Exercício-Programa
Remote Method Invocation (RMI) e Remote Procedure Call (RPC)

Norton Trevisan Roman
Escola de Artes, Ciências e Humanidades
Universidade de São Paulo

Entrega: **12 de Junho de 2022**

Descrição Geral

Vamos construir uma "versão enxuta" de um sistema de informações sobre peças ou componentes (*parts*) usando Remote Method Invocation (RMI) de Java ou Remote Procedure Call (RPC) da linguagem de sua escolha (desde que orientada a objetos). O sistema será distribuído por múltiplos servidores, cada qual implementando um repositório de informações sobre peças, conforme ilustra a Figura 1.

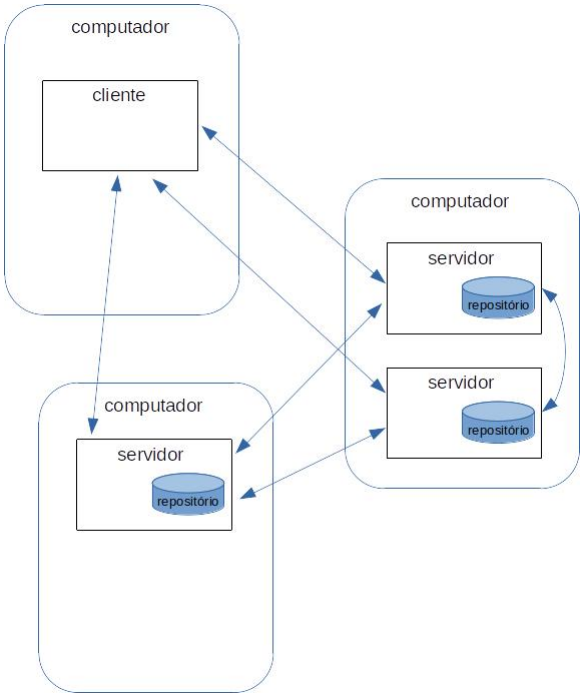

Figura 1: Esquematização do sistema distribuído.

Note que cada servidor tem um repositório (que pode ser minimalista, como uma lista ligada interna de peças nele armazenadas), fornecendo acesso a essas.

Cada peça será representada por um objeto cuja interface é `Part`. Cada servidor implementará um objeto `PartRepository`, que é essencialmente uma coleção de `Part`s. 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;
- O lista de subcomponentes da peça.

Uma peça pode ser uma agregação de subcomponentes, ou pode ser uma peça primitiva (não composta por sub-peças). 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 a 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 `Part`s 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

Para este trabalho, vocês devem se organizar em grupos de 4 (quatro) ou 5 (cinco) pessoas. 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 parâmetros 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 número 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 sub-peças.

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 sub-peças 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 sub-peças corrente.

addsubpart Adiciona à lista de sub-peças corrente n unidades da peça corrente.

addp Adiciona uma peça ao repositório corrente. A lista de sub-peças corrente é usada como lista de subcomponentes diretos da nova peça. (É só para isto que existe a lista de sub-peças 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!

Funcionamento

`PartRepository` é um objeto distribuído, que será exposto por um processo servidor e que implementa um repositório de informações sobre peças. Os clientes se conectam a servidores que possuem uma instância de `PartRepository`. O servidor, por sua vez, implementa 2 tipos de objetos o `PartRepository` (que é uma interface de acesso) e `Parts` (vários objetos remotos armazenados no servidor).

O cliente deve poder, por exemplo, se conectar ao servidor A, pegar uma peça, se conectar ao servidor B, e inserir a referência a essa peça lá. Com isso, é possível construir peças agregadas que são formadas por peças originadas em servidores diferentes. Executar os métodos nessas peças agregadas irá disparar chamadas a métodos das instâncias desses objetos que estão espalhados por todos esses servidores.

Material para Entrega

Sua entrega deverá constar de:

- Código fonte, organizado em diretórios e subdiretórios, conforme sua implementação, e **documentado**.
- Relatório **pormenorizado** sobre o sistema, descrevendo a solução implementada, além de 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 dados e suas respectivas saídas. Também devem constar do relatório a descrição das estruturas de dados utilizadas, incluindo seu uso dentro do sistema. **Não esqueçam de colocar os integrantes do grupo na capa do relatório**.

Observações

Dúvidas em relação ao EP devem ser discutidas no fórum da disciplina no edisciplinas: `https://edisciplinas.usp.br/`. Todos são **fortemente encorajados** a participar das discussões e ajudar seus colegas.

A entrega será feita unica e exclusivamente via edisciplinas, até a data final marcada. Um (e apenas um) dos integrantes do grupo deve fazer a postagem de um arquivo zip, tendo como nome o número USP desse integrante:

`número_usp.zip`

Dentro do zip devem constar seu código fonte, além do relatório final de entrega.

A responsabilidade de postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu download, por exemplo). Problemas referentes ao uso do sistema devem ser resolvidos com antecedência.

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

[1] `http://www.ime.usp.br/~reverbel/SOD-03/proj_corba.html`