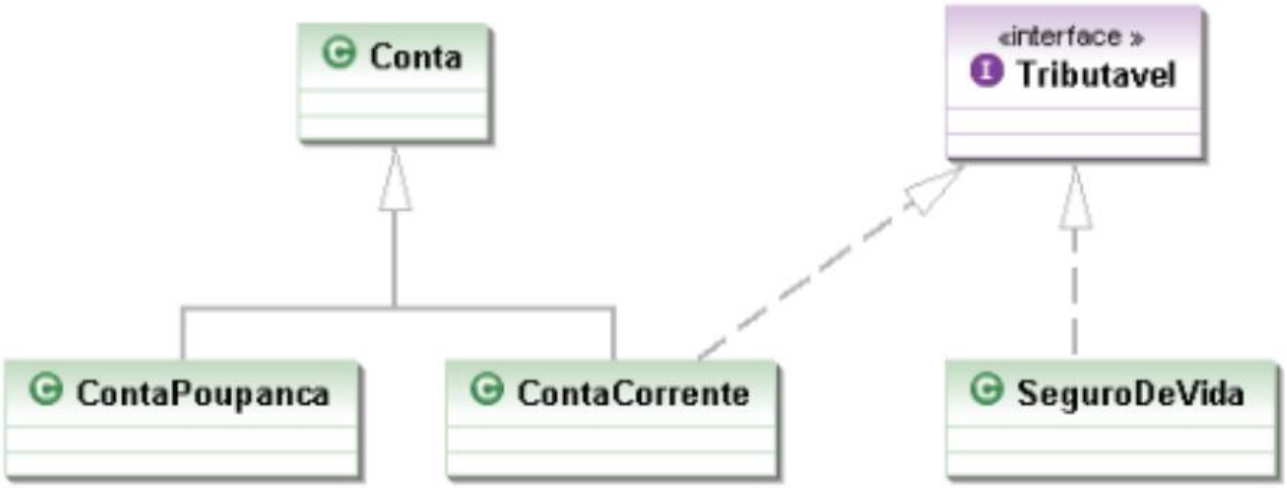
Lista de Exercícios Interfaces

1. Faça o que pede a questão.
   a) Crie um projeto interfaces e crie a interface **AreaCalculavel** com o método **calculaArea**() - sem parâmetros e que retorna um **double**.
   b) Queremos criar algumas classes que implementam **AreaCalculavel**:
      I. **Quadrado**: possui um atributo lado.
      II. **Retangulo**: possui os atributos base e altura.
      III. **Circulo**: possui o atributo raio.
   c) Crie uma classe de **Teste**. No método **main** crie um vetor de 5 posições que contém alguns objetos do tipo **AreaCalculavel**. Logo após, percorra esse vetor imprimindo a área de cada objeto.

2. Nosso 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% da conta e o **SeguroDeVida** tem uma taxa fixa de 42 reais.
   c) As classes **ContaCorrente** e **ContaPoupanca** herdam 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) Vamos criar uma classe **TestaTributavel** com um método **main** para testar o nosso exemplo.

3. Crie um **GerenciadorDeImpostoDeRenda**, que recebe todos os tributáveis de uma pessoa e soma seus valores e inclua nele um método para devolver seu total. Essa classe deve ter um atributo para calcular a soma total dos tributos e um método **adicionar**(**Tributavel**) que recebe como parâmetro um **Tributavel** e soma os tributos dele ao total. Crie um main para instanciar diversas classes que implementam **Tributavel** e passar como argumento para um **GerenciadorDeImpostoDeRenda**. Repare que você não pode passar qualquer tipo de conta para o método adiciona, apenas as que implementam **Tributavel**.

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

```
1   public interface IProduto{
2         public String getNome();
3         public float getCusto();
4   }
```

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**:

```
1   public interface IProdutoFabricado extends IProduto{
2         int getNumeroIngredientes();
3         IProduto getIngrediente (int numero);
4   }
```

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. 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).