LPOO – Lista de Exercícios

Herança e Polimorfismo

1. Crie uma classe chamada *ComissionEmployee* que contenha os seguintes membros:

   - *firstName* (String);
   - *lastName* (String);
   - *socialSecurityNumber* (String);
   - *grossSales* (double);
   - *comissionRate* (double);
   - método *earnings*, que calcula os lucros de um *ComissionEmployee*;
   - método *toString*.

   Você deve fornecer um construtor capaz de inicializar todos os atributos do objeto. Para cada atributo, utilize os modificadores de acesso que achar conveniente, lembrando de manter o encapsulamento.

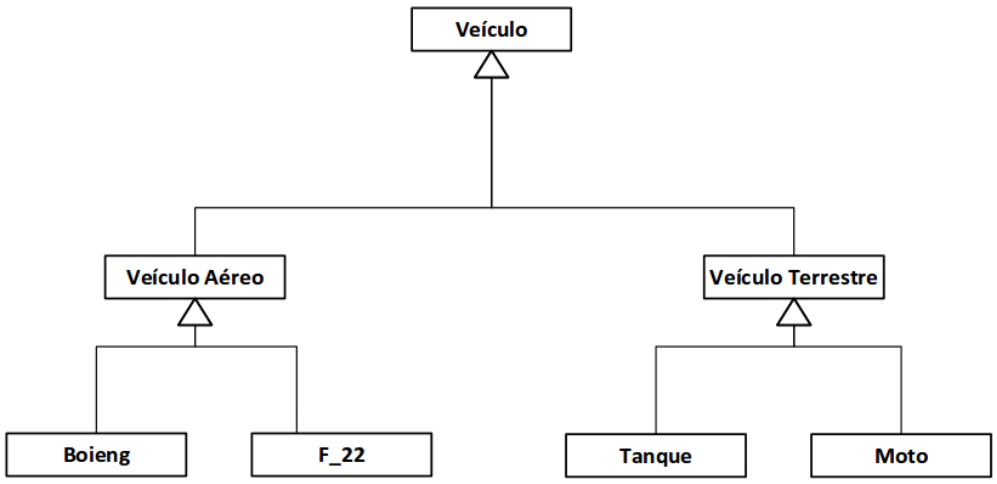
   Crie uma classe chamada *BasePlusComissionEmployee*, que contém as mesmas características de um *ComissionEmployee*, exceto:

   - Atributo *baseSalary* (double), que representa o salário base;
   - Método *earnings* somar o salário base à comissão;
   - Método *toString* deve também deve imprimir o salário base.

   Modele essa hierarquia da maneira que achar conveniente, tentando usufruir ao máximo do conceito de herança. Crie uma classe de teste que instancia e testa os objetos das classes acima.

2. Muitos programas escritos com herança podem ser escritos com composição e vice-versa. Reescreva o exercício anterior utilizando composição em vez da herança.

3. Crie a seguinte hierarquia de classes:

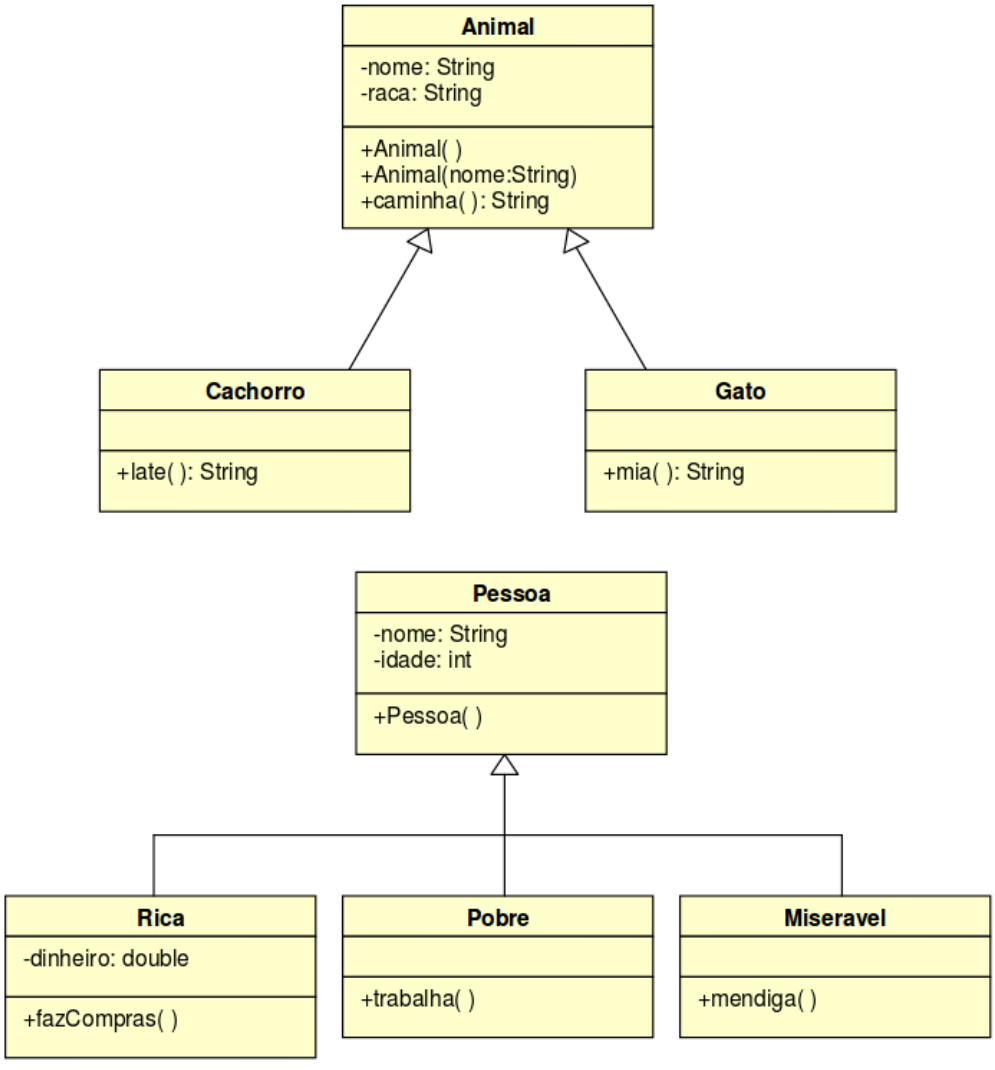


   - Veículos devem possuir o método *locomover*;
   - Veículos terrestres devem implementar o método *andar*;
   - Veículos aéreos devem implementar o método *voar*;
   - Veículos devem implementar o método bater e devem receber outro veículo como entrada;

- Crie pelo menos um atributo exclusivo para cada tpo de veículo (Moto, Boeing, Tanque...);
- Crie construtores que recebam atributos e verifique seu funcionamento;
- Deve haver uma forma de identificar unicamente os veículos;
- Utilize o método *toSring* para imprimir as informações dos veículos;
- Utilize as classe de Veículos de modo que seja possível controlar o número ao todo de veículos criados, o número de veículos terrestres e o número de veículos aéreos.
- Modele a hierarquia de classes da maneira que achar conveniente.

Ao final da modelagem, crie uma classe de teste para instanciar vários objetos e acessá-los de maneira polimórfica.

4. Implemente os diagramas de classe abaixo e crie uma classe de testes para testá-las polimorficamente:



5. Crie uma classe *Imovel*, que possui um endereço e um preço. Crie uma classe *Novo*, que é um imóvel e possui um adicional de preço, e uma classe *Velho*, que também é um imóvel e possui um desconto no preço. Crie métodos de acesso e impressão para todos os atributos das classes criadas.

6. Qual a finalidade da palavra "protected"? Há uma maneira de não utilizá-la na modelagem de hierarquia de classes? Quais as vantagens e desvantagens de usá-la?

7. A empresa XPTO necessita desenvolver um sistema para catalogar itens colecionáveis (livros, CDs, DVDs e revistas). O objetivo deste sistema é manter os itens colecionáveis, organizados por tipo. O sistema deve permitir cadastrar os dados comuns e os específicos de cada tipo de item. Os

dados comuns são: identificação única, nome, data de aquisição e lista de autores. Para os livros é importante manter também o nome da editora e o ano de publicação. Já para os CDs, é interessante manter o gênero musical e a identificação das faixas de áudio (nomes das músicas). Para os DVDs é importante armazenar o tipo (musical, filme ou dados), e uma descrição geral. Por fim, das revistas é interessante manter o ano de publicação, o volume, a editora e os principais assuntos tratados. Modele o problema acima utilizando classes e teste-as polimorficamente.