Orientação à Objetos: Introdução

CIÊNCIA DA COMPUTAÇÃO

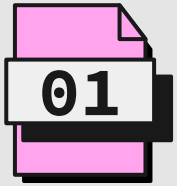
ATITUS EDUCAÇÃO

- Aula 02 - Organização e Abstração

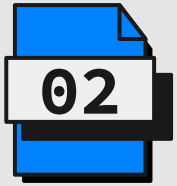
Prof. Me. Lucas R. C. Pessutto

ATITUS EDUCAÇÃO

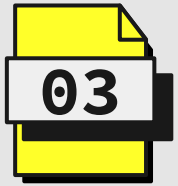
Na aula de hoje...



Paradigmas de Ling. de Programação



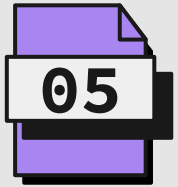
Pilares da Orientação a Objetos



Abstração

Classes e Objetos

Conceito
Implementação de uma classe em Python
Instanciação de Objetos



UML

O que é?
Diagrama de Classes

ATITUS EDUCAÇÃO

Paradigmas de Programação

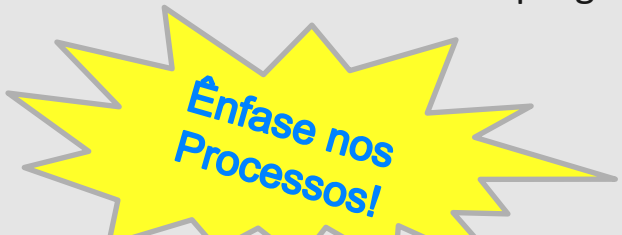
* Atrelados à forma de pensar sobre a solução de problemas
  → Como atuar sobre a realidade?

* Determinante na análise e abstração do problema

* Proveem uma estrutura fundamental para (d)escrever algoritmos, usando-se de conceitos específicos sobre as técnicas de programação disponíveis:
  → Paradigma de Programação Procedural
  → Paradigma de Programação Orientada a Objetos
  → Paradigma de Programação Funcional
  → Paradigma de Programação Lógico

Paradigma Procedural

* Estado e ações que manipulam este estado
* Emprego de funções agrupados por funcionalidades em bibliotecas
* Modelagem por fluxogramas.
* Um programa é modelado de acordo com a Arquitetura do computador: memória e operações
* Orientado a procedimentos (“orientado a verbo”):
  - → Sintaxe dada pelo nome da sub-rotina - que supostamente indica o que ela faz
  - → Trabalha com argumentos e retorno de variáveis que armazenam os estados do programa.
* Exemplos: C, Pascal, Fortran, etc.



Paradigma Orientado a Objetos

* Organizam entidades do mundo real como objetos
* Abstração de um aspecto de interesse, incluindo estrutura e comportamento
* Programas são um conjunto de classes que interagem entre si, trocando mensagens
* Um programa é modelado de acordo com os dados
* Operações e dados são pensados conjuntamente
  - → Especificação e implementação de classes
  - → Objetos são entidades dinâmicas criadas a partir de classes
* Exemplos: C++, Java, etc.

Organização e Abstração

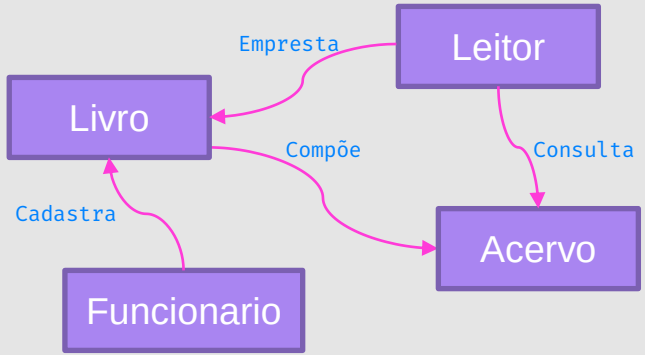
Sistema para Gerenciar uma Biblioteca

Procedural

Estado	Operações
Lista de Livros	iniciarSistema()
Lista de Autores	login()
Lista de Usuários	addLivro()
Número de Livros	emprestimo()
Saldo em Caixa	pagarMulta()
...	consultaAcervo()
	...

Orientado a Objetos

Objetos e Mensagens



Prog Estruturada x Orientação a Objetos

* Programação Estruturada utiliza o princípio da decomposição: um problema complexo é dividido em um conjunto de funções, que juntas o resolvem

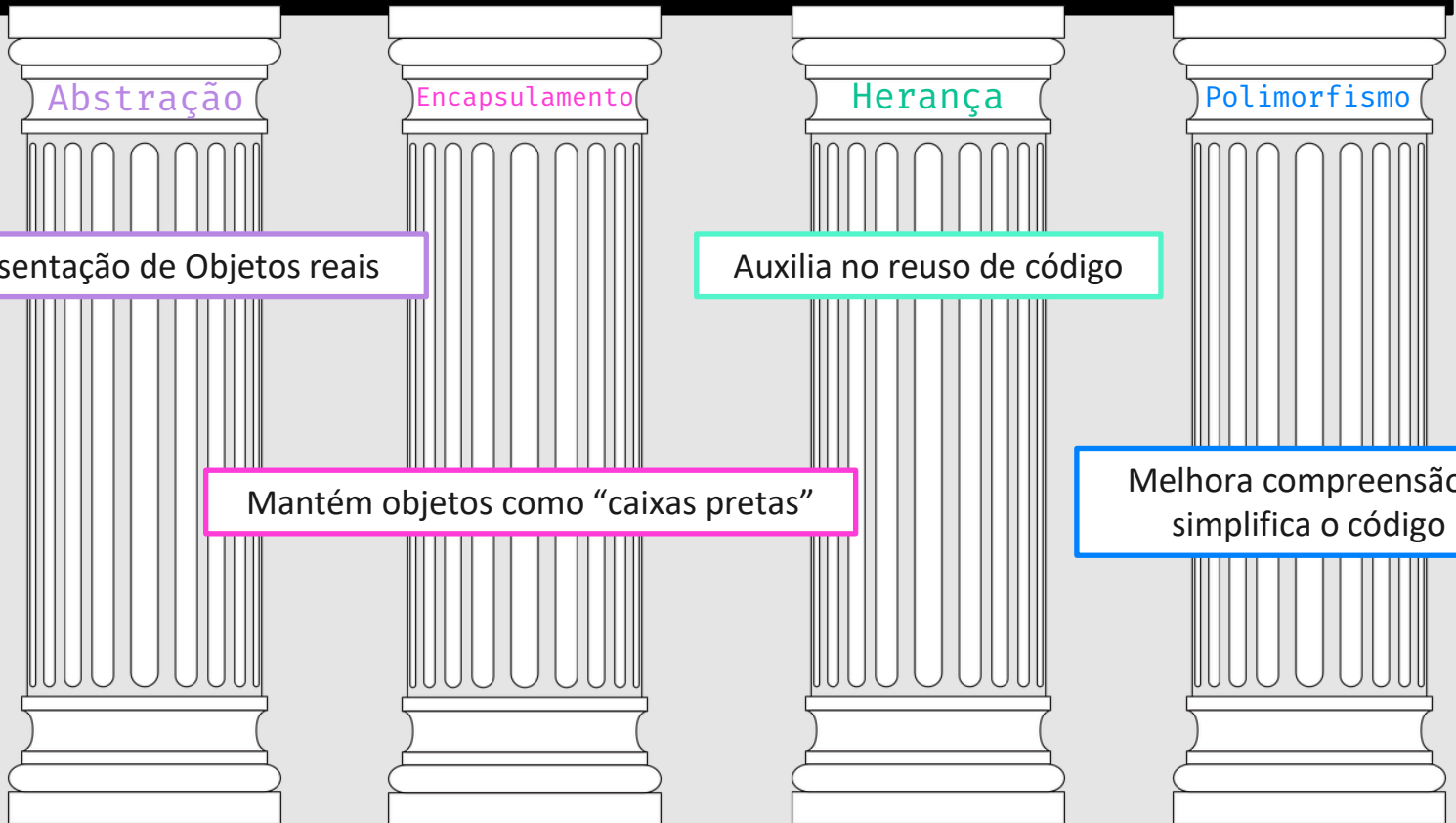
* Mas é difícil de entender e melhorar um programa que consiste em uma grande coleção de métodos

* Na Programação Orientada a Objetos, um programa consiste em um conjunto de objetos que colaboram entre si

* Cada objeto possui seu próprio conjunto de dados e alguns métodos que operam sobre esses dados

BOOCH, 1994

Pilares Orientação a Objetos



Representação de Objetos reais

Mantém objetos como “caixas pretas”

Auxilia no reuso de código

Melhora compreensão e simplifica o código

Abstração

O que pode ser visto nesta pintura?

Fantasia à Constantinople - Felix Ziem

Abstração

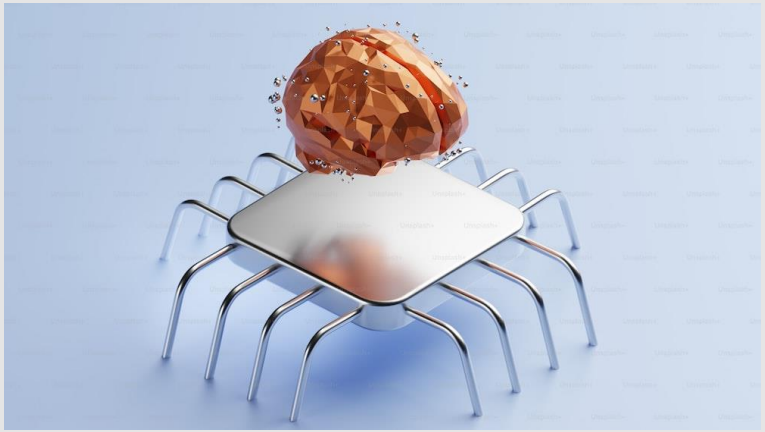
Vejamos mais de perto...

Abstração

Abstração

"Processo mental que consiste em escolher ou isolar uma aspecto determinado de um estado de coisas relativamente complexo, a fim de simplificar a sua avaliação, classificação ou para permitir a comunicação do mesmo"

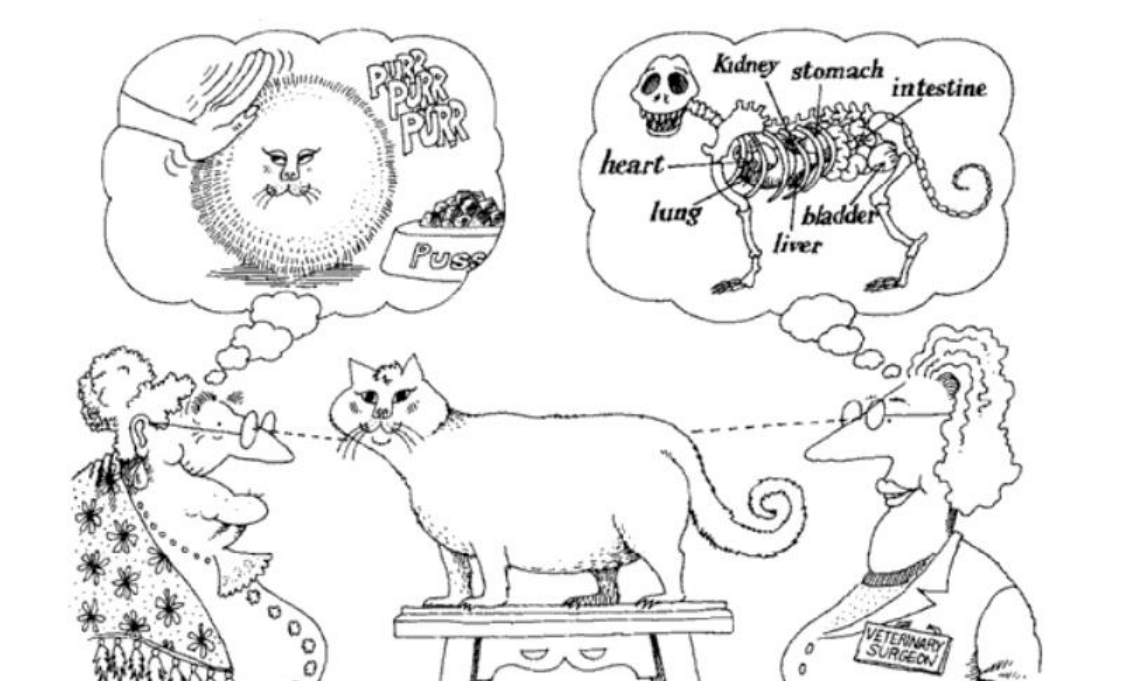
— Houaiss (2006)



BOOCH, 1994

Abstração

Abstração foca nas características principais de determinado objeto, de acordo com a perspectiva do observador



Exercício

Quais os atributos mais relevantes ao mapear a entidade **CARRO** nas aplicações abaixo?

EM DEFESA DA VIDA
DetranRS

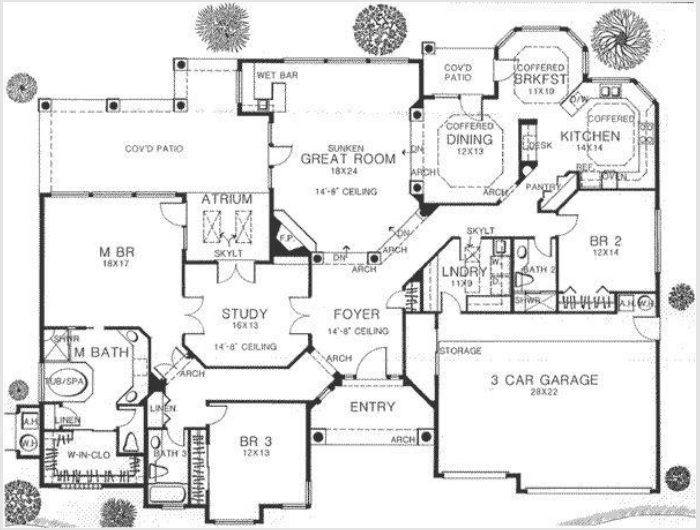
VEÍCULOS

- Consulta de veículo
- Licenciamento do veículo
- CRLV digital

MAIS

Classes e Objetos

Classe



Objeto

Classes e Objetos

Uma classe descreve um **conjunto de objetos do mesmo tipo**, que possuem o **mesmo comportamento**

Classes: Atributos + Métodos

Esta Ligado?

Marca

Cor

Marcha

Ano

Velocidade

Classe Carro

Ligar

Desligar

Andar

Mudar Marcha

Classes: Atributos + Métodos

Classe

Televisão

Objetos

Comportamento + Identidade + Estado

Carro fusca

Marca	“Fusca”
Ano	1978
Cor	“Amarelo”
Velocidade	0
Marcha	1
estaLigado	False

Carro ferrari

Marca	“Ferrari”
Ano	2013
Cor	“Vermelho”
Velocidade	150
Marcha	3
estaLigado	True

Carro gol

Marca	“Gol”
Ano	2021
Cor	“Prata”
Velocidade	50
Marcha	2
estaLigado	True

Objetos

Crie dois exemplos de instâncias da classe Televisão

Classe Televisão

Classes em Python

Um programa pode definir novas classes (tipo de objeto) utilizando a primitiva `class`.

Assim como funções (def), classes também devem ser definidas antes de usadas.

```
class Carro:
```

Comando que identifica a definição de uma nova classe

Nome da classe escolhido pelo programador

Por convenção, costuma-se usar *CamelCase* nos nomes de classes definidas pelo programador.

Classes em Python

Toda classe deve conter um método (operação) especial __init__ que normalmente define os atributos (dados) básicos de um objeto.

```
class Carro:
    def __init__(self, marca, ano, cor):
        self.marca = marca
        self.ano = ano
        self.cor = cor
        self.velocidade = 0
        self.marcha = 0
        self.esta_ligado = False
```

O primeiro argumento é sempre `self` que identifica uma **instância** da classe.

Um método é uma espécie de "função local" a uma classe, isto é, só existe neste contexto.

Outros argumentos seguem as mesmas regras de funções.

Os **argumentos** passados para o método `__init__`, em geral, são usados para inicializar **atributos**.

Os atributos (dados) internos do objeto são identificados pelo prefixo `self.`

Métodos “Dunder”

* Dunder = “Double underscore before and after”. (Também chamados de métodos mágicos)

* São métodos especiais, que não são chamados diretamente pelo programador, mas sim pelo interpretador Python em momentos especiais

* Veremos mais sobre métodos dunder quando falarmos sobre sobrecarga de operadores

Implementando Métodos

* Outros métodos adicionam funcionalidade à classe `Carro`:

```
class Carro:
    ...
    def ligar(self):
        self.esta_ligado = True
```

O argumento `self` segue sendo necessário.

Atributos do objeto definidos em outros métodos podem ser usados normalmente.

Argumentos e retorno funcionam da mesma forma como em funções.

Implementando Métodos

* Adicione outros métodos à classe `Carro`:
  - `Desligar`: muda o valor do atributo `esta_ligado` para `False`, a `velocidade` para `0` e a `marcha` para `0`
  - `Acelerar`: aumenta a `velocidade` (use parâmetros)
  - `Frear`: diminui a `velocidade` (use parâmetros). Não deixe a velocidade ficar negativa!
  - `Mudar Marcha`: altera o valor da `marcha`. Use um parâmetro opcional para indicar ré (atributo marcha = -1)

Instanciando Objetos

Agora podemos criar objetos do tipo carro e instancia-los:

```
carro1 = Carro("Fusca", 1978, "Amarelo")
carro2 = Carro("Ferrari", 2013, "Vermelho")
carro3 = Carro("Gol", 2021, "Prata")

print(f"Cor do carro 1: {carro1.cor}")
print(f"Velocidade do carro 3: {carro3.velocidade}")

carro3.acelerar(50)
print(f"Cor do Carro 1: {carro1.cor}")

carro1.ligar()
carro1.acelerar(20)
carro1.mudar_marcha()
carro1.acelerar(30)
print(f"Velocidade do carro 1: {carro1.velocidade}")
print(f"Marcha do carro 1: {carro1.marcha}")
```

```
Cor do carro 1: Amarelo
Velocidade do carro 3: 0
Carro Desligado!
Cor do Carro 1: Amarelo
Velocidade do carro 1: 50
Marcha do carro 1: 1
```

Instanciando Objetos

Código | **Interpretador Python** | **Classe Carro**

```
carro1 = Carro("Fusca", 1978, "Amarelo")
```

Aloca espaço na memória para um objeto Carro

Chama o método `__init__()`, passando o novo objeto

Executa o método `__init__()`, e seta `self` para o novo objeto inicializa o valor dos atributos

Retorna o novo objeto

Guarda o objeto criado em `carro1`

Exercício:

Implemente a Classe Televisão que você projetou anteriormente e instancie alguns objetos dessa classe

Projetando Classes

* A partir de agora, utilizaremos diagramas para representar as classes que implementamos.

* Diagramas são utilizados para especificar um software, sem a necessidade de implementar.

* Um bom profissional primeiro realiza a especificação / modelagem do sistema que irá desenvolver, para que depois ele implemente.

* Isso diminui a chances de erros do projeto, diminui a reimplementação e diminui o tempo de desenvolvimento

Unified Modeling Language (UML)

* A UML é a linguagem padrão da Object Management Group para visualização, especificação, construção e documentação de sistemas baseados em POO.

* UML oferece um padrão de projeto de sistemas que inclui aspectos abstratos (funcionalidades do sistema) e concretos (classes do Python)

* A UML usa diagramas (estruturais ou comportamentais) para mostrar visões distintas do modelo de um sistema OO

* Nos concentraremos nos Diagramas de Classe.

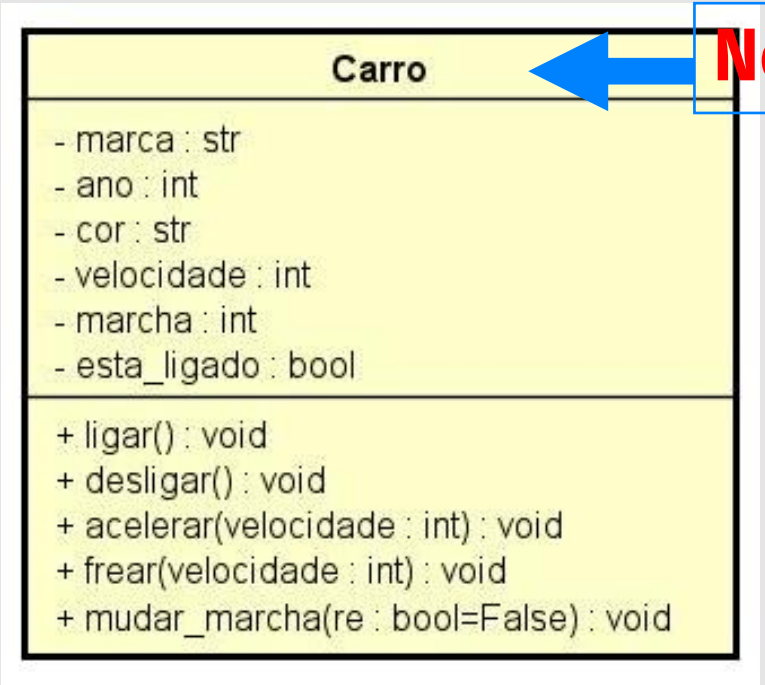
Diagrama de Classes

* Descrevem os vários tipos de objetos no sistema e os relacionamento existentes entre eles

* Usos comuns:
  - → Modelar o vocabulário do sistema em termos de quais abstrações devem ou não fazer parte do domínio
  - → Modelar as colaborações/interações entre entidades
  - → Modelar lógica dos dados do sistema (base para modelagem de banco de dados)
* Compostos por entidades (classes ou interfaces) e relacionamentos (dependência, generalização e associação)

UNIFIED MODELING LANGUAGE™

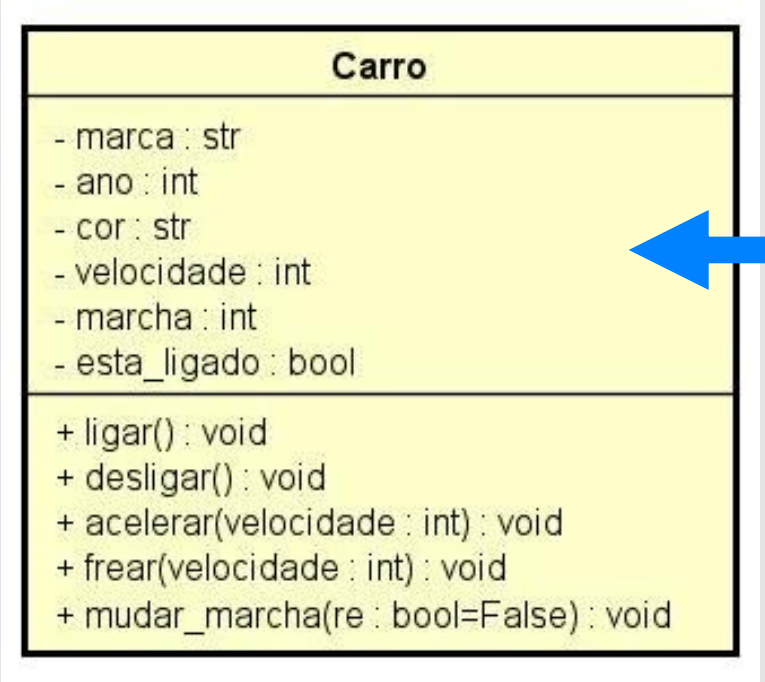
Representando uma Classe



UNIFIED MODELING LANGUAGE™

ATITUS EDUCAÇÃO

Representando uma Classe

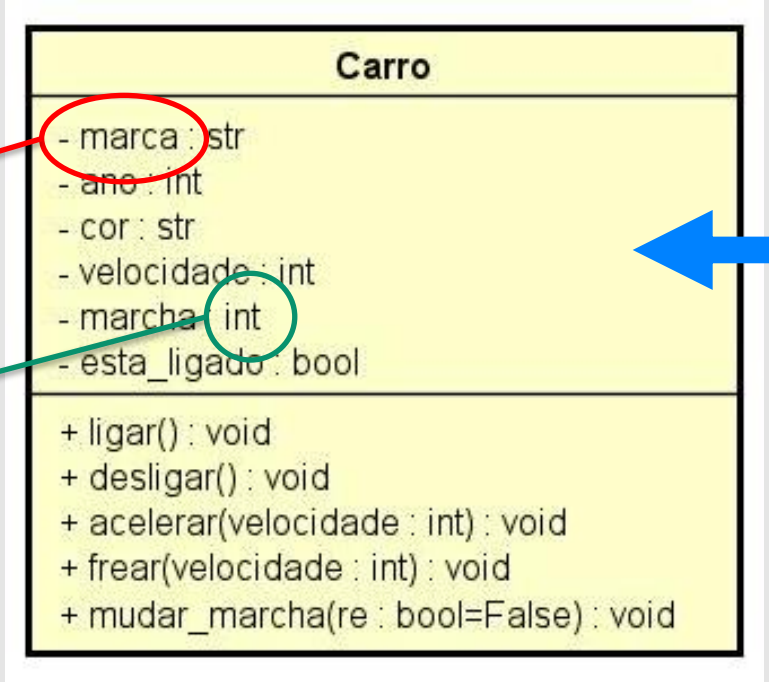


Atributos

UNIFIED MODELING LANGUAGE™

ATITUS EDUCAÇÃO

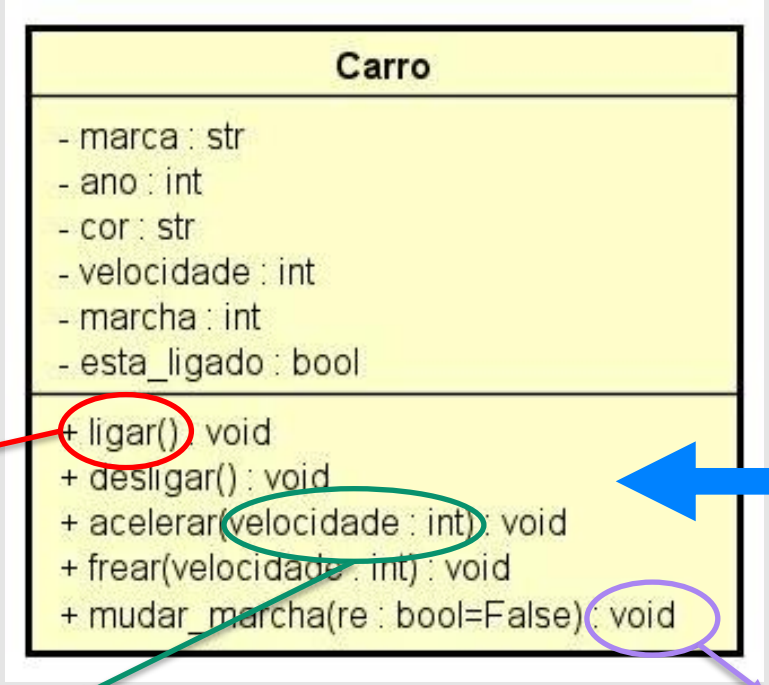
Representando uma Classe



ATITUS EDUCAÇÃO

UNIFIED MODELING LANGUAGE™

Representando uma Classe



UNIFIED MODELING LANGUAGE™

ATITUS EDUCAÇÃO

Diagrama de Classes

* Usando a Ferramenta Astah: https://astah.net/pt/
* Baixe e Instale
* Vamos recriar o diagrama da classe Carro
* Crie o diagrama de classes da Classe Televisão