MC 302EF - Atividade de Laboratório no. 3

Objetivos

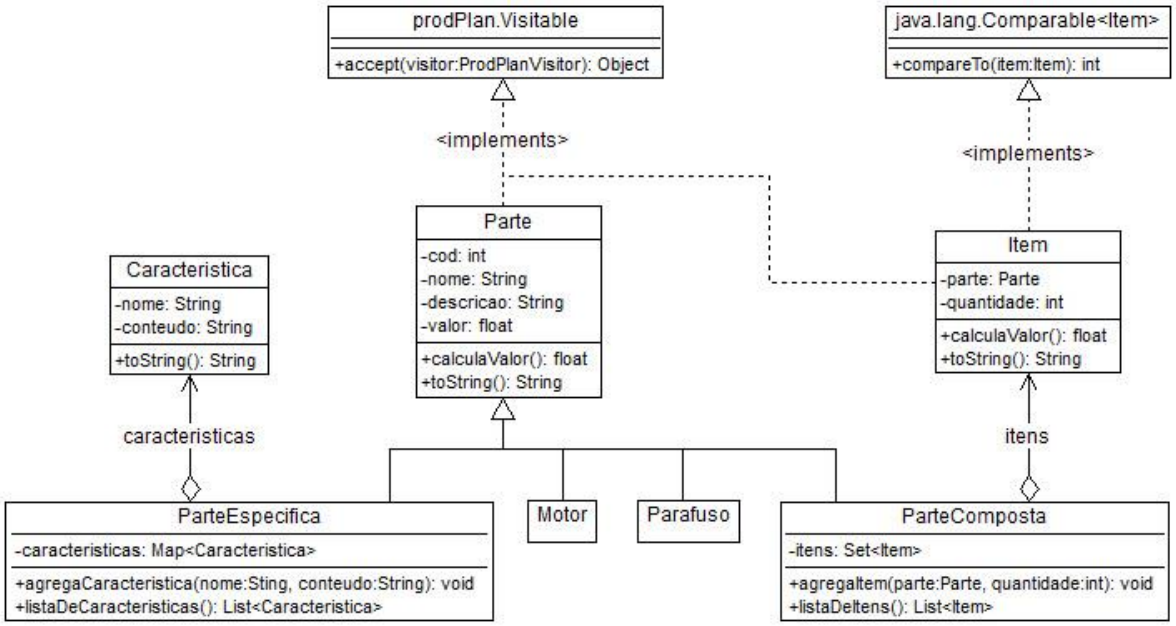
- Uso de coleções, interfaces e exceções em Java

Descrição do Problema [criado em 16/03/16]

Em continuação ao trabalho da atividade no2, esta atividade consiste em implementar as classes para descrever 'partes compostas' e 'partes específicas'. Essas classes farão uso das coleções disponíveis nas bibliotecas de apoio à linguagem Java.

As classes

A figura abaixo ilustra as classes que serão usadas nesta atividade. As classes Item, Parte, Motor e Parafuso foram criadas na atividade anterior. Nesta atividade serão criadas as classes ParteComposta, ParteEspecifica e Caracteristica, descritas a seguir.



Interfaces utilizadas

Nesta atividade, serão utilizadas duas interfaces:

- Comparable: parte integrante da linguagem Java, define o método compareTo(), e será usando nesta atividade para comparar dois itens pelo atributo quantidade. A convenção para o valor de retorno deste método é a seguinte: -1, 0 ou 1 se a comparação indicar 'menor', 'igual' ou 'maior' respectivamente.
- Visitable: esta interface, cujo código é disponibilizado como parte desta atividade, define o método accept(), cujo código é mostrado a seguir. Esse método deverá ser utilizado nas atividades subsequentes, e prevê o uso de um objeto da classe abstrata ProdPlanVisitor, cujo código também é disponibilizado como parte desta atividade.

O método accept() mostrado abaixo, deverá ser incluído nas classes concretas que implementam a interface Visitable:

```
@Override
public Object accept(ProdPlanVisitor visitor) {
	return visitor.visit(this);
}
```

Item, Motor e Parafuso

Estas classes, que são as mesmas implementadas na atividade anterior, deverão ser alteradas para implementar as interfaces mostradas no diagrama de classes acima.

ParteComposta

Esta classe derivada de Parte representa uma parte formada por outras partes (simples ou compostas). Além dos atributos herdados de Parte, os objetos desta classe tem o seguinte atributo:

- Conjunto de objetos Item que compõem este objeto ParteComposta.

Esta classe deve também implementar os seguinte métodos públicos:

```
public void agregaItem(Parte parte, int quantidade) throws Exception
```

- este método deve agregar um item ao objeto ParteComposta e tem como parâmetros a referência a um objeto Parte e a quantidade que essa parte será usada na ParteComposta. Este método deve gerar uma exceção se o objeto Parte referenciado pelo objeto Item já estiver presente ou for nula e também no caso de a quantidade ser nula.

```
List<Item> listaDeItens()
```

- este método deve retornar uma lista contendo os objetos Item que constituem a ParteComposta ordenada de acordo com a quantidade do item. Para ordenação dessa lista, pode ser usado o método Collections.sort(), definido em java.util.Collections.

Sobre os demais métodos:

```
public float calculaValor()
```

- O valor retornado pelo método, herdado de Parte, deverá corresponder ao atributo 'valor' da parte composta somado aos valores de cada uma das Partes que constituem a ParteComposta. O atributo 'valor' da parte composta pode ser entendido como o valor agregado em consequência da montagem da mesma.

```
public String toString()
```

- este método deve retornar a representação do objeto ParteComposta como um string.

O construtor desta classe tem a mesma assinatura que o construtor de Parte.

ParteEspecifica

Classe concreta derivada de Parte, usada para representar outras partes da gama de produtos da empresa. Os objetos desta classe, além dos atributos herdados de Parte, mantém um conjunto de 'características específicas' na forma de pares (nome,conteúdo).
Além dos métodos previstos ou herdados de Parte, esta classe deve oferecer os seguinte métodos:

```
public void agregaCaracteristica(String nome, String conteudo) throws Exception
```

- este método associa uma 'caracteristica específica' ao objeto ParteEspecífica, na forma (valor,conteúdo). Caso um dos parâmetros seja igual a null ou caso o objeto já tenha uma característica do mesmo nome, este método deve gerar uma exceção.

```
public String caracteristica(String nome)
```

- este método retorna o 'conteúdo' associado a uma característica, definida pelo seu nome. Caso o objeto não tenha a característica desejada, o método deve retornar null.

O construtor desta classe tem a mesma assinatura que o construtor de Parte.

Exemplo de uso

O arquivo 'TesteLab3_1.java', disponibilizado junto com este enunciado contém uma classe de testes para as classes criadas na atividade.

Saída esperada

O arquivo 'SaidaLab3_1.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: 26/03/16