

Escola de Artes, Ciências e Humanidades – Universidade de São Paulo

Exercício-Programa: Remote Method Invocation (RMI)

Prof. Norton Trevisan Roman

Matheus Percoraro de Carvalho Santos

Nº USP: 11917271

Ryan Brenno Ramos

Nº USP: 11215772

Sungwon Yoon

Nº USP: 9822261

Wendel Fernandes de Lana

Nº USP: 11796722

Código

[Url para Código Implementado](#)

Implementação

Desenvolvemos a classe `Server` que deve receber como argumento o nome que será dado ao repositório e a porta para utilização do RMI. Assim, cada processo servidor cria um registro na porta especificada e o nome passado como parâmetro é utilizado para criar o repositório. Após a criação, vincula o nome com a referência remota do repositório.

A classe `Part` possui as variáveis de código, nome, descrição, subcomponentes e o nome de qual repositório pertence. Seus métodos são para pegar tais informações e 2 outros que retornam quantos subcomponentes possui e se é ou não uma peça primitiva.

A classe `RepositoryPart` contém variáveis de nome e uma lista contendo as instâncias de `Part` que possui. Além dos métodos getters, possui um para retornar quantas peças estão contidas no repositório, para adicionar uma nova peça e para encontrar uma peça pelo seu código dentro do repositório.

Por fim, a classe `Client` possui as variáveis que salvam o repositório corrente, a peça corrente e uma lista de subpeças corrente que é utilizada quando uma nova peça é adicionada ao repositório atual. Um menu de navegação é apresentado, deixando o usuário efetuar conexões com diferentes servidores/repositórios, listar as peças do repositório atual, buscar uma peça por código, entre outras funções. Ademais, é importante lembrar que os métodos em uma peça agregada disparam chamadas a métodos das instâncias de objetos que estão espalhados por todos esses servidores/repositórios.

Funcionamento

```
### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
1
Escreva o endereco IP do repositorio (padrao: localhost):

Escreva a porta do repositorio:
1099
Escreva o nome do repositorio:
rep01
```

Estabelecendo conexão com servidor local na porta 1099 responsável pelo repositório repo1

```
### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: repo1
2
Coloque o nome da peca:
peça1
Coloque uma descricao da peca:
peça1
```

Criando uma peça nova dentro do repositório atual (repo1)

```
### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: repo1
3

Codigo Nome          Descricao
2      peça2         peça2
1      peça1         peça1
Total: 2
```

Listando todas as peças que estão no repositório atual (repo1)

```

### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: rep01
4
Digite o codigo da peca que deseja buscar:
1
Nome da peca: peça1
Descricao: peça1
Repositorio: rep01
Primitiva: Sim

```

Realizando uma busca pela peça de código 1 no repositório atual (rep01)

```

### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: rep01
5
Nome da peca: peça1
Descricao: peça1
Repositorio: rep01
Primitiva: Sim

```

Obtendo detalhes de qual é a peça corrente na aplicação

```

### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: repo1
6
Digite a quantidade da peca peça1 para adicionar:
5

```

Adicionando 5 unidades da peça corrente na lista de subpeças corrente da aplicação

```

### Menu do Gerenciador de Pecas ###
=====
| 1 - Abrir/Mudar de repositorio |
| 2 - Criar peca no repositorio  |
| 3 - Lista pecas do repositorio |
| 4 - Buscar peca por codigo     |
| 5 - Detalhes da peca atual     |
| 6 - Adicionar peca na lista    |
| 7 - Limpar a lista de subpecas |
| 0 - Finalizar aplicacao       |
=====
Repositorio atual: repo2
5
Nome da peca: peça3
Descricao: peça3
Repositorio: repo2
Primitiva: Nao
Lista de subcomponentes:

Cod.    Nome                Quant. Repositorio
1       peça1                5       repo1
Total: 1

```

Após criação de uma peça agregada (peça3) utilizando a lista de subpeças corrente no repositório repo2, realizamos uma busca e mostramos os detalhes da peça3 que apresenta a peça1 do repositório repo1 como um subcomponente.

Estruturas Utilizadas

Utilizamos HashSet para mapear as instâncias de Part dentro do repositório, dessa forma é garantido a não repetição dentro da coleção e um tempo de performance constante para as operações básicas.

Para o armazenamento dos subcomponentes de cada Part foi utilizado o HashMap armazenando a instância da subpeça Part como a chave e a quantidade de unidades utilizadas para aquela peça como o valor do HashMap. Essa estrutura também garante tempo constante de performance para as operações básicas.