Módulo Lógica de Programação com aplicações em Java

Projeto khouse Profissionalizante

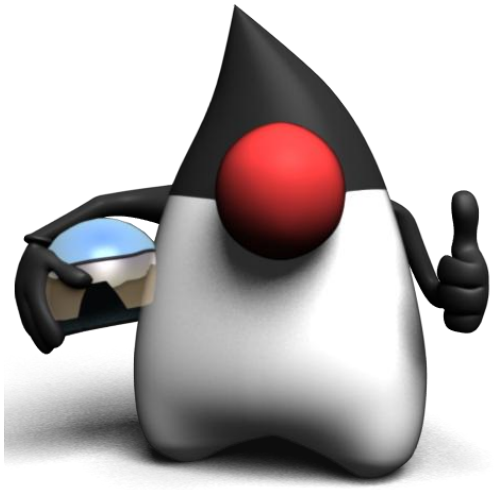
Profª Larissa Brandão

Fundamentos da Programação

Objetivos:

- Reconhecer as diferenças entre os tipos primitivos, variáveis, identificadores e operadores
- Desenvolver um simples programa em Java usando os conceitos estudados nesta lição

Instruções e Blocos em Java



Instrução

Uma instrução é composta de uma ou mais linhas terminadas por ponto-e-vírgula

Exemplo:
System.out.println(“Hello world”);

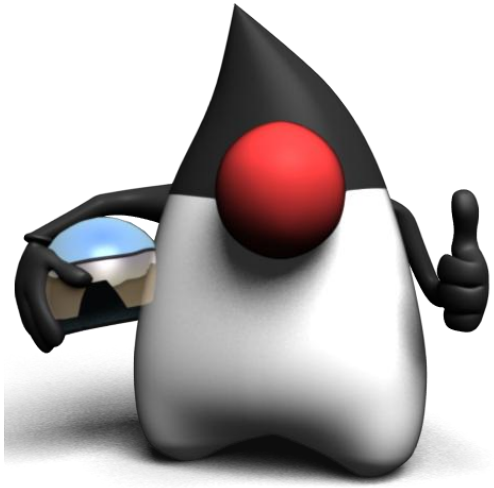
Bloco

é formado por uma ou mais instruções agrupadas entre chaves { } indicando que formam uma só unidade

Exemplo:

```
public static void main (String[] args) {
System.out.println("Hello");
System.out.println("world”);
}
```

Identificadores em Java



Fundamentos da Programação

Identificadores

são representações de nomes de variáveis, métodos, classes, etc.

Exemplo:

- Hello
- main
- System

Fundamentos da Programação

- Os identificadores são case-sensitive.
- Iniciam com Letra (A-Z, a-z), Underscore "_", ou Sinal de cifrão "$".
- Aos caracteres subseqüentes adicionam números (0-9)
- Não pode utilizar nomes iguais as palavras-chave

Palavras Chaves

- são identificadores pré-definidos e reservados por Java para propósitos específicos

Fundamentos da Programação

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

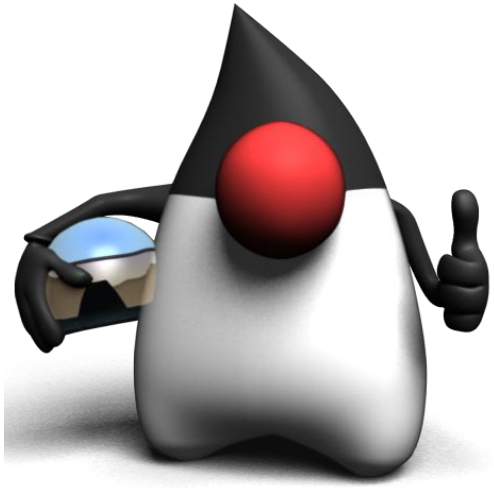
* not used

** added in 1.2

*** added in 1.4

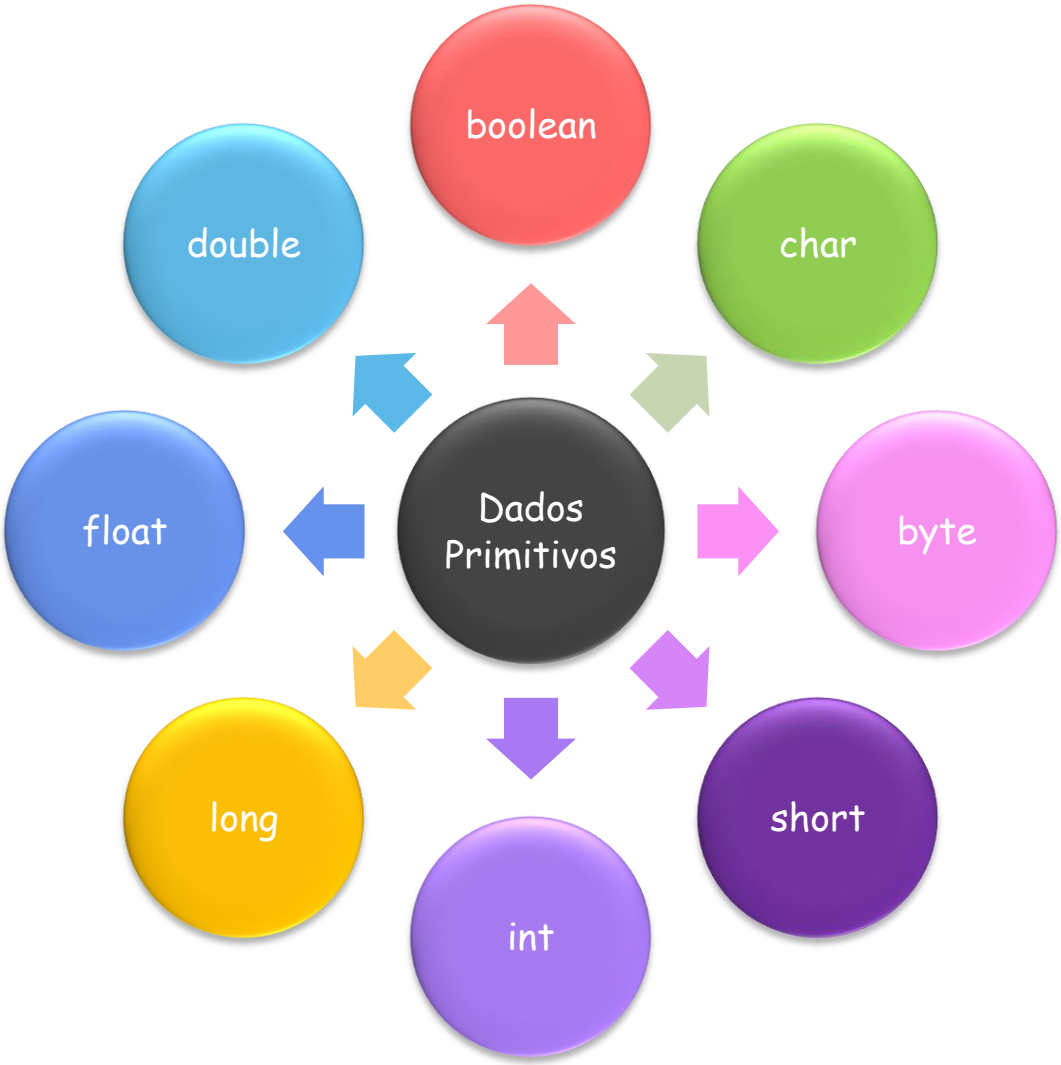
**** added in 5.0

Tipos de dados em Java




Tipos de dados em Java
Boolean (lógico)
Character (Caractere)
Integer (Inteiro)
Floating-Point (Número Flutuante)

Fundamentos da Programação



Fundamentos da Programação

Boolean

true ou false

```
boolean result = true;
```

Fundamentos da Programação

Char

16 bits - armazena Unicodes

char gato = ‘G’;

Fundamentos da Programação

Byte

8 bits –

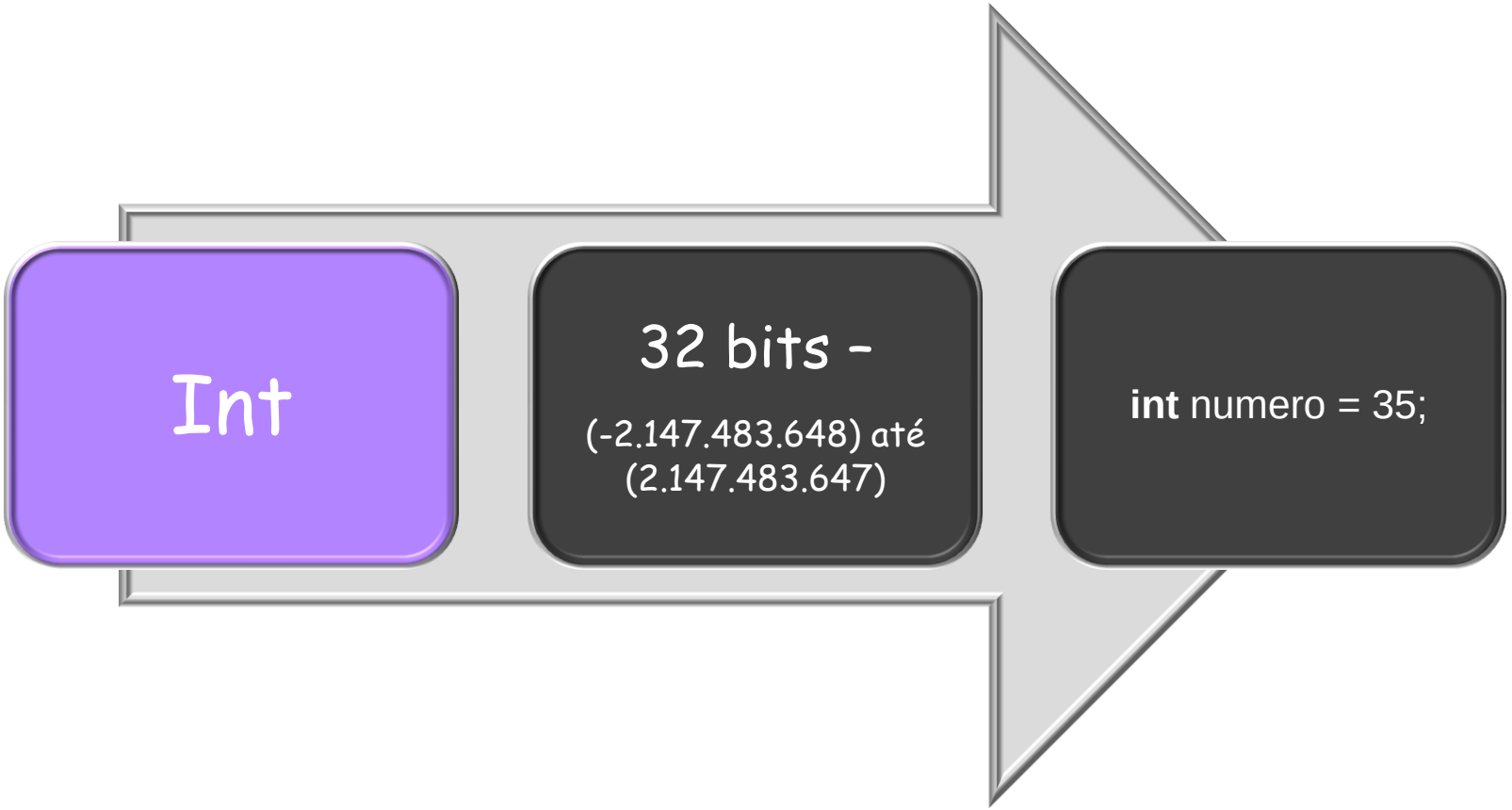
(-128) até (127)

```
byte numero = 35;
```

Fundamentos da Programação

Short	16 bits - (-32.768) até (32.767)	**short** numero = 35;

Fundamentos da Programação



Fundamentos da Programação

Long

64 bits -

(-9.223.372.036.854.775.808)
até
(9.223.372.036.854.775.807)

```
long numero = 35L;
```

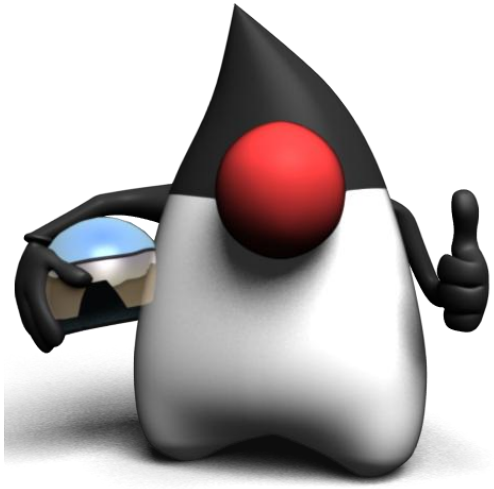
- **Os números flutuantes possuem um ponto decimal ou um dos seguintes caracteres:**
  - E ou e // expoente
  - F ou f // float
  - D ou d // double

- **São exemplos:**
  - 3.14 // tipo sem marcação(double por padrão)
  - 6.02E23 // tipo double com expoente
  - 2.718F // tipo float
  - 123.4E+306D // tipo double

Fundamentos da Programação

Tamanho em memória	*Dado primitivo*	*Faixa*
32 bits	float	-10^{38} até $10^{38}-1$
64 bits	double	-10^{308} até $10^{308}-1$

Variáveis



- **Uma variável é um espaço na memória usado para armazenar o estado de um objeto.**

- **Uma variáveis possui:**
  - Tipo que indica o tipo de dado que ela pode conter
  - Nome que deve seguir as regras para identificadores

Variável

< tipo do dado > < nome > [= valor inicial];

Exercício

```
public class Variavel {

	public static void main( String[] args ){
		boolean result;
		char option;
		option = 'C';
		double grade = 0.0;

	}
}
```

Exercício

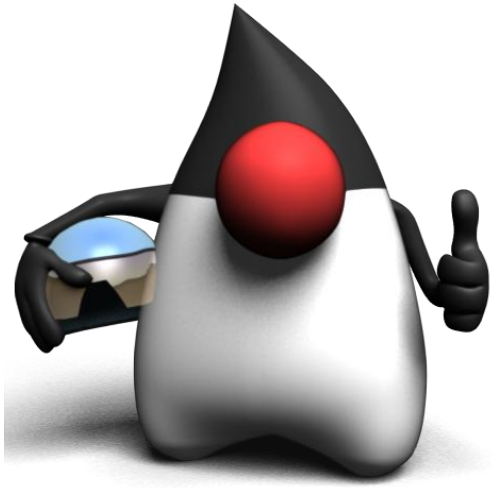
```
public class Variavel {

        public static void main( String[] args ){
                boolean resultado;
                char opcao;
                opcao = 'C';
                double number= 0.0;

                System.out.println(“Mostrando variáveis:” +
                opcao + number);

        }
}
```

Operadores



Fundamentos da Programação

Fundamentos da Programação

aritméticos

Operador	Uso	Descrição
*	op1 * op2	Multiplica op1 por op2
/	op1 / op2	Divide op1 por op2
%	op1 % op2	Resto da divisão de op1 por op2
-	op1 - op2	Subtrai op2 de op1
+	op1 + op2	Soma op1 e op2

Fundamentos da Programação

Incremento e decremento

Operador	*Uso*	*Descrição*
++	op++	Incrementa op em 1; Avalia a expressão antes do valor ser acrescido
++	++op	Incrementa op em 1; Incrementa o valor antes da expressão ser avaliada
--	op--	Decrementa op em 1; Avalia a expressão antes do valor ser decrescido
--	--op	Decrementa op em 1; Decrementa op em 1 antes da expressão ser avaliada

Fundamentos da Programação

relacionais

Operador	Uso	Descrição
>	op1 > op2	op1 é maior do que op2
>=	op1 >= op2	op1 é maior ou igual a op2
<	op1 < op2	op1 é menor do que op2
<=	op1 <= op2	op1 é menor ou igual a op2
==	op1 == op2	op1 é igual a op2
!=	op1 != op2	op1 não igual a op2

Lógicos

- Operadores lógicos avaliam um ou dois operandos **lógicos** e resultam em um único valor **lógico: true ou false**

- Os operadores lógicos são seis:
  - && (e lógico)
  - & (e binário)
  - || (ou lógico)
  - | (ou binário)
  - ^ (ou exclusivo binário)
  - ! (negação)

Lógicos

- && (e lógico) e & (e binário)

x1	*x2*	*Resultado*
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
FALSO	FALSO	FALSO

Lógicos

- || (ou lógico) e | (ou binário)

x1	*x2*	*Resultado*
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
FALSO	FALSO	FALSO

Lógicos

- ^ (ou exclusivo binário)

x1	x2	Resultado
VERDADEIRO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
FALSO	FALSO	FALSO

Lógicos

- ! (negação)

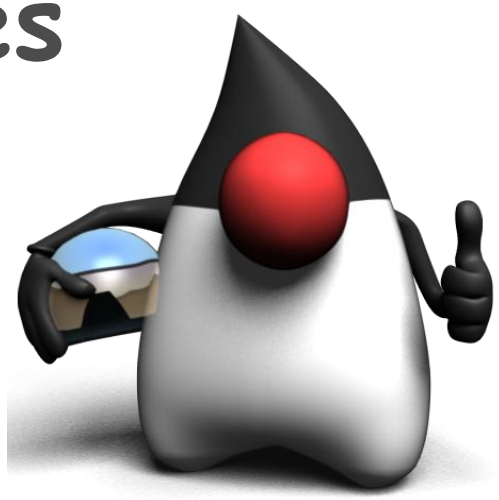
x1	*Resultado*
VERDADEIRO	FALSO
FALSO	VERDADEIRO

Fundamentos da Programação

Condicionais

expLógica**?**expCasoTrue**:**expCasoFalse

Precedência de Operadores



Ordem	Operador
1	( ) parênteses
2	++ pós-incremento e -- pós-decremento
3	++ pré-incremento e -- pré-decremento
4	! negação lógica
5	* multiplicação e / divisão
6	% resto da divisão
7	+ soma e – subtração
8	< menor que, <= menor ou igual, > maior que e >= maior ou igual
9	== igual e != não igual
10	& e binário
11	| ou binário
12	^ ou exclusivo binário
13	&& e lógico
14	|| ou lógico
15	?: condicional
16	= atribuição

Precedência de Operadores

Dada a seguinte expressão complexa:

6%2*5+4/2+88-10

pode-se fazer uso de parênteses para reescrevê-la de maneira mais clara:

((6%2)*5)+(4/2)+88-10