java.lang

Alexandre Mendonça Fava
alexandre.fava@udesc.br

Universidade do Estado de Santa Catarina – UDESC
Programa de Pós-graduação em Computação Aplicada – PPGCA

Roteiro

- `java.lang`
  - Comentários
  - Pacotes
  - Saída padrão
  - Tipos de dados
  - Operadores
    - Aritméticos
    - Relacionais
    - Lógicos
  - Declaração de variáveis
  - Estruturas de Controle
    - `if`
    - `if...else`
    - `switch`
  - Estruturas de Repetição
    - `while`
    - `do...while`
    - `for`
- Resumo
- Referências

Comentários

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           System.out.print("Olá, Mundo!");
6       }
7   }
8
9
10
11
12
13
14
```

Comentários

```
7  }
8  // Comentário de fim de linha
9  /*
10  * Comentários tradicionais
11  */
12 /**
13  * Comentários Javadoc
14  */
```

DEITEL, Paul. Java: como programar.

O Java fornece comentários de um terceiro tipo: **comentários Javadoc**. Esses são delimitados por `//**` e `*/`. O compilador ignora todo o texto entre os delimitadores. Os comentários no estilo Javadoc permitem-lhe incorporar a documentação do programa diretamente aos seus programas. Esses comentários são o formato de documentação Java preferido na indústria. O **programa utilitário `javadoc`** (parte do JDK) lê comentários Javadoc e os usa para preparar a documentação do programa no formato HTML.

java.lang

É um pacote que contém os recursos básicos da linguagem

É indispensável para executar qualquer programa Java

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          System.out.print("Olá, Mundo!");
6      }
7  }
```

java.lang

É um pacote que contém os recursos básicos da linguagem

É indispensável para executar qualquer programa Java

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
```

DEITEL, Paul. Java: como programar.

Um dos pontos fortes do Java é seu rico conjunto de classes predefinidas que você pode *reutilizar* em vez de "reinventar a roda". Essas classes são agrupadas em **pacotes** — *chamados de grupos de classes relacionadas* — e, coletivamente, são chamadas de **biblioteca de classes Java**, ou **Java Application Programming Interface** (**Java API**).

java.lang

Java Pacote = C Biblioteca

import java.lang

#include <stdio.h>

São mecanismos para a organização
e reutilização de código

java.lang

Java Pacote

import java.lang

É um esquema de diretórios responsável por organizar as **classes** fundamentais da linguagem Java

É possível importar pacotes inteiros ou apenas partes

C Biblioteca

#include <stdio.h>

É um arquivo responsável por gerenciar as **funções** da entrada e saída padrão da linguagem C

Todas as funções da biblioteca são importadas*

*após a compilação apenas as funções utilizadas são mantidas

java.lang

Java Pacote

import java.lang

C Biblioteca

#include <stdio.h>

SANTOS, Rafael. **Introdução à Programação Orientada a Objetos usando Java.**

Classes pertencentes aos pacotes devem ser obrigatoriamente importadas por classes que as queiram usar, através da palavra-chave `import`. Uma exceção são as classes pertencentes ao pacote `java.lang`, que são chamadas as classes-base de Java. Estas classes não precisam ser importadas para que possam ser usadas.

É possível importar pacotes inteiros ou apenas partes

Todas as funções da biblioteca são importadas*

*após a compilação apenas as funções utilizadas são mantidas

java.lang

Java Pacote

import java.lang

System

C Biblioteca

#include <stdio.h>

scanf()

printf()

java.lang

Java Pacote

import java.lang

System

Biblioteca

#include <stdio.h>

fgetc() fclose()
puts()
getc()
fread()
tmpfile() sprintf()
scanf() putc() fwrite()
fdopen() ungetc() perror()
printf() tmpnam() fgets()
remove() sscanf() fprintf()
fscanf() fputc() ferror(
fread() fsetpos() fflush()
vscanf() putchar() ftell()
fseek() vsscanf() setbuf()
fopen() getchar() vprintf()

java.lang

Process Enum Class
Thread
Void
Runtime
ClassLoader Character
ClassValue Compiler String
StrictMath StringBuilder
SecurityManager System
StackTraceElement Number
Long StringBuffer Math
InheritableThreadLocal
ThreadLocal ProcessBuilder
Throwable Package Object
Double RuntimePermission

Java Pacote

import java.lang

fgetc() fclose()
puts()
getc()
fread()
tmpfile() sprintf()
scanf() putc() fwrite()
fdopen() ungetc() perror()
printf() tmpnam() fgets()
remove() sscanf() fprintf()
fscanf() fputc() ferror(
fread() fsetpos() fflush()
vscanf() putchar() ftell()
fseek() vsscanf() setbuf()
fopen() getchar() vprintf()

C Biblioteca

#include <stdio.h>

java.lang

https://docs.oracle.com/javase/8/docs/api/java/lang/package-summary.html

Class Summary

Class	Description
Boolean	The Boolean class wraps a value of the primitive type `boolean` in an object.
Byte	The `Byte` class wraps a value of primitive type `byte` in an object.
Character	The `Character` class wraps a value of the primitive type `char` in an object.
Character.Subset	Instances of this class represent particular subsets of the Unicode character set.
Character.UnicodeBlock	A family of character subsets representing the character blocks in the Unicode specification.
Class<T>	Instances of the class `Class` represent classes and interfaces in a running Java application.
ClassLoader	A class loader is an object that is responsible for loading classes

java.lang

	trigonometric functions.
String	The `String` class represents character strings.
StringBuffer	A thread-safe, mutable sequence of characters.
StringBuilder	A mutable sequence of characters.
System	The `System` class contains several useful class fields and methods.
Thread	A *thread* is a thread of execution in a program.
ThreadGroup	A thread group represents a set of threads.
ThreadLocal<T>	This class provides thread-local variables.
Throwable	The `Throwable` class is the superclass of all errors and exceptions in the Java language.
Void	The `Void` class is an uninstantiable placeholder class to hold a reference to the `Class` object representing the Java keyword void.

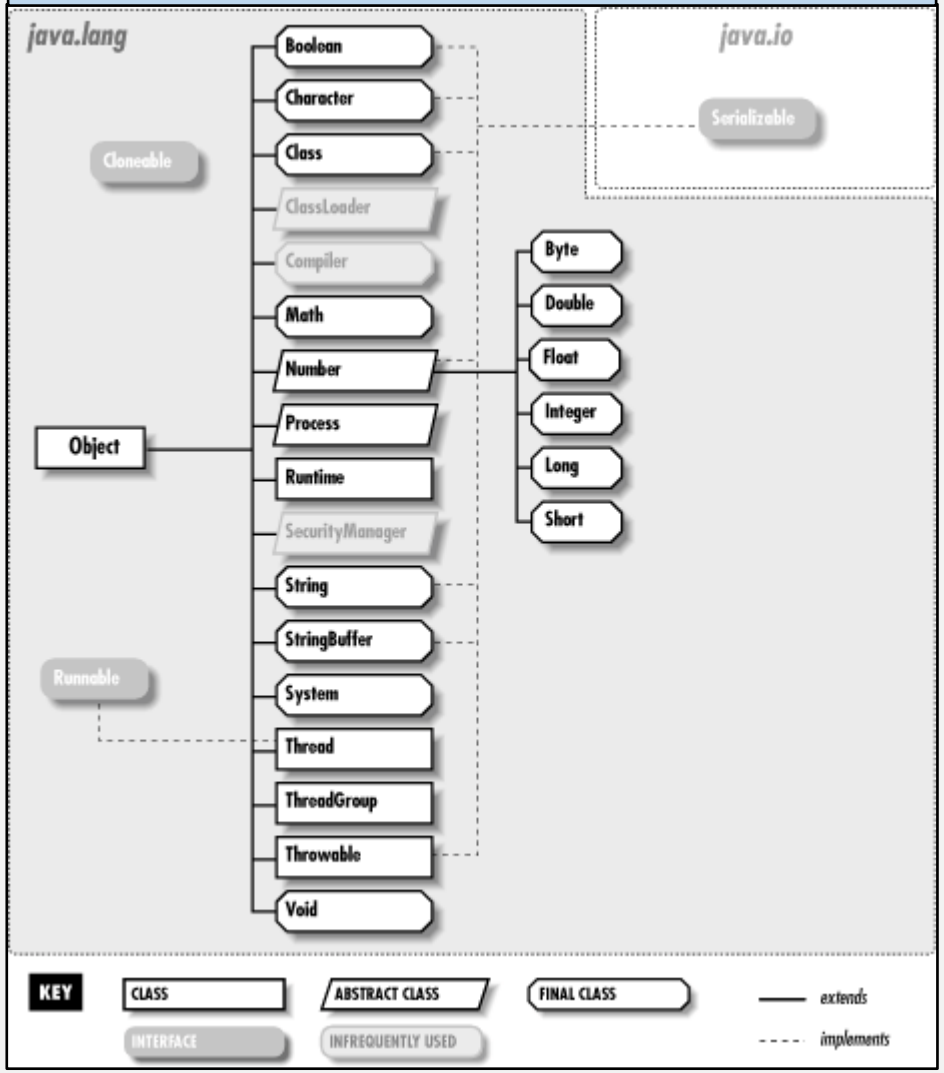
java.lang

DEITEL, Paul. **Java: como programar.**

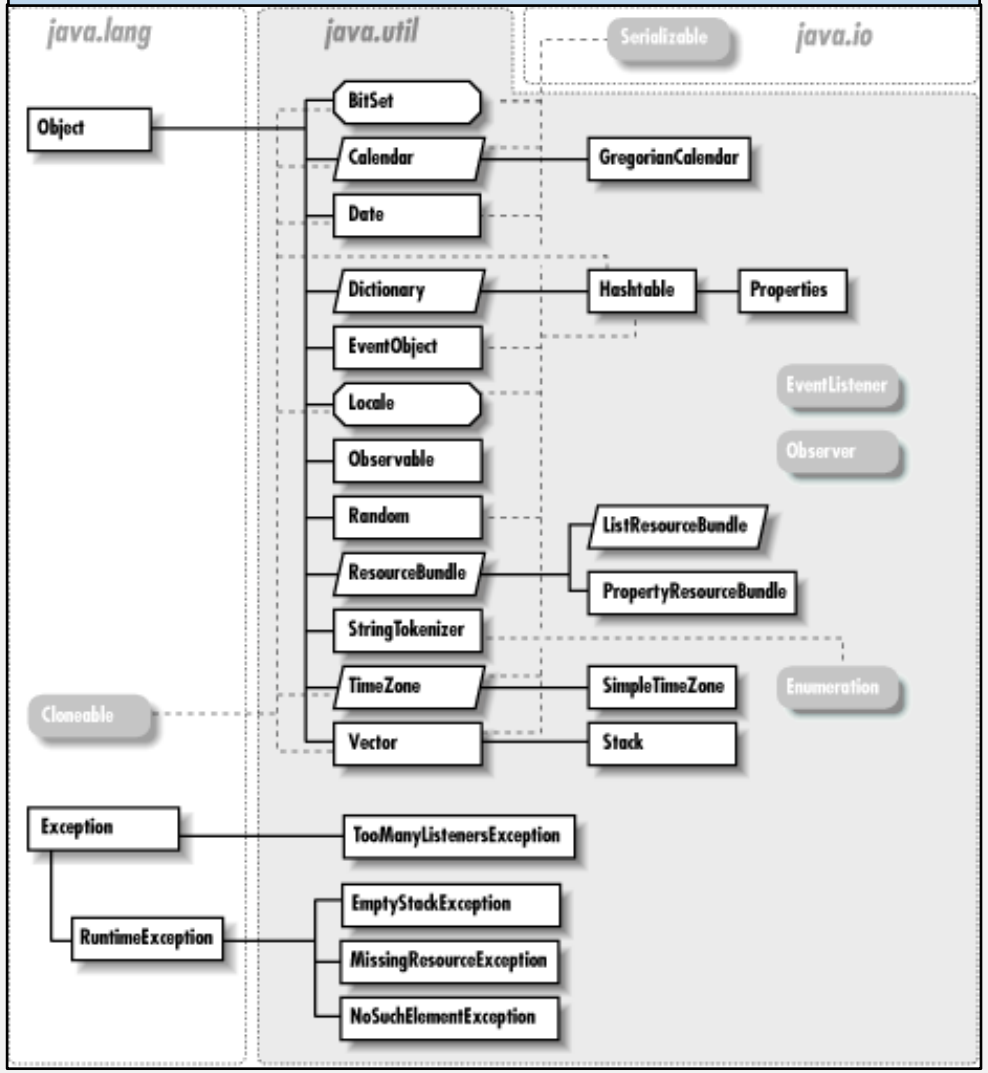
Pacote	Descrição
`java.awt.event`	O **Java Abstract Window Toolkit Event Package** contém classes e interfaces que permitem tratamento de evento para componentes GUI em nos pacotes `java.awt` e `javax.swing` (Veja o Capítulo 12, "Componentes GUI: parte 1", e o Capítulo 22, "Componentes GUI: parte 2".)
`java.awt.geom`	O **Java 2D Shapes Package** contém classes e interfaces para trabalhar com as avançadas capacidades gráficas bidimensionais do Java. (Veja o Capítulo 13, "Imagens gráficas e Java 2D".)
`java.io`	O **Java Input/Output Package** contém classes e interfaces que permitem aos programas gerar entrada e saída de dados. (Veja o Capítulo 15, "Arquivos, fluxos e serialização de objetos".)
`java.lang`	O **Java Language Package** contém classes e interfaces (discutidas por todo este texto) que são exigidas por muitos programas Java. Esse pacote é importado pelo compilador em todos os programas.
`java.net`	O **Java Networking Package** contém classes e interfaces que permitem aos programas comunicar-se via redes de computadores, como a internet. (Veja o Capítulo 28, em inglês, na Sala Virtual do livro.)
`java.security`	O **Java Security Package** contém classes e interfaces para melhorar a segurança do aplicativo.
`java.sql`	O **JDBC Package** contém classes e interfaces para trabalhar com bancos de dados. (Veja o Capítulo 24, "Acesso a bancos de dados com JDBC".)

java.lang

https://www.oreilly.com/



https://www.oreilly.com/



Os pacotes são um recurso tanto organizacional, quanto de controle de acesso

Saída padrão

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          System.out.print("Olá, Mundo!");
6      }
7  }
```

Saída padrão

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          System.out.print("Olá, Mundo!");
6      }
7  }
```

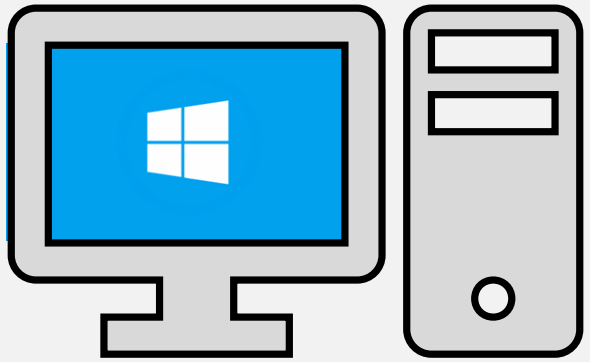
SCHILDT, Herbert. Java para iniciantes.

A linha começa com **System.out**. Embora seja muito complicada para explicarmos com detalhes nesse momento, em resumo, **System** é uma classe predefinida que dá acesso ao sistema, e **out** é o fluxo de saída que está conectado ao console. Portanto, **System.out** é um objeto que encapsula a saída do console. O fato de o Java usar um objeto para definir a saída do console é mais uma evidência de sua natureza orientada a objetos.

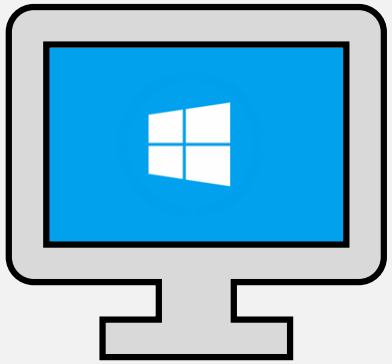
Saída padrão

Código (Main.java)

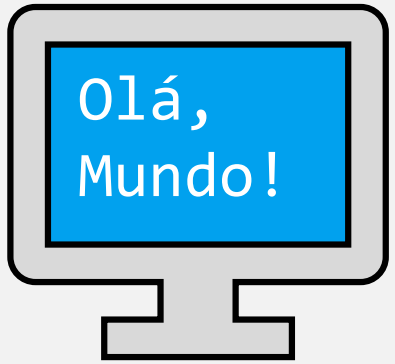
```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          System.out.print("Olá, Mundo!");
6      }
7  }
```



Sistema



Sistema.saída



Sistema.saída.imprima

Saída padrão

DEITEL, Paul. **Java: como programar.**

Sequência de escape	Descrição
`\n`	Nova linha. Posiciona o cursor de tela no início da *próxima* linha.
`\t`	Tabulação horizontal. Move o cursor de tela para a próxima parada de tabulação.
`\r`	Retorno de carro. Posiciona o cursor da tela no início da linha *atual* — *não* avança para a próxima linha. Qualquer saída de caracteres depois do retorno de carro *sobrescreve* a saída de caracteres anteriormente gerada na linha atual.
`\\`	Barras invertidas. Utilizadas para imprimir um caractere de barra invertida.
`\"`	Aspas duplas. Utilizadas para imprimir um caractere de aspas duplas. Por exemplo, `System.out.println("\"entre aspas\"");` exibe `"entre aspas"`.

Figura 2.5 | Algumas sequências de escape comuns.

Saída padrão

DEITI … amar.

Sequência de escape	
\n	na.
\t	ada de tabulação.
\r	tual — *não* avança para a le carro *sobrescreve* a saída de
\\	ra invertida.
\"	uplas. Por exemplo,

Figura 2.5 | Algumas sequênc…

https://docs.oracle.com/javase/tutorial/java/data/characters.html

Sequência de escape	Descrição
\t	Insira uma guia no texto neste ponto.
\b	Insira um backspace no texto neste ponto.
\n	Insira uma nova linha no texto neste ponto.
\r	Insira um retorno de carro no texto neste ponto.
\f	Insira um feed de formulário no texto neste ponto.
\'	Insira um único caractere de aspas no texto neste ponto.
\"	Insira um caractere de aspas duplas no texto neste ponto.
\\	Insira um caractere de barra invertida no texto neste ponto.

Saída padrão

SCHILDT, Herbert. **Java para iniciantes.**

Tabela 2-2 **Sequências de escape de caracteres**

Sequência de escape	Descrição
\'	Aspas simples
\"	Aspas duplas
\\	Barra invertida
\r	Retorno de carro
\n	Nova linha
\f	Avanço de página
\t	Tabulação horizontal
\b	Retrocesso
ddd	Constante octal (onde *ddd* é uma constante octal)
uxxxx	Constante hexadecimal (onde *xxxx* é uma constante hexadecimal)

Saída padrão: Exemplos

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5         System.out.print("Linha 1\n");
6         System.out.println("Linha 2\n");
7         System.out.print("Linha \t3\n");
8         System.out.print("Linha 4");
9         System.out.print("\"Linha 5\"");
10        System.out.print("\n\nLinha 6");
11        System.out.printf("%n%s%n%s", "Linha", "7");
12      }
    }
```

Saída

Saída padrão: Exemplos

Código (Main.java)

```java
import java.lang.System;

public class Main{
    public static void main(String[] args){
      System.out.print("Linha 1\n");
      System.out.println("Linha 2\n");
      System.out.print("Linha \t3\n");
      System.out.print("Linha 4");
      System.out.print("\"Linha 5\"");
      System.out.print("\n\nLinha 6");
      System.out.printf("%n%s%n%s", "Linha", "7");
    }
}
```

Saída

```
Linha 1
Linha 2

Linha      3
Linha 4"Linha 5"

Linha 6
Linha
7
```

Saída padrão: Exemplos

Código (Main.java)

```
1
2
3
4
5
6
7
8   System.out.print("Linha 4");
9   System.out.print("\"Linha 5\"");
10  System.out.print("\n\nLinha 6");
11  System.out.printf("%n%s%n%s", "Linha", "7");
12  }
}
```

Saída

```
a 5"
Linha
7
```

DEITEL, Paul. **Java: como programar.**

O método `System.out.printf` (f significa "formato") exibe os dados *formatados*. Especificadores de formato iniciam com um sinal de porcentagem (%) seguido por um caractere que representa o *tipo de dados*. Por exemplo, o especificador de formato `%s` é um marcador de lugar para uma string.

Tipo de dados

O que é um dado?

O que é um dado?

Um dado é uma
valor ou atributo
sem significado

Tipo de dados

PINTO, Rafael; et. al. **Estrutura de Dados.**

Você sabe a diferença entre dados e informação?

Tipo de dados

- Número Inteiro
- Cadeia de Caracteres
- Número Fracionário
- Comando Lógico

*existem outros mas esses são os principais

Console

```
>: 10
>: JAVA
>: 10.5
>: ESC
```

Tipo de dados

- Número Inteiro
- Cadeia de Caracteres
- Número Fracionário
- Comando Lógico

*existem outros mas esses são os principais

Console

```
>: 10
>: JAVA
>: 10.5
>: ESC
```

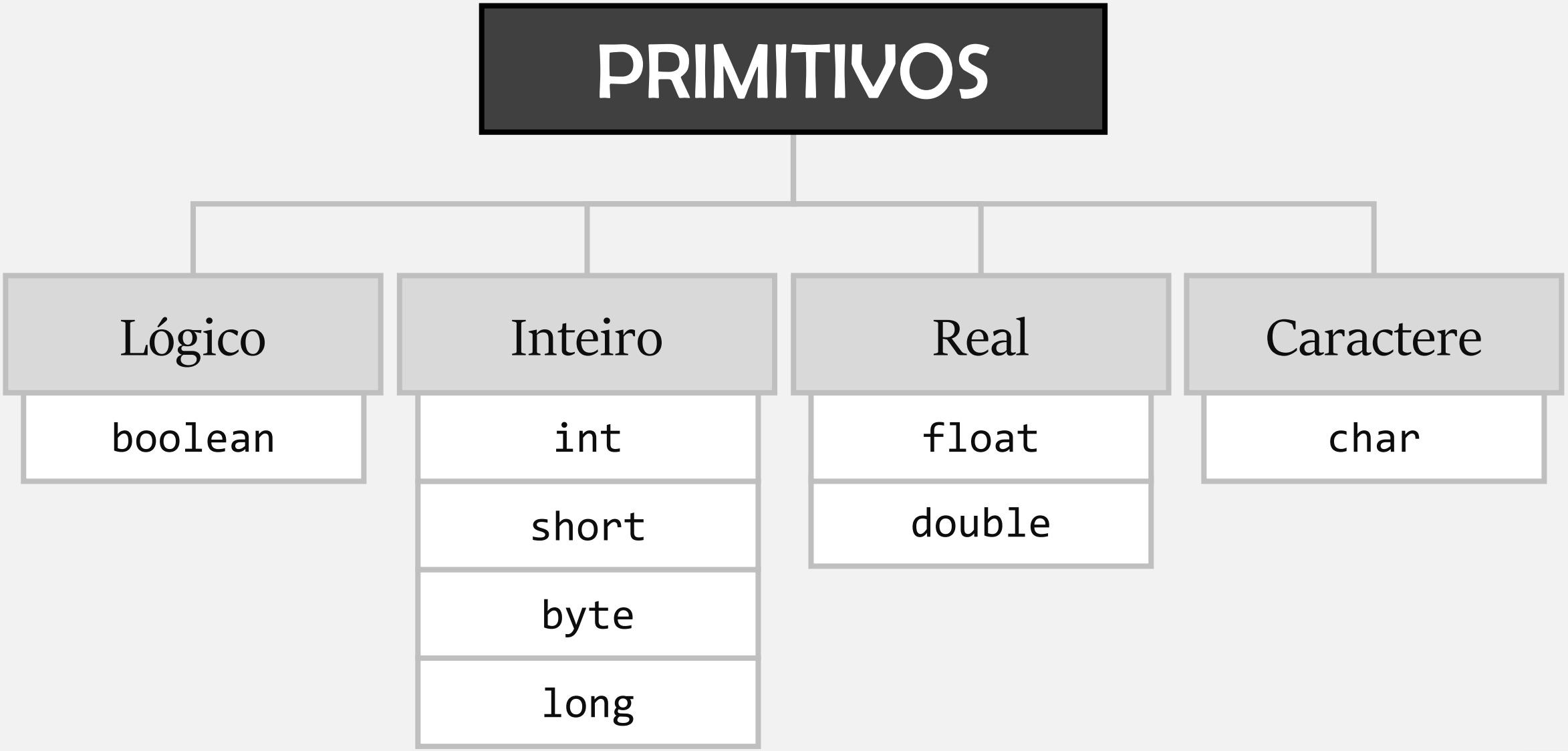
SCHILDT, Herbert. Java para iniciantes.

Os tipos de dados são especialmente importantes em Java porque essa é uma linguagem fortemente tipada. Ou seja, todas as operações têm a compatibilidade de seus tipos verificada pelo compilador. Operações inválidas não serão compiladas.

Tipo de dados

PRIMITIVOS	ORIENTADOS A OBJETOS
	String
	Array
	List
	...

Tipo de dados



Tipo de dados

DEITEL, Paul. Java: como programar.

Os tipos do Java são divididos em primitivos e **por referência**.

Os tipos `int`, `float`, `double` e `char` são chamados de **tipos primitivos**.

Os nomes dos tipos primitivos são palavras-chave e devem aparecer em letras minúsculas.

Tipo

Tipo	Tamanho em bits	Valores	Padrão
`boolean`		`true` ou `false`	
[*Observação*: a representação de um `boolean` é específica à Java Virtual Machine em cada plataforma.]			
`char`	16	`'\u0000'` a `'\uFFFF'` (0 a 65535)	(conjunto de caracteres Unicode ISO)
`byte`	8	–128 a +127 (-2^7 a $2^7 - 1$)	
`short`	16	–32.768 a +32.767 (-2^{15} a $2^{15} - 1$)	
`int`	32	–2.147.483.648 a +2.147.483.647 (-2^{31} a $2^{31} - 1$)	
`long`	64	–9.223.372.036.854.775.808 a +9.223.372.036.854.775.807 (-2^{63} a $2^{63} - 1$)	
`float`	32	*Intervalo negativo:* –3,4028234663852886E+38 a –1,40129846432481707e–45 *Intervalo positivo:* 1,40129846432481707e–45 a 3,4028234663852886E+38	(IEEE 754, ponto flutuante)
`double`	64	*Intervalo negativo:* –1,7976931348623157E+308 a –4,9406564584124654 4e–324 *Intervalo positivo:* 4,9406564584124654 4e–324 a 1,7976931348623157E+308	(IEEE 754, ponto flutuante)

Figura D.1 | Tipos primitivos do Java.

Lóg ... ere

bool ...

Os tipos do ...

Os tipos in ...

Os nomes ... úsculas.

Tipo

Tipo	Tamanho em bits	Valores	Padrão
`boolean`		`true` ou `false`	
[*Observação*: a representação de um `boolean` é específica à Java Virtual Machine em cada plataforma.]			
`char`	16	`'\u0000'` a `'\uFFFF'` (0 a 65535)	(conjunto de caracteres Unicode ISO)
`byte`	8	–128 a +127 (-2^7 a $2^7 - 1$)	
`short`	16	–32.768 a +32.767 (-2^{15} a $2^{15} - 1$)	
`int`	32	–2.147.483.648 a +2.147.483.647 (-2^{31} a $2^{31} - 1$)	
`long`	64	–9.223.372.036.854.775.808 a +9.223.372.036.854.775.807 (-2^{63} a $2^{63} - 1$)	
`float`	32	*Intervalo negativo:* –3,4028234663852886E+38 a –1,40129846432481707e–45	(IEEE 754, ponto flutuante)
		Intervalo positivo: 1,40129846432481707e–45 a	

Lóg ... ere

bool ...

SCHILDT, Herbert. Java para iniciantes.

Como a tabela mostra, todos os tipos inteiros são valores de sinal positivo e negativo. Java não suporta inteiros sem sinal (somente positivos). Outras linguagens de computador suportam inteiros com e sem sinal. No entanto, os projetistas de Java decidiram que inteiros sem sinal eram desnecessários.

Tipo de dados

Tipo	Tamanho	Limite	Inicialização Padrão
boolean	1 bit* *depende do sistema operacional (JVM)	true – false	false
byte	1 byte	-2^{7} – $2^{7}-1$	0
char	2 bytes	0 – $2^{16}-1$	'\u0000'
short	2 bytes	-2^{15} – $2^{15}-1$	0
int	4 bytes	-2^{31} – $2^{31}-1$	0
float	4 bytes	-2^{149} – $(2-2^{-23})\cdot 2^{127}$	0.0f
long	8 bytes	-2^{63} – $-2^{63}-1$	0L
double	8 bytes	-2^{1074} – $(2-2^{-52})\cdot 2^{1023}$	0.0d

Tipo de dados

Tipo	Tamanho	Limite	Inicialização Padrão
int	[illegible]	[illegible]	[illegible]
float	4 bytes	$-2^{149} - (2-2^{-23}) \cdot 2^{127}$	0.0f
long	8 bytes	$-2^{63} - -2^{63}-1$	0L
double	8 bytes	$-2^{1074} - (2-2^{-52}) \cdot 2^{1023}$	0.0d

SCHILDT, Herbert. **Java para iniciantes.**

P: **Por que Java tem tipos de dados diferentes para inteiros e valores de ponto flutuante? Isto é, por que não são todos valores numéricos do mesmo tipo?**

R:

O que é um operador?

Operadores são símbolos especiais que executam operações específicas sobre um certo conjunto de operandos, de modo a retornar algum resultado

Operadores

SCHILDT, Herbert. **Java para iniciantes.**

Java fornece um ambiente rico em operadores. Um *operador* é um símbolo que solicita ao compilador que execute uma operação matemática ou lógica específica. Java tem quatro classes gerais de operadores: aritmético, bitwise, relacional e lógico. Também define alguns operadores adicionais que tratam certas situações especiais.

Operadores são símbolos especiais que executam operações específicas sobre um certo conjunto de operandos, de modo a retornar algum resultado

Operadores: Aritméticos

ADIÇÃO (+)

MULTIPLICAÇÃO (*)

SUBTRAÇÃO (-)

DIVISÃO (/)

MÓDULO OU RESTO (%)

Operadores: Aritméticos

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5         System.out.println(5+3); // ADIÇÃO
6         System.out.println(5*3); // MULTIPLICAÇÃO
7         System.out.println(5-3); // SUBTRAÇÃO
8         System.out.println(5/3); // DIVISÃO
9         System.out.println(5%3); // RESTO
10
11
12      }
13  }
```

Saída

Operadores: Aritméticos

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5         System.out.println(5+3); // ADIÇÃO
6         System.out.println(5*3); // MULTIPLICAÇÃO
7         System.out.println(5-3); // SUBTRAÇÃO
8         System.out.println(5/3); // DIVISÃO
9         System.out.println(5%3); // RESTO
10        System.out.println(5.0/3.0); // DIVISÃO
11        System.out.println(0.1+0.2); // ADIÇÃO
12      }
13  }
```

Saída

```
8
15
2
1
2
1.6666666666666667
0.30000000000000004
```

Operadores: Relacionais

MAIOR QUE (>)

MENOR QUE (<)

MAIOR OU IGUAL (>=)

MENOR OU IGUAL (<=)

IGUAL A (==)

DIFERENTE (!=)

Operadores: Relacionais

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5         System.out.println(5>3);  // MAIOR
6         System.out.println(5<5);  // MENOR
7         System.out.println(5>=3); // MAIOR IGUAL
8         System.out.println(5<=3); // MENOR IGUAL
9         System.out.println(5==3); // IGUAL
10        System.out.println(5!=3); // DIFERENTE
11
12      }
13  }
```

Saída

Operadores: Lógicos

E (&&)

OU (| |)

NÃO (!)

E (&)

OU (|)

OU EXCLUSIVO (^)

*bitwise

Operadores: Lógicos

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5         System.out.println(true && true);  // E
6         System.out.println(true && false); // E
7         System.out.println(true || true);  // OU
8         System.out.println(true || false); // OU
9         System.out.println(!true);  // NEGAÇÃO
10        System.out.println(!false); // NEGAÇÃO
11
12      }
13  }
```

Saída

Operadores: Lógicos *bitwise

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5        System.out.println(true & true);    // E
6        System.out.println(true & false);   // E
7        System.out.println(true | true);    // OU
8        System.out.println(true | false);   // OU
```

Saída

```
true
false
true
true
false
true
```

SCHILDT, Herbert. **Java para iniciantes.**

O operador AND de curto-circuito é **&&** e o operador OR de curto-circuito é ||. Seus equivalentes comuns são **&** e |. A única diferença entre as versões comum e de curto-circuito é que a versão comum sempre avalia cada operando e a versão de curto-circuito só avalia o segundo operando quando necessário.

Operadores: Lógicos *bitwise

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5        System.out.println(true ^ true);    // XOU
6        System.out.println(true ^ false);   // XOU
7        System.out.println(true | true);    // OU
8        System.out.println(true | false);   // OU
```

Saída

```
false
true
true
true
false
true
```

SCHILDT, Herbert. **Java para iniciantes.**

O operador AND de curto-circuito é **&&** e o operador OR de curto-circuito é ||. Seus equivalentes comuns são **&** e |. A única diferença entre as versões comum e de curto-circuito é que a versão comum sempre avalia cada operando e a versão de curto-circuito só avalia o segundo operando quando necessário.

Operadores: Precedência

maior

precedência

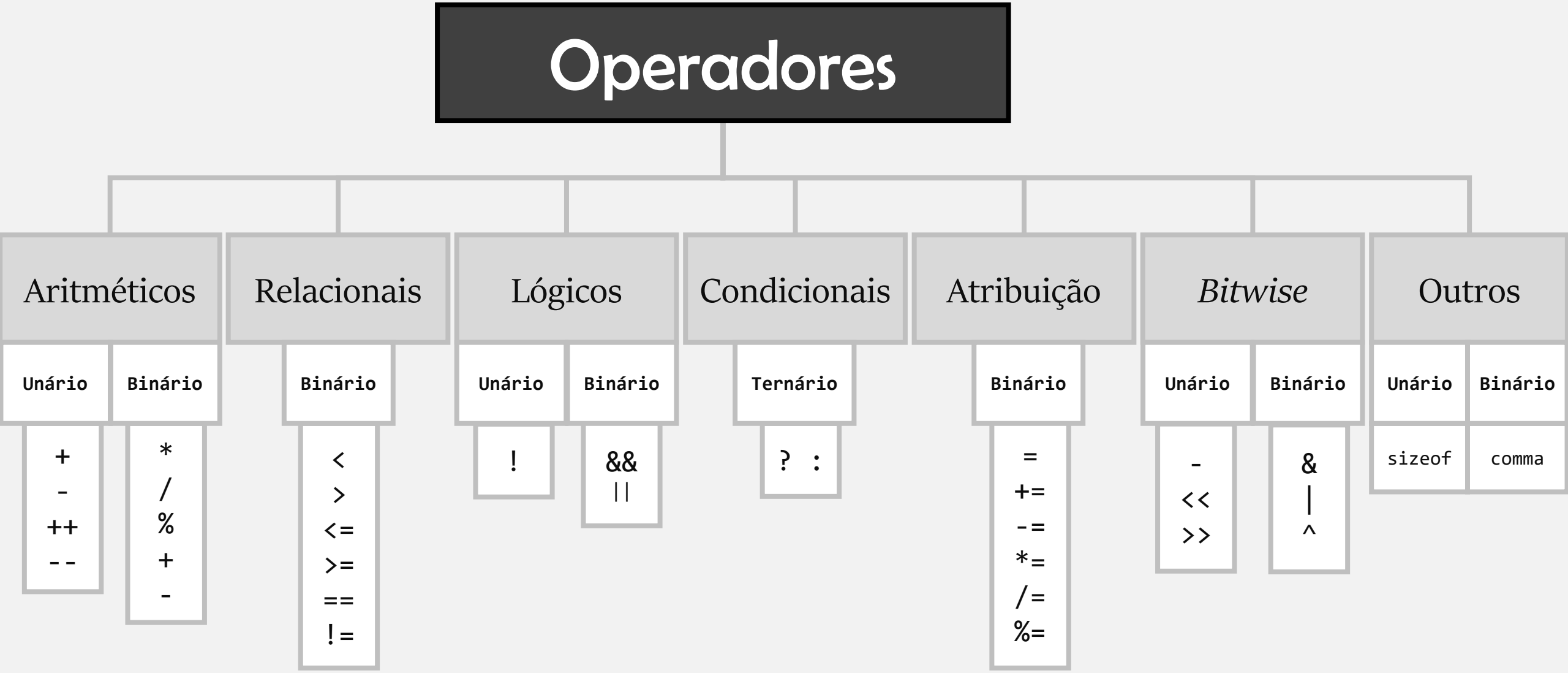
menor

Level	Operators	Description	Associativity
15	() [] -> . ++ --	Function Call Array Subscript Member Selectors Postfix Increment/Decrement	Left to Right
14	++ -- + - ! ~ (type) * & sizeof	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting Dereferencing Address of Find size in bytes	Right to Left
13	* / %	Multiplication Division Modulo	Left to Right
12	+ -	Addition / Subtraction	Left to Right
11	>> <<	Bitwise Right Shift Bitwise Left Shift	Left to Right
10	< <= > >=	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To	Left to Right
9	== !=	Equality Inequality	Left to Right
8	&	Bitwise AND	Left to Right
7	^	Bitwise XOR	Left to Right
6	|	Bitwise OR	Left to Right
5	&&	Logical AND	Left to Right
4	||	Logical OR	Left to Right
3	?:	Conditional Operator	Right to Left
2	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operators	Right to Left
1	,	Comma Operator	Left to Right

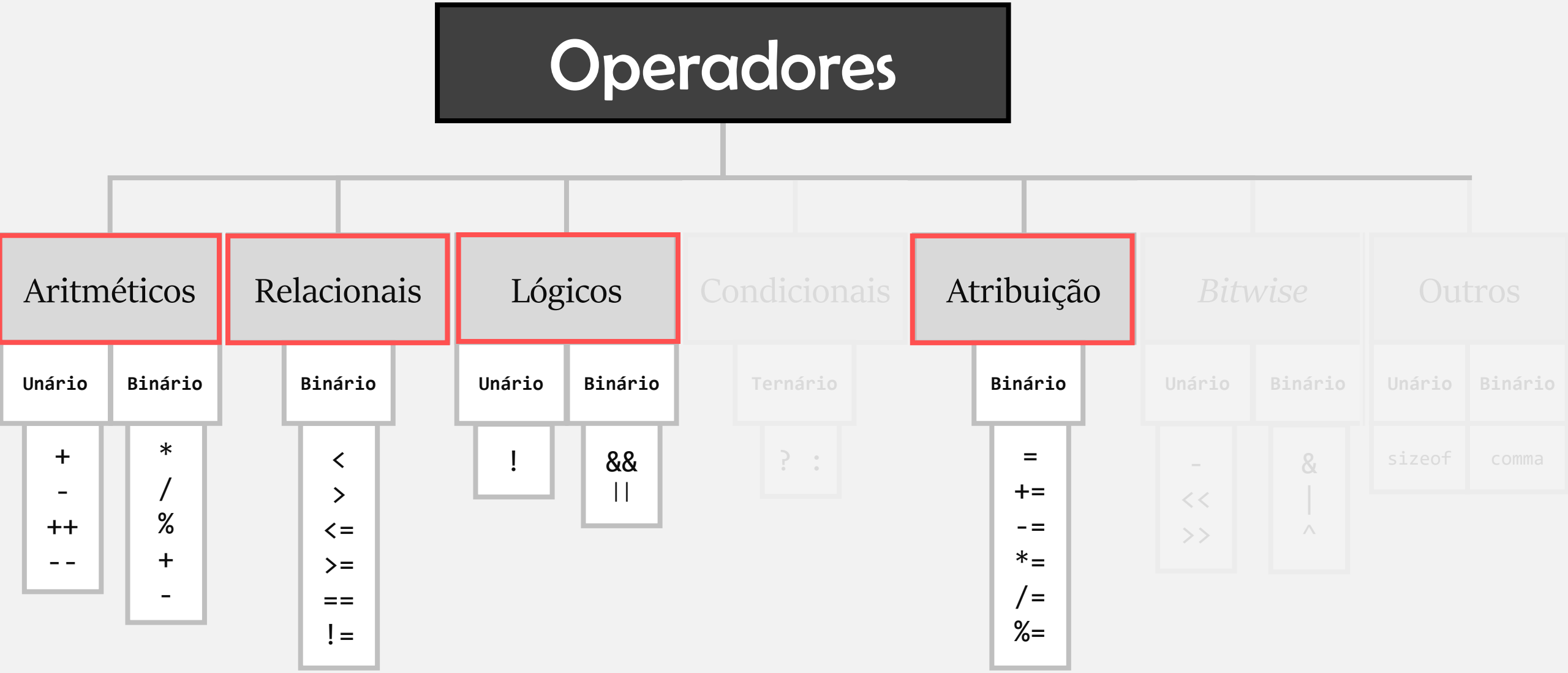
Java

Level	Operators	Description	Associativity
15	() [] .	Function Call Array Subscript Member Selection	Left to Right
14	++ --	Postfix Increment / Decrement	Right to Left
13	++ -- + - ! ~ (type)	Prefix Increment / Decrement Unary plus / minus Logical negation / bitwise complement Casting	Right to Left
12	* / %	Multiplication Division Modulo	Left to Right
11	+ -	Addition / Subtraction	Left to Right
10	<< >> >>>	Bitwise Left Shift Bitwise Right Shift with sign extension Bitwise Right Shift with zero extension	Left to Right
9	< <= > >= instance of	Relational Less Than / Less than Equal To Relational Greater / Greater than Equal To Type Comparison for objects	Left to Right
8	== !=	Equality Inequality	Left to Right
7	&	Bitwise AND	Left to Right
6	^	Bitwise XOR	Left to Right
5	|	Bitwise OR	Left to Right
4	&&	Logical AND	Left to Right
3	||	Logical OR	Left to Right
2	?:	Conditional Operator	Right to Left
1	= += -= *= /= %= &= ^= |= <<= >>=	Assignment Operators	Right to Left

Operadores: Resumo



Operadores: Resumo



Declaração de Variáveis

Java proíbe o uso de suas palavras-chave na declaração de nomes de variáveis

SCHILDT, Herbert. **Java para iniciantes.**

Tabela 1-1 **As palavras-chave Java**

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

Declaração de Variáveis

DEITEL, Paul. **Java: como programar.**

Palavras-chave do Java

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

Palavras-chave que não são atualmente utilizadas

const	goto

Figura C.1 | Palavras-chave do Java.

Declaração de Variáveis

Case Name		Example
	Camel	camelCase
	Pascal	PascalCase
	Snake	snake_case
	Kebab	kebab-case

Declaração de Variáveis

É uma convenção, NÃO UMA REGRA

Case Name	Example
Camel	camelCase
Pascal	PascalCase

Normalmente usada em:

- nomes de classes
- nomes de método
- nomes de variável

DEITEL, Paul. **Java: como programar.**

Por convenção, identificadores de nomes de variáveis iniciam com uma letra minúscula e cada palavra no nome depois da primeira começa com uma letra maiúscula. Por exemplo, o identificador de nome da variável `firstNumber` *inicia a sua segunda palavra,* `Number`*, com uma letra* `N` *maiúscula. Essa convenção de nomenclatura é conhecida como* ***notação camelo****, porque as letras maiúsculas destacam-se como corcovas desse animal.*

Declaração de Variáveis

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          int variavel = 10;
6          System.out.print("Olá, Mundo!");
7      }
8  }
```

Variáveis guardam **literais**

Declaração de Variáveis

Variáveis guardam **literais**

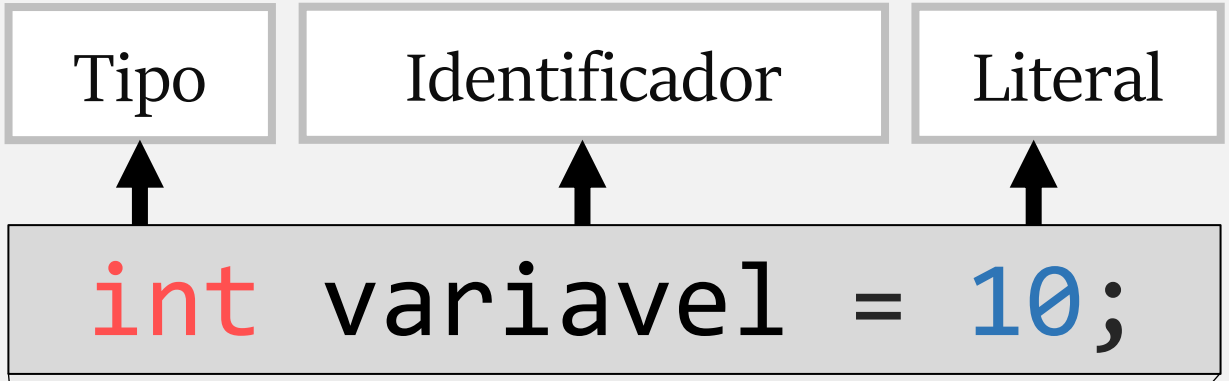
Variáveis são nomes simbólicos definidos pelo programador que serão associados a **endereços de memória**

Literais são os valores numéricos, textuais ou lógicos, normalmente atribuídos a variáveis ou constantes

SCHILDT, Herbert. **Java para iniciantes.**

Em Java, os *literais* são valores fixos representados em sua forma legível por humanos. Por exemplo, o número 100 é um literal. Normalmente, os literais também são chamados de *constantes*.

Declaração de Variáveis



Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          int variavel = 10;
6          System.out.print("Olá, Mundo!");
7      }
8  }
```

Declaração de Variáveis

Java diferencia maiúsculas de minúsculas

Tipo | Identificador | Literal

```
int variavel = 10;
```

Variáveis devem ser declaradas antes de serem usadas

Código (Main.java)

SCHILDT, Herbert. **Java para iniciantes.**

Em Java, um identificador é o nome dado a um método, uma variável ou qualquer outro item definido pelo usuário. Os identificadores podem ter de um a vários caracteres. Os nomes de variável podem começar com qualquer letra do alfabeto, um sublinhado ou um cifrão. Em seguida pode haver uma letra, um dígito, um cifrão ou um sublinhado.

Declaração de Variáveis

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int     variavel_1 = 10;
6           byte    variavel_2 = 10;
7           short   variavel_3 = 10;
8           float   variavel_4 = 10.0F;
9           double  variavel_5 = 10.0D;
10          long    variavel_6 = 10L;
11          char    variavel_7 = 'A';
12          boolean variavel_8 = true;
13      }
14  }
```

Declaração de Variáveis

*coerção (ou *casting*)

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int     variavel_1 = (int)    10;
6           byte    variavel_2 = (byte)   10;
7           short   variavel_3 = (short)  10;
8           float   variavel_4 = (float)  10.0;
9           double  variavel_5 = (double) 10.0;
10          long    variavel_6 = (long)   10;
11          char    variavel_7 = (char)   'A';
12          boolean variavel_8 = (boolean) true;
13      }
14  }
```

Declaração de Variáveis

*coerção (ou *casting*)

De: \ Para:	byte	char	short	int	long	float	double
byte		(char)	implícito	implícito	implícito	implícito	implícito
char	(byte)		(short)	implícito	implícito	implícito	implícito
short	(byte)	(char)		implícito	implícito	implícito	implícito
int	(byte)	(char)	(short)		implícito	implícito	implícito
long	(byte)	(char)	(short)	(int)		implícito	implícito
float	(byte)	(char)	(short)	(int)	(long)		implícito
double	(byte)	(char)	(short)	(int)	(long)	(float)	

Algumas conversões podem gerar perda de precisão

Estruturas

❑ Estruturas de Controle

SCHILDT, Herbert. **Java para iniciantes.**
Você pode executar seletivamente parte de um programa com o uso da instrução condicional de Java: a instrução **if**. A instrução **if** da linguagem Java funciona de maneira semelhante à instrução IF de qualquer outra linguagem.

❑ Estruturas de Repetição

SCHILDT, Herbert. **Java para iniciantes.**
Você pode executar repetidamente uma sequência de código criando um *laço*. Java funciona de maneira semelhante à instrução **for** de qualquer outra linguagem.

Estruturas de Controle

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6           if(x > 6){
7               System.out.println("1");
8           }else{
9               if(y < 8){
10                  System.out.println("2");
11              }else{
12                  System.out.println("3");
13              }
14          }
```

Saída

```
3
```

Estruturas de Controle

A solução dos `if...else` aninhados

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6           if(x > 6)
7               System.out.println("1");
8           else if(y < 8)
9               System.out.println("2");
10          else
11              System.out.println("3");
12
13      }
14  }
```

Saída

```
3
```

Estruturas de Controle

Problema do `else` oscilante

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6           if(x > 3)
7               if(y < 8)
8                   System.out.println("AQUI");
9           else
10              System.out.println("ALI");
11
```

A indentação pode confundir

Saída

```
ALI
```

SCHILDT, Herbert. **Java para iniciantes.**

Java é uma linguagem de forma livre, ou seja, não importa onde inserimos as instruções em uma linha em relação umas às outras.

Estruturas de Controle

Código (Main.java)

```
1  import java.lang.System;
2
3  public class Main{
4      public static void main(String[] args){
5          int x = 5, y = 10;
           switch(x){
               case 5:
                   System.out.println("Faz");
9                  break;
10
11
12
13
14
```

Sem isso, ocorre falling through

Saída

```
Faz
```

DEITEL, Paul. Java: como programar.

A instrução `break`, quando executada em um `while`, `for`, `do...while` ou `switch`, ocasiona a saída *imediata* dessa instrução.

Estruturas de Repetição

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6
7           while(x < y){
8               x = x+1;
9           }
10
11          do{
12              x++;
13          }while(x < y);
14
```

Testar antes de executar o bloco de código

Executar o bloco de código, depois testar

Estruturas de Repetição

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6
7           for(;;){
8               x = x+1;
9           }
10
11
12
13
14
```

DEITEL, Paul. **Java: como programar.**

Todas as três expressões em um cabeçalho `for` são opcionais.

Estruturas de Repetição

```
5        int x = 5, y = 10;
6
7        for(;;){
8            x = x+1;
9        }
10
11
12
13
14
```

DEITEL, Paul. Java: como programar.

Todas as três expressões em um cabeçalho `for` são opcionais.

SCHILDT, Herbert. Java para iniciantes.

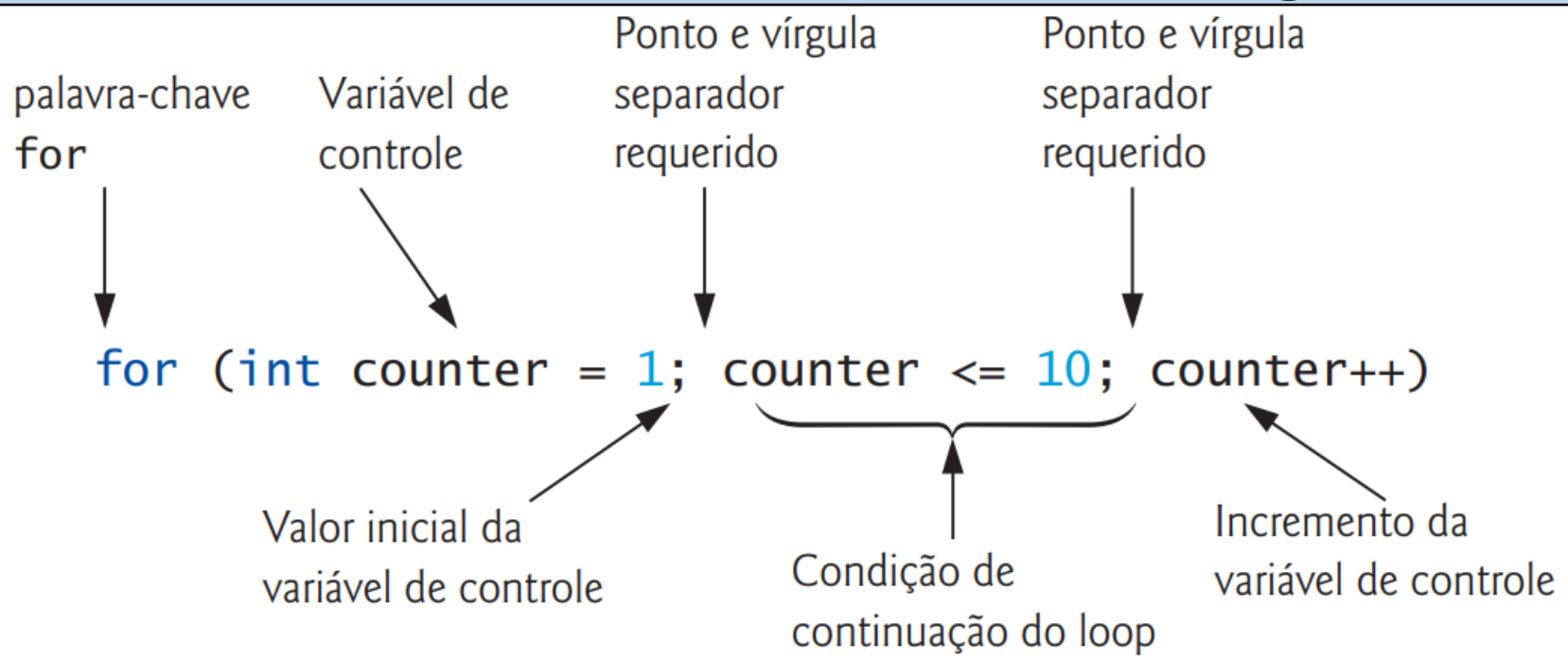
Algumas variações interessantes do laço **for** são criadas quando deixamos vazias partes da definição do laço. Em Java, podemos deixar algumas ou todas as partes referentes à inicialização, condição ou iteração do laço **for** em branco.

Estruturas de Repetição

SCHILDT, Herbert. Java para iniciantes.

Algumas variações interessantes do laço **for** são criadas quando deixamos vazias partes da definição do laço. Em Java, podemos deixar algumas ou todas as partes referentes à inicialização, condição ou iteração do laço **for** em branco.

DEITEL, Paul. Java: como programar.



Estruturas de Repetição

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int x = 5, y = 10;
6
7           for(x = 0, y = 0; (x<3 || y<5); x++, y++){
8               System.out.printf(x + " " + y + "\n");
9           }
10
11
12
13
14
```

Estruturas de Repetição

Código (Main.java)

```
1  import java.lang.System;
2
   class Main{
   blic static void main(String[] args){
       int x = 5, y = 10;

       for(x = 0, y = 0; (x<3 || y<5); x++, y++){
           System.out.printf(x + " " + y + "\n");
       }
10
14
```

Saída

```
0 0
1 1
2 2
3 3
4 4
```

SCHILDT, Herbert. **Java para iniciantes.**

O laço **for** é uma das instruções mais versáteis da linguagem Java porque permite muitas variações. Por exemplo, podem ser usadas diversas variáveis de controle.

Estruturas de Repetição

Código (Main.java)

```
1  import java.lang.System;
2
   class Main{
   blic static void main(String[] args){
       int x = 0;

       for(x = 0; x<5; x++){
           if (x%2 == 0){
               continue;
10         }
11         System.out.printf("Fazendo:" + x + "\n");
```

Saída

```
Fazendo:1
Fazendo:3
```

DEITEL, Paul. **Java: como programar.**

A instrução `continue`, quando executada em um `while`, `for` ou `do...while`, pula as instruções restantes no corpo do loop e prossegue com a *próxima iteração* do loop.

Estruturas de Repetição APRIMORADA

A instrução `for` aprimorada itera sobre cada um dos item de um arranjo sem extrapolar seu limite

DEITEL, Paul. **Java: como programar.**

7.7 A instrução for aprimorada

A **instrução for aprimorada** itera pelos elementos de um array *sem* usar um contador, evitando assim a possibilidade de "pisar fora" do array.

A sintaxe de uma instrução `for` aprimorada é:

```
for (parâmetro : nomeDoArray)
    instrução
```

Estruturas de Repetição APRIMORADA

Código (Main.java)

```
1   import java.lang.System;
2
3   public class Main{
4       public static void main(String[] args){
5           int[] arranjo = {87, 68, 94, 100};
6           int soma = 0;
7
8           for(int item : arranjo){
9               soma = soma + item;
10              //soma += item;
11          }
12  }
13
14
```

Atribuição Abreviada

+=	-=	*=	/=
%=	&=	|=	^=

SCHILDT, Herbert. Java para iniciantes.

Java fornece operadores especiais de atribuição *abreviada* que simplificam a codificação de certas instruções de atribuição. Comecemos com um exemplo. A instrução de atribuição mostrada aqui

```
x = x + 10;
```

pode ser escrita, com o uso da atribuição abreviada Java, como:

```
x += 10;
```

Resumo

- **Pacotes:** A base da linguagem é o `java.lang`, porém outros pacotes fornecem muitos outros **recursos adicionais**.
- **Linguagem livre:** Java não se preocupa com indentação.
- **Estruturas:** No Java tanto o `if`, `if...else` e `switch` quanto o `while`, `do...while` e `for` são praticamente idênticos a qualquer outra linguagem.
- **Fortemente tipada:** Quaisquer variáveis devem pertencer a **algum tipo**.
- **Sensível ao texto:** A linguagem Java diferencia **maiúsculas** de **minúsculas** (`case-sensitive`).
- **Declaração:** Além da declaração de variáveis, Java permite a declaração de objetos (**será visto em aulas futuras**).

Referências

SCHILDT, Herbert. **Java para iniciantes: crie, compile e execute**. 6. ed. Porto Alegre: Bookman, 2015. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/9788582603376>.

SILVA, Fabricio; LEITE, Márcia; OLIVEIRA, Diego. **Paradigmas de Programação**. Porto Alegre: SAGAH, 2019. Disponível em: <https://app.minhabiblioteca.com.br/reader/books/9788533500426>.

DEITEL, Paul. **Java: como programar**. 4. ed. Porto Alegre: Bookman, 2003.

DEITEL, Paul. **Java: como programar**. 10. ed. São Paulo: Pearson Education do Brasil, 2017.

java.lang

Alexandre Mendonça Fava
alexandre.fava@udesc.br

Universidade do Estado de Santa Catarina – UDESC
Programa de Pós-graduação em Computação Aplicada – PPGCA