INSTITUTO FEDERAL
São Paulo
Câmpus Caraguatatuba

EXERCÍCIOS DE FIXAÇÃO 5
CARPRC1 – 2024
Prof. Ederson / Prof. Juliana

---

1. Crie uma classe calculadora. Esta classe deve ser abstrata e implementar as operações básicas (soma, subtração, divisão e multiplicação). Utilizando o conceito de herança crie uma calculadora cientifica que implementa os seguintes cálculos: raiz quadrada e a potência. Dica utilize a classe *Math* do pacote *java.lang*.

2. Uma empresa controla seus produtos na forma de dois tipos de objetos: produtos comprados e produtos fabricados. Produtos comprados implementam a interface *IProduto*:

```
public interface IProduto {
    public String getNome();
    public float getCusto();
}
```

O método *getNome* retorna o nome do produto. Se o objeto for um produto comprado, o método getCusto retorna o valor de compra do produto (este valor representa seu custo).

Produtos fabricados são feitos a partir de uma combinação de ingredientes. Para fins de simplificação considere que sempre será usada uma unidade de cada ingrediente. Tais produtos fabricados implementam a interface *IProdutoFabricado*:

```
public interface IProdutoFabricado extends IProduto {
    int getNumeroIngredientes();
    IProduto getIngrediente(int numero);
}
```

Para fins de simplificação, o custo de um produto fabricado é apenas a soma dos custos de seus ingredientes. Portanto, se o objeto for produto fabricado, seu método *getCusto* retorna esta soma.

Objetos de produtos comprados não implementam a interface IProdutoFabricado.

O sistema define uma classe auxiliar *GerenteProdutos* com os seguintes métodos estáticos:

ingredientes	Recebe como parâmetro uma String contendo o nome do produto e retorna um vetor de Strings com os ingredientes do mesmo. Retorna null se este produto não puder ser fabricado.
valorCompra	Retorna como parâmetro uma String contendo o nome do produto e retorna seu valor de compra.

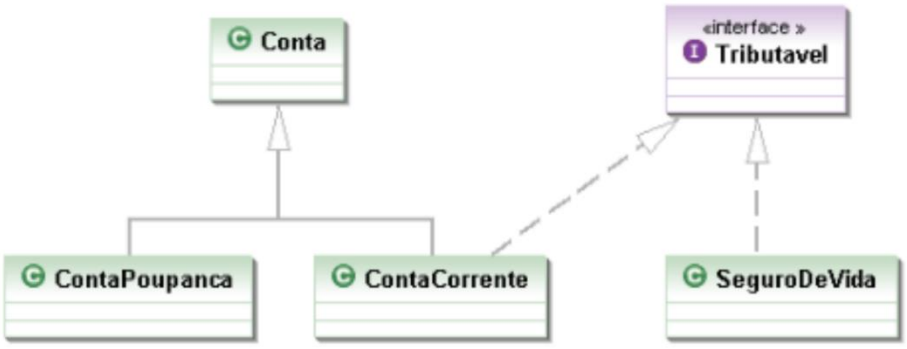
Codifique uma classe que implemente a interface *IProdutoFabricado*.

Codifique um método em Java que receba como parâmetro um objeto representando um produto comprado (implementando a interface *IProduto*). Este método deverá executar o processo de redução de custo do produto, baseado nos seguintes passos.

a) Recupera os ingredientes do produto e calcula o seu custo fabricado. Se o custo fabricado for menor que o comprado, transforma o objeto produto comprado em um objeto produto fabricado.
b) Se o produto se converter em fabricado, este método deve aplicar o mesmo processo de redução de custo a cada um dos ingredientes. O mesmo acontece com os sub-ingredientes e assim sucessivamente.

   O método deve retornar o objeto produto com a modificação aplicada (se houver).

3. Um banco precisa tributar dinheiro de alguns bens que nossos clientes possuem. Para isso, vamos criar um sistema para isso.



a) Crie uma interface Tributavel que possui o método calculaTributos(), que retorna um double.
b) Alguns bens são tributáveis e outros não, ContaPoupanca não é tributável, já para ContaCorrente você precisa pagar 1% do saldo da conta e o SeguroDeVida tem uma taxa fixa de 42 reais.
c) As classes ContaCorrente e ContaPoupanca são subcalsses de uma classe Conta. Essa classe Conta possui um saldo e os métodos sacar(double), depositar(double) e obterSaldo() que retorna o saldo da conta.
d) Crie uma classe TestaTributavel com um método main para testar a sua aplicação.

4. Escreva uma classe abstrata chamada Produto que implementa a interface `Comparable`:

- Veja a API do Java para informações sobre essa interface
- A comparação entre produtos deverá ser implementada considerando seu custo-benefício

a) Um produto deve conter: nome, preço e métodos que achar necessário.
b) Escreva as classes Shampoo, Biscoito e Leite, filhas de Produto

- Shampoo contém um campo que indica a irritabilidade do shampoo para peles normais (`int`)
- Biscoito contém um campo que indica quantidade de componentes cancerígenos em sua fórmula (`int`)
- Leite contém um campo que indica quantos dias o leite dura após ser embalado (`int`)
- Cada instância terá um valor para esses campos
- Crie uma fórmula para cada produto que combine seu preço e as características individuais de cada um para calcular o custo-benefício. Essa fórmula deve ser usada na implementação do método `compareTo` de cada classe.

c) Escreva uma classe concreta chamada Supermercado:

- Essa classe não precisa ter atributos
- Contém o método `main()`

d) No método `main`:

- Crie um array para cada tipo específico de produto – Shampoo[], Biscoito[], Leite[]
- Crie algumas instâncias e coloque dentro dos arrays
- Compare todos os produtos de um mesmo tipo entre si, indicando quem tem maior custo-benefício – `compareTo`

OBS: Sobre a interface `Comparable`. A interface `Comparable` da API do Java utiliza o conceito de Generics, do qual trataremos mais adiante. Se o tipo generic não for definido quando utilizamos a interface, o método `compareTo` dessa interface terá como parâmetro um tipo Object: `int compareTo (Object obj)`. Para utilizar o recurso generic, basta colocar entre <> o tipo de classe que se deseja comparar: `public class MyClass implements Comparable <MyClass>`. Isso faz com que o método `compareTo` tenha a assinatura abaixo `int compareTo (MyClass obj)`.