



TÉCNICAS DE PROGRAMAÇÃO I

Fatec – Votorantim

Prof.º Me. Rodrigo de Paula Diver



Emenda do Curso

■ Programação Orientada a Objeto.

- *O que é POO ?*
- *Conceitos: Classes, Métodos, Atributos, Instâncias, etc.*
- *Conceitos de MVC (Model-View-Controller).*

■ Linguagem JAVA.

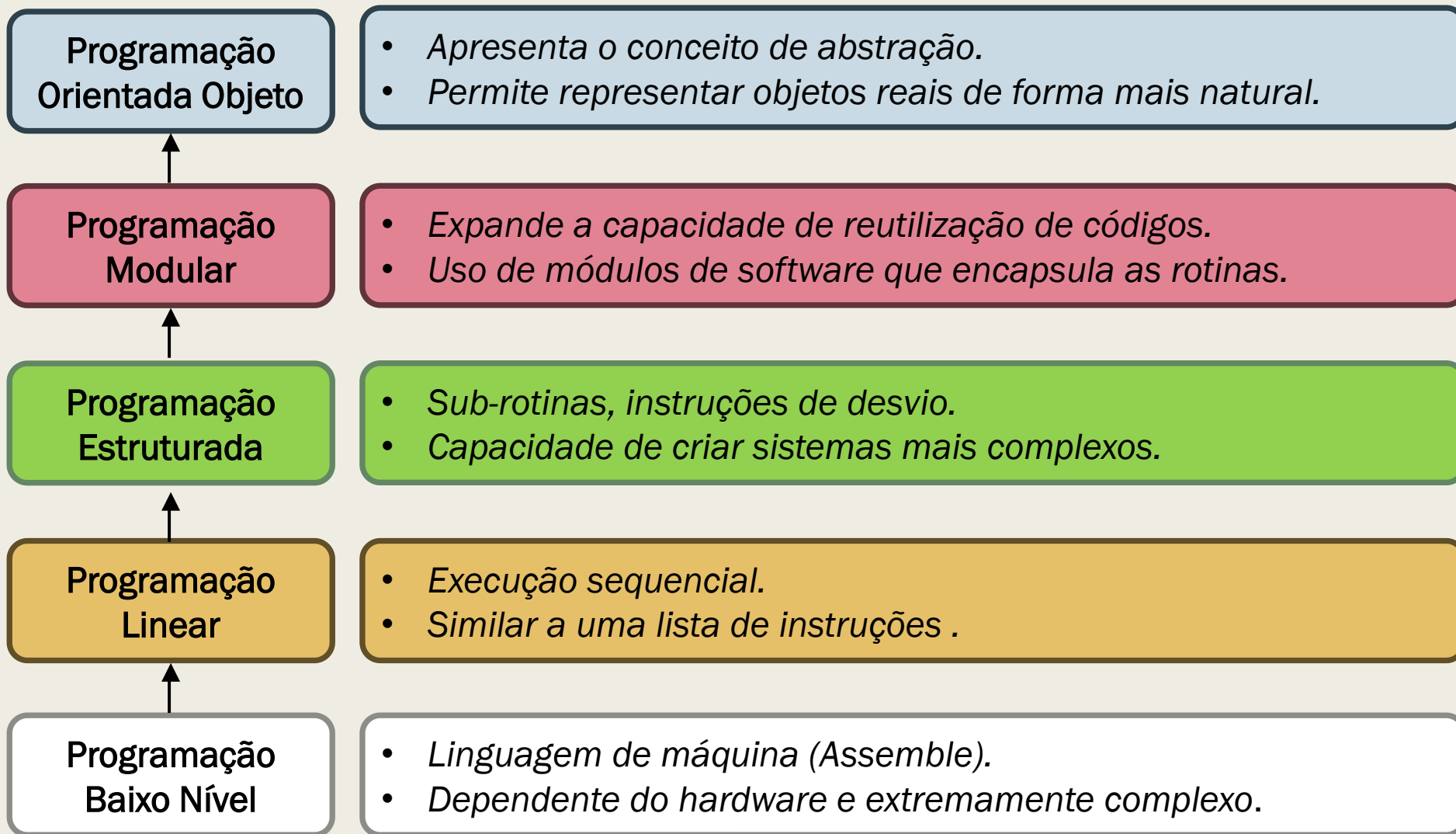
- *O que é JAVA?*
- *Declaração de variáveis, classes, métodos, etc.*
- *Padronização de código e documentação.*
- *Introdução a interfaces gráficas.*
- *Tratamento de Exceções.*



Avaliações

- **Média (P1 + P2 + EA + PF)**
- **P1 e P2** – Prova Teórica
- EA = Exercícios em Sala.
- PF = Projeto Final.

Origem dos Paradigmas de Programação



Paradigmas de Programação

Programação Estruturada

- É um paradigma de programação com ênfase em sequência, decisão e, iteração.
- Normalmente é formado por um único bloco de códigos com sub-rotinas, laços de repetição e condicionais.

Programação Orientada a Objeto

- É um modelo de programação onde objetos reais são representados por classes.
- Com as classes possuindo atributos que definem características dos objetos, como cor, tamanho ou idade.
- E métodos que definem ações ou comportamentos dos objetos, como andar, correr ou calcular.

Paradigmas de Programação

Programação Estrutura

Um grande bloco de instruções:

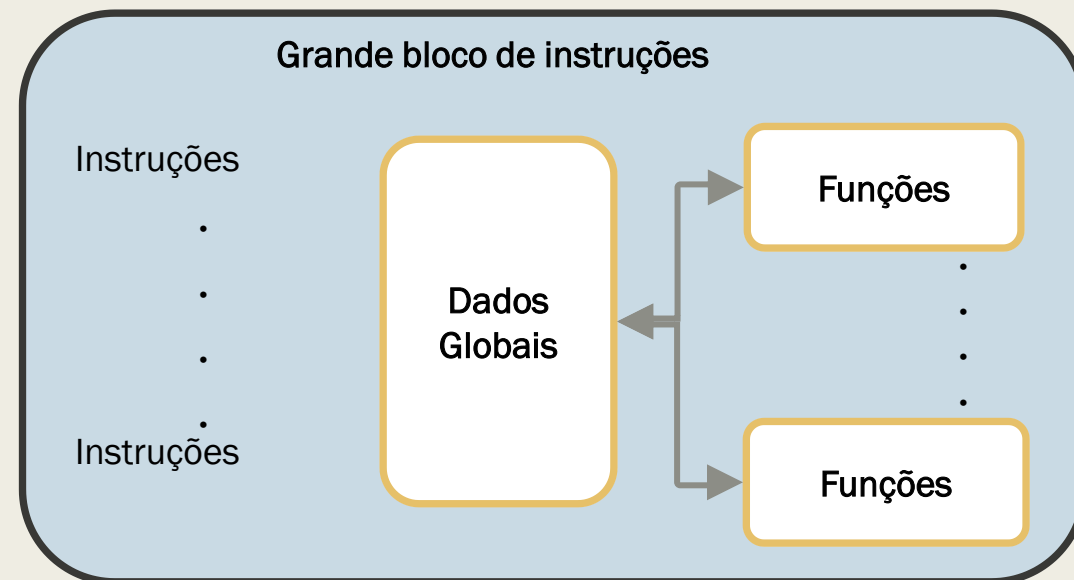
Estruturas de Decisão, Interação, Condicionais.

Execução sequenciais (mais rápidas).

Grande domínio do Hardware.

Linguagem de baixo nível.

Adequado para aplicação embarcadas.



Paradigmas de Programação

Programação Orientada a Objeto

Divisão por Classes

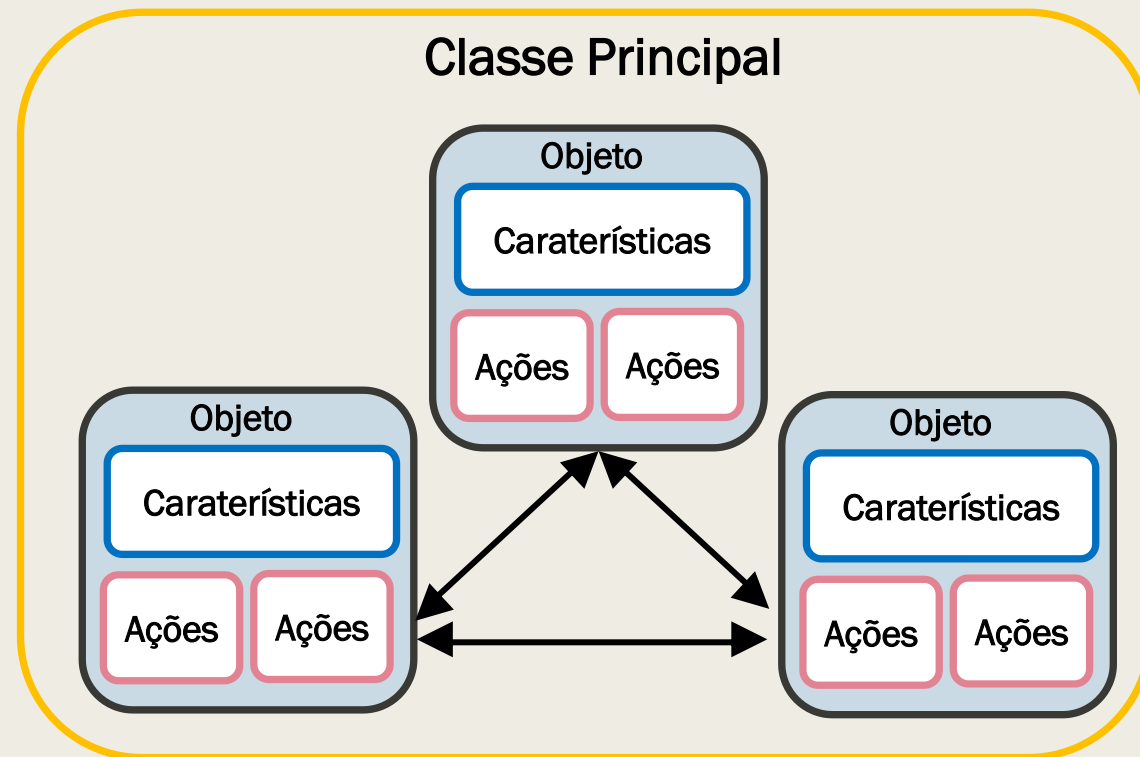
- Separação por camadas

Conceito de Objetos

- Atributos e Métodos.

Reaproveitamento de códigos.

- Abstração.
- Encapsulamento.
- Herança.
- Polimorfismo.



Programação Orientada a Objetos

Características

Confiável.

- Isolamento entre as partes gera um software seguro
- Alterações em parte do código não afeta o todo.

Escalável.

- A separação do todo em partes menores, com funcionalidades bem definidas permite o desenvolvimento em paralelo das diversas partes.

Reutilizável.

- O baixo acoplamento entre as classes permite a reutilização de classes inteiras sem alterações.

Manutenção.

- É possível substituir partes inteiras sem a necessidade de validar todo o software

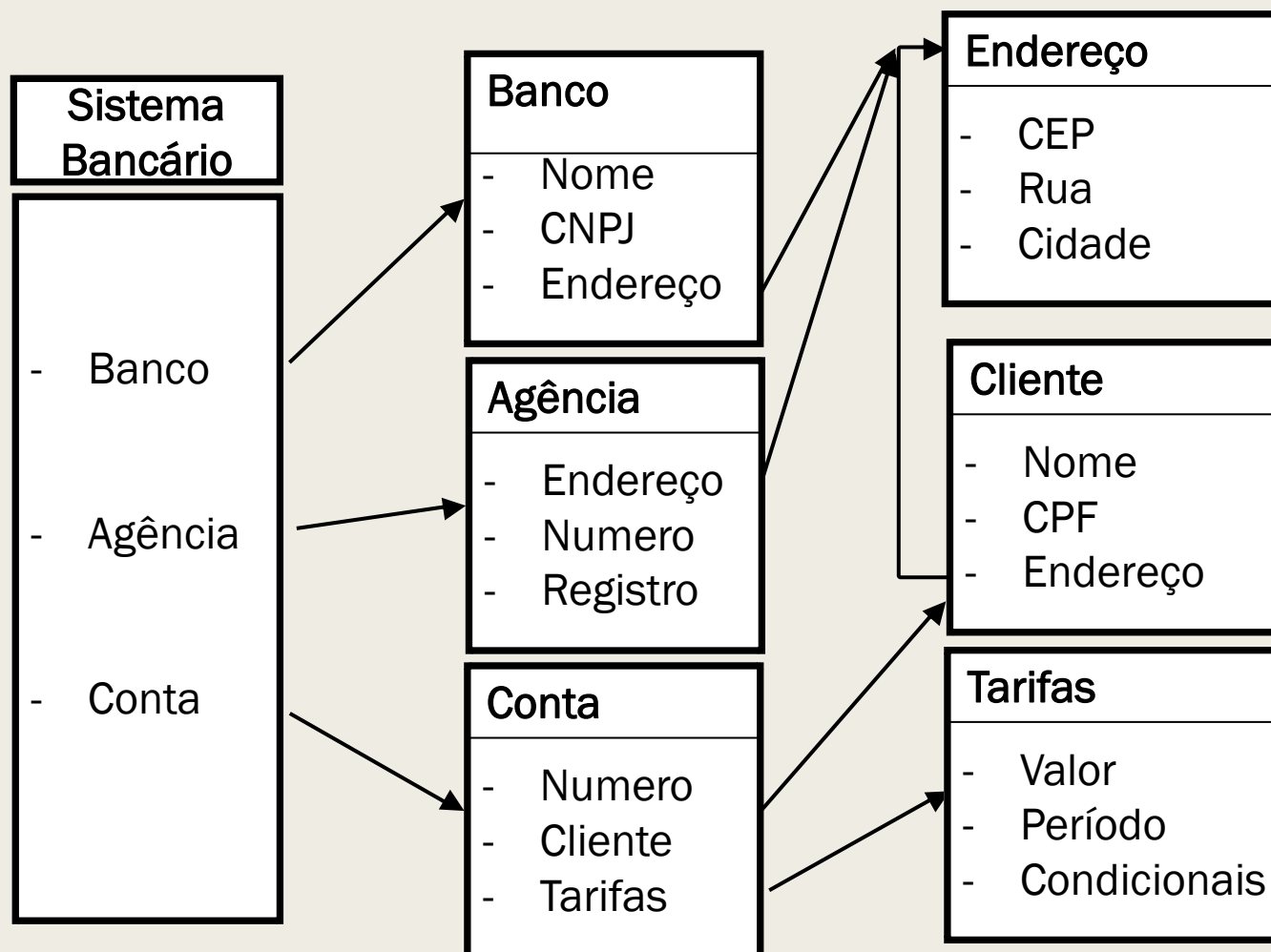
Extensível

- É possível ampliar as funcionalidades facilmente.

Abstração

- Simplifica o aprendizado, pois evita a necessidade de conhecer em detalhe todas as partes do software.

Exemplo – Sistema Bancário



Funcionalidades

- Abrir Conta.
- Fechar Conta.
- Realizar Deposito.
- Realizar Saques.
- Consultar Saldo.
- Calcular Saldo.
- Autenticar acessos.
- Etc

Exemplo – Sistema Bancário

Preciso transferir R\$ 10,00 da conta A para conta B:

- I. **Buscar os dados das contas.**
 - I. Quem poderá ter acesso as variáveis ?
 - II. Como controlar o acesso aos dados ?
- II. **Verifica se conta possui saldo.**
 - I. Necessário aplicar regras de validação.
 - II. A onde armazena essas regras ?
- III. **A operação é válida ?**
 - I. Sim: desconta o valor de A e soma em B.
 - II. Não: retorna um aviso de erro.
 - III. Se ocorrer um erro durante a execução como controlar o estorno do dinheiro ?



Exemplo – Sistema Bancário

Preciso transferir R\$ 10,00 da conta A para conta B:

Programação Estruturada

// Declaração de variáveis Globais.

Lista contas[index];

Contas{

- Numero;
- Lista Clientes;
- Saldo; }

Clientes{

- Nome;
- CPF;
- Endereço; }

contas = buscaDados (A,B);

If contas[1,3] > 10,00 then

contas[1,3] = contas[1,3] - 10;

contas[2,3] = contas[2,3] + 10;

Else

Mensagem (Saldo insuficiente!);

Lista contas [.....]			
Index	(1) conta	(2) cliente	(3) saldo
1	A	João	100
2	B	Maria	20

Programação Orientada a Objeto

Classe Principal

// Instancia objetos.

conta A = new conta (cpf_cliente);

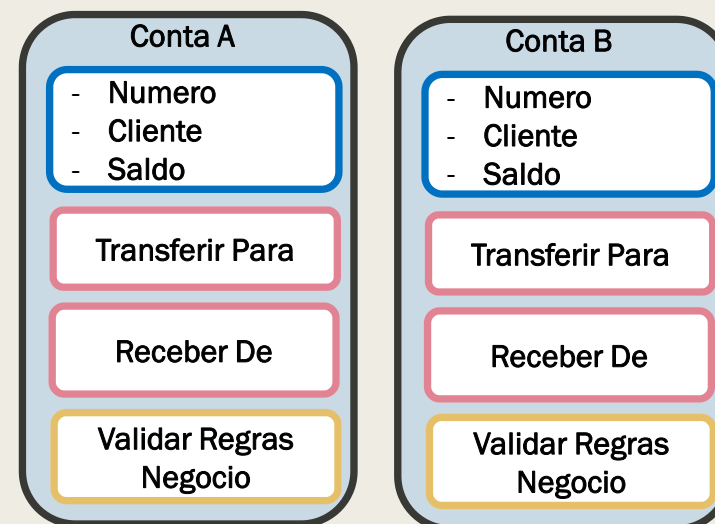
conta B = new conta (cpf_cliente);

If (A.transferirPara (B, 10) = valida)

Mensagem (“Valor transferido!”);

Else

Mensagem (“Saldo insuficiente !”);



O que é um objeto?

“Coisa material ou abstrata que pode ser percebida pelos sentidos e descrita por meio das suas características, comportamento e estado atual.”

“Atributos são as características ou estados inerentes aos objetos.”

“Métodos são os comportamentos ou funções inerentes aos objetos.”



Atributos: Cor, Peso, Potência... Etc

Métodos: Acelerar, Frear, Estacionar... Etc.



Atributos: Nome, Idade, CPF... Etc

Métodos: Falar, Pular, Correr, ... Etc.







Atributos: Nome, Tipo, Chave.... Etc

Métodos: Abrir, Fechar, Vibrar, ... Etc.

O que é um objeto?

Uma aula é um objeto ?

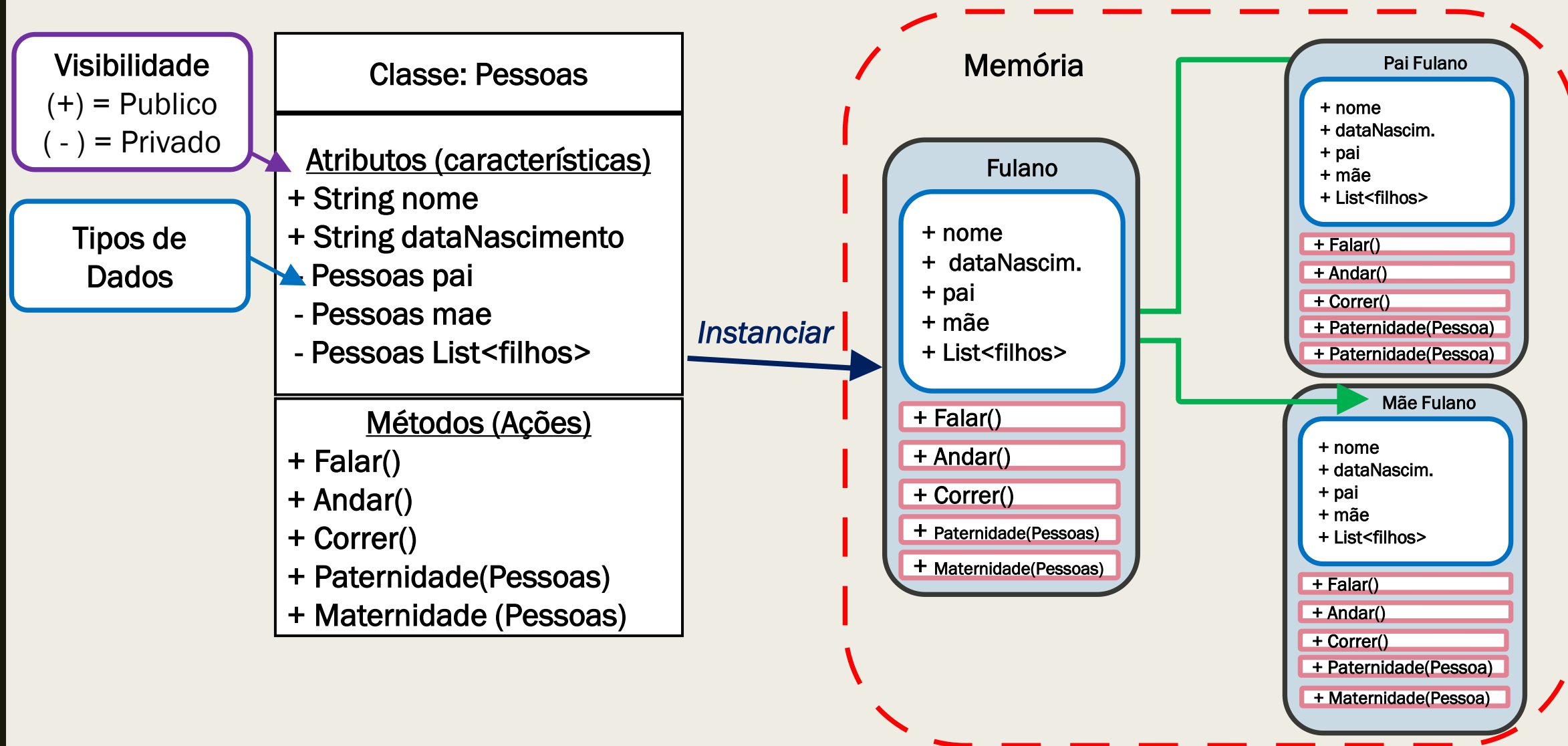
Quais os atributos ou características de uma aula?

- *Nome.*  *String nome;*
- *Tempo de Duração.*  *double duração;*
- *Professor.*  *Funcionário professor;*
- *Alunos.*  *Aluno aluno[];*

Quais os métodos ou comportamentos de uma aula?

- *Inicio.*  *Método inicioAula(String dataAtual);*
- *Fim.*  *Método fimAula(String dataAtual);*
- *Estado.*  *Método estadoAtual(String status)*

Classes, Atributos e Métodos



Linguagem JAVA



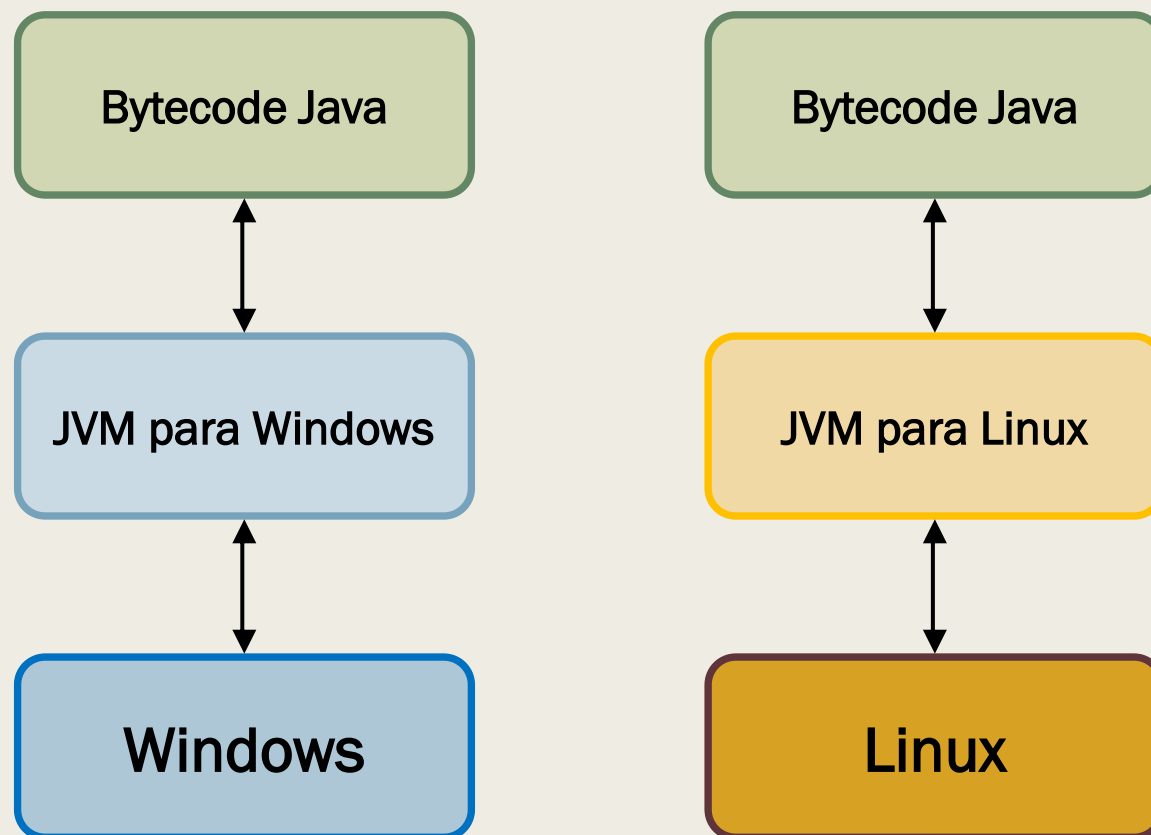
- O Java como linguagem de programação nasceu em 1991 pelas mãos de um pesquisador da Sun Microsystems, James Gosling.
- A ideia original do projeto era desenvolver uma linguagem de programação que fosse independente de plataforma e que pudesse ser executada em diferentes dispositivos eletrônicos, como geladeiras, televisões ou fogões.
- Inicialmente, o projeto não foi prioridade para a Sun, já que alguns executivos acharam absurda a ideia de ter uma linguagem que pudesse ser processada numa televisão. Com a chegada da TV digital interativa, podemos ver que não estavam tão errados assim...
- Sendo uma linguagem de programação orientada a objetos, robusta, elegante, podendo ser utilizada nos mais diversos ambientes.

Máquina Virtual Java (JVM)

- A forma de execução da linguagem Java é baseada na interpretação por meio de uma máquina virtual, a **Java Virtual Machine (JVM)**. Ela proporciona um ambiente de aplicação completo para execução de aplicativos Java.
- **Java Virtual Machine (JVM)**
 - *Imagine uma máquina criada por meio de um software emulador dentro de uma máquina real, como se fosse um minicomputador dentro do seu próprio computador, e que possui gerenciamento de memória, cache, threads e outros processos de um computador com seu sistema operacional.*
- A compilação de um código fonte Java é o **bytecode**, que é uma linguagem de máquina inteligível para a **JVM**.
- O **bytecode** é independente de hardware; assim, basta o computador ou o dispositivo eletrônico ter o interpretador adequado (a **JVM**) que poderá executar um programa Java.

Máquina Virtual Java (JVM)

- O conceito de máquina virtual garante a independência do código em relação ao sistema operacional.
- E aumenta a segurança da aplicação, principalmente em servidores.
- O Bytecode é gerado por um compilador Java, como o **javac**.
- A **JVM** compila dinamicamente durante a execução do código, podendo aplicar estratégias de otimização.



Exercício

Classes, Atributos e Métodos



Exercício Aula 1

Considere uma família formada por Pai, Mãe e 4 Irmãos. Monte uma única classe chamada “Pessoa” com as características básicas de uma pessoa e que possa responder as seguintes perguntas:

- Quantos filhos a pessoa possui ?
- Qual o nome desses filhos ?
- Quantos irmãos a pessoa possui ?
- Qual o nome desses irmãos ?

Bibliografia

- **Java Básico e Orientação a Objeto (livro);**

- Clayton Escouper das Chagas; Cássia Blondet Baruque; Lúcia Blondet Baruque.
- Fundação CECIERJ, 2010: <https://canal.cecierj.edu.br/012016/d7d8367338445d5a49b4d5a49f6ad2b9.pdf>

- **Java e Orientação a Objetos (apostila);**

- Caelum, 2023: <https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf>

- **Os 4 pilares da Programação Orientada a Objetos (artigo);**

- DevMedia, 2023: <https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>

- **Programação Orientada a Objetos e Programação Estruturada (artigo);**

- DevMedia, 2023: <https://www.devmedia.com.br/programacao-orientada-a-objetos-e-programacao-estruturada/32813>



Aula 2 – Métodos Especiais

- **Modificadores de Acesso.**

- *Tipos de modificadores.*

- **Métodos Construtores.**

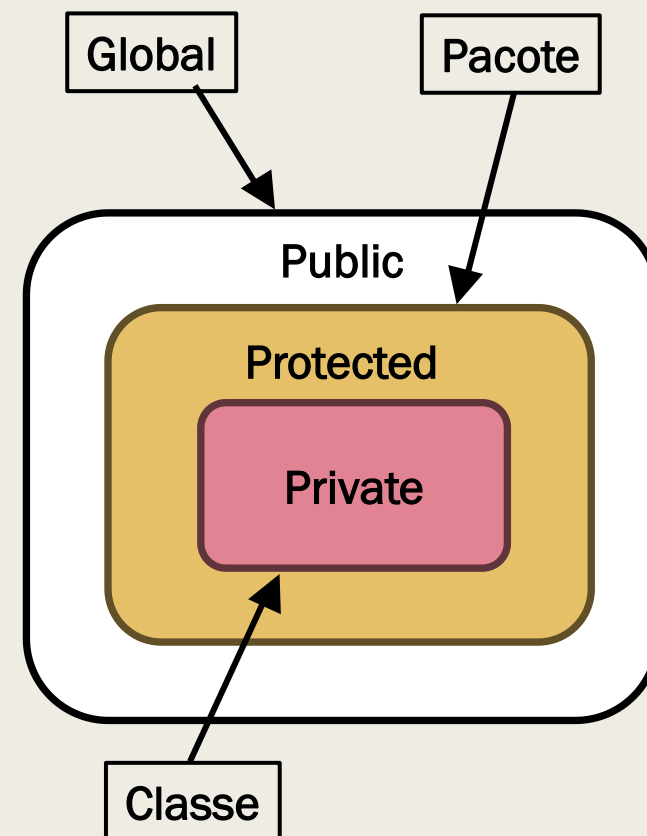
- *Declaração de Métodos Construtores.*
- *Sobrecarga em Métodos Construtores.*

- **Métodos Gettter e Setter.**

- *Encapsulamento de Atributos.*
- *Métodos Getters.*
- *Métodos Setters.*

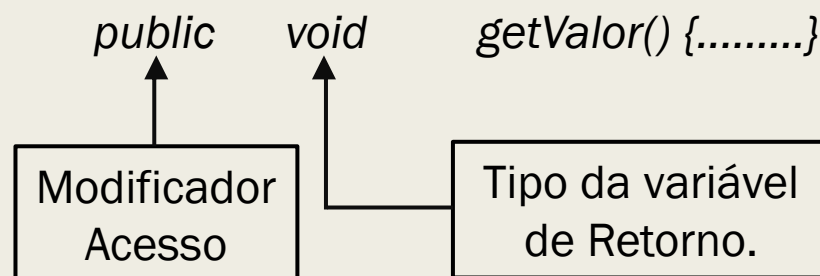
Modificadores de Acesso em Java

- Os modificadores de acesso são palavras-chave na linguagem Java, que definem a visibilidade que determinada classe ou membro terá diante das outras.
- Existem quatro modificadores de acesso em Java:
 - Public:** os elementos da classe são acessíveis tanto pela própria classe quanto por qualquer classe que tente acessá-los.
 - Private:** os elementos da classe são acessíveis somente pela própria classe que os definiu.
 - Protected:** os elementos da classe são acessíveis somente por classes no mesmo pacote da classe dos elementos.
 - Default:** os elementos da classe são acessíveis somente pelos métodos internos da classe e das suas subclasses. É acionado quando não se escreve nenhum dos outros modificadores de acesso antes das definições de classes, atributos ou métodos.

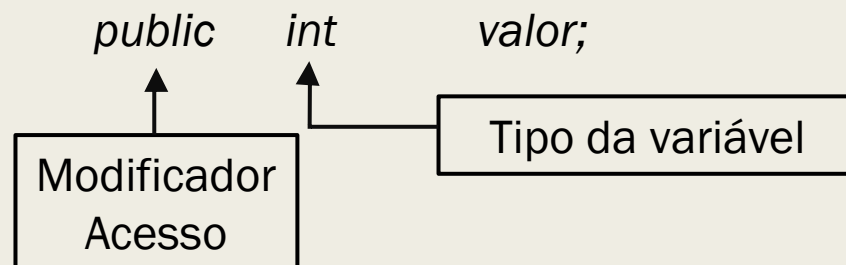


Modificadores de Acesso em Java.

- Declaração de Métodos:



- Declaração de Variáveis:



Tipos primitivos de dados	
boolean	Valores lógicos (true / false)
char	Um único caractere Unicode
byte	Número inteiro de 8 bits
short	Número inteiro de 16 bits.
int	Número inteiro de 32 bits
long	Número inteiro de 64 bits
float	Números com ponto flutuante de 32 bits
double	Números com ponto flutuante de 64 bits

Construtores

- Quando utilizamos a palavra-chave **new** para instanciar um objeto, o construtor da classe referente ao objeto é chamado.
- O construtor tem a função de inicializar as variáveis do objeto instanciado.
- O valor das variáveis a serem inicializadas são passadas como argumento para o construtor.
- Por padrão toda classe em Java possui um construtor **default**.
- O construtor sempre possui o mesmo nome da classe.
- Podem existir diversos construtores isso é conhecido como sobrecarga de construtores.

```
public class Pessoa {
```

```
//construtor default não recebe argumentos
```

```
public pessoa(){.....}
```

```
//Uma classe pode possuir múltiplos  
construtores, sendo obrigatório alterar o tipo e  
quantidade de argumentos.
```

```
public pessoa(String nome){.....}
```

```
public pessoa(String nome, int idade){.....}
```

```
public pessoa(String nome, int idade, int  
peso){.....}
```

```
}
```

Construtores

- **Porque eu preciso de um construtor ??**

Para garantir a inicialização de variáveis obrigatórias.

Por exemplo: se todo cadastro de pessoa precisa de um CPF. A classe Pessoa deverá possuir um construtor que passa como argumento o número do CPF.

```
public class Cadastro {
    Pessoa p = new Pessoa("João", "35.....");
}
```

```
public class Pessoa {
    String nome, CPF;
    public Pessoa(String nome, String CPF){
        this.nome = nome;
        this.CPF = CPF;
    }
}
```

- *Um construtor só pode ser chamado durante a instancia do objeto. Porém é possível chamar um construtor dentro do outro.*

```
public class Pessoa {
    //{..... Variáveis.....}
    public pessoa(String nome){
        this.nome = nome;
    }
    public pessoa(String nome, int idade){
        this(nome);
        this.idade = idade;
    }
    public pessoa(String nome, int idade, int peso){
        this(nome, idade);
        this.peso = peso;
    }
}
```




Encapsulamento

O encapsulamento é um dos pilares da Orientação a Objeto:

Possibilita proteger os atributos e métodos de uma classe, ao criar uma “caixa preta”, que não está suscetível a modificações externas.

Acelerar o processo de escrita do software, uma vez que o programador não precisa conhecer todos os procedimentos internos da classe utilizada.

Facilita a manutenção do código, pois os atributos e métodos de um objeto estão alocados dentro de uma única classe.

Para garantir o encapsulamento, os atributos e métodos correspondentes as regras de negócio de uma classe deverão ser **private**. Somente métodos de **interface** deverão ser **public**.

Os acessos aos atributos deverão ser realizados através de métodos modificadores **Getters** e **Setters**.

Métodos Getters

- A utilização de métodos **Getters** permitem controlar o acesso aos atributos de uma classe e facilita a manutenção do código.
- Métodos Getters podem ser públicos ou privados, e sempre devem retornar o mesmo tipo de dados da variável encapsulada.
- A variável encapsulada deverá ser privada.
- Seu nome deve iniciar com “get”+ “nome da variável”.

Exemplo:

```
private double saldoConta;  
  
public double getSaldoConta(){  
    return this.saldoConta;  
}
```

Métodos Setters

- A utilização de métodos **Setters** permitem controlar a atribuição de variáveis de uma classe, facilitando a manutenção do código.
- Métodos Setters podem ser públicos ou privados, e não retornam nenhum dado sendo do tipo **void**, recebendo como argumento um valor do mesmo tipo da variável.
- Seu nome deverá iniciar com “set”+ “nome da variável”.
- A variável encapsulada deverá ser privada.

Exemplo:

```
private double saldoConta;  
  
public void setSaldoConta(double saldoConta){  
    this.saldoConta = saldoConta;  
}
```

Exercício



1. Aplique os conceitos de encapsulamento na classe “*Pessoa*” criada na aula anterior, modificando a visibilidade de todos as variáveis para privado.
2. Modifique o acesso aos atributos da classe “*Pessoa*” adicionado métodos *Getters* e *Setters* para todas as variáveis.
3. Crie três construtores que irão receber respectivamente as seguintes variáveis:
 1. (String nome);
 2. (String nome, Pessoa mae);
 3. (String nome, Pessoa mae, Pessoa pai);Deverão ser utilizados os conceitos de sobre-carga de construtores (overload).
4. Modifique a classe principal para utilizar os métodos *getters* e *setters* e implemente os métodos necessários para responder as perguntas?
 1. Quantos filhos o objeto pessoa possui ?
 2. Qual o nome desses filhos ?
 3. Quantos irmãos o objeto pessoa possui ?
 4. Qual o nome desses irmãos ?

Bibliografia

■ Java Básico e Orientação a Objeto (livro);

- Clayton Escouper das Chagas; Cássia Blondet Baruque; Lúcia Blondet Baruque.
- Fundação CECIERJ, 2010: <https://canal.cecierj.edu.br/012016/d7d8367338445d5a49b4d5a49f6ad2b9.pdf>

■ Java e Orientação a Objetos (apostila);

- Caelum, 2023: <https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf>

■ Modificadores de acesso em Java(artigo);

- DevMedia, 2023: <https://www.devmedia.com.br/modificadores-de-acesso-em-java/18822>

■ Get e Set - Métodos Acessores em Java(artigo);

- DevMedia, 2023: <https://www.devmedia.com.br/get-e-set-metodos-acessores-em-java/29241>

■ Sobrecarga e sobreposição de métodos em orientação a objetos(artigo);

- DevMedia, 2023: <https://www.devmedia.com.br/sobrecarga-e-sobreposicao-de-metodos-em-orientacao-a-objetos/33066>