📚 Glossário Completo de Java para Iniciantes

Este glossário reúne os principais conceitos, termos e práticas da linguagem Java, explicados de forma clara e objetiva, com exemplos de código prático para facilitar seu aprendizado.

📌 Variáveis

São usadas para armazenar valores temporários na memória durante a execução do programa.

```
java
CopiarEditar
int idade = 25;
double salario = 3500.50;
String nome = "Carlos";
```

📌 Constantes

São valores que não podem ser alterados após serem definidos.

```
java
CopiarEditar
final double PI = 3.1415;
```

📌 Tipos de Dados

Tipos primitivos em Java incluem:

- `int` — números inteiros
- `double` — números decimais
- `float` — números decimais (menos preciso)
- `boolean` — verdadeiro ou falso
- `char` — caractere único
- `String` — texto (não é primitivo)

```
java
CopiarEditar
boolean ativo = true;
char letra = 'A';
String texto = "Olá!";
```

📌 Operadores

Utilizados para realizar operações matemáticas, lógicas e relacionais.

```
java
CopiarEditar
int total = 5 + 3 * 2;
boolean resultado = 5 > 3 && 2 < 4;
```

📌 Condicionais

Controlam o fluxo de execução baseado em condições.

```
java
CopiarEditar
if (idade >= 18) {
    System.out.println("Maior de idade");
} else {
    System.out.println("Menor de idade");
}
```

📌 Loops

Permitem repetir ações várias vezes.

```
java
CopiarEditar
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

📌 Arrays

Armazena múltiplos valores do mesmo tipo.

```
java
CopiarEditar
int[] numeros = {1, 2, 3, 4, 5};
```

📌 ArrayList

Coleção dinâmica que pode crescer ou diminuir.

```
java
CopiarEditar
import java.util.ArrayList;
```

```
ArrayList<String> nomes = new ArrayList<>();
nomes.add("Ana");
nomes.add("João");
```

📌 Métodos

Funções que pertencem a uma classe.

```
java
CopiarEditar
void saudacao() {
    System.out.println("Bem-vindo(a)!");
}
```

📌 Classes

Estruturas que organizam atributos e comportamentos.

```
java
CopiarEditar
class Pessoa {
    String nome;

    void falar() {
        System.out.println("Oi, eu sou " + nome);
    }
}
```

📌 Objetos

Instâncias criadas a partir de classes.

```
java
CopiarEditar
Pessoa p = new Pessoa();
p.nome = "Lucas";
p.falar();
```

📌 Herança

Permite que uma classe herde características de outra.

```
java
CopiarEditar
class Estudante extends Pessoa {
    String curso;
}
```

📌 Encapsulamento

Protege os atributos, permitindo acesso seguro via métodos.

```
java
CopiarEditar
class Conta {
    private double saldo;

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double valor) {
        saldo = valor;
    }
}
```

📌 Interfaces

Contratos que definem métodos obrigatórios.

```
java
CopiarEditar
interface Animal {
    void fazerSom();
}
```

📌 Try-Catch (Tratamento de Exceções)

Captura e trata erros durante a execução.

```
java
CopiarEditar
try {
    int resultado = 10 / 0;
} catch (Exception ex) {
    System.out.println("Erro: " + ex.getMessage());
}
```

📌 Lambda Expressions

Funções anônimas escritas de forma concisa.

```
java
CopiarEditar
lista.forEach(nome -> System.out.println(nome));
```

📌 Streams

Manipulam coleções de forma funcional.

```
java
CopiarEditar
lista.stream().filter(x ->
x.startsWith("A")).forEach(System.out::println);
```

📌 Threads

Permitem executar múltiplas tarefas simultaneamente.

```
java
CopiarEditar
Thread t = new Thread(() -> System.out.println("Executando..."));
t.start();
```

📌 Pacotes (Packages)

Organizam o código e evitam conflitos de nomes.

```
java
CopiarEditar
package meu.projeto;
```

📌 Modificadores de Acesso

Controlam a visibilidade de classes, métodos e atributos:

- `public`
- `private`
- `protected`
- *sem modificador* (package-private)