Herança Simples

Prof. Fábio Procópio

Prof. João Nascimento

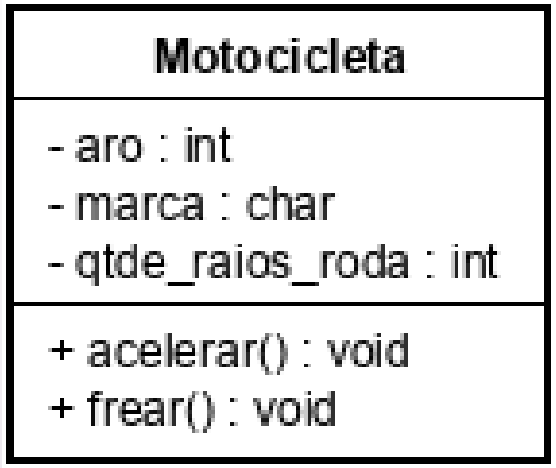
Técnico Integrado em Informática – Programação Estruturada e Orientada a Objetos

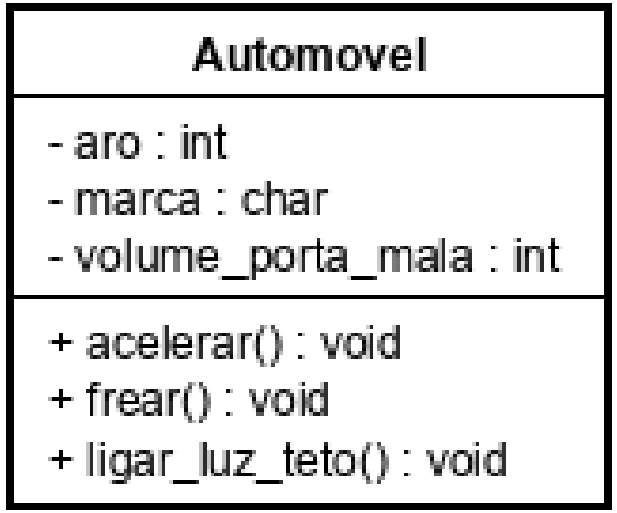
Relembrando...

- Na aula passada, implementamos os conceitos iniciais de orientação a objetos: classes, objetos, atributos e métodos;
- Nesta aula, continuaremos nossos estudos falando sobre um dos pilares da Orientação a Objetos: o conceito de herança.

Exercício Resolvido 01 – 1 de 3

- Implemente as classes `Motocicleta` e `Automovel`.





Os métodos **`acelerar()`**, **`frear()`** e **`ligar_luz_teto()`** devem imprimir a string "Acelerando...", "Freando..." e "Luz ligada.", respectivamente.

Exercício Resolvido 01 – 2 de 3

```
class Motocicleta:
    def __init__(self, aro, marca, qtde_raios_roda):
        self.aro             = aro
        self.marca           = marca
        self.qtde_raios_roda = qtde_raios_roda

    def acelerar(self):
        print("Acelerando...")

    def frear(self):
        print("Freando...")
```

Exercício Resolvido 01 – 3 de 3

```
class Automovel:
    def __init__(self, aro, marca, volume_porta_mala):
        self.aro               = aro
        self.marca             = marca
        self.volume_porta_mala = volume_porta_mala

    def acelerar(self):
        print("Acelerando...")

    def frear(self):
        print("Freando...")

    def ligar_luz_teto(self):
        print("Luz ligada.")
```

Comentários

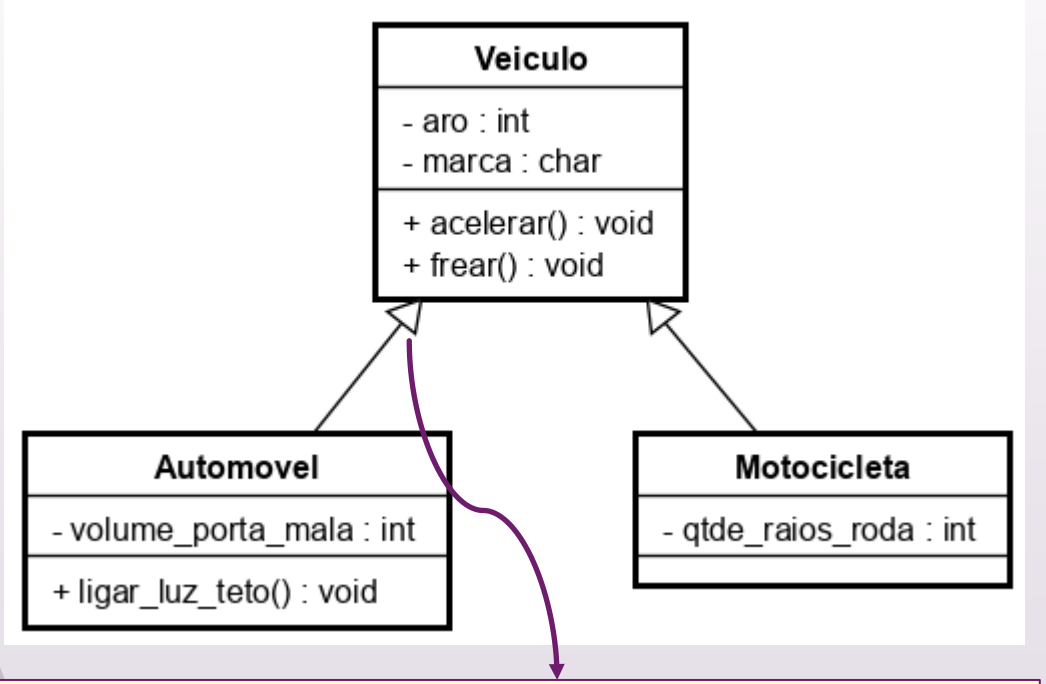
- Você percebeu que os atributos `aro` e `marca` são comuns às classes `Motocicleta` e `Automovel`?
  - Veja que **`qtde_raios_roda`** (na classe `Motocicleta`) e **`volume_porta_mala`** (na classe `Automovel`) são os atributos que não são comuns entre essas classes
- Observou também que os métodos acelerar() e frear() são idênticos nas duas classes?
  - Veja que **`ligar_luz_teto()`** (na classe `Automovel`) é o único método que diferencia as duas classes
- Resumindo:
  - Fizemos um **CTRL + C / CTRL + V** e modificamos pequenas partes no código
  - Se, eventualmente, os método **`acelerar()`** e **`frear()`** fossem modificados, precisaríamos modificá-los nas duas classes
  - Será que CTRL + C / CTRL + V é uma **boa prática** de programação? Não!

Introdução – 1 de 2

- Herança é um mecanismo que permite que **características** e **comportamentos** comuns de várias classes sejam **especificados em apenas uma classe**, isto é, uma superclasse;
- A utilização do conceito de herança pode evitar retrabalho quando os métodos são implementados em uma superclasse.
- A partir de uma **superclasse** (também chamada de classe base, de classe mãe ou de classe ancestral), **outras classes podem ser especificadas** (chamadas de **subclasses**);
- Uma **subclasse** (também chamada de classe derivada, de classe filha ou de classe descendente) **herda atributos** e **métodos** da sua superclasse;

Introdução – 2 de 2

- As classes `Motocicleta` e `Automovel`, apresentadas anteriormente, podem ser apresentadas em um diagrama de classes, como o exemplo abaixo.



Classe **`Veiculo`**: possui todos os atributos e métodos que são comuns às classes `Automovel` e `Motocicleta`

Classe **`Automovel`**: possui apenas o atributo e o método que a distingue de `Motocicleta`

Classe **`Motocicleta`**: possui apenas o atributo que a distingue de `Automovel`

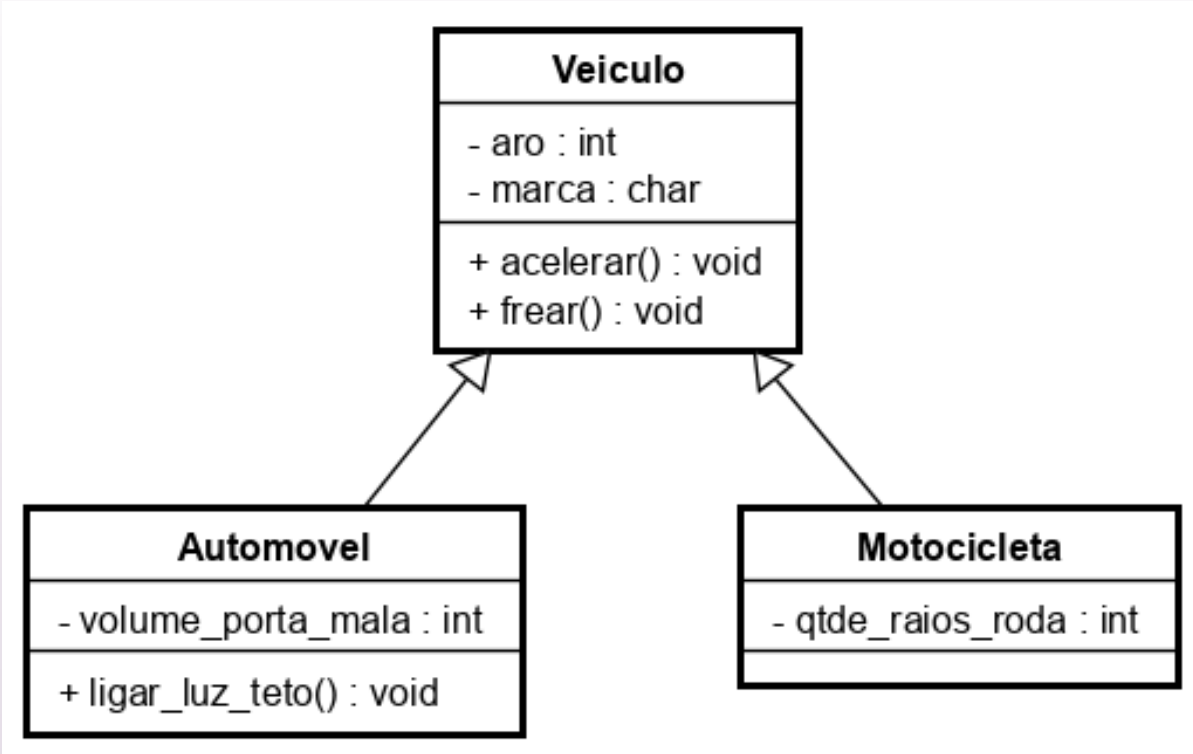
O modelo nos "diz" que:

a) Além de possuir o atributo **`volume_porta_mala`** e o método **`liga_luz_teto()`**, `Automovel` também possui os **atributos** e **métodos** definidos em **`Veiculo`**
b) Idem para a classe `Motocicleta`

Essa seta "aberta" significa **Herança**, a qual indica que Veiculo é uma superclasse e Automovel e Motocicleta são subclasses de Veiculo.

Exercício Resolvido 02 – 1 de 4

- Depois de modificarmos o modelo de representação das classes Motocicleta e Automovel, vamos implementar o modelo abaixo.



Exercício Resolvido 02 – 2 de 4

```
class Veiculo:
    def __init__(self, aro, marca):
        self.aro   = aro
        self.marca = marca

    def acelerar(self):
        print("Acelerando...")

    def frear(self):
        print("Freando...")
```

Na classe **`Veiculo`**, implementamos apenas os atributos e métodos que são comuns às classes `Automovel` e `Motocicleta`

Exercício Resolvido 02 – 3 de 4

```
from Veiculo import *

class Automovel(Veiculo):
    def __init__(self, aro, marca, volume_porta_mala):
        Veiculo.__init__(self, aro, marca)
        self.volume_porta_mala = volume_porta_mala

    def ligar_luz_teto(self):
        print("Luz ligada.")
```

Na classe **Automovel**, deixamos apenas o atributo (`volume_porta_mala`) e o método (`ligar_luz_teto()`) que a distingue da classe `Motocicleta`.

```
from Veiculo import *

class Motocicleta(Veiculo):
    def __init__(self, aro, marca, qtde_raios_roda):
        Veiculo.__init__(self, aro, marca)
        self.qtde_raios_roda = qtde_raios_roda
```

Em **Motocicleta**, definimos apenas o atributo que não é comum à `Automovel`: qtde_raios_roda.

Exercício Resolvido 02 – 4 de 4

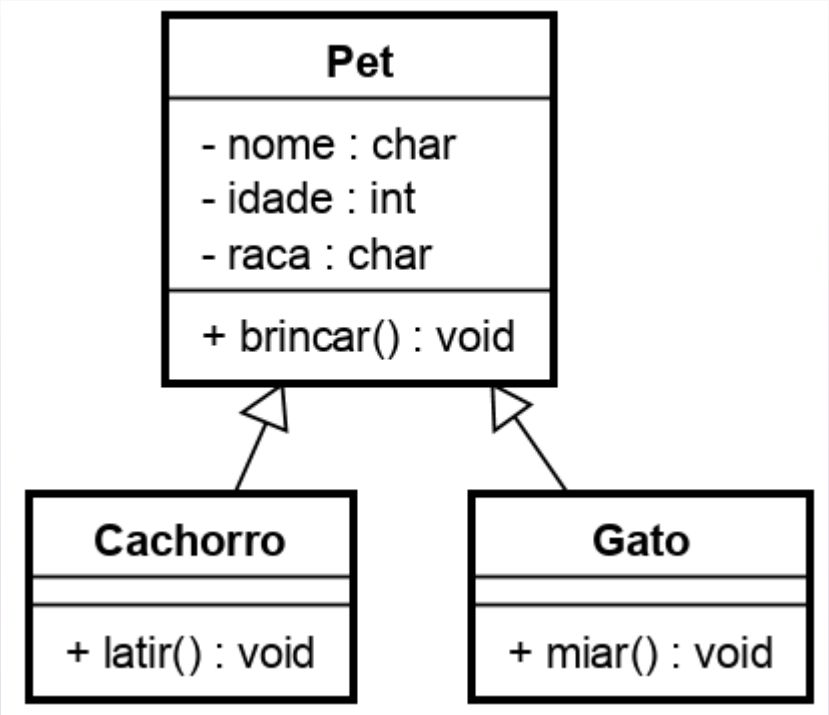
```
from Automovel import *
from Motocicleta import *

carro = Automovel(17, "Hyundai", 500)
carro.ligar_luz_teto()
carro.acelerar()
carro.frear()

moto = Motocicleta(14, "Honda", 50)
moto.acelerar()
moto.frear()
```

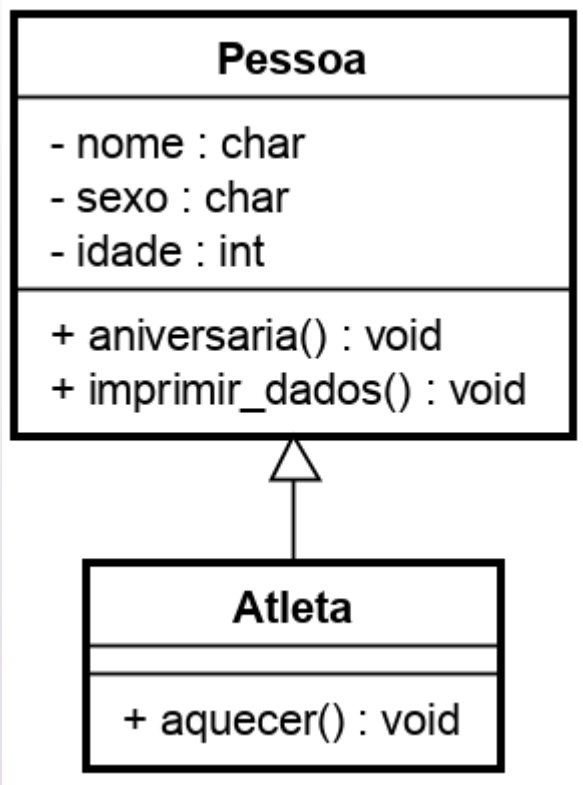
Exercício de Fixação 01

- Anteriormente, você implementou as classes `Cachorro` e `Gato`. Agora, adapta-as e implemente o modelo abaixo. Em seguida, teste-as.



Exercício de Fixação 02

- Anteriormente, você implementou a classe `Pessoa`. Reaproveite-a e implemente o modelo abaixo. Em seguida, teste a classe `Atleta`.



Canal no Youtube

Em nosso canal no Youtube, você pode acessar uma playlist que preparamos com várias vídeo-aulas falando sobre o assunto. Veja:

- Herança Simples

Referências

1. DEVMEDIA, **Conceitos e Exemplos – Herança: Programação Orientada a Objetos – Parte 1**. Disponível em: http://www.devmedia.com.br/conceitos-e-exemplos-heranca-programacao-orientada-a-objetos-parte-1/18579. Acessado em: 13 ago. 2019.

2. Ricarte, Ivan. **Programação Orientada a Objetos: herança**. Disponível em: http://pt.slideshare.net/ivanricarte/programao-orientada-a-objetos-herana. Acessado em: 13 ago. 2019.

3. Castro, Marcos. **Python – Utilizando herança**. Disponível em: https://www.youtube.com/watch?v=UIsb8bpCnSc. Acessado em: 13 ago. 2019.