Universidade Federal do Ceará - Campus de Crateús
Ciência da Computação e Sistemas de Informação
6º Laboratório de Programação Orientada a Objetos
31/10/2017
Prof. Lívio Freire

1. Crie uma classe para representar uma conta corrente, com métodos para depositar uma quantia, sacar uma quantia e obter o saldo. Para cada saque será debitada também uma taxa de operação equivalente à $0,5\%$ do valor sacado. Crie, em seguida, uma subclasse desta classe anterior para representar uma conta corrente de um cliente especial. Clientes especiais pagam taxas de operação de apenas $0,1\%$ do valor sacado. Faça testes com as duas classes e verifique seus resultados.

2. Implementar uma hierarquia de classes que descrevem as **partes** a serem utilizadas numa linha de produção, conforme modelo a seguir:

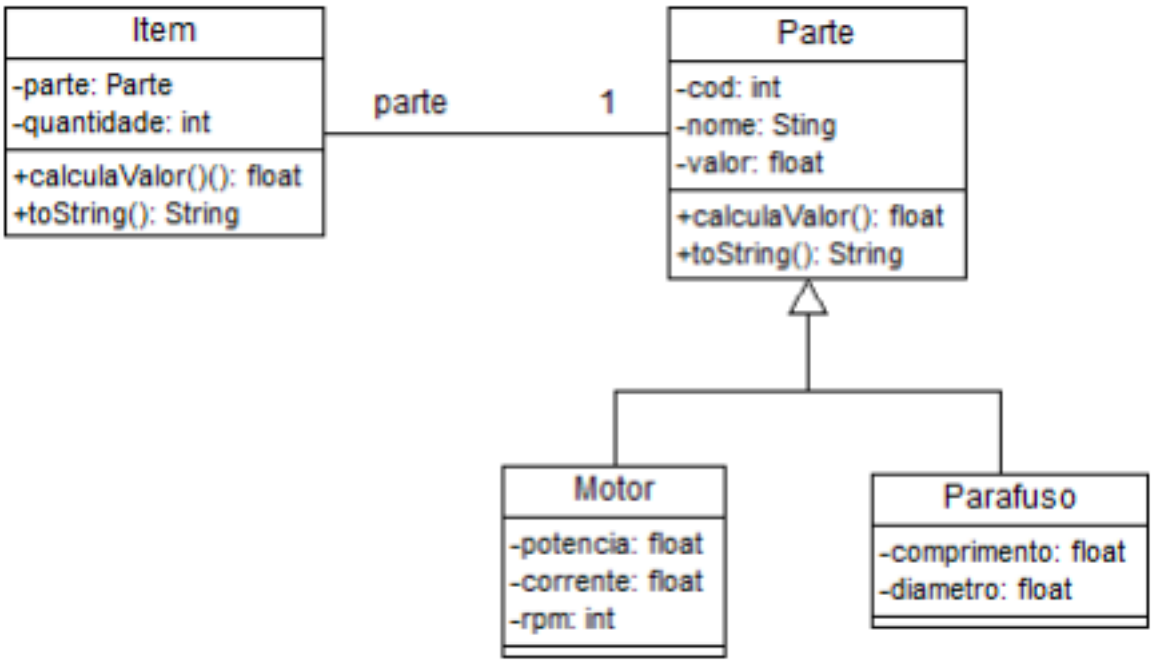

Figura 1: Modelo

Sobrescreva o método `toString()` e crie construtores para as classes. O método abstrato `calculaValor()`, na classe `Parte`, retorna o valor correspondente à parte descrita. O método `calcularValor`, em `Item`, calcula o valor do item em função do valor de suas partes. O valor do `Parafuso` ou `Motor` deve ser definido em função de seus atributos, da seguinte forma:

- Preço do motor: $valorBase \times (0.7potencia + 0.003rpm)$
- Preço do parafuso: $valorBase \times (Math.PI \times (diametro/2)^2 \times comprimento)$

3. Implemente uma classe chamada `Aleatorio` que representa um número aleatório (como aquela utilizada nos slides do curso). Esta classe deve possuir:

- Um objeto da classe `java.util.Random` compartilhado por todas as instâncias da classe. Ao criar este objeto, passe o valor `System.currentTimeMillis()` como parâmetro ao seu construtor;
- Uma constante `VALOR_MAXIMO_DEFAULT` (a ser utilizada somente pela própria classe) que define o valor máximo padrão caso um não seja especificado no construtor de `Aleatorio`;
- Dois construtores: um sem parâmetros e outro que recebe um valor inteiro como parâmetro, que deve ser utilizado como valor máximo para o número aleatório gerado;
- Um atributo que armazene o número aleatório gerado durante a construção do objeto e um método que retorne este valor.

Crie uma outra classe com um método `main()` que gere e imprima 10 números aleatórios utilizando a classe `Aleatorio`.

4. No exercício anterior, uma vez que um número aleatório é gerado e armazenado em uma instância da classe `Aleatorio`, ele não é mais modificado. Agora, gostaríamos de adicionar um método `renovar()` a esta classe que troca o valor numérico armazenado dentro dela, porém obedecendo o valor máximo especificado (ou não) no momento da construção daquele objeto. Modifique a classe nesse sentido.

5. Crie uma hierarquia de classes de domínio para uma loja que venda livros, CDs e DVDs. Sobrescreva o método `toString()` para que imprima:

   - Para livros: nome, preço e autor;
   - Para CDs: nome, preço e número de faixas;
   - Para DVDs: nome, preço e duração.

   Evite ao máximo repetição de código utilizando a palavra super no construtor e no método sobrescrito. Em seguida, crie uma classe Loja com o método `main()` que adicione 5 produtos diferentes (a sua escolha) a um vetor e, por fim, imprima o conteúdo do vetor.

6. Modifique o código do programa anterior, da seguinte forma:

   (a) Adicione um atributo que represente o código de barras do produto (é um valor obrigatório e, portanto, deve ser pedido no construtor);

   (b) Sobrescreva o método `equals()` retornando `true` se dois produtos possuem o mesmo código de barras;

   (c) Na classe `Loja`, implemente um simples procedimento de busca que, dado um produto e um vetor de produtos, indique em que posição do vetor se encontra o produto especificado ou imprima que o mesmo não foi encontrado;

   (d) No método `Loja.main()`, após a impressão do vetor (feita na questão 2a), escolha um dos 5 produtos e crie duas novas instâncias idênticas a ele: uma com o mesmo código de barras e outra com o código diferente. Efetue a busca deste produto no vetor utilizando as duas instâncias e verifique o resultado.

(e) Ainda modificando o código do programa anterior, faça com que `Produto` implemente a interface `Comparable`, e implemente a comparação por nome. Ao final do método `Loja.main()`, ordene o vetor utilizando o método `java.util.Arrays.sort()` e imprima-o novamente. Depois altere a implementação da comparação para ordenar por preço e verifique o resultado.

7. Crie a seguinte hierarquia de classes:

   (a) Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;

   (b) Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;

   (c) Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;

   (d) Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.

   No programa principal, crie diferentes formas geométricas. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os dados (lados ou raio); (b) os perímetros; e (c) as áreas de todas as formas. Para (b) e (c), tire vantagem do polimorfismo, enquanto que para (a) utilize instanceof e downcast.

Divirta-se!