

# PROGRAMAÇÃO BÁSICA

## CONCEITOS IMPORTANTES

- 1º O QUE SÃO VARIÁVEIS;
- 2º COMO DECLARAR VARIÁVEIS;
- 3º CONVENÇÃO DE NOMENCLATURA;
- 4º TIPOS PRIMITIVOS.

### O QUE SÃO VARIÁVEIS

NO NOSSO DIA-A-DIA É COMUM TERMOS QUE GUARDAR DADOS;

NA MAIORIA DAS VEZES USAMOS UM DISPOSITIVO PARA ISSO (EX: PENDRIVE);

NA PROGRAMAÇÃO NÃO É DIFERENTE, OS DADOS FICAM EM MEMÓRIA;

VARIÁVEIS SÃO REFERÊNCIAS PARA ENDEREÇOS NA MEMÓRIA.

## EXEMPLOS DE USO

GUARDAR O NOME DE UMA PESSOA;

GUARDAR A IDADE DE UMA PESSOA;

GUARDAR UM VALOR PARA USO POSTERIOR.

ÁREA DE MEMÓRIA ASSOCIADA A UM NOME, QUE PODE ARMazenar VALORES DE UM TIPO.

### Como DECLARAR UMA VARIÁVEL EM JAVA

<TIPO> <NOME DA VARIÁVEL>;

<TIPO> <NOME DA VARIÁVEL> = <VALOR>;

↳ DIZEMOS QUE ESTAMOS CRIANDO E INICIALIZANDO A VARIÁVEL.

## CONVENÇÃO DE NOMENCLATURA

NUNCA USAR PALAVRAS RESERVADAS (AQUELAS JÁ USADAS PELA LINGUAGEM);

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

NAO É NECESSÁRIO DECORAR,  
COM O TEMPO JÁ É AUTOMÁTICO;  
A PRÓPRIA IDE NÃO DEIXA! ;

## CONVENÇÃO = BOAS PRÁTICAS = REGRAS

LETRA INICIAR DEVE COMEÇAR COM: a-z A-Z \_ &

APÓS A SEGUNDA LETRA: a-z A-Z \_ & 0-9

O PRIMEIRO NOME DE UMA VARIÁVEL COMEÇA COM MINÚSCULO: camelCase

NAO HÁ LIMITE DE CARACTERES PARA O SEU TAMANHO.

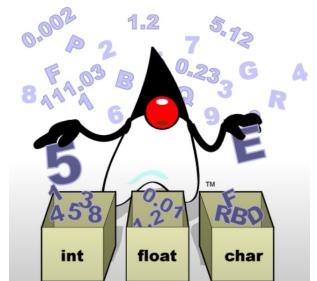
## TIPOS PRIMITIVOS.

INT = NÚMEROS INTEIROS POSITIVOS E NEGATIVOS;

FLOAT/DOUBLE = PONTOS FLUTUANTES (NÚMEROS COM VÍRGULA);

CHAR = CARACTERE ÚNICO;

BOOLEAN = VERDADEIRO OU FALSO.



## PACOTES

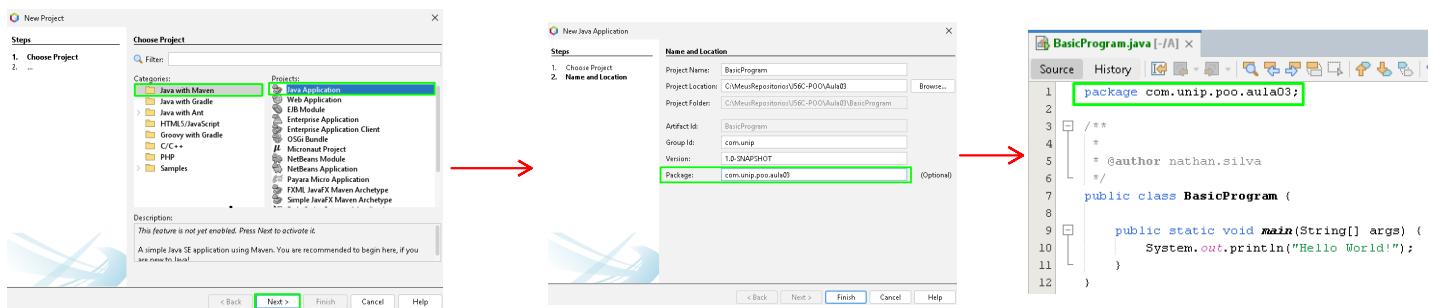
PERMITE ORGANIZAR MELHOR O CÓDIGO;

AS CLASSES PASSAM A PERTENCER A UM PACOTE;

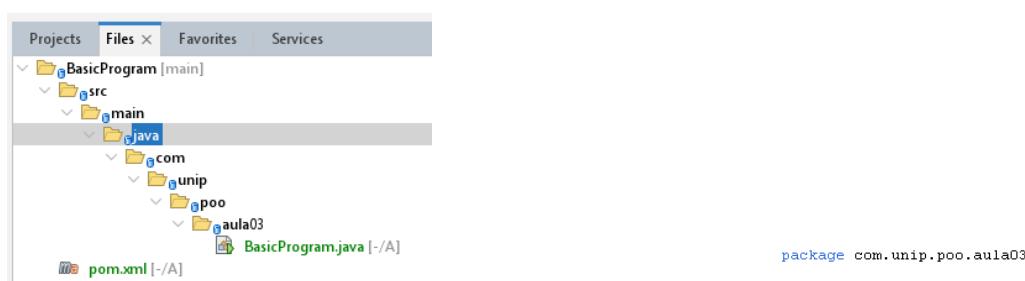
GERALMENTE REPRESENTADO PELO DOMÍNIO DA EMPRESA;

CADA PARTE (SEPARADA POR PONTO) REPRESENTA UMA PASTA.

## HANDS ON COM NETBEANS



VEJA COMO O PACOTE FICA ORGANIZADO EM PASTAS:



```
package com.unip.poo.aula03;

/*
 * @author nathan.silva
 */
public class BasicProgram {

    public static void main(String[] args) {

        //usando convenção corretamente
        int idade = 22;
        String nome = "Nathan C.";
        String nomeDoCachorro = "Mei";
        boolean temDinheiro = false;
        char melhorLetra = 'N';
        double peso = 84.7;

        //funciona, mas fora da convenção
        String _nome = "errado";
        String Nome = "muito errado";
        String nome_da_cachorro = "não precisa nem falar...";

        System.out.println("Nome: " + nome);
    }
}
```

## DECLARANDO ALGUMAS VARIÁVEIS

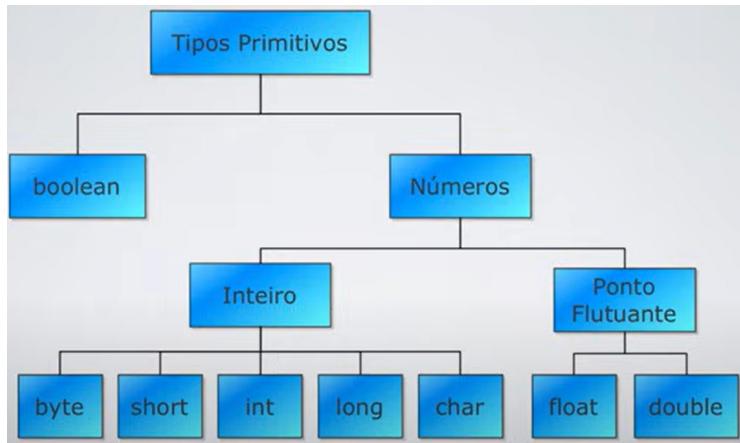
NOTE QUE A IDE FAZ ALGUMAS

MARCAÇÕES (AMARELO). ISSO INDICA

QUE ELA NÃO ESTÁ SENDO LIDA, OU  
SEJA, O SEU VALOR NÃO É CONSUMIDO.

## APROFUNDANDO TIPOS PRIMITIVOS

JAVA EMBORA SEJA ORIENTADA A OBJETOS, ELA NÃO É 100% ORIENTADA; UM EXEMPLO DISSO SÃO OS PRÓPRIOS TIPOS PRIMITIVOS; CADA TIPO OCUPA UM ESPAÇO (TAMANHO) CONTECIDO NA MEMÓRIA; EXISTEM BASICAMENTE DOIS GRUPOS: BOOLEAN E NÚMEROS.



### Sobre os Números Inteiros

NO JAVA EXISTEM 4 TIPOS PRINCIPAIS QUE REPRESENTAM INTEIROS; O QUE VARIA NESSES TIPOS É O SEU TAMANHO (CAPACIDADE ARMazenamento). ATENÇÃO: QUANTO MAIOR MAIS ESPAÇO DE MEMÓRIA SERÁ USADO; OS MAIS USADOS NO DIA-A-DIA SÃO INT E LONG.

Tipo	Tamanho (bits)	Intervalo de Valores	
byte	8	-128 a 127	-(2 <sup>7</sup> ) a 2 <sup>7</sup> -1
short	16	-32.768 a 32.767	-(2 <sup>15</sup> ) a 2 <sup>15</sup> -1
int	32	-2.147.483.648 a 2.147.483.647	-(2 <sup>31</sup> ) a 2 <sup>31</sup> -1
long	64	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807	-(2 <sup>63</sup> ) a 2 <sup>63</sup> -1

### Sobre os Pontos Flutuantes

PODEM SER DE DOIS TIPOS: FLOAT E DOUBLE; ARMazenam NÚMEROS COM CASAS DECIMais; O QUE OS DIFERENCIAM É O ESPAÇO OCUPADO; NO DIA-A-DIA O MAIS USADO É O DOUBLE.

Tipo	Tamanho (bits)
float	32
double	64

QUANDO USAR O FLOAT, O QUE NÃO É MUITO CONUM, DEVEMOS EXPRESSAR A INTENÇÃO DE USO PÔR MEIO DO LITERAL F.

```
float saldo1 = 100.30f;  
double saldo2 = 100.30;
```

## SOBRE O TIPO CHAR

NADA MAIS É DO QUE UM ÚNICO CARACTERE;  
IMAGINE COMO EXEMPLO UMA LETRA DO ALFABETO;  
COMO O CHAR PODE SER UM TIPO INTEIRO? **TABELA ASCII**  
CADA CARACTERE POSSUE UM VALOR EM DECIMAL PARA REPRESENTAÇÃO.

<https://www.ascii-code.com/>

ARMAZENAR O CARACTERE:

```
char o = 'o';  
char i = 'i';
```

ARMAZENAR O DECIMAL:

```
char o = 111;  
char i = 105;  
System.out.println(""+ o + i);
```

ARMAZENAR O HEXADECIMAL:

```
char o = 111;  
char i = 105;  
char interrogacao = 0x003F;  
System.out.println(""+ o + i + interrogacao);
```

## SOBRE O TIPO BOOLEAN

CONSISTE EM VERDADEIRO OU FALSO;  
É MUITO ÚTIL QUANDO PRECISAMOS DEFINIR FLAGS;

```
boolean verdadeiro = true;  
boolean falso = false;
```

## CURIOSIDADES

EN NOSSO DIA-A-DIA ESTAMOS ACOSTUMADOS COM A CONTAGEM DECIMAL, ONDE OS VALORES VÃO DE 0 A 9.

PORÉM É IMPORTANTE SALIENTAR QUE CONSEGUIMOS REPRESENTAR TAMBÉM OUTRAS BASES, COMO: HEXADECIMAS, OCTAIS E BINÁRIOS.

```
int decVal = 26;  
int hexVal = 0x1a;  
int octVal = 032;  
int binVal = 0b11010; // JDK 7
```

QUANDO O NÚMERO É MUITO GRANDE, O JAVA PERMITE USAR O CARACTERE UNDERSCORE PARA SEPARA-LOS, O QUE MELHORA A SUA LEGIBILIDADE.

```
long creditCardNumber = 1234_5678_9012_3456L;  
long cpf = 101_134_156_68L;  
float pi = 3.14_15F;  
long hexBytes = 0xFF_EC_DE_5E;  
long hexWords = 0xCAFE_BABE;  
long maxLong = 0x7fff_ffff_ffff_ffffL;  
byte nybbles = 0b0010_0101;  
long bytes = 0b11010010_01101001_10010100_10010010;
```

## CARACTERES DE ESCAPE

USADOS QUANDO PRECISAMOS FORMATAR O OUTPUT DO PROGRAMA.

EXEMPLO :

```
System.out.println("\\"Hello World!\"");  
System.out.println("Linha um. \nLinha dois.");
```

Seqüência de Escape	Descrição
\t	tab
\b	backspace
\n	nova linha
\r	retorno de carro
\f	avanço de página
'	aspas simples
"	aspas duplas
\\"	barra invertida
\ddd	constante octal
\uxxxx	constante hexadecimal

## LEITURA DE DADOS VIA CONSOLE

PARA FAZER A LEITURA DE UM DADO DO TECLADO, USAMOS A CLASSE SCANNER.

```
Scanner scan = new Scanner(System.in);
```

IMPORTANTE: A API DO JAVA É ORGANIZADA EM PACOTES. ENTÃO PARA USAR A CLASSE SCANNER, UMA IMPORTAÇÃO DEVE SER REALIZADA.

```
import java.util.Scanner;  
  
public class LeituraDadosTeclado {  
}
```

LENDENDO UMA LINHA INTEIRA:

É USADO O MÉTODO `nextLine()` DA CLASSE SCANNER. NÃO IMPORTA O QUE VOCÊ COLOCAR NA LÍMINA AO DIGITAR `ENTER` O CONTEÚDO SERÁ CAPTURADO.

```
String nome = scan.nextLine();
```

LENDENDO UM TIPO ESPECÍFICO:

`next()` = LÊ UMA SEQUÊNCIA DE CARACTERES ATÉ O PRÓXIMO DELIMITADOR (ESPAÇO, TAB OU QUEBRA DE LINHA);

`nextInt()` = REALIZA A LEITURA DE UM VALOR INTEIRO;

`nextDouble()` = REALIZA A LEITURA DE UM VALOR DOUBLE;

... = OS DEMAIS TIPOS PRIMITIVOS TAMBÉM POSSUEM.

```
String primeiroNome = scan.next();  
int idade = scan.nextInt();  
double altura = scan.nextDouble();
```

## OPERADORES DO JAVA

OPERADORES ARITMÉTICOS

OPERADORES RELACIONAIS

OPERADORES LÓGICOS

OPERADORES DE ASSIGNMENT

PRECEDÊNCIA



TEM POR OBJETIVO SOLICITAR AO

COMPILEADOR ALGUMA OPERAÇÃO.

EXEMPLO: SOMAR DOIS NÚMEROS.

## OPERADORES ARITMÉTICOS

USADOS PARA REALIZAR OPERAÇÕES MATEMÁTICAS ENTRE NÚMEROS.

Operador	Descrição
+	adição (e mais unário)
-	subtração (e menos unário)
*	multiplicação
/	divisão
%	módulo
++	incremento (pos ou pré fix)
--	decremento (pos ou pré fix)

```
int resultado = 1 + 2;  
System.out.println(resultado);  
  
resultado = resultado - 1;  
System.out.println(resultado);  
  
resultado = resultado * 2;  
System.out.println(resultado);  
  
resultado = resultado / 2;  
System.out.println(resultado);  
  
resultado = resultado % 8;  
resultado = resultado % 7;  
System.out.println(resultado);
```

```
int i = 3;  
i++;  
// output 4  
System.out.println(i);  
++i;  
// output 5  
System.out.println(i);  
// output 6  
System.out.println(++i);  
// output 6  
System.out.println(i++);  
// output 7  
System.out.println(i);
```

## ATENÇÃO ESPECIAL SOBRE O OPERADOR +

ELE TAMBÉM PODE SER USADO PARA CONCATENAR STRINGS E/OU VARIÁVEIS

```
String primeiraString = "Esta é";
String segundaString = " uma String concatenada.";
String terceiraString = primeiraString + segundaString;
System.out.println(terceiraString);
```

## OPERADORES RELACIONAIS

COMO O PRÓPRIO NOME SUGERE, ELES SÃO USADOS PARA ESTABELECER COMPARAÇÕES, OU SEJA, RELACIONAR VALORES.

Operador	Descrição
==	igual a
!=	diferente de
>	maior que
<	menor que
>=	maior ou igual que
<=	menor ou igual que

```
int valor1 = 10;
int valor2 = 5;

System.out.println("valor1 == valor2: " + (valor1 == valor2));
System.out.println("valor1 != valor2: " + (valor1 != valor2));
System.out.println("valor1 > valor2: " + (valor1 > valor2));
System.out.println("valor1 < valor2: " + (valor1 < valor2));
System.out.println("valor1 >= valor2: " + (valor1 >= valor2));
System.out.println("valor1 <= valor2: " + (valor1 <= valor2));
```

## OPERADORES LÓGICOS

OPERADORES LÓGICOS SÃO MUITO USADOS EM CONDICIONAIS EM CONJUNTO COM OS OPERADORES RELACIONAIS.

Operador	Descrição
&	AND
	OR
^	XOR
	OR curto circuito
&&	AND curto circuito
!	NOT

Tabela Verdade

a	b	a & b	a   b	a ^ b	!a
Falso	Falso	Falso	Falso	Falso	Verdadeiro
Verdadeiro	Falso	Falso	Verdadeiro	Verdadeiro	Falso
Falso	Verdadeiro	Falso	Verdadeiro	Verdadeiro	Verdadeiro
Verdadeiro	Verdadeiro	Verdadeiro	Verdadeiro	Falso	Falso