UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

PROGRAMA DE GRADUAÇÃO EM REDES DE COMPUTADORES

JOÃO FERREIRA LIMA NETO

VINICIUS GABRIEL RODRIGUES DO NASCIMENTO

RELATÓRIO DA SEGUNDA ENTREGA DO TRABALHO FINAL DE SISTEMAS DISTRIBUÍDOS COM O TEMA DE “LOJA DE PEÇAS DE VEÍCULOS”

QUIXADÁ

INICIE O CÓDIGO COM:

```
javac PecaServer.java
   java PecaServer
   java PecaClient
```

ADIÇÃO DOS REQUISITOS

1 - Comunicação Cliente-Servidor via RMI:

O código utiliza RMI para comunicação entre cliente e servidor, sem o uso direto de sockets. Isso atende ao requisito.

2 - Protocolo de Requisição-Resposta:

O código implementa o protocolo de requisição-resposta através dos métodos doOperation, getRequest e sendReply.

O método doOperation envia uma requisição ao servidor e retorna a resposta.

Os métodos getRequest e sendReply são simulados para exemplificar o recebimento de requisições e o envio de respostas.

3 - Representação Externa de Dados:

O código utiliza JSON para serialização e desserialização de objetos Peca por meio da biblioteca Gson. Isso atende ao requisito de usar uma representação externa de dados (JSON, XML ou Protocol Buffers).

Demais Requisitos:

4 Classes POJOs: Peca, Amortecedor, Motor, Pneu.

2 Composições do Tipo Agregação ("tem-um"):

PecaServiceImpl tem uma lista de Peca.

RMIClient tem uma referência para o Registry.

2 Composições do Tipo Extensão ("é-um"):

Amortecedor, Motor e Pneu são subclasses de Peca.

4 Métodos para Invocação Remota:

getPecas, addPeca, findPecaByCodigo, getQuantidadeTotal.

Passagem por Referência: O RMI utiliza passagem por referência para objetos remotos.

Passagem por Valor: O RMI serializa objetos locais (como Peca) para enviá-los ao servidor, utilizando JSON como representação externa.

EXPLICANDO O MÉTODO DE INVOCAÇÃO REMOTA

1. Interface Remota (PecaService):

A interface PecaService define os métodos que podem ser invocados remotamente pelo cliente:

getPecas(): Retorna a lista de todas as peças cadastradas.

addPeca(Peca peca): Adiciona uma nova peça à lista.

findPecaByCodigo(String codigo): Busca uma peça pelo código.

getQuantidadeTotal(): Retorna a quantidade total de peças cadastradas.

Esses métodos são implementados no servidor e podem ser chamados remotamente pelo cliente.

2. Implementação do Serviço Remoto (PecaServiceImpl)
A classe PecaServiceImpl implementa a interface PecaService e gerencia a lista de peças:

A lista pecas armazena todas as peças cadastradas.

Os métodos implementam a lógica para adicionar, buscar e listar peças.

3. Classes POJOs (Peca, Amortecedor, Motor, Pneu)
As classes POJOs representam as peças de automóveis:

Peca: Classe base que representa uma peça genérica.

Amortecedor, Motor, Pneu: Subclasses que representam tipos específicos de peças.

4. Protocolo de Requisição-Resposta (Request, Response)
As classes Request e Response encapsulam as mensagens de requisição e resposta:

Request: Contém o nome do objeto remoto (objectReference), o ID do método (methodId) e os argumentos (arguments).

Response: Contém a resposta do servidor (reply).

5. Cliente RMI (RMIClient)
O RMIClient é responsável por enviar requisições ao servidor e receber respostas:

O método doOperation envia uma requisição ao servidor e retorna a resposta.

O cliente usa o RMI para invocar métodos remotos no servidor.

6. Servidor RMI (PecaServer)
O servidor registra o serviço remoto e fica aguardando requisições:

O servidor cria um registro RMI na porta 1099 e associa o serviço PecaService a ele.

7. Cliente Interativo (PecaClient)
O cliente interativo permite ao usuário adicionar, listar e consultar peças:

O cliente interage com o usuário para adicionar peças e consultar informações.

Ele usa o RMIClient para enviar requisições ao servidor.

FLUXO DE FUNCIONAMENTO

Servidor:

Inicia o registro RMI na porta 1099.

Registra o serviço PecaService.

Aguarda requisições dos clientes.

Cliente:

Conecta-se ao servidor.

Envia requisições para adicionar, listar ou consultar peças.

Recebe respostas do servidor e exibe os resultados.

Protocolo de Requisição-Resposta:

O cliente envia uma requisição serializada em JSON.

O servidor processa a requisição e retorna uma resposta serializada em JSON.