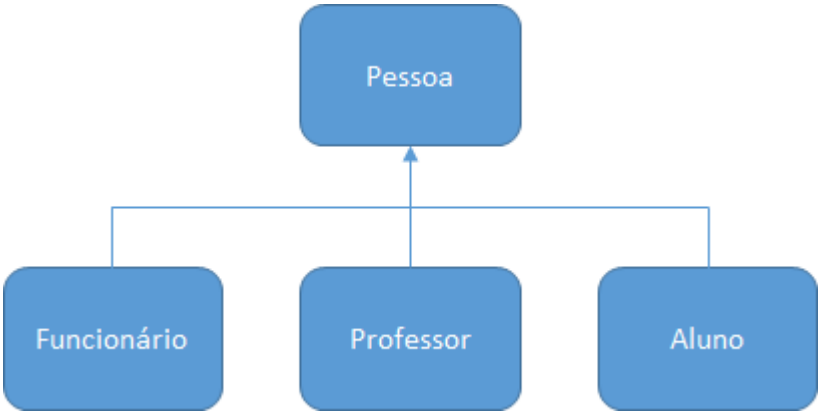
PUC Minas
Instituto de Ciências Exatas e Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ALGORITMOS E ESTRUTURAS DE DADOS I
PROF.: HENRIQUE BATISTA DA SILVA
LISTA DE EXERCÍCIOS 14

Herança x Composição

Herança é artifício usado no Programação Orientada à Objetos que permite ao programador evitar a repetição de código desnecessária, podemos criar classes de forma que herdem atributos e métodos de uma classe superior. No entanto a herança deve ser usada em um contexto muito restrito, pois uma de suas desvantagens é o alto acoplamento, ou seja, qualquer alteração realizada na superclasse afeta também todas as subclasses. Outro problema relacionado à herança é que este relacionamento entre a superclasse e as suas subclasses é estático e não pode ser alterado em tempo de execução (veremos um exemplo em que esta funcionalidade seria muito útil).

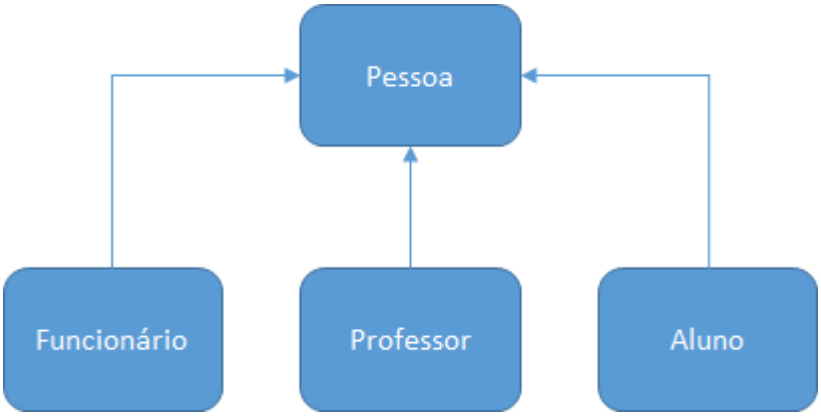
Suponha como exemplo o diagrama abaixo representando uma universidade:



A universidade contém Funcionário, Professor e Aluno, e todos eles são do tipo Pessoa, pois compartilham dos mesmo métodos e atributos de pessoa, como por exemplo, nome, endereço, telefone, data de nascimento, etc.. Este relacionamento entre as classes não pode mudar durante a execução do programa.

Na composição uma classe possui um objeto de outra classe. O relacionamento é diferente da herança, uma vez que na composição uma classe **usa** a outra classe e na herança o uma classe é um tipo da outra classe (no exemplo acima, Professor é uma Pessoa).

Na figura abaixo, usando o mesmo exemplo da universidade, podemos notar a composição sendo usada para mapear a mesma ideia do relacionamento entre as classes ao invés da herança.



As classes Funcionário, Professor e Aluno usam a classe Pessoa, que, por sua vez, continua tendo seu métodos e atributos que a caracterizam.

Neste exemplo há um fraco acoplamento, pois, alterar a classe pessoa não altera os objetos das classes Funcionário, Professor e Aluno. Estas classes não são mais do tipo pessoa (como na herança) e sim elas usam o tipo pessoa. Portanto, quando alguma funcionalidade mais geral precisa ser utilizada, como por exemplo, calcular a idade da pessoa por meio de sua data de nascimento, qualquer uma das classes apenas invocam este método delegando este cálculo da idade para o objeto Pessoa.

Outra vantagem da composição sobre a herança está relacionada aos diferentes papéis que as entidades podem assumir. Neste nosso exemplo, um professor também pode ser aluno da instituição, ao mesmo tempo. Por meio da composição, esta modificação pode ser facilmente realizada, bastando apenas instanciar um novo objeto de Aluno que receberá o objeto de Pessoa (mesmo objeto que também pertence a uma instância de Professor). Esta atribuição pode ser feita em tempo de execução, sem ter que alterar as estruturas das classes.

Exercícios

1. Crie uma classe abstrata chamada Ingresso que possui um valor em reais e um método imprimeValor().
   a. Crie uma classe abstrata IngressoVIP, que herda Ingresso e possui um valor adicional. Crie um método que retorne o valor do ingresso VIP (com o adicional incluído).
   b. Crie uma classe IngressoNormal, que herda Ingresso e possui um método que imprime: "Ingresso Normal".
   c. Crie uma classe CamaroteInferior (que possui a localização do ingresso e métodos para acessar e imprimir esta localização) e uma classe CamaroteSuperior, que é mais cara (possui valor adicional). Esta última possui um método para retornar o valor do ingresso. Ambas as classes herdam a classe IngressoVIP.

Crie uma classe de Teste com o método *main*. Neste método crie um ingresso. Peça para o usuário digitar 1 para normal e 2 para VIP. Conforme a escolha do usuário, diga se o ingresso é do tipo normal ou VIP. Se for VIP, peça para ele digitar 1 para camarote superior e 2 para camarote inferior. Conforme a escolha do usuário, diga se que o VIP é camarote superior ou inferior. Imprima o valor do ingresso.