**Descrição do Problema**

Implementar uma hierarquia de classes que descrevem as *partes* a serem utilizadas numa linha de produção. As classes devem fazer parte de um pacote chamado ‘prodPlan’ cujo projeto deve ser nomeado por ‘Polimorfismo\_seunome’. Cada aluno coloca seu primeiro nome no projeto.

**As classes**

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

Parte

-cod: int

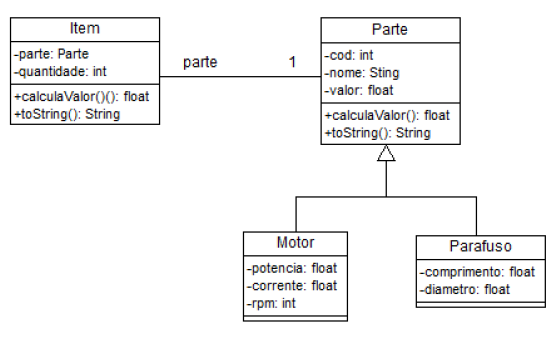
-nome: String

-descricao: String

-valor: float

+calculaValor( ): float

+toString( ): String



:

**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 (acesso padrão).

**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**

A classe Teste faz os testes para as classes criadas na atividade é descrita abaixo:

/\*\*

\* MÉTODO estático do tipo void que escreve na saída padrão o conteúdo de um vetor de

\* objetos Parte devidamente convertidos para String.

\* @param titulo texto a ser mostrado antes da listagem

\* @param partes vetor de objetos Parte a ser listado

\*/

static void listaPartes( String titulo, Parte[ ] partes){

}

/\*\*

\* MÉTODO estático que cria um vetor de objetos Item a ser usado nos testes

\* @param partes vetor de objetos Parte a partir do qual serao criados os objetos Item que

\* formarão o vetor

\* @return vetor de objetos Item

\*/

static Item[ ] criaItens( Parte[ ] partes ){

}

/\*\*

\* MÉTODO estático que escreve na saída padrão o conteúdo de um vetor de objetos Item,

\* devidamente convertidos para String.

\* @param titulo texto apresentado antes da listagem.

\* @param itens vetor de objetos Item a ser listado.

\*/

static void listaItens( String titulo, Item[ ] itens ){

}

/\*\*

\* MÉTODO principal que dispara os testes.

\* @param args o de sempre.

\*/

public static void main( String[ ] args ){

Parte[ ] partes = criaPartes( );

Item[ ] itens = criaItens( partes );

listaPartes( "\*\*\* Partes utilizadas na producao \*\*\*\*", partes );

listaItens( "\*\*\* Itens solicitados \*\*\*", itens );

}

}

**Saída esperada**

A saída esperada é mostrada abaixo:

\*\*\* Partes utilizadas na produção \*\*\*\*

codigo:112 nome:motor m112 descricao:motor de avanco do cabecote valor:100.0 potencia:1.2 corrente:1.1 rpm:1250

codigo:114 nome:motor m114 descricao:motor auxiliar valor:60.0 potencia:0.6 corrente:0.8 rpm:1250

codigo:111 nome:motor m111 descricao:motor de ventilador valor:70.0 potencia:1.0 corrente:1.0 rpm:3000

codigo:110 nome:motor m110 descricao:motor principal valor:120.0 potencia:1.8 corrente:1.5 rpm:1250

codigo:231 nome:parafuso p1 descricao:parafuso de fixacao do cabecote valor:2.5 comprimento:100.0 diametro:8.0

codigo:232 nome:parafuso p2 descricao:parafuso de fixacao do motor valor:2.5 comprimento:80.0 diametro:6.0

codigo:233 nome:parafuso p3 descricao:parafuso de fixacao do ventilador valor:2.0 comprimento:60.0 diametro:6.0

codigo:234 nome:parafuso p4 descricao:parafuso de uso geral valor:3.0 comprimento:120.0 diametro:12.0

\*\*\* Itens solicitados \*\*\*

cod:112 nome:motor m112 quantidade:10 valor unitario:100.0 valor:1000.0

cod:232 nome:parafuso p2 quantidade:50 valor unitario:2.5 valor:125.0

cod:234 nome:parafuso p4 quantidade:30 valor unitario:3.0 valor:90.0

cod:111 nome:motor m111 quantidade:5 valor unitario:70.0 valor:350.0

Valor total:1565.0