MC 302EF - Atividade de Laboratório no. 2

Objetivos

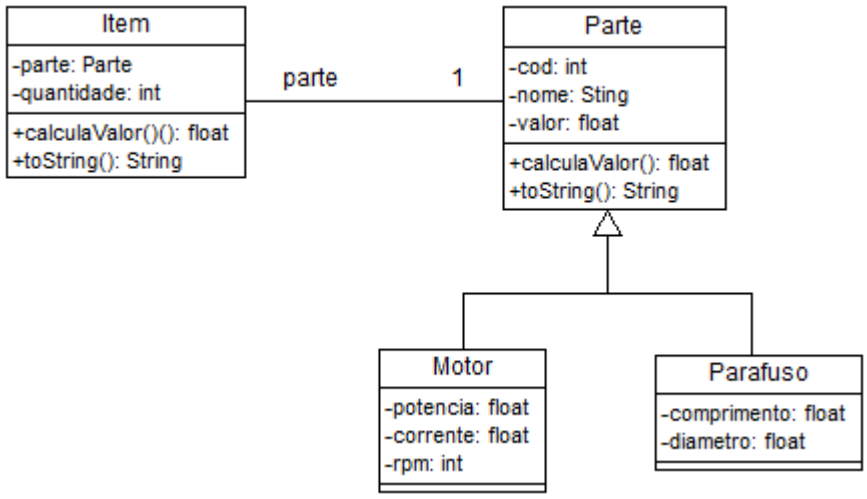
- Uso de herança, polimorfismo e classes abstratas em Java

Descrição do Problema [criado em 11/03/16]

Implementar uma hierarquia de classes que descrevem as *partes* a serem utilizadas numa linha de produção. A hierarquia de classes implementada nesta atividade será utilizada em atividades futuras. As classes devem fazer parte de um pacote chamado 'prodPlan'.

As classes

A figura abaixo ilustra as classes a serem implementadas nesta atividade. Cada uma delas é descrita a seguir.



Parte

Esta é uma classe abstrata que define os elementos comuns aos objetos Parte utilizados na produção. Ela define os seguintes atributos (acessíveis apenas às classes do mesmo pacote):

- código do item, como inteiro
- nome do item, como string
- descrição do item como string
- valor do item, como float

Esta classe prevê a criação dos seguintes métodos públicos:

- construtor (usado para definir o valor dos atributos descritos acima – os parâmetros devem manter a ordem apresentada).
- calculaValor(): método abstrato que retorna o valor correspondente à parte descrita pelo objeto.

- toString(): método abstrato que retorna uma representação da parte como um string (o formato desse string é mostrado no exemplo de uso, abaixo). É importante que este método gere a representação como String exatamente da forma como apresentada .
- métodos auxiliares podem ser criados nesta classe desde que acessíveis apenas às classes do mesmo pacote.

Motor

Classe concreta derivada de Parte, que acrescenta os seguintes atributos:

- potencia como float.
- corrente como float.
- rpm (‘rotações por minuto’) como inteiro.

O método toString() desta classe deve seguir o padrão mostrado no exemplo de uso disponibilizado junto com este enunciado. A lista de parâmetros do construtor deve ser mesma prevista para Parte, e seguida dos atributos acima, nessa ordem.

Parafuso

Classe derivada de Parte, que acrescenta os seguintes atributos:

- comprimento como float
- diâmetro como float

O método toString() para a classe Parafuso deve seguir o padrão mostrado no exemplo de uso. A lista de parâmetros do construtor deve ser a mesma prevista para Parte, seguida dos valores para os atributos acima, nessa ordem.

Item

A classe Item associa uma Parte a uma quantidade. Ela será usada em várias situações como por exemplo para definir um lote de produção ou para representar o estoque disponível. Seus atributos são (acessíveis apenas às classes do mesmo pacote):

- referência ao objeto Parte
- quantidade como inteiro.

Exemplo de uso

O arquivo ‘TesteLab2_1.java’, disponibilizado junto com este enunciado contém uma classe de testes para as classes criadas na atividade. É importante notar que a aplicação de testes se baseia no conceito de pacotes em Java, que está descrito na apresentação ‘Classes & Pacotes’ na página do curso.

Saída esperada

O arquivo ‘SaidaLab2_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:

A entrega deverá ser feita até 19/03/16 e a submissão será feita através do Run.Codes. Para validação no Run.Codes será utilizada uma classe de testes específica.

Desafio opcional

1. Escreva um método que gera num arquivo a representação como String de um vetor de objetos Parte e de um vetor de objetos Item.
2. Escreva um método que leia o arquivo gerado no item anterior e crie dois vetores equivalentes aos originais.
3. Nas classes Item, Parte, Motor e Parafuso crie o o método `equals()` que compara o objeto ao qual o mesmo é aplicado ao objeto passado como parâmetro, retornando um valor booleano.
4. Use os métodos criados no item anterior para verificar se os vetores reconstruídos no item 2, acima, são iguais aos vetores originais.

Assim como no lab anterior, o item opcional não deve ser submetido e sim enviado por email diretamente ao professor.