

# Exercício Programa 1

## Java™ Remote Method Invocation (RMI)

Prof. Dr. Daniel Cordeiro  
Escola de Artes, Ciências e Humanidades  
Universidade de São Paulo

Entrega: 22 de abril de 2017

### Descrição Geral

Vamos construir uma “versão enxuta” de um sistema de informações sobre “peças” ou “componentes” (*parts*) usando Remote Method Invocation (RMI) de Java. O sistema será distribuído por múltiplos servidores, cada qual implementando um repositório de informações sobre peças. Cada peça será representada por um objeto cuja interface é *Part*. Cada servidor implementará um objeto *PartRepository*, que é essencialmente uma coleção de *Parts*.

As interfaces *Part* e *PartRepository* devem ser definidas por vocês e são parte da implementação do EP.

**Part** Cada objeto *Part* encapsula as seguintes informações:

- o código da peça, um identificador automaticamente gerado pelo sistema na ocasião da inserção das informações sobre a peça;
- o nome da peça;
- a descrição da peça;
- a lista de subcomponentes da peça.

Uma peça pode ser uma agregação de subcomponentes ou pode ser uma peça primitiva (não composta por subpeças). Sua lista de subcomponentes contém pares (*subPart*, *quant*), onde *subPart* referencia um subcomponente da

peça, e `quant` indica quantas unidades do subcomponente aparecem na peça. Uma peça primitiva tem sua lista de componentes vazia.

Os subcomponentes de um objeto `Part` agregado são também objetos `Part`. Esses objetos não são necessariamente implementados pelo mesmo servidor que implementa a peça agregada. Eles podem estar distribuídos por múltiplos servidores.

**PartRepository** Os objetos `PartRepository` devem implementar repositórios de peças, isso é, servidores para o acesso à conjuntos de peças. Em particular, você deve ser capaz de inserir uma nova `Part` ao repositório, recuperar uma `Part` pelo seu código e obter uma lista de todas as `Parts` que estão armazenadas em um dado repositório.

Neste EP, apenas os servidores implementados por `PartRepository` devem ser registrados e recuperados do serviço de nomes do Java RMI.

## Instruções

Cada equipe de projeto escreverá um “programa servidor” e um “programa cliente”.

### O Servidor

O programa servidor implementará as interfaces `PartRepository` e `Part`. Escreva-o tendo em mente que poderão ocorrer várias execuções simultâneas do programa servidor: cada “processo servidor” (uma execução do programa servidor) implementará um objeto `PartRepository`, mais a correspondente coleção de objetos `Part`. Isto significa que o programa servidor deve receber como argumentos, na linha de comando, certos parametros que devem variar de um processo servidor para outro (o nome do servidor, por exemplo).

### O Cliente

O programa cliente será usado para exercitar o sistema. Ele deve permitir que o usuário:

- estabeleça uma conexão com um (processo) servidor;
- interaja com o repositório implementado pelo servidor:
  - examinando o nome do repositório e o numero de peças nele contidas,

- listando as peças no repositório,
- buscando uma peça (por código de peça) no repositório,
- adicionando ao repositório novas peças (primitivas ou agregadas);
- tendo uma referência a uma peça, referência essa previamente obtida como resultado de uma busca num repositório, interaja com a peça:
  - examinando o nome e a descrição da peça,
  - obtendo o (nome do) repositório que a contém,
  - verificando se a peça é primitiva ou agregada,
  - obtendo o número de subcomponentes diretos e primitivos da peça,
  - listando suas subpeças.

Fique à vontade para definir como seu programa cliente vai fazer a interface com os usuários. O único requisito é que a interface com o usuário permita que o sistema seja exercitado da forma descrita acima. Em particular, o programa cliente deve possibilitar que um usuário crie (de modo razoavelmente conveniente) peças agregadas cujas subpeças estejam distribuídas por vários repositórios. Provavelmente o mais fácil é escrever um cliente com uma interface tipo linha de comando. Uma possibilidade é um cliente “linha de comando” que mantenha três variáveis:

- o “repositório corrente”, uma referência ao repositório com o qual toda interação ocorre;
- a “peça corrente”, uma referência à peça com a qual toda interação ocorre;
- a “lista de subpeças corrente”, usada exclusivamente quando uma nova peça é adicionada ao repositório corrente.

Tal cliente apresentaria um *prompt* e ficaria esperando comandos do usuário. Ele aceitaria comandos como:

**bind** Faz o cliente se conectar a outro servidor e muda o repositório corrente. Este comando recebe o nome de um repositório e obtém do serviço de nomes uma referência para esse repositório, que passa a ser o repositório corrente.

**listp** Lista as peças do repositório corrente.

**getp** Busca uma peça por código. A busca é efetuada no repositório corrente. Se encontrada, a peça passa a ser a nova peça corrente.

**showp** Mostra atributos da peça corrente.

**clearlist** Esvazia a lista de subpeças corrente.

**addsubpart** Adiciona à lista de subpeças corrente *n* unidades da peça corrente.

**addp** Adiciona uma peça ao repositório corrente. A lista de subpeças corrente é usada como lista de subcomponentes diretos da nova peça. (É só para isto que existe a lista de subpeças corrente.)

**quit** Encerra a execução do cliente.

A lista acima tem a finalidade de ilustrar como um cliente “linha de comando” poderia funcionar. Tome-a como uma sugestão (incompleta, por sinal), que pode ser seguida ou não. Se você tiver gás para escrever um cliente com uma interface com o usuário mais elaborada e amigável (GUI), vá em frente!

## Instruções

As instruções abaixo devem ser seguidas à risca:

1. O EP deve ser feito em *equipes de 2 ou 3 pessoas*.
2. Dúvidas em relação ao EP devem ser discutidas no fórum do e-Disciplinas: <https://edisciplinas.usp.br/mod/forum/view.php?id=1454135>. Todos são **fortemente encorajados** a participar das discussões e ajudar seus colegas.
3. Entregue junto com o EP um relatório detalhado descrevendo a solução implementada e alguns exemplos de uso da interface do EP. Por exemplo, se você implementar um *prompt* de comando, cada exemplo consistiria de um sessão com os comandos necessários e suas respectivas saídas. Os cenários serão testados durante, portanto assegure-se que eles estão completos e autoexplicativos.
4. Cuidado ao seguir tutoriais desatualizados na Internet. O arcabouço que implementa o Java RMI sofreu várias modificações ao longo dos anos. Sua capacidade de encontrar informações confiáveis e atualizadas será levada em consideração na atribuição da nota.

Esse EP é uma adaptação para Java RMI do EP<sup>1</sup> proposto pelo prof. Francisco Reverbel (IME/USP) para seu curso com CORBA.

---

<sup>1</sup>[http://www.ime.usp.br/~reverbel/SOD-03/proj\\_corba.html](http://www.ime.usp.br/~reverbel/SOD-03/proj_corba.html)