Conceitos de Programação Orientada a Objetos

Instituto Federal de Educação, Ciência e Tecnologia do Triângulo Mineiro
Prof. Edwar Saliba Júnior
Dezembro de 2019

Conceitos Fundamentais

- Classe,
- Encapsulamento,
- Ocultamento de informação,
- Atributo,
- Método,
- Mensagem e
- Objeto.

É bom saber!

- Palavra-chave no paradigma de Orientação a Objetos (OO):

ORGANIZAÇÃO

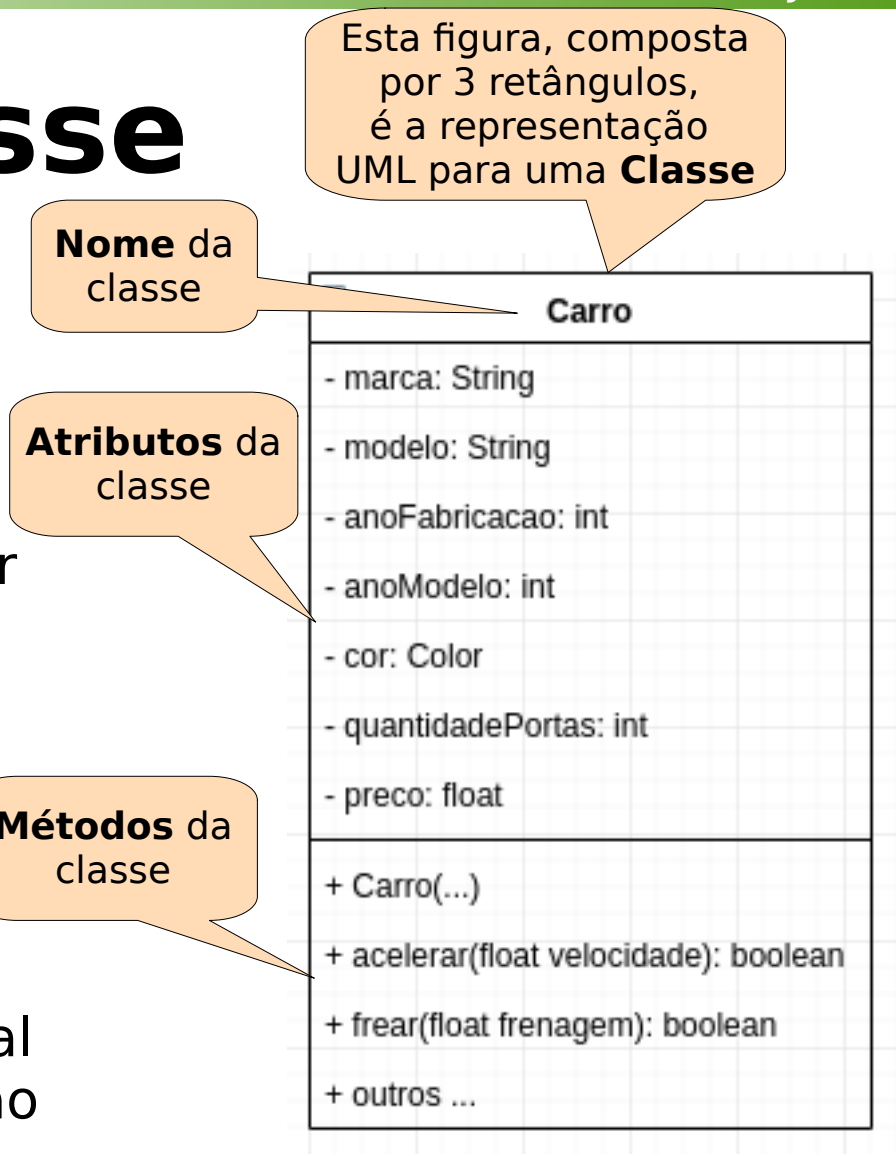
do código-fonte.

Carro X OO

- Suponha que você queira guiar um carro e em determinados momentos fazê-lo andar mais rápido pisando no acelerador;
- O que deve acontecer antes que você possa fazer isto?
  - Alguém tem que projetar e construir o carro, pois o carro é um objeto.

Classe

- Em geral, os carros 'nascem' em um desenho de engenharia semelhante às plantas utilizadas nos projetos de casas;
- Nesta planta, procura-se detalhar o máximo possível, de forma genérica, todos os **atributos** (características) e **métodos** (funcionalidades) do que está sendo projetado;
- Neste desenho de engenharia (o do carro) está o desenho do pedal do acelerador, do freio, da direção e etc.



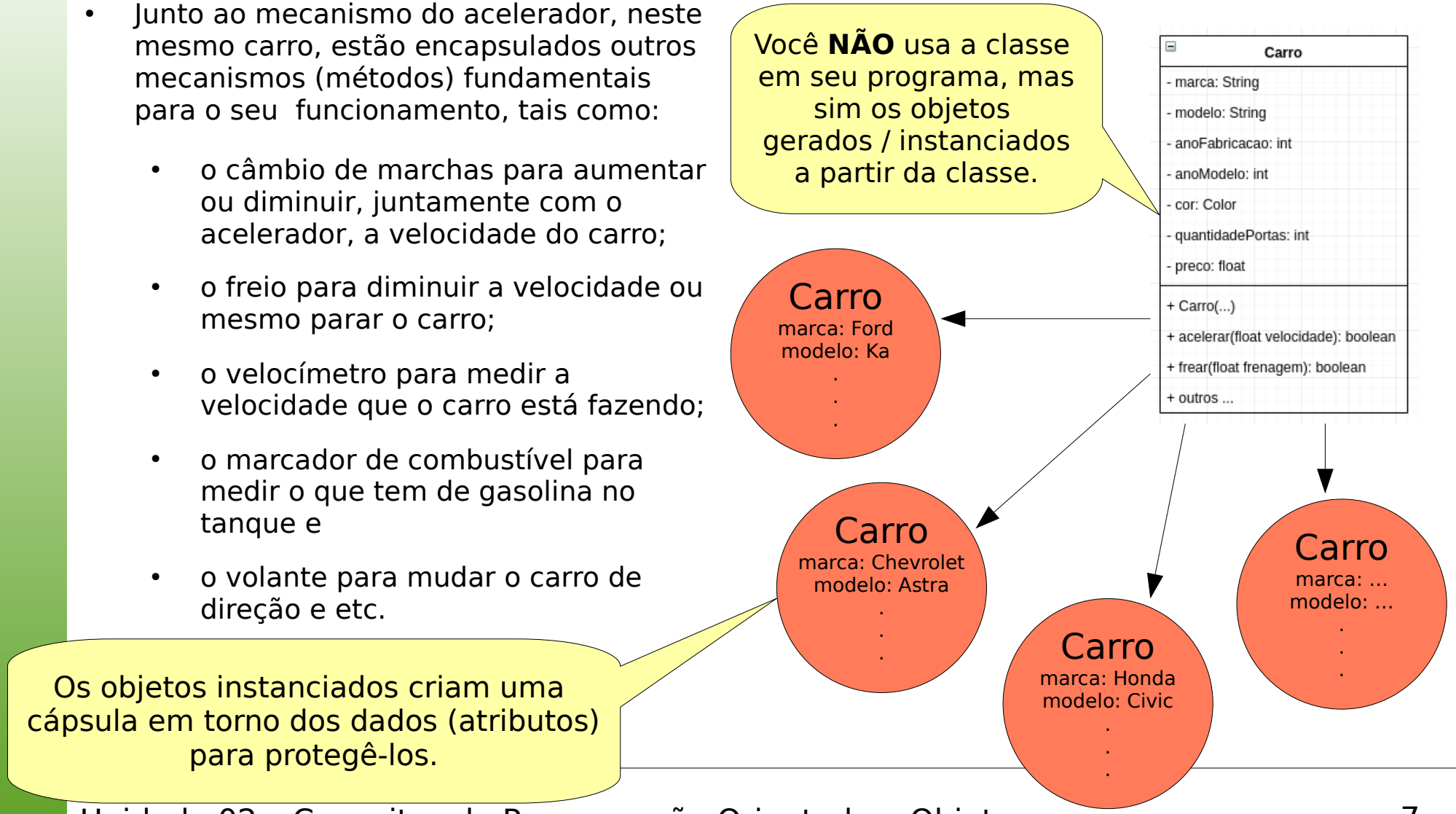
Encapsulamento

- Um objeto, empacota tanto os dados quanto os procedimentos (funções) que operam sobre eles.
- As funções são tipicamente chamadas de métodos ou operações;
- Um objeto executa uma operação quando recebe uma solicitação (ou mensagem) de um cliente (outro objeto);
- As solicitações (*requests*) são a única maneira de conseguir que um objeto execute uma operação;
- As operações são a única maneira de mudar os dados internos de um objeto;
- Por causa destas restrições, diz-se que o estado interno de um objeto está encapsulado; ele não pode ser acessado diretamente e sua representação é invisível do exterior do objeto.

Gamma et al. (2000, p. 27-28)

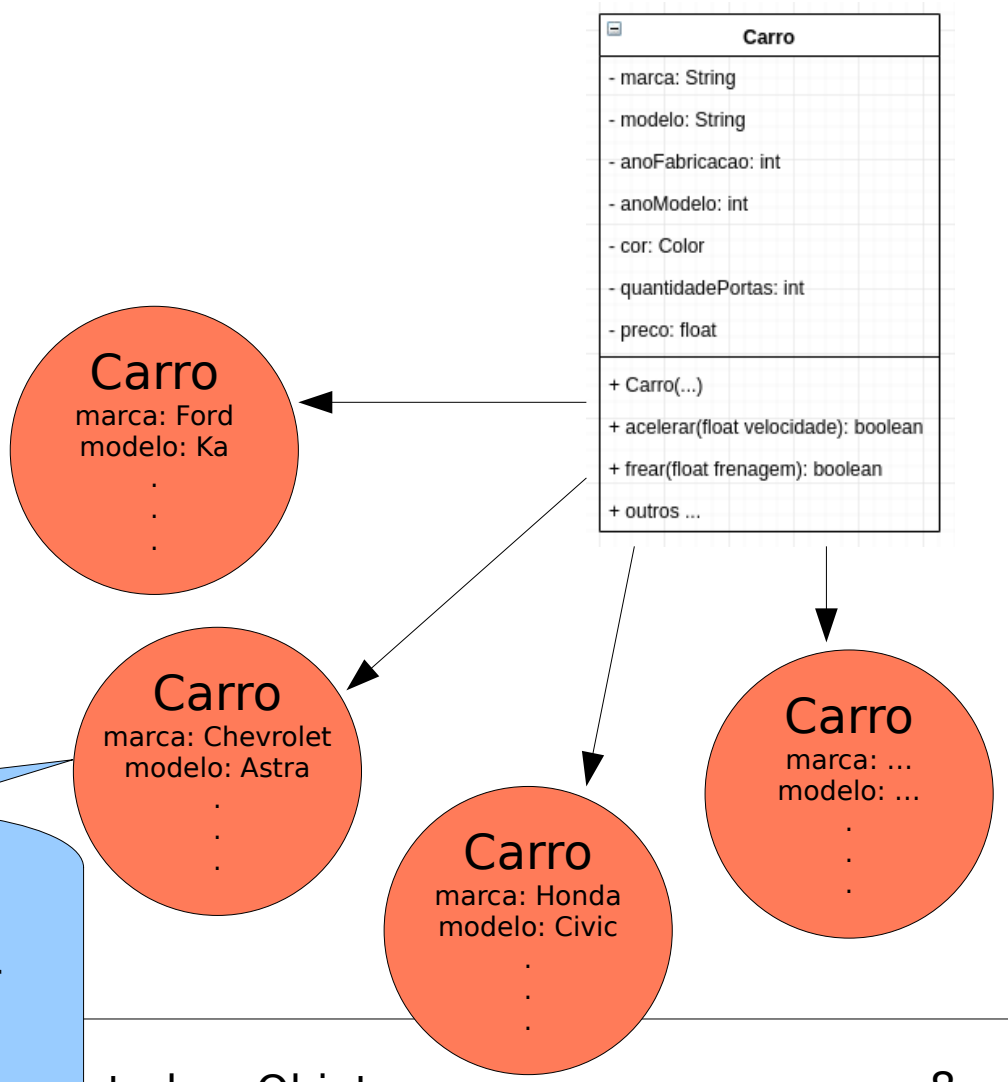
Encapsulamento

- Junto ao mecanismo do acelerador, neste mesmo carro, estão encapsulados outros mecanismos (métodos) fundamentais para o seu funcionamento, tais como:
  - o câmbio de marchas para aumentar ou diminuir, juntamente com o acelerador, a velocidade do carro;
  - o freio para diminuir a velocidade ou mesmo parar o carro;
  - o velocímetro para medir a velocidade que o carro está fazendo;
  - o marcador de combustível para medir o que tem de gasolina no tanque e
  - o volante para mudar o carro de direção e etc.



Ocultamento de Informações

- O pedal do acelerador, o pedal do freio e a direção, **ocultam** de seus usuários os complexos mecanismos que os fazem funcionar;
- este processo de ocultamento permite que pessoas com pouco ou nenhum conhecimento de como os motores e as engrenagens funcionam, dirijam um carro facilmente.

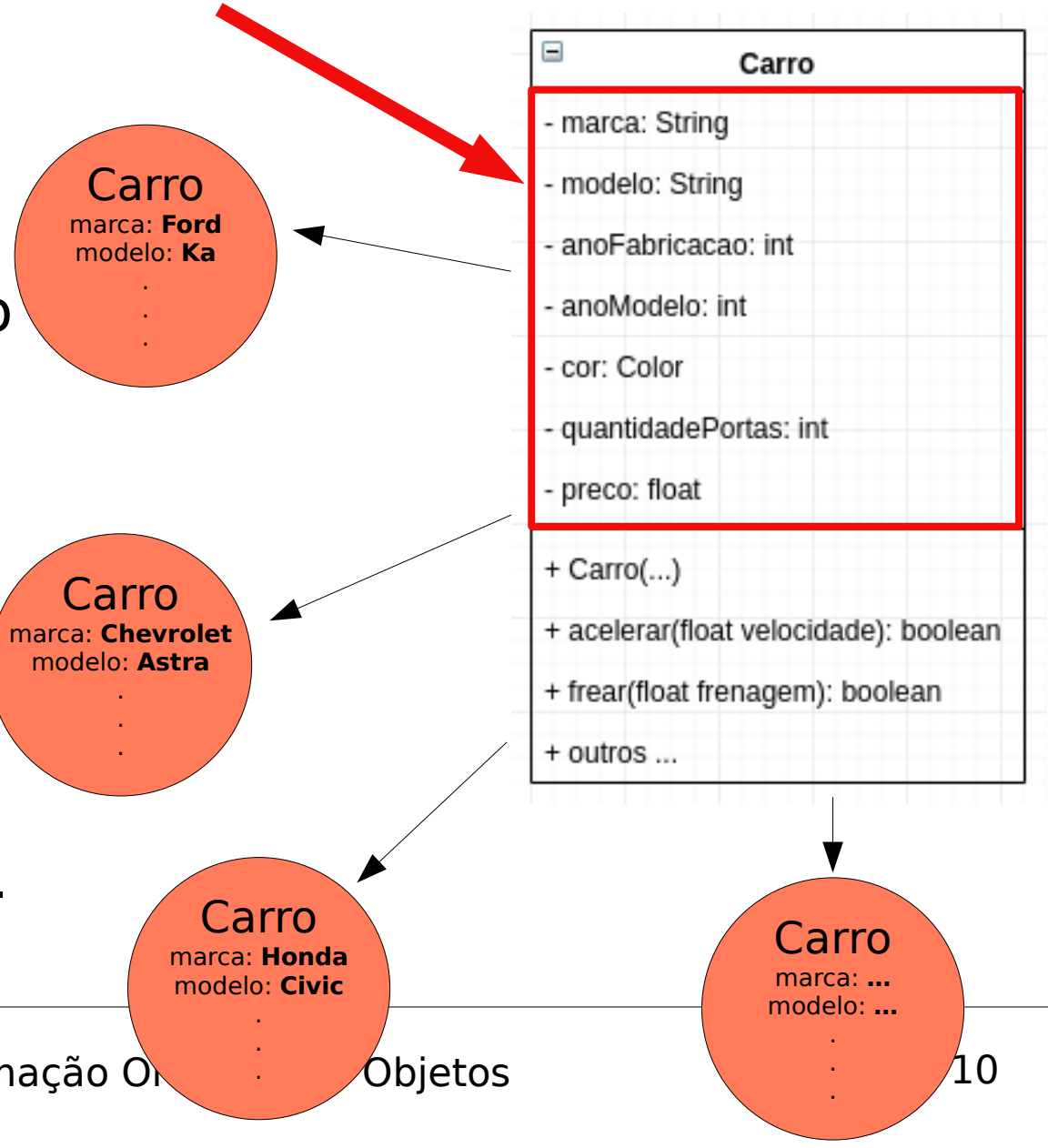


Para “manusear” os objetos instanciados, usa-se os métodos criados: acelerar, frear, virar e etc. Sem a necessidade de se saber como foi feito o código-fonte, basta fazer a chamada do método e pronto.

Reforçando!

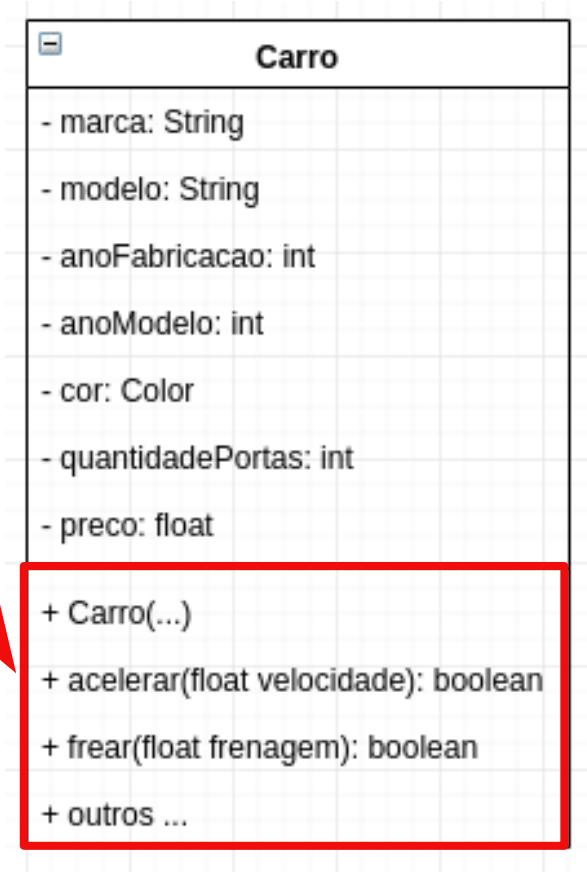
Atributos

- São variáveis que guardarão os valores **individuais** de cada objeto que for gerado de uma determinada classe;
- **atributos** também são chamados de:
  - campos,
  - propriedades ou
  - variáveis de instância.



Métodos

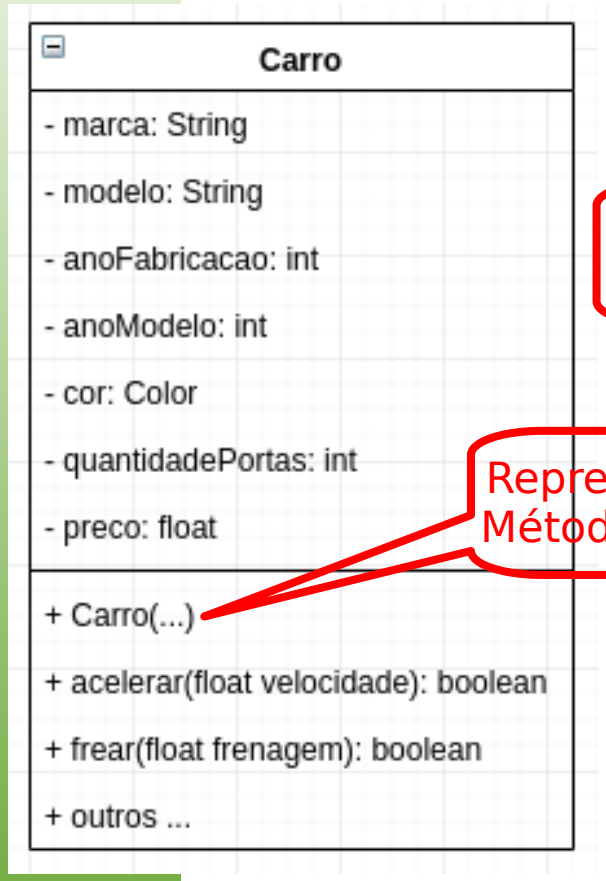
- São “funções” que manipularão, armazenarão e/ou modificarão os valores nos atributos que compõem o objeto;
- em POO funções são chamadas de **métodos**.



Método Construtor

- O método construtor é um método especial que serve para instanciar (inicializar) um objeto e seus atributos;
- O método construtor pode ou não ser declarado explicitamente. Mesmo que não seja, toda classe possui um método construtor implícito que é chamado na instanciação (criação) do objeto;
- O método construtor **possui o mesmo nome da classe**, porém, **não possui tipo de retorno**;
- Podem existir diversos métodos construtores em uma mesma classe, desde que a assinatura de cada qual seja diferente uma da outra.

Método Construtor



Representação do Método Construtor.

```
5  public class Carro {
6      private String marca;
7      private String modelo;
8      private int anoFabricacao;
9      private int anoModelo;
10     private Color cor;
11     private int quantidadePortas;
12     private float preco;
13
14     public Carro() {
15
16     }
17
18     public Carro(String marca, String modelo, int anoFabricacao,
19             int anoModelo, Color cor, int quantidadePortas, float preco) {
20         this.marca = marca;
21         this.modelo = modelo;
22         this.anoFabricacao = anoFabricacao;
23         this.anoModelo = anoModelo;
24         this.cor = cor;
25         this.quantidadePortas = quantidadePortas;
26         this.preco = preco;
27     }
28
29     public String getMarca() {
30         return marca;
31     }
32
33     public void setMarca(String marca) {
34         this.marca = marca;
35     }
36
37     public String getModelo() {
38         return modelo;
39     }
40
41     public void setModelo(String modelo) {
```

Método Construtor com parâmetros.

Método Construtor sem parâmetros.

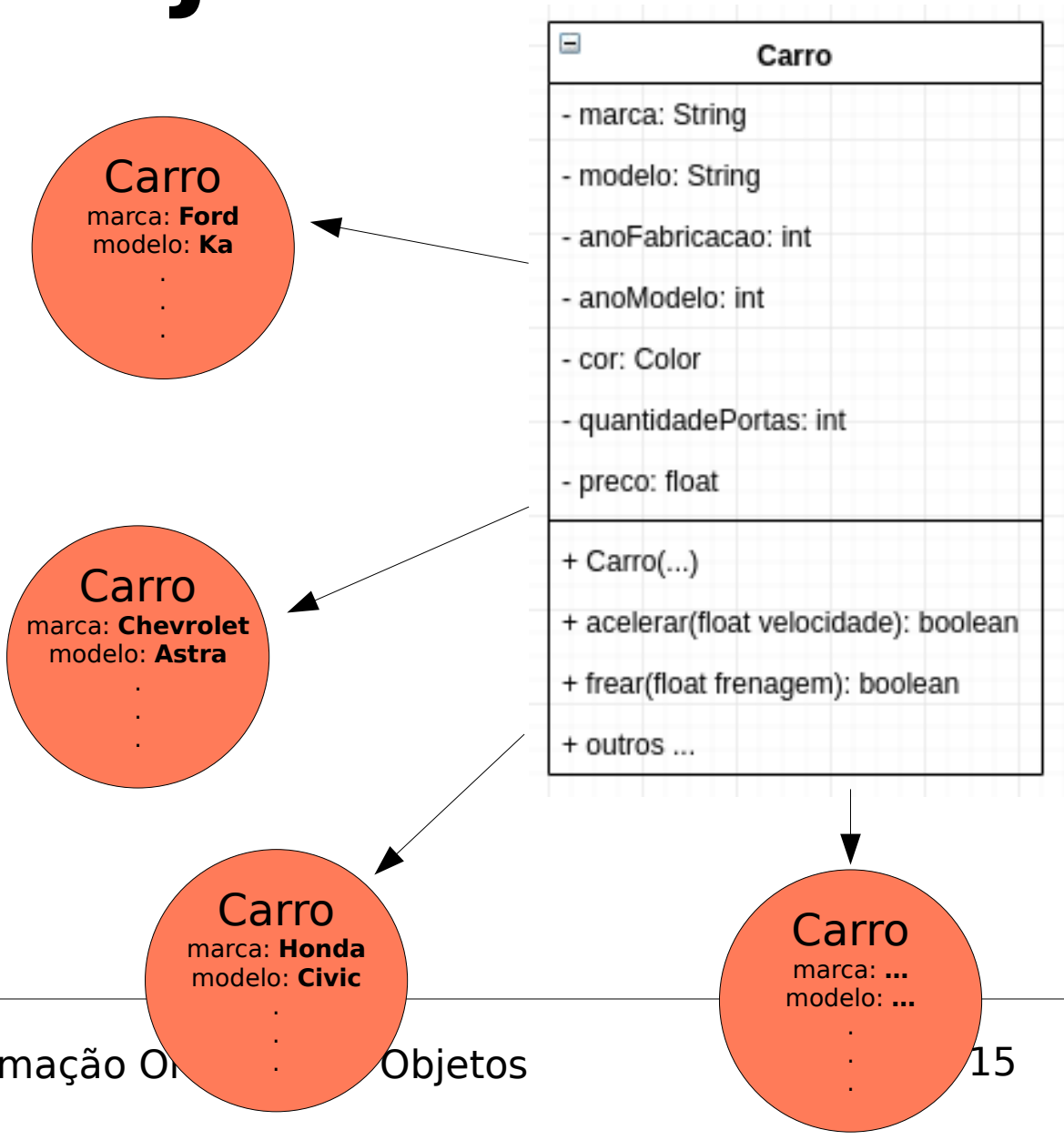
Mensagem

- Ao ato do programador de chamar um método de um objeto no código-fonte dá-se o nome de: “mensagem” ou
- diz-se que o objeto está enviando uma “mensagem”.

- **Isto é puramente conceitual.**

Objeto

- Depois de **criado / instanciado**, o objeto torna-se uma entidade única dentro do programa;
- esta entidade **possui valores individuais** que geralmente são acessíveis via métodos *get* e *set*;
- a interação de um objeto com o programa ao qual pertence se dá por meio de seus métodos.



Classe x Objeto

Classes

Fonte: <https://pt.aliexpress.com/item/32797936162.html>. Acesso em: 23 Out. 2019.

Objetos

Objetos

Objetos

Objetos

Classes

Fonte: <https://pt.aliexpress.com/item/32849196131.htm>. Acesso em: 23 Out. 2019.

Objeto

Tipo Abstrato de Dados (TAD)

TAD é a representação encapsulada de um tipo definido pelos seus atributos e suas operações.

Comparação TAD

Progamação Estruturada (Linguagem C)
X
Programação Orientada a Objetos (Java)

TAD em Linguagem C

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct{
    int matricula;
    char nome[100];
    char curso[50];
} Aluno;

int main()
{
    Aluno paulo, vanessa;

    printf("Digite a matrícula: ");
    scanf("%d",&paulo.matricula);
    __fpurge(stdin);
    printf("Digite o nome: ");
    gets(paulo.nome);
    printf("Digite o curso: ");
    gets(paulo.curso);

    printf("\nMatrícula: %d", paulo.matricula);
    printf("\nNome     : %s", paulo.nome);
    printf("\nCurso    : %s\n\n", paulo.curso);

    return 0;
}
```

TAD

Programa

TAD em Java

TAD

```java
package principal;

public class Aluno {
    private int matricula;
    private String nome;
    private String curso;

    public Aluno() {
    }

    public int getMatricula() {
        return matricula;
    }

    public void setMatricula(int matricula) {
        this.matricula = matricula;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getCurso() {
        return curso;
    }

    public void setCurso(String curso) {
        this.curso = curso;
    }
}
```

Programa

```java
package principal;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Aluno paulo, vanessa;

        String nome;
        int matric;

        paulo = new Aluno();

        System.out.println("Digite a matrícula: ");
        matric = sc.nextInt();
        sc.skip("\n"); // Semelhante ao "fflush" ou "__fpurge"
        paulo.setMatricula(matric);
        System.out.println("Digite o nome: ");
        nome = sc.nextLine();
        paulo.setNome(nome);
        System.out.println("Digite o curso: ");
        paulo.setCurso(sc.nextLine());

        System.out.println("Matrícula: " + paulo.getMatricula());
        System.out.println("Nome     : " + paulo.getNome());
        System.out.println("Curso    : " + paulo.getCurso());

        sc.close();
    }
}
```

Partes Básicas de Um Programa

```
class NomeDaClasse{
    public static void main(String[] args){
        …
        comandos
        …
    }
}
```

Exemplo Criação de Programa

- Se você abrir o aplicativo “Bloco de Notas” (no Windows) ou “GEdit” (GNU/Linux) e digitar o programa:

```
class Hello{
   public static void main(String[] args){
      System.out.println(“Hello world!”);
   }
}
```

- Ele funcionará perfeitamente. Assim que você compilá-lo e executá-lo.

Comandos: Compilação e Execução

- Vamos supor que o arquivo mostrado no *slide* anterior tenha sido salvo com o nome: **hello.java**. Então no "Terminal" (GNU/Linux) ou no "Prompt de Comandos" (Windows):
- para compilação:

  `javac hello.java`

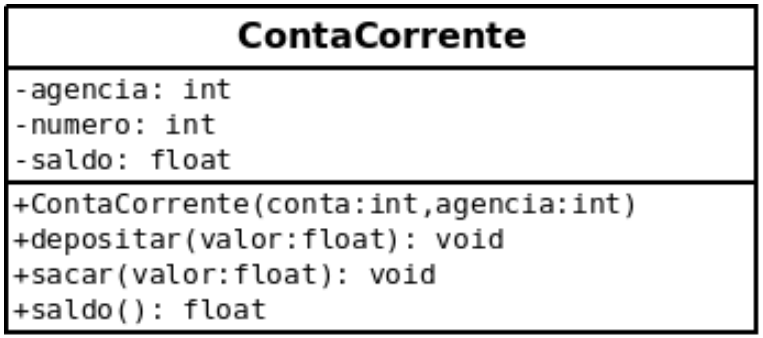
  - será criado o arquivo "hello.class";
- para execução:

  `java hello` (*não colocar hello.class*)

  - será executado o programa.

Outro Exemplo

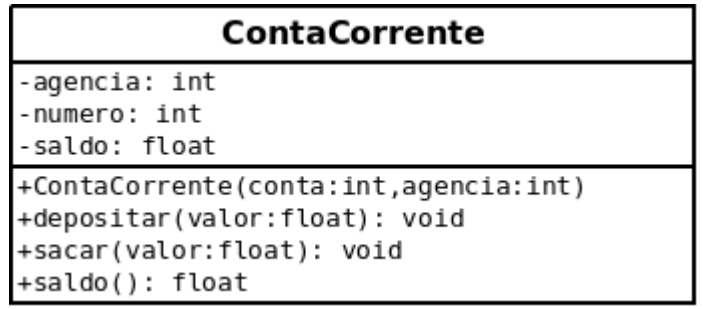
- No contexto de automação bancária, identifica-se, dentre outras, a seguinte classe:
  - Conta Corrente
- Estrutura da classe Conta Corrente (representação UML):



Código da Classe Conta Corrente

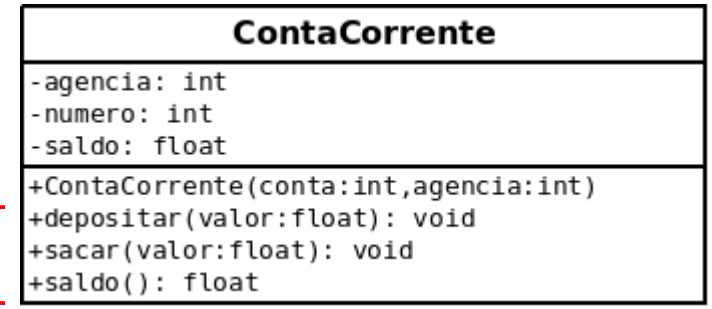
```
public class ContaCorrente {
    private int agencia;
    private int numero;
    private float saldo;

    public ContaCorrente(int conta, int agencia) {
        numero = conta;
        this.agencia = agencia;
        saldo = 0;
    }
```

Método Construtor



Código da Classe Conta Corrente

```
    public void sacar(float valor){
        if((saldo - valor) >= 0)
            saldo = saldo – valor;
        else
            System.out.println(“Saldo insuficiente!”);
    }
    public void depositar(float valor){
        if(valor > 0)
            saldo = saldo + valor;
    }
    public float saldo(){
        return(saldo);
    }
}
```



Observações

- No exemplo anterior, o método **ContaCorrente** (de nome idêntico ao da classe) é denominado **construtor** da classe. O método construtor:
  - só é executado uma única vez, sempre que um objeto da classe é criado;
  - seu objetivo é colocar valores aos atributos do objeto na sua instanciação e
  - não especifica um tipo de retorno, pois não há.
- Em Java objetos são instanciados (criados) utilizando-se o comando ***new***.

Criação de Um Objeto

```
Exemplo;
ContaCorrente minhaConta;
minhaConta = new ContaCorrente(12345, 236);
```

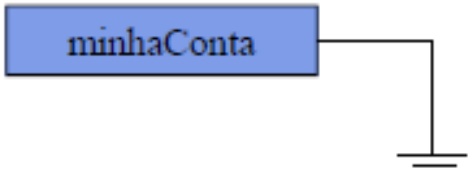
Tipo da Variável (classe)

Variável

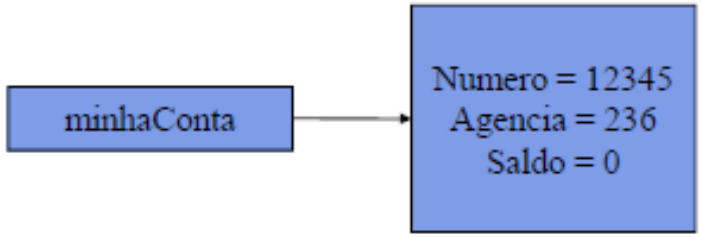
Método Construtor

Comando para Criação de Objetos

- Na primeira linha do exemplo, é criada uma área na memória, que é uma referência para um objeto da classe **ContaCorrente**;



- Na segunda linha é instanciado (criado), um objeto da classe **ContaCorrente,** e este é atribuído a variável **minhaConta**;



Comunicação entre Objetos

- Programas orientados a objetos são constituídos por objetos que trocam mensagens entre si;
- o envio de uma **mensagem** a um objeto corresponde a invocar (chamar) um **método** de tal objeto;
- em:

```
minhaConta.depositar(540.00);
```

- o método **depositar** do objeto **minhaConta** é invocado. Em outras palavras, é enviada uma **mensagem** para o objeto **minhaConta** para que este realize a operação depositar.

Para Casa

- Pegar o livro na biblioteca:

  DEITEL, P.; DEITEL, H. **Java Como Programar**. 8. ed. São Paulo: Person, 2010

  - e ler o capítulo 3 para a próxima aula.

Exercícios

- Para os sistemas a seguir, identifique 3 classes, seus respectivos métodos e atributos:
  - Sistema de Vídeo Locadora;
  - Sistema de Gestão de Biblioteca;
  - Agenda de Compromissos.

Bibliografia

- DEITEL, H. M.; DEITEL, P. J. **Java Como Programar**; tradução Edson Furmankiewicz; revisão técnica Fábio Lucchini. 6. ed., São Paulo: Pearson, 2005.
- FERREIRA, Kecia Aline Marques. *Slides* da disciplina de Programação de Computadores II. CEFET-MG, 2009.
- GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projeto: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 2000.
- HUBBARD, John R. **Teoria e Problemas da Programação com Java**. 2. ed. Porto Alegre: Bookman, 2006.
- MIZRAHI, Victorine Viviane. **Treinamento em Linguagem C++ - Módulo 2**. 2. ed., São Paulo: Pearson, 2006.