

Lição 4



Fundamentos da programação

Objetivos

Ao final desta lição, o estudante será capaz de:

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



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
6     public static void main(String[] args) {
7         // Mostra a mensagem "Hello world" na tela
8         System.out.println("Hello world");
9     }
10 }
```



Entendendo meu primeiro programa em Java

```
1  public class Hello
2  {
3      /**
4       * Meu primeiro programa em Java
5       */
```

- O nome da classe é Hello
- Em Java todo e qualquer código deve pertencer a uma classe
- Esta classe usa um identificador de acesso **public**. Indica que está acessível para outras classes de diferentes pacotes (pacotes são coleções de classes)



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
```

- A próxima linha contém uma chave { e indica o início de um bloco de instruções



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
```

- As próximas 3 linhas indicam um comentário em Java
- Um comentário:
 - Explicação usada para a documentação do código
 - Não faz parte do programa em si, mas sim de sua documentação
 - É uma boa prática de programação adicionar comentários aos programas



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
6     public static void main(String[] args) {
```

- A linha 6 indica a declaração de um método em Java, neste caso, o método main
- O método main é o ponto de partida dos programas em Java
- Todos os programas, com exceção de applets, escritos em Java, se iniciam pelo método main



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
6     public static void main( String[] args ){
7         // Mostra a mensagem "Hello World" na tela
```

- A linha seguinte é um comentário de linha



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * My first Java program
5      */
6     public static void main( String[] args ){
7         // Mostra a mensagem "Hello World" na tela
8         System.out.println("Hello world");
```

- A instrução `System.out.println()`, mostra, na saída padrão, o texto descrito entre as aspas



Entendendo meu primeiro programa em Java

```
1 public class Hello
2 {
3     /**
4      * Meu primeiro programa em Java
5      */
6     public static void main( String[] args ){
7         // Mostra a mensagem "Hello World" na tela
8         System.out.println("Hello world");
9     }
10 }
```

- As últimas duas linhas, que contêm somente uma chave em cada, simbolizam, respectivamente, o fechamento do método main e da **classe**



Comentários em Java

- Notas escritas para fins de documentação
- Estas notas não fazem parte do programa e não afetam seu fluxo
- Java possui 3 tipos de comentários:
 - Comentário de linha
 - Comentário de bloco
 - Comentário estilo Javadoc



Instruções e Blocos em Java

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

```
System.out.println("Hello world");
```



Instruções e Blocos em Java

- Um bloco é formado por uma ou mais instruções agrupadas entre chaves { } indicando que formam uma só unidade
- Blocos podem ser organizados em estruturas aninhadas indefinidamente
- Qualquer quantidade de espaços em branco é permitida
- Exemplo:

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



Identificadores em Java

- Identificadores são representações de nomes de variáveis, métodos, classes, etc
- Exemplos de identificadores podem ser: Hello, main, System, out
- Os identificadores são case-sensitive.
 - Isto significa que o identificador **Hello** não é o mesmo que **hello**.



Identificadores em Java

- 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-chave 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

* not used

** added in 1.2

*** added in 1.4

**** added in 5.0

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



Tipos de dados em Java

- Java Possui 4 tipos de dados, divididos em:
 - Boolean
 - Character
 - Integer
 - Floating-Point



Tipos de dados primitivos

- A linguagem Java possui 8 tipos de dados primitivos

Representação	Tipo de Dado	Dado Primitivo
lógico	boolean	boolean
inteiro	integer e character	char, byte, short e int
inteiro longo	integer	long
número fracionário	float-point	float e double

Tipos de dados primitivos: lógico

- O tipo booleano pode representar dois estados: **true** ou **false**

`boolean result = true;`

- Na instrução acima, é declarada uma variável chamada **result** do tipo **boolean** e lhe é atribuída o valor **true**



Tipos de dados primitivos: inteiro

```
byte b = 97;  
char c = 97;  
short s = 97;  
int i = 97;
```

- Nas instruções acima, são declaradas variáveis chamadas **b**, **c**, **s** e **i**, cada uma representando um determinado tipo **inteiro** e lhes são atribuídas o valor **97**
- Se a instrução `System.out.println(c)` for executada; o resultado mostrado será o caractere 'a'



Tipos de dados primitivos: inteiro longo

```
long z = 10L;
```

- Na instrução acima, é declarada uma variável chamada **z**, do tipo **inteiro longo**, e lhe é atribuída o valor **10**
- Representa-se um **inteiro longo** adicionando-se a letra “L” após o número, preferencialmente em maiúscula para evitar confusão com o número 1



Tamanhos dos tipos: byte, char, short, int e long

<i>Tamanho em memória</i>	<i>Dado primitivo</i>	<i>Faixa</i>
8 bits	byte	-2^7 até 2^7-1
16 bits	char	0 até $2^{16}-1$
16 bits	short	-2^{15} até $2^{15}-1$
32 bits	int	-2^{31} até $2^{31}-1$
64 bits	long	-2^{63} até $2^{63}-1$



Tipos de dados primitivos: float e double

- 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
```



Tamanho dos tipos: float e double

- Os dados de tipo ponto-flutuante representam as seguintes faixas de valores:

<i>Tamanho em memória</i>	<i>Dado primitivo</i>	<i>Faixa</i>
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 variável possui:
 - Tipo que *indica o tipo de dado que ela pode conter*
 - Nome que *deve seguir as regras para identificadores*



Declarando e inicializando variáveis

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



Declarando e inicializando variáveis: Exemplo

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



Exibindo o valor de uma variável

- Para exibir na tela, ou em outro dispositivo, o valor de uma variável, pode-se fazer uso das seguintes instruções:

```
System.out.println()
```

```
System.out.print()
```

Exibindo o valor de uma variável: Exemplo

```
1 public class OutputVariable {  
2     public static void main(String[] args) {  
3         int value = 10;  
4         char x;  
5         x = 'A';  
6         System.out.println(value);  
7         System.out.println("The value of x=" + x);  
8     }  
9 }
```



System.out.println() e System.out.print()

- **System.out.println()**
 - Ao final da exibição do seu conteúdo, inicia uma nova linha
- **System.out.print()**
 - não inicia uma nova linha.



Referência de variáveis e Valor de variáveis

- Temos dois tipos de acesso suportados:
 - Por valor
 - Por referência
- Valor
 - armazenam dados no exato espaço de memória onde a variável está.
- Referência
 - armazenam o endereço de memória onde o dado está armazenado



Referência de variáveis: Exemplo

- Supondo que existam estas duas variáveis dos tipos **int** e **String (variável de classe)**.

```
int num = 10;  
String nome = "Hello"
```



Referência de variáveis: Exemplo

- O quadro abaixo representa a memória do computador, com seus endereços de memória, o nome das variáveis e os tipos de dados nela armazenados

Endereço de memória

1001

:

1563

:

:

2000

Nome da variável

num

nome

dado

10

:

endereço(2000)

:

:

"Hello"

10
:
endereço(2000)
:
:
"Hello"



Operadores

- Os diferentes tipos de operadores são:
 - Operadores aritméticos
 - Operadores relacionais
 - Operadores lógicos
 - Operadores condicionais
- Estes operadores obedecem uma ordem de precedência para que o compilador saiba em qual seqüência executar as operações



Operadores Aritméticos

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

Operadores de Incremento e Decremento

- Operador unário de incremento (++)
- Operador unário de decremento (--)
- Operadores de incremento ou decremento somam ou subtraem em 1 o valor da variável
- Por exemplo, a expressão:

```
count = count + 1;
```

ou:

```
count++;
```



Operadores de Incremento e Decremento

<i>Operador</i>	<i>Uso</i>	<i>Descrição</i>
++	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

Operadores de Incremento e Decremento

- Os operadores de incremento e decremento podem ser usados tanto antes quanto após o operando
- Quando usado antes do operando, provoca o acréscimo ou decréscimo de seu valor antes da avaliação da expressão em que este operador é utilizado
- Por exemplo:

```
int i = 10;  
int j = 3;  
int k = 0;  
k = ++j + i; //resultará em k = 4+10 = 14
```



Operadores de Incremento e Decremento

- Quando usado depois do operando, provoca acréscimo ou decréscimo de seu valor somente após a avaliação da expressão em que este operador é utilizado
- Por exemplo:

```
int i = 10;  
int j = 3;  
i = j++ + i; //resultará em k = 3+10 = 13
```



Operadores Relacionais

- Os operadores relacionais são usados para comparar dois valores e determinar o relacionamento entre eles
- A saída desta avaliação será dada com um **valor lógico**: true ou false

<i>Operador</i>	<i>Uso</i>	<i>Descrição</i>
>	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



Operadores 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)



Operadores Lógicos: && (e lógico) e & (e binário)

<i>x1</i>	<i>x2</i>	<i>Resultado</i>
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	FALSO
FALSO	VERDADEIRO	FALSO
FALSO	FALSO	FALSO

Operadores Lógicos:

|| (ou lógico) e | (ou binário)

<i>x1</i>	<i>x2</i>	<i>Resultado</i>
VERDADEIRO	VERDADEIRO	VERDADEIRO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
FALSO	FALSO	FALSO



Operadores Lógicos:

\wedge (ou exclusivo binário)

<i>x1</i>	<i>x2</i>	<i>Resultado</i>
VERDADEIRO	VERDADEIRO	FALSO
VERDADEIRO	FALSO	VERDADEIRO
FALSO	VERDADEIRO	VERDADEIRO
FALSO	FALSO	FALSO

Operadores Lógicos:

! (negação)

- A **negação** pode ser utilizada para avaliar um argumento. Este argumento pode ser uma expressão, variável ou constante

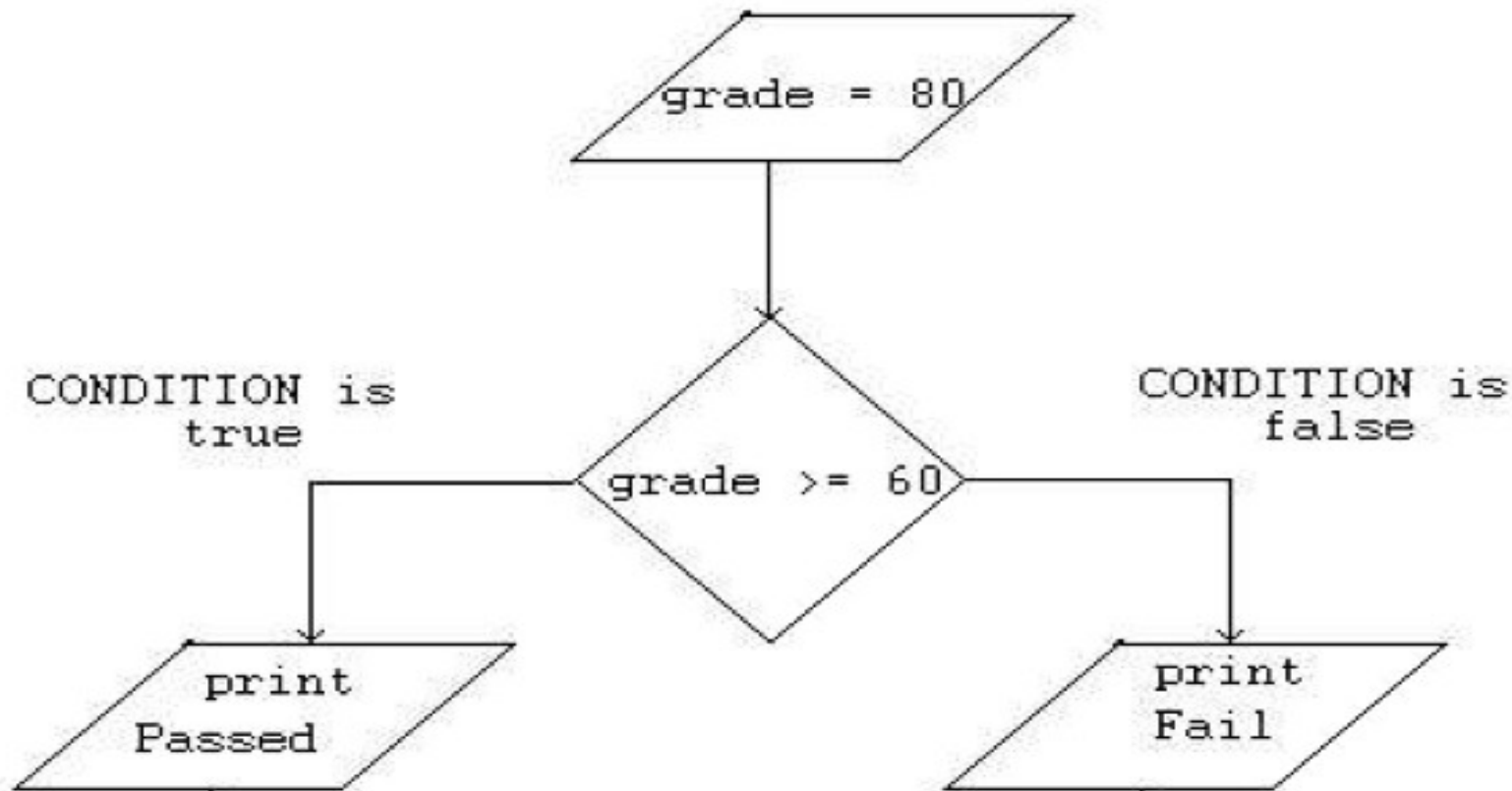
<i>x1</i>	<i>Resultado</i>
VERDADEIRO	FALSO
FALSO	VERDADEIRO



Operadores Lógicos: ?: (Condicional)

`expLógica?expCasoTrue:expCasoFalse`

Operadores Lógicos: ?: (Condicional)



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:

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



Sumário

- Comentários em Java (estilo C++, C e Javadoc)
- Instruções, blocos, identificadores e palavras-chave
- Tipos de dados (integer, ponto-flutuante, boolean e char)
- Tipos de dados primitivos (boolean, char, byte, short, int, long, float e double)
- Variáveis (declaração, inicialização, saída)
- `System.out.println()` vs. `System.out.print()`
- Referência vs. Valor
- Operadores (aritméticos, de incremento e decremento, relacionais, lógicos, condicional)
- Precedência dos operadores



Parceiros

- Os seguintes parceiros tornaram JEDI possível em Língua Portuguesa:

