MC 302EF - Atividade de Laboratório no. 4

Objetivos

- Uso do padrão Visitor
- Uso de '*getters*' e '*setters*'

Descrição do Problema [criado em 30/05/16]

Em continuação ao trabalho das atividades anteriores, esta atividade gerar uma representação em XML para uma lista de objetos da classe Item, usada na atividade anterior. Para gerar a representação em XML dessa lista, deverá ser utilizado o padrão Visitor descrito em sala. Um exemplo de Visitor foi disponibilizado como um dos testes para a atividade 3.

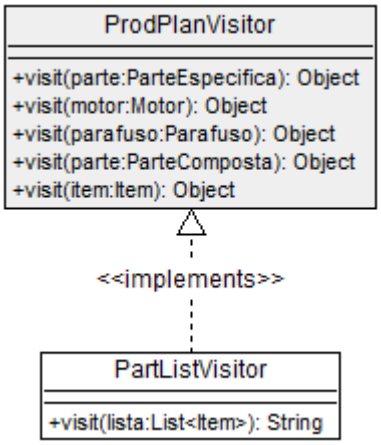
As classes

As classes usadas nesta atividade serão as mesmas usadas na atividade 3, com as seguintes modificações:

- Todos os atributos de todas as classes só devem ser acessíveis à própria classe, devendo portanto ser declarados como '*private*'.
- O acesso aos atributos das classes por outras classes deverá ser feito sempre através de métodos do tipo '*getter*' ou '*setter*', seguindo o padrão de nomenclatura adotado pela 'comunidade Java' (conforme mencionado em sala, o Eclipse oferece a opção de gerar automaticamente esses métodos). A única exceção é quanto ao método ParteComposta.getItens() que deverá retornar o mesmo valor que o método ParteComposta.listaDeItens(), implementado na atividade 3 (e que retorna uma lista de itens ordenada pela quantidade).

A classe PartListVisitor

Além das classes descritas acima, nesta atividade deverá ser implementada a classe PartListVisitor.java. Essa classe deverá implementar a interface ProdPlanVisitor, definida na atividade 3 e usada em exemplos e seu objetivo é, a partir de uma lista de itens, gerar uma listagem contendo a quantidade total de cada um dos objetos Parte usados nessa lista. A listagem gerada deve conter apenas os objetos das classes Motor, Parafuso e ParteEspecifica, relacionados direta ou indiretamente na 'lista de entrada'. No caso de objetos ParteComposta, os objetos das demais classes usadas (Motor, Parafuso e ParteEspecifica) devem ser totalizados na listagem gerada. A figura abaixo, ilustra a classe PartListVisitor.



Na figura, o método visit(lista: List<Item>), não faz parte da interface ProdPlanVisitor e é o responsável por disparar a visita aos itens que compõem a lista passada como parâmetro, retornando a listagem com os totais por parte, na forma de um string.
.
O texto gerado pela visita à lista consiste basicamente numa sequência de pares com o código e quantidade de cada parte usada na lista.

A figura abaixo mostra o texto gerado a partir da lista de itens usada na atividade 3.

```
cod:231 quant:22
cod:511 quant:44
cod:233 quant:12
cod:232 quant:35
cod:321 quant:65
cod:322 quant:24
cod:401 quant:26
cod:512 quant:33
cod:112 quant:38
cod:402 quant:12
```

Observações importantes:

1. Cada código relacionado direta ou indiretamente na lista de entrada deve aparecer uma única vez na listagem de saída. Todos os códigos relacionados direta ou indiretamente pela lista de entrada, que não se referem a objetos ParteComposta, devem aparecer na listagem de saída.
2. As quantidades devem representar as quantidades totais de cada objeto Parte usada na lista, exceto aqueles da classe ParteComposta.
3. Os objetos da classe ParteComposta, relacionados direta ou indiretamente pela lista, devem ser visitados de forma que suas partes (Motor, Parafuso e ParteEspecifica) sejam totalizados na listagem de saída.

Sugestão: Criar um mapa do tipo Map<Integer, Integer> associando cada código de Parte à quantidade usada na lista. Nessa linha, durante a visita à lista esse mapa é preenchido e ao final, o a listagem é gerada a partir dos pares (código, quantidade) contidos no mapa.

Exemplo de uso

O arquivo ‘TesteLab4_2.java’, disponibilizado junto com este enunciado contém uma classe de testes para as classes criadas na atividade.

Saída esperada

O arquivo ‘SaidaLab4_2.txt’, disponibilizado junto com este enunciado mostra a saída esperada ao se executar a classe de testes disponibilizada como exemplo de uso.

Data de entrega: 09/04/16