente
  - Os clientes podem invocar métodos nesse objeto remoto como se ele estivesse executando localmente.
  - RMI esconde o mecanismo subjacente de transporte, via rede, de argumentos dos métodos e valores de retorno.

Programação com RMI

- Defina uma **`interface`** que declara os métodos remotos
- O programa *servidor*
  - deve incluir uma **`class`** that implementa essa **`interface`**.
  - deve criar um objeto remoto e registrá-lo no serviço de nomes (*rmi register*)
- O programa *cliente*
  - deve perguntar ao serviço de nomes pela referência do objeto remoto.
  - deve invocar o método sobre o objeto remoto

Interface

- Similar a classe
- Não há implementação, apenas declaração de métodos
- Tudo é público
- É uma API que pode ser implementada por uma classe

```
public interface BDados{
  public string ler();
  public void escreve(string);
}
```

```
public interface HelloWorld{
    public string hello();
}
```

Servidor
Interface com métodos remotos

- A interface definida pelo servidor deve declarar que os métodos da interface serão invocados por clientes remotos
  - A *interface* deve estender a interface Java ***Remote*** que o pacote **`java.rmi`** oferece

```
import java.rmi.*
public interface HelloWorld extends Remote{
    public string hello() throws RemoteException;
}
```

Indica que a interface é remota

Necessária para Sinalizar erros da Chamada remota

Servidor
Implementação da Classe

- Criar uma classe que *implementa* a interface. A classe deve estender **`UnicastRemoteObject`***
- O código **`main()`** deve:
  - Criar um objeto remoto
  - Registrar esse objeto no serviço de Nomes
- A classe precisa definir um construtor para **`RemoteException`** !
- A classe será usada pelo compilador **`rmic`** para criar o código do stub e do skeleton.

Servidor
Implementação da Classe

```
import java.rmi.*
import java.rmi.server.*
public class HelloWorldImpl extends UnicastRemoteObject
    implements HelloWorld{
    public HelloWorldImpl() throws RemoteException
    { ….  }
    public string hello() {
         return "==== Hello World!!!==== ";
    }
    public static void main (String args[]) {
    try {
         // criando o objeto remoto
         HelloWorldImpl obj = new HelloWorldImpl();
         // registrando esse objeto no serviço de nomes
          Naming.bind("HelloWorldServer", obj);
    } catch (RemoteException e) {
      System.out.println("ERRO")
   }
  }
}
```

Gerando Stubs e Skeletons

- Compile a interface remota e a implementação:

```
> javac HelloWorld.java HelloWorldImpl.java
```

- Use **`rmic`** para gerar **`HelloWorldImpl_stub.class`**, **`HelloWorldImpl_skel.class`**

```
> rmic HelloWorldImpl
```

Cliente

- O cliente precisa obter, do serviço de nomes, a referência para o objeto remoto
- O serviço de nomes usa URLs para identificar objetos remotos.

Cliente
Implementação

```
import java.rmi.*
    try {
         // recuperando o objeto remoto via o servidor de nomes
         Object obj= Naming.lookup("uri://pontanegra.dimap.ufrn.br
"                                          /HelloWorldServer);
         // invocando o método hello do servidor remoto
         mensagem = obj:hello()
    } catch (RemoteException e) {
      …
   }
```

Executando...

- Inicialmente deve-se executar o servidor de Nomes

```
start rmiregistry
```

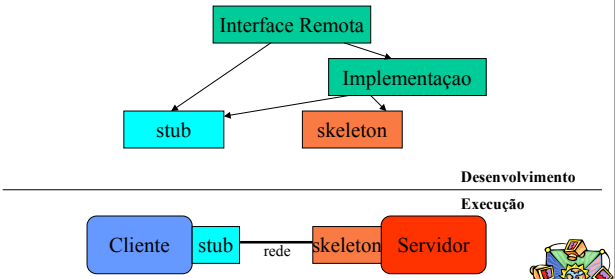
- Em seguida deve-se executar o servidor:

```
java HelloWorld
```

Passos do RMI

- 1. Defina a interface do servidor (estendendo a interface java.rmi.remote)
- 2. Escreva o código do servidor que implementa a interface (herdando da classe java.rmi.UnicastRemoteObject)
- 3. Compile o servidor
  - Use o compilador ***javac*** para produzir o arquivo .class
  - Execute o compilador ***rmic*** para obter o stub e skeleton
- 4. Execute o programa servidor (o rmiregister tem de estar executando antes)
- 5. Execute o cliente

RMI



RMI

- Implementa uma idéia "*elementar*" de Sistema baseado em Middleware