

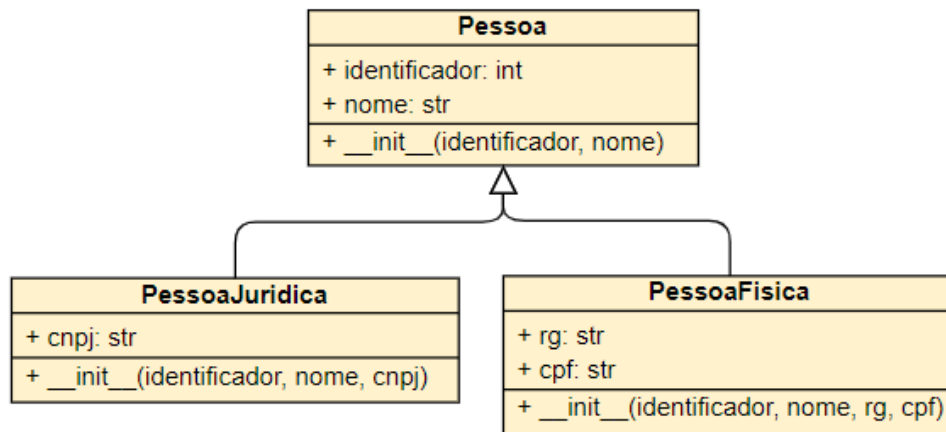
Exercício 01

O diagrama abaixo fornece uma hierarquia de classes onde a classe Pessoa é a superclasse (classe mãe), e as classes PessoaFisica e PessoaJuridica são as subclasses (classes filhas).

Crie a classe Pessoa com os atributos identificador e nome.

Crie a classe PessoaJuridica que herda da classe Pessoa e acrescenta o atributo cnpj.

Crie a classe PessoaFisica que herda da classe Pessoa e acrescenta os atributos rg e cpf.



Utilize o trecho de programa abaixo para testar as classes

```

pessoa1 = Pessoa(1, "Nome da Pessoa")
p_juridica = PessoaJuridica(2, "Nome da Pessoa Juridica", "1111111111")
p_fisica = PessoaFisica(3, "Nome da Pessoa Fisica", "22222222", "33333333")

print(pessoa1.identificador)      # 1
print(pessoa1.nome)               # Nome da Pessoa

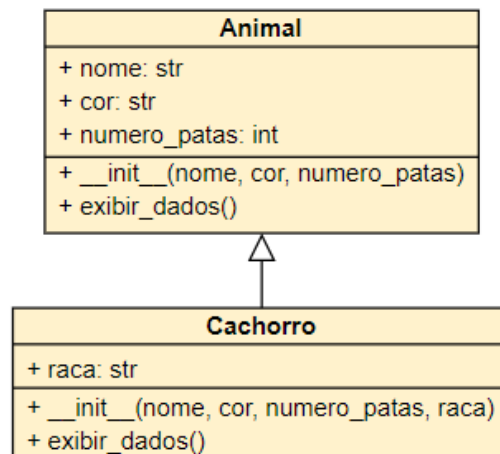
print(p_juridica.identificador)   # 2
print(p_juridica.nome)            # Nome da Pessoa Juridica
print(p_juridica.cnpj)            # 1111111111

print(p_fisica.identificador)     # 3
print(p_fisica.nome)              # Nome da Pessoa Fisica
print(p_fisica.rg)                # 22222222
print(p_fisica.cpf)               # 33333333
  
```

Exercício 02

Crie a classe `Animal` com os atributos `nome`, `cor` e `numero_patas`. Crie também o método `exibir_dados`, que imprime na tela os dados do animal (`nome`, `cor` e `numero_patas`).

Crie a classe `Cachorro` que herda da classe `Animal` e que possui como atributo adicional a `raça` do cachorro. Crie também o método `exibir_dados`, que imprime na tela os dados do cachorro (`nome`, `cor`, `numero_patas` e `raça`).



Utilize o trecho de programa abaixo para testar as classes

```
animal = Animal("Passarinho", "Azul", 2)
animal.exibir_dados()          # exibe os atributos do animal

dog = Cachorro("Rex", "Marrom", 4, "Vira lata")
dog.exibir_dados()            # exibe os atributos do cachorro
```

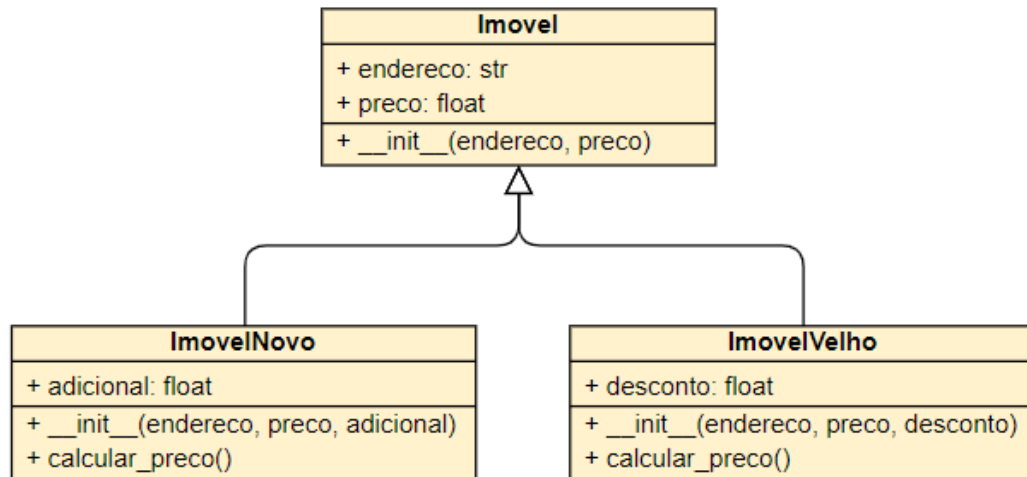
Exercício 03

Crie a classe `Imovel`, que possui um endereço e um preço.

Crie a classe `ImovelNovo`, que herda de `Imovel` e possui um adicional no preço.

Crie a classe `ImovelVelho`, que herda de `Imovel` e possui um desconto no preço.

O método `calcular_preco` das classes deve retornar o preço atualizado de acordo com o adicional ou desconto.



Utilize o programa abaixo para testar as classes

```
imovel = Imovel("Rua Silva, 123", 300000.0)
imovel_novo = ImovelNovo("Rua Joaquim, 999", 250000.0, 20000.0)
imovel_velho = ImovelVelho("Av. Brasil, 777", 500000.0, 35000.0)

print(imovel.endereco)                # Rua Silva, 123
print('Preço:', imovel.preco)          # 300000.0

print(imovel_novo.endereco)            # Rua Joaquim, 999
print('Preço:', imovel_novo.preco)     # 250000.0
print('Preço Atualizado:', imovel_novo.calcular_preco()) # 270000.0

print(imovel_velho.endereco)           # Av. Brasil, 777
print('Preço:', imovel_velho.preco)    # 500000.0
print('Preço Atualizado:', imovel_velho.calcular_preco()) # 465000.0
```

Exercício 04

Escreva um programa para armazenar dados de veículos.

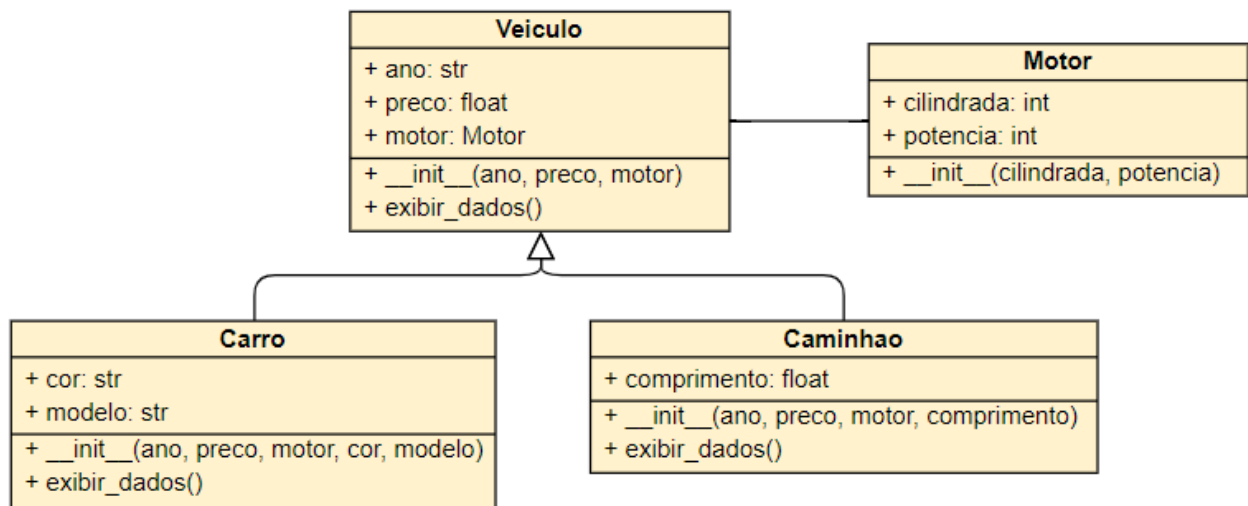
Crie a classe Motor que contém cilindrada e potencia.

Crie a classe Veiculo contendo ano de fabricação, preco e motor. Crie também o metodo exibir_dados para mostrar os dados do Veículo.

Crie a classe Carro, que herda da classe Veiculo e adiciona os atributos cor e modelo. Crie também o metodo exibir_dados para mostrar os dados do Carro.

Crie a classe Caminhão, que herda da classe Veiculo e adiciona o atributos comprimento (em metros). Crie também o metodo exibir_dados para mostrar os dados do Caminhão.

Obs.: A classe Motor não possui relação de herança com a classe Veiculo, possui apenas uma relação de associação (o veiculo possui um motor)



Utilize o programa abaixo para testar as classes

```
motor1 = Motor(1000, 500)
motor2 = Motor(8000, 900)
carro = Carro(2010, 20000, motor1, "branca", "gol")
caminhao = Caminhao(2015, 80000, motor2, 10)

carro.exibir_dados()          # imprime os valores de todos os atributos do carro
caminhao.exibir_dados()      # imprime os valores de todos os atributos do caminhão
```