Java
Fundamentos e
Orientação a
Objetos

Java – Aula 5

Java
Fundamentos e
Orientação a
Objetos

Programação Orientada a Objetos

Elementos Básicos do Modelo OO

Classes

A POO tem como princípio básico categorizar e concentrar tudo relacionado a determinado item num único local, chamado de classe.

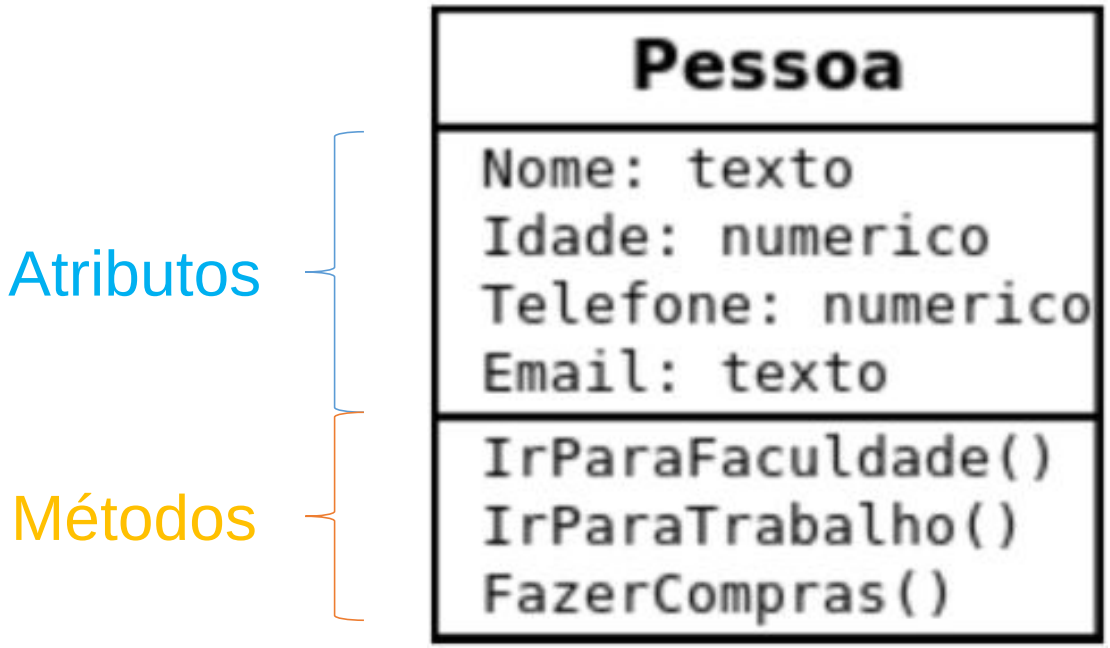
A classe, então, concentra todas as características de uma entidade qualquer e os chama de atributos.

Tudo o que essa entidade pode realizar, é chamado de método.

Elementos Básicos do Modelo OO

Classe “Pessoa”

São conjuntos de objetos com as mesmas características (atributos e métodos).



Elementos Básicos do Modelo OO

Objeto

É o principal elemento do Modelo Orientado a Objeto. Será visto como uma cópia de uma “classe”.

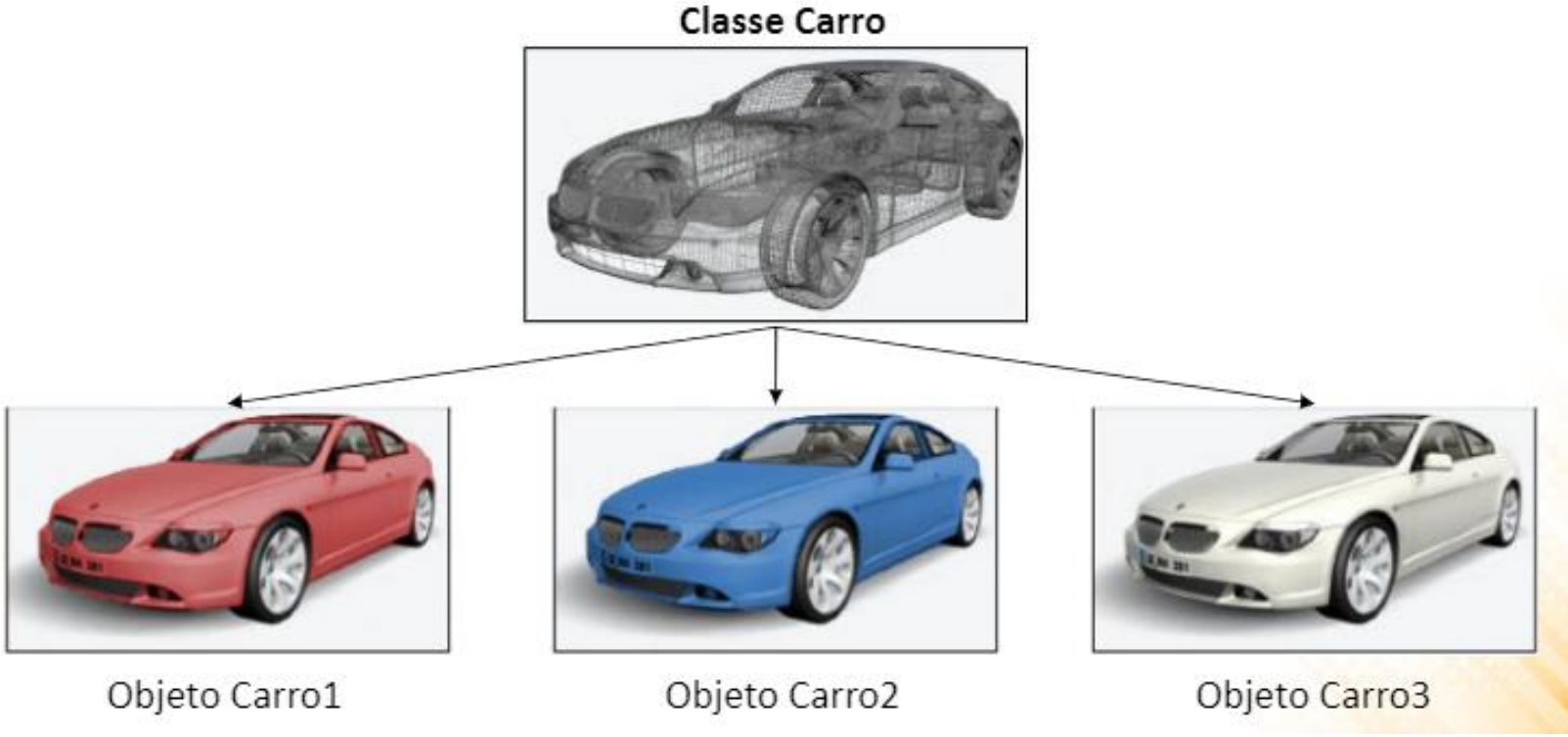
Os objetos representam as “coisas” a serem modeladas do mundo real. Um objeto pode ser algo concreto como um carro, um aluno ou algo abstrato como uma disciplina.

Cada objeto possui os dados inerentes a ele, como por exemplo, Nome (José) e Matrícula (20110127220l) de um aluno e possui as operações que ele executa, como incluir novo aluno ou alterar dados de um aluno existente.

Programação Java

Objetos

Trata-se de uma cópia de uma classe



Programação Java

Código em java

```java
public class Carro {

    String modelo;
    String chassi;  // atributos da classe Carro
    int qtdPortas;

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public String getChassi() {
        return chassi;
    }

    public void setChassi(String chassi) {
        this.chassi = chassi;
    }

    public int getQtdPortas() {
        return qtdPortas;
    }

    public void setQtdPortas(int qtdPortas) {
        this.qtdPortas = qtdPortas;
    }

    public void acelerar() {

        // código do método acelerar aqui
    }

    public void frear() {

        // código do método frear aqui
    }

}
```

Programação Java

Código em java para criar os Objetos

```
Carro carro1 = new Carro();
carro1.modelo = "Gol";
carro1.qtdPortas = 4;
carro1.chassi = "9c2xx250xxx003931";
carro1.acelerar();
carro1.frear();

Carro carro2 = new Carro();
carro2.modelo = "Fusca";
carro2.qtdPortas = 2;
carro2.chassi = "7c2yy255xxy002225";
carro2.acelerar();
carro2.frear();
```

```
Carro carro3 = new Carro();
carro3.modelo = "Fiat uno";
carro3.qtdPortas = 2;
carro3.chassi = "2c2mm255tyr001111";
carro3.acelerar();
carro3.frear();
```

Teremos apenas uma classe carro (molde). Cada objeto que criarmos do tipo carro terá seus valores próprios e ações, ou seja, cada carro terá o seu próprio modelo, quantidade de portas e chassi. Além dos seus métodos de acelerar e frear.

Programação Java

Atividade Prática

Criar uma classe “Carro”

1. Definir os atributos da classe Carro;
    - 1.1 modelo;
    - 1.2 chassi;
    - 1.3 qtdPortas;
2. Definir o método acelerar()

Criar uma classe “CarrosTipos”

1. Criar um objeto que represente cada carro (são 3 objetos)
2. Preencher cada objeto com informações pertinentes aos atributos;
3. Chamar o método Acelerar();

Programação Java

Métodos

São funções para realizar tarefas específicas e pode ser chamado por qualquer outro método ou classe, para realizar uma função num determinado contexto.

Os métodos possuem algumas características como:

- Podem ou não retornar um valor;
- Podem ou não aceitar argumentos (valores de entrada e de saída dos métodos); e
- Após encerrar sua execução, o método retorna ou fluxo de controle do programa para quem o chamou.

Programação Java

Palavra “This”

É uma referência para o próprio objeto.

Usos comuns:

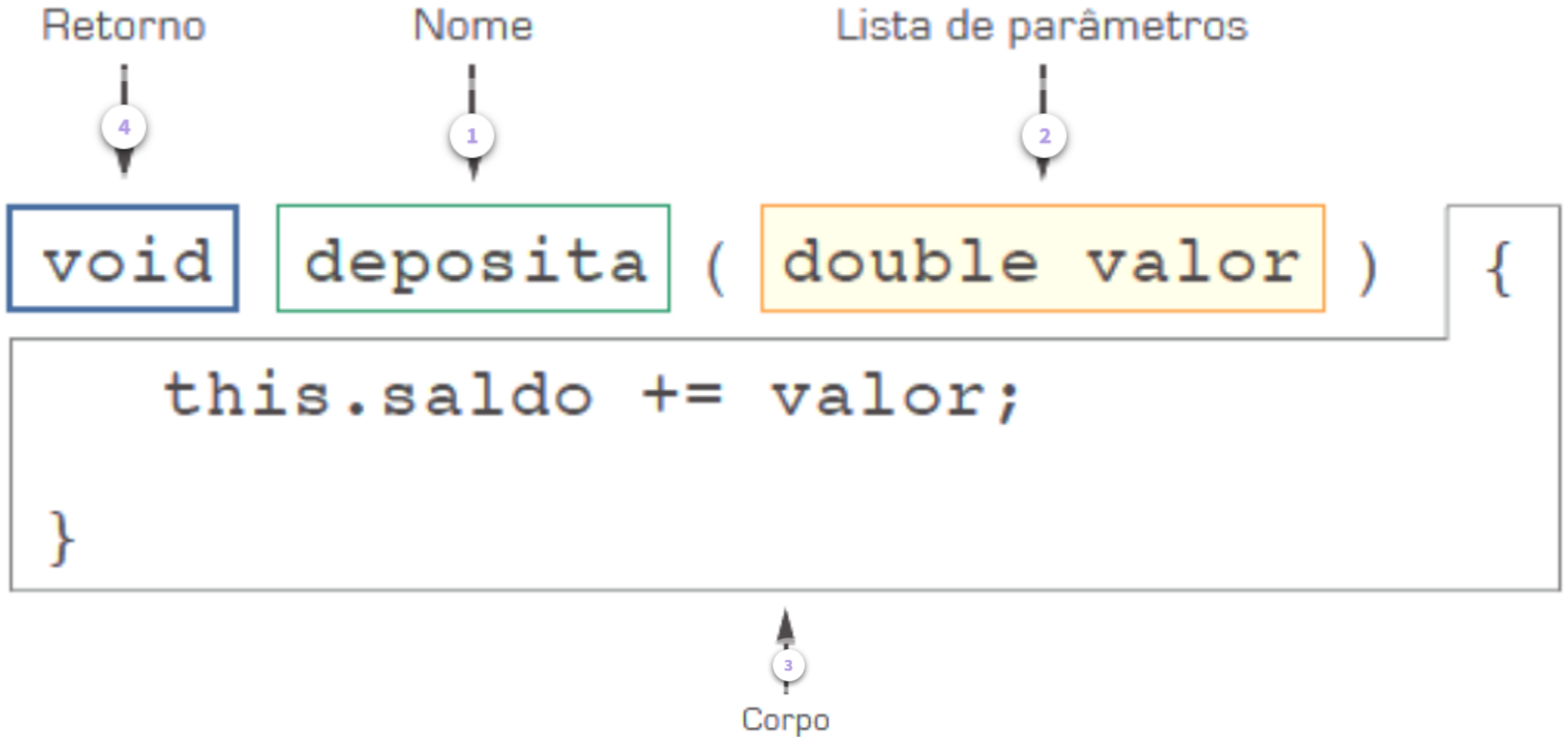
- Diferenciar atributos de variáveis locais.
- Passar o próprio objeto como argumento na chamada de um método ou construtor.

Podemos usar uma palavra reservada isto para mostrar que esse é um atributo da classe.

Ainda que seja opcional, é sempre uma boa prática usar o this em atributos para evitar futuros problemas de ambiguidade e para deixar claro que este é um atributo da classe, e não uma simples variável.

Programação Java

Podemos dividir um método em quatro partes



Programação Java

Nesse exemplo em Java o método de depósito, recebe um valor como parâmetro e soma mais o valor do atributo saldo, porém não retorna nenhum valor para o programa que chamou.

```
void deposita(double valor){
        this.saldo += valor ;
}
```

Programação Java

Nesse exemplo em Java o método consultaSaldoDisponivel, devolve para o programa principal o valor do atributo saldo do tipo double, desta forma quem chamou o método pode pegar o valor do saldo e usar para outras operações.

```
double consultaSaldoDisponivel(){
    return this.saldo;
}
```

Programação Java

Atividade Prática (complementação da atividade do carro)

Classe Carro

1. **Definir o método acelerar()**
    - **1.1 no método ter uma estrutura de decisão para verificar:**
        - **1.1.1 Se a quantidade de portas é igual a 2, se for mensagem "carro não pode passar do 60 Km!".**
        - **1.1.2 Se a quantidade de portas é igual a 4, se for mensagem "carro não pode passar do 110 Km!".**
        - **1.1.3 Se a quantidade de portas é igual a 5, se for mensagem "carro não pode passar do 140 Km!".**

Criar uma nova classe "CarrosTipos"

1. **Criar um objeto que represente cada carro (são 3 objetos)**
2. **Preencher cada objeto com informações pertinentes aos atributos;**
3. **Chamar o método acelerar();**

Programação Java

Atividade Prática

1. Criar uma classe “Caixa”;
2. Definir um atributo de nome saldo;
    2.1 Atribuir 1000 para o atributo saldo;
3. Criar um método sacar:
    3.1 Fazer um código para que o saldo seja atualizado conforme valor recebido pelo parâmetro. Fazer subtração de valores;
4. Criar um método depositar:
    4.1 Fazer um código para que o saldo seja atualizado conforme valor recebido pelo parâmetro. Fazer soma de valores;
5. Criar um método exibirsaldo;
6. Dentro do método principal **public static void main (String [] args)**
    6.1 Criar um objeto do tipo Caixa
        6.1.1 Chamar o método depositar e passar um valor de 500 por parâmetro;
        6.1.2 Chamar o método exibirSaldo () para mostrar o valor atualizado do atributo saldo que passa a ser de R$1.500,000.
        6.1.3 Chamar o método sacar e passar um valor de 300 por parâmetro;
        6.1.2 Chamar o método exibirSaldo () para mostrar o valor atualizado do atributo saldo que passa a ser de R$700,000.

Programação Java

Gabarito da Atividade Prática

Código em java

```java
public class Caixa {

    public double saldo = 1000;

    void sacar(double valor) {
        this.saldo = saldo - valor;
    }

    void depositar(double valor) {
        this.saldo = saldo + valor;
    }

    double exibirSaldo() {
        return this.saldo;
    }
}
```

Trecho abaixo da criação da classe Caixa

```java
Caixa caixa = new Caixa(); // criando objeto da classe
caixa.depositar(500); // chamando o método depositar
System.out.println(caixa.exibirSaldo());
```

Pilares do Modelo OO

Encapsulamento

O encapsulamento é uma das principais e mais importantes técnicas do Programação Orientada a Objeto.

Como as propriedades e os métodos ficam encapsulados na classe, elas podem ficar protegidos e seguros, sem que o código fique visível para outros programadores.

O encapsulamento é uma técnica para minimizar a interdependência entre as classes, pois apenas os métodos da respectiva classe podem alterar seus dados (atributos), facilitando a identificação de erros e a alteração dos programas.

Pilares do Modelo OO

Encapsulamento

Encapsular os dados de uma aplicação significa **evitar que estes sofram acessos indevidos**. Para isso, é criada uma estrutura que contém métodos que podem ser utilizados por qualquer outra classe, sem causar inconsistências no desenvolvimento de um código.

Na prática, isso é feito por meio de dois métodos: **os getters e os setters**.

Getters (get) → Tem por objetivo retornar o valor que lhe foi pedido, mas de forma a não prejudicar a integridade do dado em si. Exemplo: getNome();

Setters (set) → Recebe como argumento uma informação, que pode ser qualquer tipo de dados suportados pela linguagem. Dessa forma, não haverá o risco de ocorrerem acessos indevidos. Exemplo: setIdade();

Pilares do Modelo OO

Visibilidade do Encapsulamento

Público (public)

Os atributos e métodos são buscados por qualquer outra classe, podendo ser acessado e promovido.

Protegido (protected)

Os atributos e métodos são buscados apenas por objetos da própria classe ou de suas classes filhas.

Privado (private)

Os atributos e métodos são buscados apenas pela própria classe, como se fosse reservado para o uso interno.