**SINTAXE JAVA**

**ANATOMIA DAS CLASSES**

**1)**

- A escrita de códigos de um programa é feito através da composição de palavras pré-defnidas pela linguagem usada (no caso java) com as expressões que usamos p determinar o nome dos arquivos, classes, atributos e métodos.

-Uma classe em java (na IDE escolhida, por ex ItelliJ), sempre começa com o nome “public class” e em seguida o nome iterativo, intuitivo e coerente que vc escolher, tipo “MinhaClass”.

-Ao iniciar um projeto, ele fica na pasta “src”, e o nome de cada projeto deverá ser com a primeira letra de cada palavra em maiúsculo sempre, tipo “MinhaClass”

- O QUE ESCREVER DENTRO DO ARQUIVO JAVA NA IDE? Se a minha classe é executável, então precisa ter um método especial/ principal – denominado MAIN. Então na primeira linha deve-se escrever “public static void main (string [] args) {}

- STRING = TEXTO

-Tudo que está entre parêntese quer dizer que é um MÉTODO, que vai ser executado. Mas p/ que ele possa ser executado ele precisa de um parâmetro conforme suas expectativas.

- Na prox. linha Escrever uma classe chamada System (que tem inúmeras operações de input e output): “ System.out.print (“Olá turma”); “ – e assim temos a primeira classe criada tendo um método executável

- As chaves são o CORPO do que está sendo escrito, tipo: { -> abre o corpo , } -> fecha o corpo

- Na hora na CONCATENAÇÃO, o “ “ é um espaço

- Ao declarar VARIÁVEIS sempre terminar com ;

- Se chama “imprimir” quando queremos ver o resultado do código sendo executado

**2) Padrões e convenções em JAVA:**

- Existem alguns padrões e convenções de escrita em JAVA:

- **Arquivo .java**: Todo arquivo .java deve começar com letra maiúscula. Se a palavra for composta, a segunda palavra também deve ser escrita com a primeira letra em maiúsculo, ex: Calculadora.java, CalculadoraCientifica.java

-**Nome da classe no arquivo**: A classe deve possuir o mesmo nome do arquivo .java, ex: Se o seu nome do arquivo java for CalculadoraCientifica.java, a primeira instrução na IDE deve ser: “public class CalculadoraCientifica {

}”

**-Nome de variável**: Toda variável deve ser escrita com letra minúscula. Porém, se a palavra for composta, a primeira letra da segunda palavra deverá ser maiúscula. Ex: ano / anoFabricacao

O nome dessa prática para nomear variáveis dessa forma é camelCase.

**- Regra Adicional / Convenção**: Quando queremos identificar em uma variável que ela não sofrerá alterações de valor, ex: queremos determinar que uma variável de nome “br” sempre representará “Brasil” e nunca mudará seu valor, logo, determinamos como escrita o código abaixo:

1 String BR = “Brasil” (qdo vemos o nome da variável assim, todo maiúsculo, pressupõe q não poderá ser alterada nunca, sempre vai ter aquele valor)

2 double PI = 3.14

3 int ESTADOS\_BRASILEIROS = 27

4 int ANO\_2000 = 2000

**-Declaração de variáveis:**

Deve conter apenas letras, \_ (underline), $ ou os números de 0 a 9.

Devem obrigatoriamente se iniciar por uma letra (preferencialmente), ou por $ - mas jamais com número

De iniciar com uma letra minúscula (boa prática)

Não pode conter espaços

Não pode usar as palavras-chave de linguagem para as variáveis

O nome deve ser único dentro de um escopo

**Exemplos declarações INVALIDAS de variáveis:**

int numero&um = 1 (está inválida pq os únicos símbolos que pode usar são \_ e $)

int 1numero = 1 (não pode começar com números)

int numero um = 1 (não pode ter espaço no nome da variável)

int long = 1 ( não pode usar palavras-chave/reservadas da linguagem)

**3) Declarando variáveis e métodos**

-Declarar uma variável em java segue sempre essa estrutura:

Tipo NomeDaVariavel = Atribuição (opcional em alguns casos específicos)

Ex: int idade = 23;

double altura = 1.62;

string meuNome = Lorena;

int anoDeFabricacao = 2012

boolean verdadeira = true

- Além de declarar variáveis contendo números e caracteres, também podemos trabalhar com valores lógicos (do tipo SIM/NÃO/TALVEZ, VERDADEIRO/FALSO)

- Depois de atribuir um valor a determinada variável, é possível diante do percurso da escrita do código mudar o valor dela. Para isso basta digitar o nome da variável e seu novo valor.

Ex:

int idade = 23;

double altura = 1.62;

string meuNome = Lorena;

idade = 69

**-Declarando métodos em Java:**

A estrutura é sempre: O tipo de operação + nomeobjetivonoinfinitivo + (parâmetro) (se tiver)

Os parâmetros (se for mais de 1), tem que estar entre parêntese, ser separados por vírgula e deve-se escrever o tipo de parâmetro tb, ex: int numeroDois / long cep / String primeiroNome

TipoRetorno NomeObjetivoNoInfinitivo Parametro(s) 🡪 o método pode exigir parâmetro OU NÃO

Ex:

int somar (int numeroUm, int numeroDois)

String formatarCep (long cep) 🡪 um método pode ter parâmetros diferentes e tipo diferentes

String primeiroNome ( ) -> um método sem parâmetro. Na escrita ainda fica os parênteses, mesmo que não tenha nada dentro! É assim que se diferencia um método de uma variável, pois o método sempre tem os parênteses contendo o parâmetro

**Para declarar um método:**

public static String nomeCompleto (String primeiroNome, String segundoNome) {

return primeiroNome.concat(“ “).concat(segundoNome);

}

Para imprimir o resultado da execução do código:

Escrever no corpo do método main:

System.out.println(NomeCompleto);

**4) IDENTAÇÃO**

- IDENTAR é escrever o código do programa de forma hierárquica, facilitando a visualização e entendimento do programa.

- É preciso TABULAR o código para que seja melhor visualizado/organizado

- Pressiona a tecla TAB na linha que for tabular

- A chave final ( { } ) do corpo estará sempre alinhada com o começo do comando de início

**5) ORGANIZANDO ARQUIVOS**

- Um projeto tende a ter várias classes, vários arquivos com extensão .java. É preciso que seja organizado esses arquivos através dos pacotes – PACKAGES, são subdiretórios

- o dev pode criar packages de acordo com a demanda, por ex: para a área comercial, organizacional, opensource, etc.. áreas diferentes de uma empresa em que se esteja trabalhando os projetos.

- Qdo salvar esses packages, geralmente se usa esses prefixos p identificar doq se trata os subdiretórios:

- comercial (com)

-organizacional (org)

-opensource (opensource)

\* Organizei meus arquivos .java no package EDU. Dentro dele está separado todos os módulos do curso

**6) JAVA BEANS**

- É uma iniciativa conceitual da comunidade pra que a gente possa ter uma estruturação de escrita, como expressamos a declaração das variáveis, classes, códigos,métodos, etc.

- é preciso que outras pessoas do time e vc mesmo no futuro compreenda os algoritmos escritos, por isso existe uma forma universal de escrita – JAVA BEANS

**-VARIÁVEIS:**

Deve ser clara, sem abreviações ou definição sem sentido

Ex:

Double salMedio = 1500.23 // não se deve abreviar, o correto seria salarioMedio

É sempre no singular, exceto qdo se referir a um array ou coleção

Ex:

String emails = [aluno@escola.com](mailto:aluno@escola.com) // fica confuso e o correto seria “email” no singular

Defina um idioma único p suas variáveis – inglês ou português?

**- MÉTODOS**

Os métodos deverão ser nomeados como verbos através de uma mistura de letras maiúsculas e minúsculas. O nome inteiro deve ser minúsculo, menos a primeira letra da segunda palavra e da terceira, quarta, etc.

Nomes sugeridos p nomenclatura de métodos:

somar (int n1, int n2) {}

abrirConexao () {}

concluirProcessamento () {}

findById (int id) {}

calcularImprimir () {} 🡪 Aqui tem algo de errado no nome do método que apesar de estar escrito corretamente, deveria ter uma única finalidade. Ou imprime ou calcula

* **TIPOS E VARIÁVEIS**

1 – Criar um repositório no github e integrar ele com sua maquina local através do VSCODE

2- Criar um projeto chamado tiposevariaveis

**TIPOS DE DADOS**

-No Java existem algumas palavras reservadas p/ representar os tipos de dados básicos q precisam ser manipulados p a construção de programas. Esses tipos de dados são conhecidos como TIPOS PRIMITIVOS (PRIMITIVE TYPES)

- São 8 tipos primitivos, q não são considerados objetos e portanto representam valores brutos. São armazenados diretamente na pilha de memória:

Int

Byte

Short

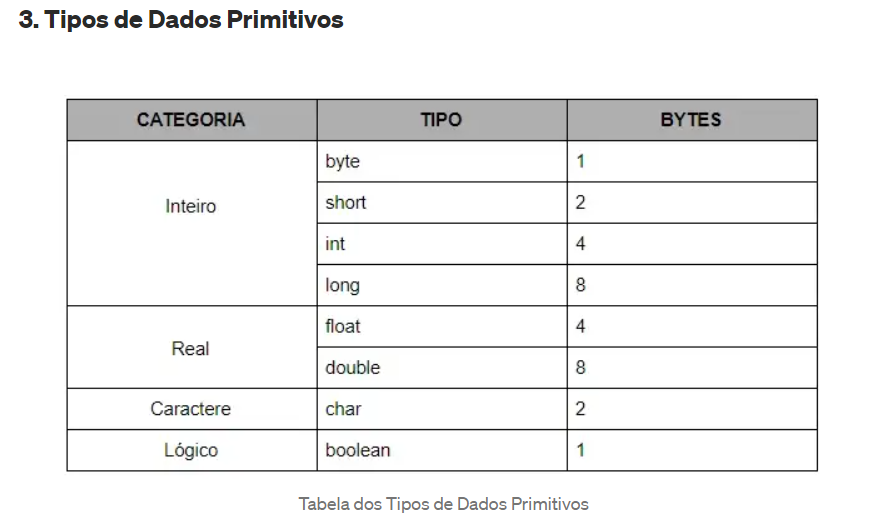
Long

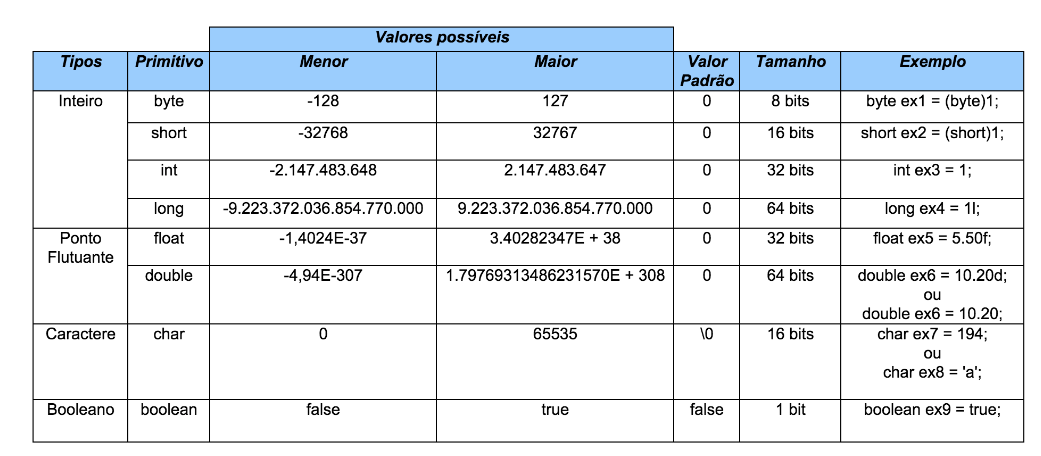
Float – p/ números com partes fracionárias

Double - p/ números com partes fracionárias ( + usado)

Boolean

Char





- No Java é comum usar o int para números inteiros (independentemente do valor ser maior ou menor, pq a linguagem tenta converter para int)

- Quando se cria uma variável é preciso pensar no TIPO DE DADO que vc vai representar. Ex: que tipo de dado usaria p representar o salário (variável) de uma pessoa? Seria double salario = 1550.32

-Cada tipo de dado tem uma forma específica de declarar o valor da variável. Ex: se for uma String o valor deve ser escrito entre “ “; se for double o valor deve ser escrito com .; se for char o valor deve ser escrito com uma aspa ‘ ‘; se for

**DECLARAÇÃO DE VARIAVEIS**

- É sempre composta da seguinte forma: <Tipo> + <nomeVariavel> = <atribuicaoDeValorOperacional>

- A forma de usar o tipo double:

Double salarioMinimo = 1500 (ou seja, n tem ponto)

Double salarioMinimo = 1500.43 (é sempre ponto e não virgula)

- No tipo float tem que colocar um F no final

- No tipo long tem que terminar com L no final

**VARIÁVEL X CONSTANTE**

- CONSTANTE: Valores armazenados na memória que NÃO PODEM SER MODIFICADOS DEPOIS DE DECLARADOS. No Java, esses valores são representados pela palavra reservada final, depois o tipo, depois o valor.

- Constantes são sempre escritas em CAIXA ALTA

Ex: final double VALOR\_DE\_PI = 3.14;

**4) OPERADORES**

São símbolos especiais p utilizar em operações matemáticas ( +, - , \*, /)

**Operadores: aritméticos, unários, relacionais, lógicos, ternários**

**Operador de atribuição** é representado pelo =

**Operador aritmético:** +, - , \*, /, % (modulo)

ATENÇÃO: O operador + qdo usado em variáveis do tipo texto, realizará a concatenação de textos

**Operador unário**: Aplicados junto com um outro operador aritmético. Realizam trabalhos básicos como incrementar, decrementar, inverter valores numéricos e booleanos.

Ex:

(+): operador unário de valor positivo

(-): operador unário de valor negativo

(++): operador unário de incremento de valor – incrementa o valor em 1 unidade (ex: incrementar um mesmo valor de 2 em 2, 3 em 3, 4 em 4..)

(--): operador de decremento de valor – decrementa o valor em 1 unidade

(!): operador unário logico de negação – nega o valor de uma expressão booleana

- para mudar o valor de uma variável que estava positiva pra negativa: nomeVariavel \* -1;



**Operador Ternário**: O operador de condição ternária é uma forma resumida p definir uma condição e escolher por dentre dois valores. Deve-se pensar na condição ternaria como se fosse um IF normal, porem de uma forma em que toda a sua estrutura estará escrita em uma única linha.

Operador ternário é representado por símbolos ?: usados na seguinte estrutura de sintaxe:

<Expressao Condicional> ? <Caso condição seja true> : <caso condição seja false>

**Operadores relacionais**:

== igual >= maior ou igual <=menor ou igual

!= diferente

>Maior que

<menor que

Operadores lógicos: recurso que nos permite criar expressões logicas maiores a partir da junção de duas ou mais expressões. Pode ser valores booleanos ou expressões relacionais

&& - operador lógico “e”

|| - operador lógico “ou”

**5) MÉTODOS**

- Todos os métodos são dispostos e devem ser definidos por uma classe

- todas As ações da aplicação são consideradas métodos

- Uma classe é definida por atributos (variáveis de diferentes tipos e valores) e métodos (funções disponíveis dentro de nossas classes)

- Existe um padrão para criação de métodos:

Deve ser nomeado como verbo

Seguir o padrão camelCase

Ex:

Somar (int n1, int n2) {}

abrirConexao () {}

concluirProcessamento () {}

**Pensar nas seguintes perguntas ao criar um método:**

1. Qual a proposta principal do método?
2. Qual o tipo de retorno esperado após executar o método?
3. Caso o método não retorne nenhum valor, ele será representado pela palavra-chave “void”
4. Quais os parâmetros serão necessários para a execução do método? Os métodos as vezes precisam de argumentos como critérios p execução
5. O método possui risco de apresentar alguma exceção?
6. Qual a visibilidade do método? Será necessário que o método seja visível a toda aplicação, somente em mesmos pacotes, através de herança ou somente a nível da própria classe?

- Quando se tem um método que vai nos retornar alguma coisa, ele precisa finalizar com a palavra “return” + o tipo correspondente

- “throws Exception”: indica que o método ao ser utilizado pode gerar uma exceção.

- private void metodoPrivado () {} 🡪 método privado, não estará visível para outras classes

1. **ESCOPO**

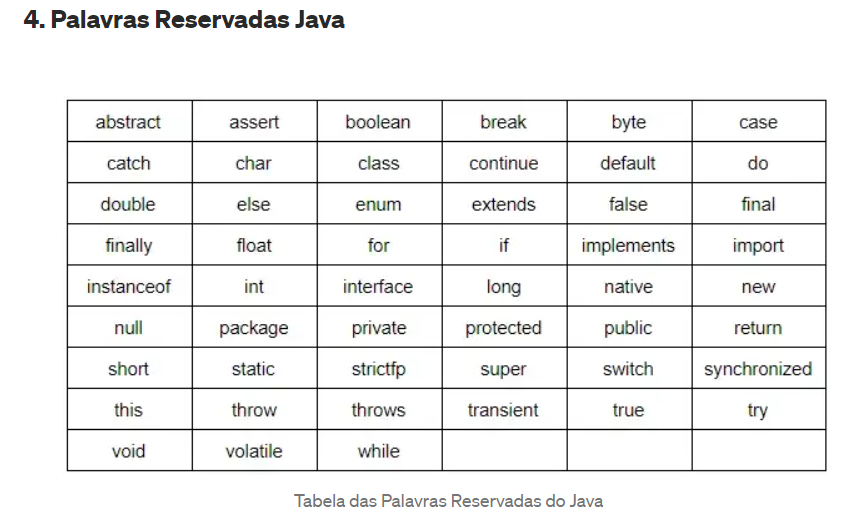
- O escopo de variáveis vai de acordo com o bloco que ela foi declarada ( as chaves {} )

**- ESCOPO DE CLASSE:**

**- ESCOPO DE MÉTODO**

**-ESCOPO DE FLUXO**

1. **PALAVRAS RESERVADAS**



Modificadores de acesso: public, private, protected

1. **DOCUMENTAÇÃO EM JAVA**

- É importante ter domínio da documentação em JAVA e os frameworks q se usa nela

- A documentação JAVA é composta por tags q representam dados relevantes p a compreensão da proposta de uma classe e o conjunto de seus métodos e atributos

- os comentários são importantes qdo vc está compartilhando o projeto com outras pessoas. Eles não são compilados , servem mais pro usuário/devs

- o do tipo “documentation” serve p elaborar um comentário a nível de documentacao

// - para colocar comentários em uma única linha [online]

/\* p começar o comentário na primeira linha [mult-line]

\*nas linhas seguintes

\*/ p fechar o corpo de comentários

/\*\* p começar o comentário na primeira linha [documentation]

\*nas linhas seguintes

\*/ p fechar o corpo de comentarios

JAVADOC

-É um gerador de documentação para documentar a API dos programas em JAVA, a partir do código-fonte. O resultado é expresso em arquivo .html

São marcações inseridas nos comentários dos programas

**7) TERMINAL E ARGUMENTOS**

- Nem sempre executamos nosso programa pela IDE, usamos tb o terminal – prompt de comando

- Com a JVM devidamente configurada podemos criar um executável do nosso programa e disponibilizar o instalador para qualquer sistema operacional

-ARGUMENTOS: quando executamos uma classe q contenha o método main, o mesmo permite que passemos um array [] de argumentos do tipo String. Logo podemos após a definição da classe a ser executada informar esses parâmetros, ex:

SCANNER: