Computação Distribuída

Projeto – Parte da avaliação somativa do 1º bimestre de 2020/2

O projeto consiste em implementar uma situação "real" que envolva um sistema computacional constituído de "peças" ou "componentes" (vamos chamar de *parts*) usando uma das seguintes tecnologias de programação distribuída:

- Remote Procedure Call (RPC)
- Remote Method Invocation (RMI) – JRMP ou IIOP
- Webservices

O sistema será distribuído por múltiplos servidores, cada qual implementando um repositório de informações sobre peças. Cada peça será representada por um objeto/struct cuja interface é **Part**. Cada servidor implementará um objeto/struct **PartRepository**, que é essencialmente uma coleção de **Parts**. As interfaces **Part** e **PartRepository** devem ser definidas por cada um de vocês e são parte da implementação do projeto.

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.

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.

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;

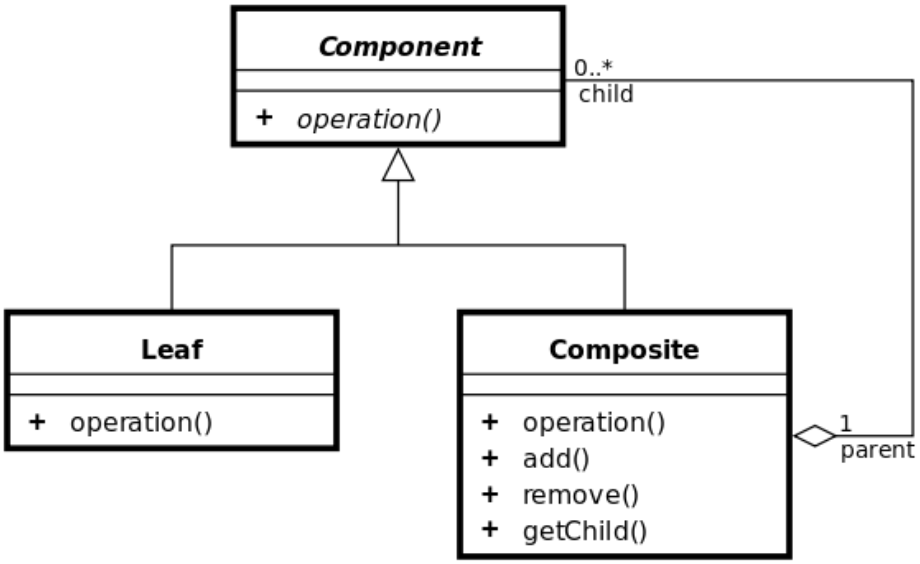
• a lista de subcomponentes da peça.

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.

Sugestões:

Uma sugestão é estudar o padrão GoF Composite para representar peças com subpeças:



Sobre o cliente, possibilidade de implementação é ter um cliente “linha de comando” que mantenha três variáveis:

• o “repositório atual”, uma referência ao repositório com o qual toda interação ocorre;

• a “peça atual”, uma referência à peça com a qual toda interaçao ocorre;

• a “lista de subpeças atual”, usada exclusivamente quando uma nova peça é adicionada ao repositório atual.

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 atual. 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 atual.

listp Lista as peças do repositório atual.

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

showp Mostra atributos da peça atual.

clearlist Esvazia a lista de subpeças atual.

addsubpart Adiciona à lista de subpeças atual n unidades da peça atual.

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

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!

Lembrete: o projeto pode ser feito em duplas, desde que uma distribuição física seja implementada.

Apresentação: durante a aula de 8/10/2020