Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação

SISTEMAS DISTRIBUÍDOS

MARCOS RODRIGO MOMO
marcos.rodrigomomo@gmail.com

Roteiro

- Comunicação entre objetos distribuídos
  - Socket, RMI, RPC, Eventos
- Web Services
- Redes Peer-to-Peer
- Midleware orientado a mensagens
- Memória compartilhada

Objetos Distribuídos
Orientação a Objetos

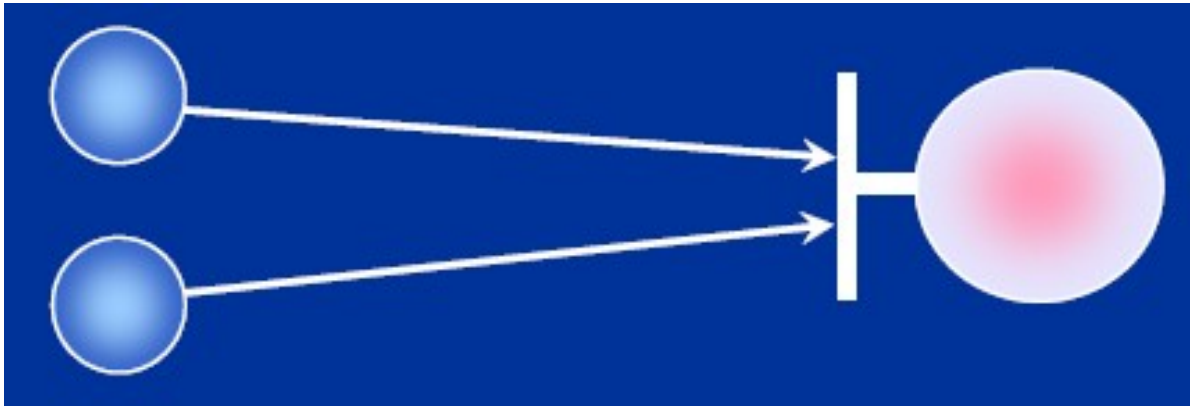
- Encapsulamento:
  - Parte interna (privada) dos objetos
    - Implementação: métodos
    - Estado: atributos, variáveis, constantes e tipos
  - Parte externa (pública) dos objetos
    - Interface: conjunto bem definido de métodos públicos que podem ser acessados externamente

Objetos Distribuídos
Orientação a Objetos

- Herança: de interfaces e implementações
- Polimorfismo: a mesma interface pode ter várias implementações
- Interação entre objetos
  - Troca de mensagens (chamadas de métodos)
  - Mensagens podem ser locais ou remotas
    - Mensagens locais: objetos no mesmo espaço de endereçamento
    - Mensagens remotas: objetos em máquinas diferentes, objetos distribuídos

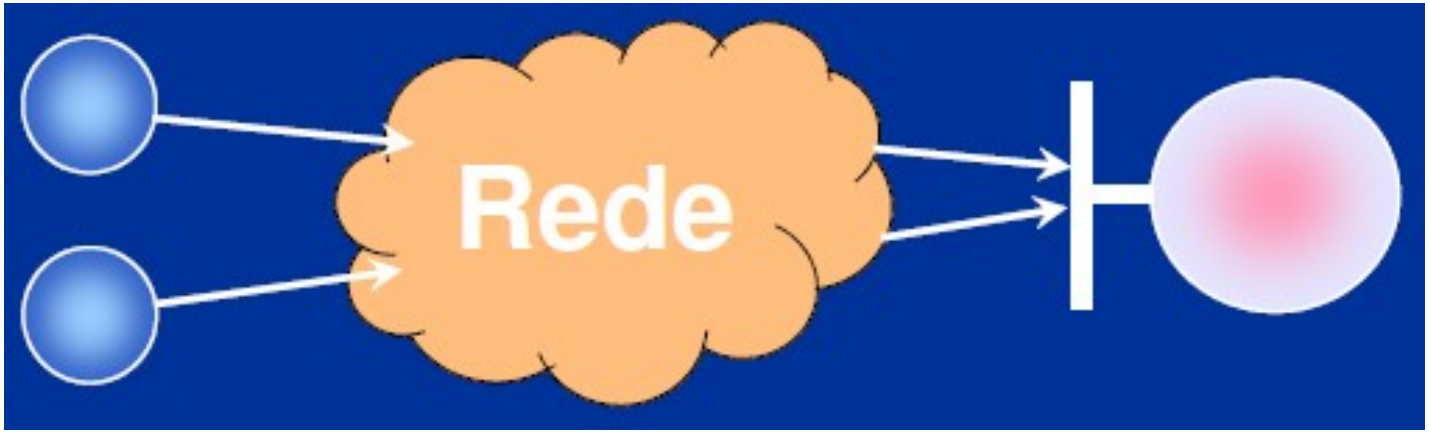
Objetos Distribuídos
Orientação a Objetos

- Referência do objeto -> Ponteiro de memória
- O acesso ao estado do objeto é feito através dos métodos da interface
  - Única parte visível do objeto
- Implementação independente da interface
- Métodos são acessados por outros objetos



Objetos Distribuídos
Orientação a Objetos

- Interagem através da rede
- Colaboram para atingir um objetivo
- Fornecem serviços (métodos) uns aos outros
- Apenas a interface do objeto é visível
- Referência do objeto possui endereço de rede



Objetos Distribuídos
Problemas

- Como compartilhar referências de objetos?
- Como gerenciar o ciclo de vida dos objetos?
- Como gerenciar o acesso concorrente aos objetos?
- Como trabalhar num ambiente heterogêneo?
  - Máquinas podem ter arquiteturas diferentes
  - Máquinas podem estar em redes diferentes
  - Máquinas podem rodar S.O.’s diferentes
  - Objetos podem ser implementados em linguagens diferentes

Objetos Distribuídos
Problemas

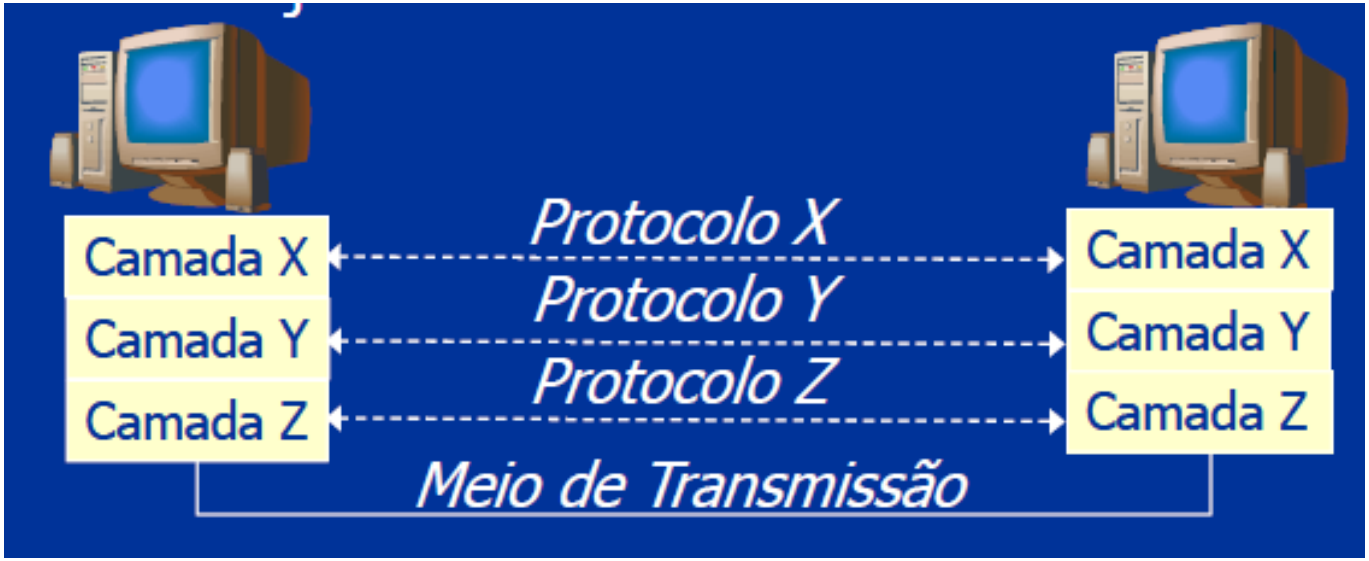
- Comunicação não confiável e não-determinista: depende da dinâmica do sistema e da rede

- Custo da comunicação: latência e largura de banda são fatores críticos em aplicações de tempo real, multimídia, etc.

- Comunicação insegura: sem controle de autorização e sem proteção das mensagens

Objetos Distribuídos
Protocolos de Comunicação

- Estabelecem caminhos virtuais de comunicação entre duas máquinas
- Devem usar os mesmos protocolos para trocar informações



Objetos Distribuídos
Protocolos de Comunicação

- Serviço sem Conexão: cada unidade de dados é enviada independentemente das demais

- Serviço com Conexão: dados são enviados através de um canal de comunicação

Objetos Distribuídos
Protocolos de Comunicação

- Protocolos de alto nível são necessários para interação entre objetos distribuídos
- Escolha natural: usar TCP/IP
  - Cria conexões entre processos para trocar mensagens
  - Amplamente disponível, confiável e robusto
  - Relativamente simples e eficiente

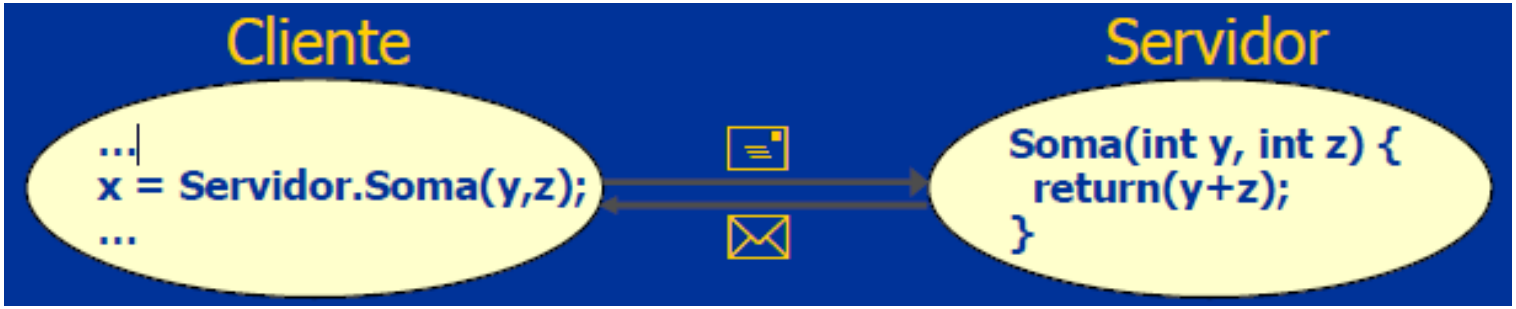
Objetos Distribuídos
Protocolos de Comunicação entre Objetos

- Trata questões não resolvidas pelo TCP/IP
  - Formato comum dos dados
  - Localização de objetos
  - Segurança

- Oferece ao programador abstrações próprias para aplicações orientadas a objetos
  - Chamada Remota de Procedimento (RPC) ou Invocação Remota de Métodos (RMI)
  - Notificação de Eventos

Objetos Distribuídos
RPC- Chamada Remota de Procedimento

- Segue o modelo Cliente/Servidor
- Muito usado na interação entre objetos
- Objeto servidor possui interface com métodos que podem ser chamados remotamente
- Objetos clientes usam serviços de servidores



Objetos Distribuídos
RPC- Chamada Remota de Procedimento

- Características
  - Em geral as requisições são ponto-a-ponto e síncronas
  - Dados são tipados
    - Parâmetros da requisição
    - Retorno do procedimento/método
    - Exceções
  - Um objeto pode ser cliente e servidor em momentos diferentes

Objetos Distribuídos
RPC- Chamada Remota de Procedimento

- Funcionamento
  - Chamada é feita pelo cliente como se o método fosse de um objeto local
  - Comunicação é feita transparentemente por código gerado automaticamente pelo compilador (stub, proxy, skeleton, ...)
  - O código gerado faz a serialização e desserialização de dados usando um formato padrão, que compatibiliza o formato de dados usado por diferentes máquinas, linguagens e compiladores

Objetos Distribuídos
RPC- Chamada Remota de Procedimento

- Funcionamento do Cliente
  - Acessa objeto local gerado automaticamente que implementa interface do servidor remoto

```
Public class HelloServerStub {
    Public String hello(String nome) {
     // Envia pela rede o identificador do método e o valor dos
     // parâmetro(s) da chamada serializados para o servidor
     // Recebe do servidor o valor do retorno da chamada pela
     // rede, o deserializa e retorna o valor recebido ao cliente
     }
// Outros métodos ...
}
```

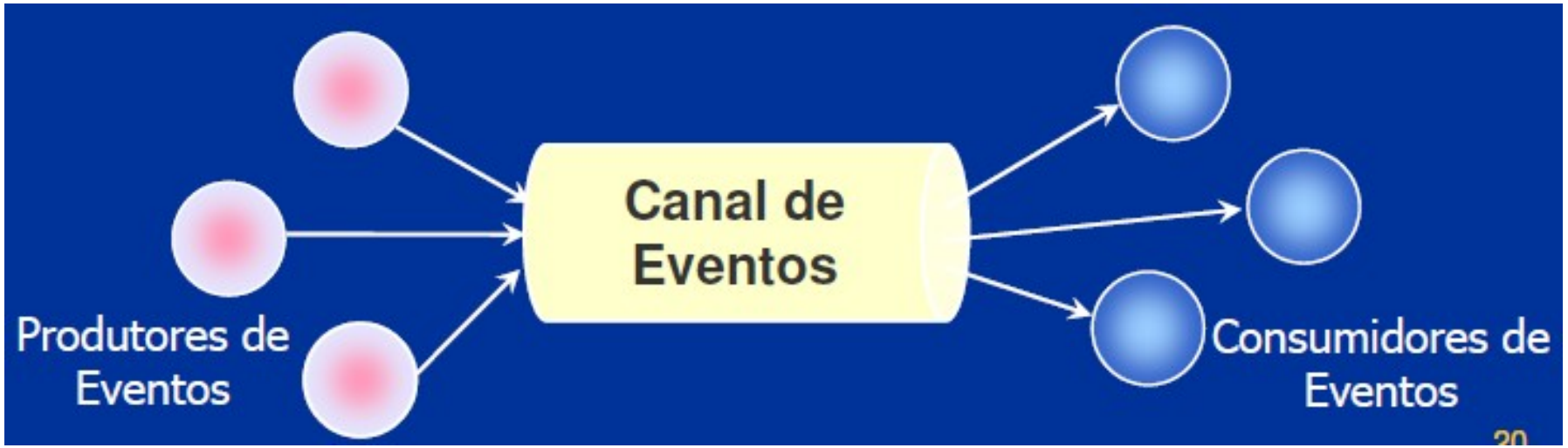
Objetos Distribuídos
RPC- Chamada Remota de Procedimento

- Funcionamento do Servidor
  - O código gerado automaticamente recebe as chamadas pela rede e as executa

```
while (true) {
    // Recebe pela rede o identificador do método chamado
    // pelo cliente e os parâmetros da chamada serializados
// Deserializa os parâmetros enviados pelo cliente
// Chama o método no objeto servidor e aguarda a execução
// Serializa o valor do retorno da chamada e envia ao cliente
```

Objetos Distribuídos
Notificação de Eventos

- Eventos ocorridos são difundidos por produtores e entregues a consumidores
- Canal de eventos permite o desacoplamento – produtor e consumidor não precisam se conhecer



Objetos Distribuídos
Notificação de Eventos

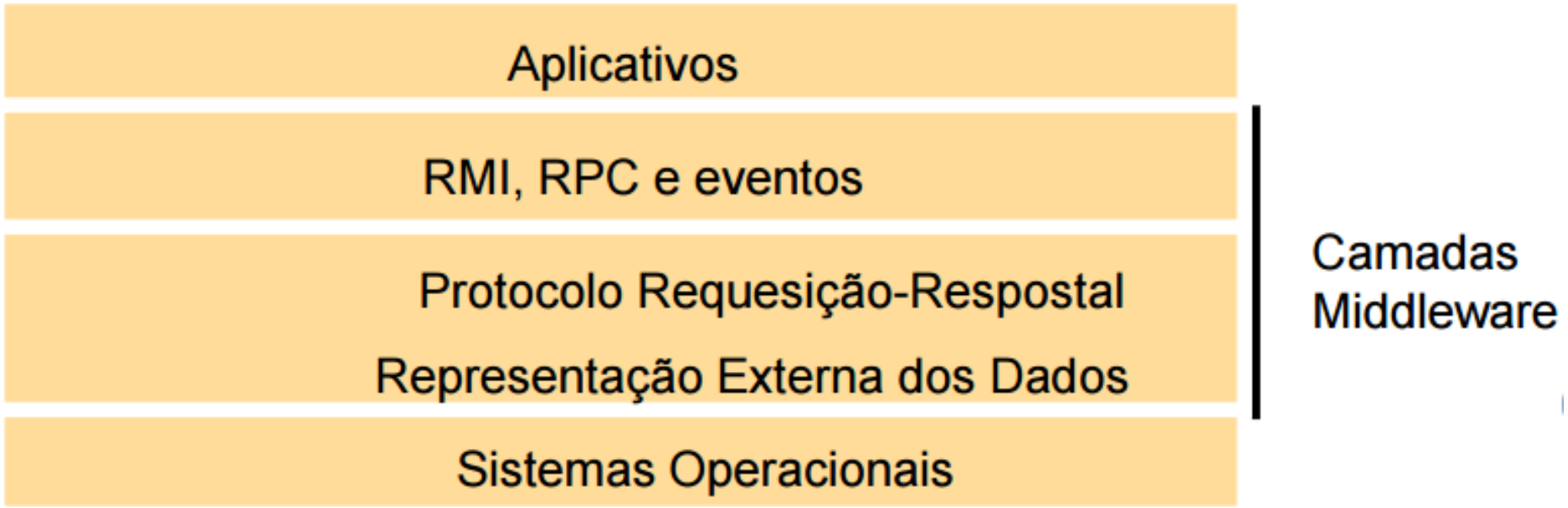
- Características
  - Envio de eventos é completamente assíncrono
  - Produtor não precisa aguardar fim do envio
  - Evento é armazenado no canal de eventos
  - Comunicação pode ser feita através:
    - De UDP multicast ou
    - Fazendo múltiplos envios unicast com TCP, UDP ou
    - Com um suporte de RPC
  - Os eventos podem ter tamanho fixo ou variável, limitado ou ilimitado
  - Eventos podem ser tipados ou não

Introdução - Middleware

Middleware para objetos distribuídos permite:

- ✓ Transparência de localização dos objetos
- ✓ Criação de objeto local e remoto idêntica
- ✓ Migração de objetos transparente
- ✓ Facilidade para ligação *(binding)* de interfaces dinamicamente
- ✓ Independência de protocolos de comunicação
- ✓ Independência do hardware de computador
- ✓ Ocultação do sistema operacional
- ✓ Uso de várias linguagens de programação
- ✓ Diversos serviços de suporte:
  - ✓ Nomes, Transação, Tempo, etc.

Introdução - Middleware



Objetos Distribuídos

Principais suportes de Middleware para Objetos Distribuídos:

- Java RMI (Remote Method Invocation) Sun Microsystems
- DCOM (Distributed Component Object Model) Microsoft Corporation
- CORBA (Common Object Request Broker Architecture) da OMG (Object Management Group)

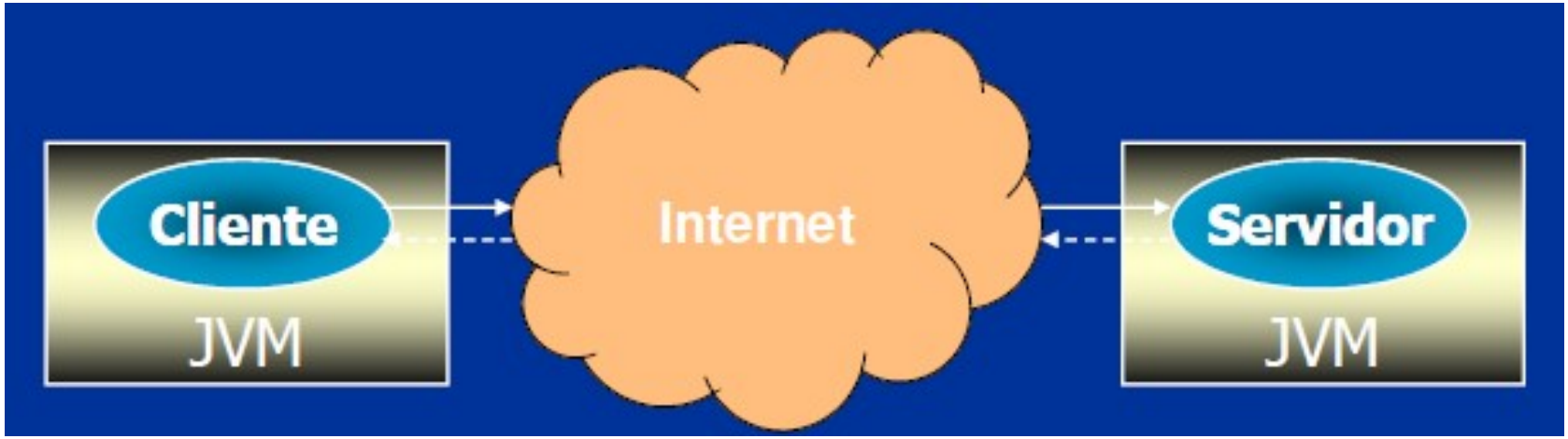
Java Orientada a objetos

JAVA:

- Orientada a objetos
- Possui diversas APIs amigáveis
- Multi-plataforma: Java Virtual Machine (JVM)
- Integrada à Internet: applets, JavaScript, JSP e Servlets
- Suporte a componentes: JavaBeans e EJB
- De fácil aprendizagem
- Bem aceita pelos programadores
- Suportada por diversos fabricantes de SW

Java RMI (Remote Method Invocation):

- Fornece um suporte um suporte simples para RPC/RMI
- Permite que um objeto Java chame métodos de outro objeto Java rodando em outra JVM
- Solução específica para a plataforma Java



Java RMI (Remote Method Invocation):

- Arquitetura RMI:
  - Stub e Skeleton
  - Camada de referência remota
  - Camada de transporte

Atividades de laboratório

Implementar uma aplicação com RMI

Pegar as classes de exemplo no AVA3

Atividades de laboratório - RMI Comunicação entre Objetos Distribuídos

- O Servidor RMI
  - Objetivo aceitar as chamadas remotas de métodos
  - O Servidor deverá estender a interface java.rmi.Remote e declarar todos os métodos que poderão ser acessados remotamente.
- Na interface RMI
  - Será definido um método, que retorna uma saudação quando é chamado
- No Cliente RMI
  - O cliente obterá uma referência para o servidor no registro RMI
  - Chamará o método

Acesso ao Serviço de Cálculo de Taxa de Comissão

O acesso ao serviço utiliza o protocolo RMI (Java). Os dados para acesso ao serviço são:

JNDI Initial Factory	com.sun.jndi.rmi.registry.RegistryContextFactory
JNDI Provider URL	rmi://127.0.0.1:11099
Security Policy	${app.home}/java_policy
Nome do método	calcularValorComComissao
Argumentos do método	float

O cliente do serviço RMI precisa definir um arquivo com a "*security policy*" de acesso. Abaixo fornece-se um exemplo desse arquivo:

Arquivo: java_policy

```
grant {
    permission java.security.AllPermission;
};
```

Atenção! **Esse arquivo é um exemplo ilustrativo**. Num ambiente real, ele seria extremamente permissivo e pouco aconselhável, já que representa uma falha de segurança.

TRABALHO FINAL