TIPOS PRIMITIVOS, CONVERSÕES E CLASSE MATH

Declaração de Variáveis

Variáveis são contêineres para armazenar valores. Em Java, existem diferentes **tipos** de variáveis, como:

- **String:** Armazena texto. Ex: "Olá, Mundo".
- **Int:** Armazena números inteiros (positivos e negativos), sem decimais: Ex: 123.
- **Float:** Armazena números de ponto flutuante (positivos e negativos), com decimais. Ex: -19,99f.
- **Char:** Armazena caracteres únicos. Ex: 'a' ou 'B'.
- **Boolean:** Armazena valores com dois estados: *true* (verdadeiro) ou *false* (falso).

Declaração de Variáveis

Para criar uma variável, você deve especificar o tipo e atribuir um valor a ela, no formato:

tipo nomedavariavel = valor;

```
String nome = "John";
int num = 15;
char  categoria = ‘a’;
float nota = 7.5f;
boolean cadastro = true;
```

Variáveis de mesmo tipo podem ser declaradas juntas:
int x = 5, y = 6, z = 50;

Valores de string são cercados por aspas duplas. Valores char são cercados por aspas simples.

Valores de float precisam ser definidos com a letra **f** no final

Declaração e Exibição de Variáveis

A variável pode ser declarada sem o valor, que pode ser atribuído depois no código:

```
int total;
total = n1 + n2;

String nome;
nome = “Rafael”;
```

O comando println / print costuma ser usado para exibir variáveis. Para combinar texto e variáveis (concaternação), use o caractere +

```
String nome = "John";
String sobrenome = “Nash”;

System.out.println (“Olá " +
nome + sobrenome);
```

Variáveis em Java

- Todas as **variáveis** devem ser **identificadas** com **nomes exclusivos (identificadores).**

- Identificadores podem ser nomes curtos (como x e y) ou descritivos (idade, soma, totalVolume).

- Os nomes podem conter letras, dígitos, sublinhados e cifrões.

Variáveis em Java

- A linguagem diferencia letras maiúsculas de minúsculas (case sensitive).

- Os nomes devem começar com uma letra minúscula e não podem conter espaço.

- Palavras reservadas não podem ser usadas como nomes.

Constantes

- São declaradas quando precisamos lidar com dados que não devem ser alterados durante a execução do programa.

- Para isso, utilizamos a palavra reservada final para que a variável seja inicializada uma única vez.

```
final float PI = 3.1416F;
final String NOME_PAGINA = “home”;
```

Palavras Reservadas em Java

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

Você **não** pode usar nenhum dos itens acima como identificadores em seus programas.

Tipos de Dados Primitivos

- Também chamados de variáveis primitivas, especificam o tamanho e o tipo de valores de variáveis.
- Java é uma linguagem **fortemente tipada,** e possui oito tipos de dados primitivos: **short, int, long, float, double, boolean**;
- Dados primitivos não possuem métodos adicionais.
- Em Java, String não é um tipo de dado, mas sim uma **classe**. Suas variáveis são, na verdade, objetos dessa classe com métodos que podem executar determinadas operações.

Tipos de Dados Primitivos

TIPO	TAMANHO	DESCRIÇÃO
byte	1 byte	Armazena valores entre -128 e 127
short	2 bytes	Armazena valores entre -32,768 to 32,767
int	4 bytes	Armazena valores entre -2,147,483,648 to 2,147,483,647
long	8 bytes	Armazena valores entre -9,223,372,036,854,775,808 e 9,223,372,036,854,775,807
float	4 bytes	Armazena números fracionados de 6 a 7 dígitos decimais
double	8 bytes	Armazena números fracionados de até 15 dígitos decimais
boolean	1 bit	Armazena valores *true* ou *false*

Tipos Primitivos x Memória

- Por ser uma linguagem feita para rodar em todos os tipos de dispositivos, alguns deles podem ter pouca memória.

- A memória pode ser pequena e talvez seja preciso guardar um número grande. Não adianta ter um tipo **int** para guardar um único número, por exemplo.

- Tipos primitivos são importantes para economizar memória e desenvolver uma boa aplicação.

Conversão de Tipos - Typecast

- Ocorre quando é atribuido um valor de um tipo de dados primitivo a outro tipo.
  - **Automático (Implícito):** Do menor para o maior
    byte -> short -> char -> int -> long -> float -> double
  - **Manual (Explícito):** Do maior para o menor
    double -> float -> long -> int -> char -> short ->byte
- O tipo boolean não pode ser convertido para nenhum outro tipo.

Conversão de Tipos - Typecast

- **Indução de Tipo:** O typecast dita ao compilador como tal dado deve ser interpretado e manipulado.

- **Typecast Implícito:** Feito automaticamente pelo compilador quando um tipo de dado pode ser facilmente manipulado no lugar de outro tipo de dado.

- **Typecast Explícito:** Feito diretamente no algorítmo para indicar ao compilador qual a forma de interpretação de um dado quando ele entende que há ambiguidade ou formatos incompatíveis.

Conversão de Tipos - Typecast

```
int num1 = 9;
double valor = num1;
```

→ **Automática**

implícita

```
double n1 = 9.78;
int n2 = (int) n1;
```

→ **Manual**

explícita

```
double sal = (float) 1825.54;
```

→ **O f do valor Float torna-se dispensável neste caso.**

Conversão de Tipos - Typecast

De / Para	byte	short	char	int	long	float	double
byte		Implícito	(char)	Implícito	Implícito	Implícito	Implícito
short	(byte)		(char)	Implícito	Implícito	Implícito	Implícito
char	(byte)	(short)		Implícito	Implícito	Implícito	Implícito
int	(byte)	(short)	(char)		Implícito	Implícito	Implícito
long	(byte)	(short)	(char)	(int)		Implícito	Implícito
float	(byte)	(short)	(char)	(int)	(long)		Implícito
double	(byte)	(short)	(char)	(int)	(long)	(float)	

Adaptado de: https://www.caelum.com.br/apostila-java-orientacao-objetos/variaveis-primitivas-e-controle-de-fluxo/#casting-e-promoo

Wrapper Class

- Cada um dos oito tipos de dados primitivos do Java tem uma classe dedicada a ele, conhecidas como wrappers class porque "envolvem" o tipo de dados primitivo em um objeto dessa classe.

- **Wrapper Class:** Boolean, Byte, Character, Double, Float, Integer, Long, Short.

- Uma de suas funções é fazer conversões.

Método parse<Tipo>()

Colocar um valor String em int

String valor = “30”;
int idade = valor;

String não pode ser convertido para int, nem mesmo com o typecast

```
String valor = “30”;
int idade = (int)valor;
```

SOLUÇÃO: int idade = **Integer.parseInt**(valor);

Método parse<Tipo>()

- Para fazer uma conversão de String para o tipo primitivo correspondente, pode-se usar um método parse com o tipo correspondente.
  - Byte.parseByte(String S);
  - Double.parseDouble(String S);
  - Float.parseFloat(String S);
  - Integer.parseInteger(String S);
  - Long.parseLong(String S);
  - Short.parseShort(String S);
- Todas as Wrapper Class, exceto Character, têm métodos com essa função

Método toString()

Colocar um valor int numa String

```
int idade = 30;
String valor = idade;
```

Int não pode ser convertido para String, nem mesmo com o typecast

```
int idade = 30;
String valor = (String) idade
```

SOLUÇÃO: String valor = **Integer.toString**(idade);

Método toString()

- A ideia é retornar uma representação em String de um dado objeto. Dessa forma, todas as Wrapper Class têm um método toString().

  - Byte.toString();
  - Double.toString();
  - Float.toString();
  - Integer.toString();
  - Long.toString();
  - Short.toString();
  - Character.toString();

Classe Math

Proporciona uma série de operações e constantes matemáticas facilmente acessadas, sem ser necessário instanciar um objeto da classe para podermos usar seus métodos, sendo os mais utilizados:

- ✓ **Máximo e Mínimo**
- ✓ **Potências e Raízes**
- ✓ **Logaritmo**
- ✓ **Arredondamentos e Valores Absolutos**
- ✓ **Trigonometria**
- ✓ **Números Randômicos**

Classe Math

pow	Exponenciação	Math.pow(5,2)	**25**
sqrt	Raiz Quadrada	Math.sqt(81)	**9**
cbrt	Raiz Cúbica	Math.cbrt(64)	**4**
abs	Valor Absoluto	Math.abs(-5)	**5**
floor	Arrendondamento para Baixo	Math.floor(3.9)	**3**
ceil	Arrendondamento para Cima	Math.ceil(4.2)	**5**
round	Arredondamento Aritimético	Math.round(5.6)	**6**
min	Valor Mínimo	Math.min(9,3,2)	**2**
max	Valor Máximo	Math.max(7,9,3)	**9**
PI	Número PI	Math.PI	**3,141592653589793**

Mais Métodos: https://www.w3schools.com/java/java_ref_math.asp

Geração de Números Aleatórios

As instâncias da classe “Random” são objetos geradores de números aleatórios, que produzem estes números em resposta a solicitações. Ela possue alguns métodos para isso:

Random gerador = new Random();

gerador.nextInt(); **gerador.**nextFloat();

gerador.nextInt(n); **gerador.**nextDouble();

Nota: A classe Math possui um método para gerar números aleatórios: Math.randon(). No entanto, eles se limitam a serem números decimais maiores ou igual a 0 e menor que 1. Para outros, será necessário realizar cálculos matemáticos.

Método Randon.nextInt()

- Chamado sem nenhum parâmetro, o método nestInt() retorna um inteiro diferente da sequência de aleatórios. Serão retornados números positivos ou negativos dentre a faixa de valores inteiros do Java.

- O método nextInt() quando chamado com parâmetros, obterá números entre um intervalo. **Ex: nextInt(6) retorna números entre 0 e 5.**

- Os métodos **nextDouble()** e **nextFloat()** não possuem parâmetros e retornam números decimais entre 0 e 1.

Intervalo entre 2 Números com Math.random()

O primeiro numero do intervalo mais o número randômico (gerado por Math.random()), vezes o limite do intervalo (último número menos o primeiro).

Número entre 5 e 10

```
int n1 = (int) (5 + (Math.random() * (10-5)));
System.out.println(n1);
```

Número entre 15 e 50

```
double aleatorio = Math.random();
int n2 = (int) (15 + aleatorio * (50-15));
System.out.println(n2);
```

Referências Bibliográficas

https://www.w3schools.com/java/default.asp

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

https://www.caelum.com.br/apostila-java-orientacao-objetos

http://www.javaprogressivo.net/2012/09/classe-math-constantes-principais.html

https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

https://www.devmedia.com.br/numeros-aleatorios-em-java-a-classe-java-util-random/26355

http://www.tiexpert.net/programacao/java/string.php

http://www.tiexpert.net/programacao/java/math.php

Material desenvolvido pelo
Prof. Rafael da Silva Polato
rafael.polato@etec.sp.gov.br