Unicesumar

Programação I

Tipos de Dados

Prof. Me. Fábio Perfetto

Tipos de Dados

Assim como em outras linguagens, antes de utilizar variáveis é necessário saber definir um tipo de dado. Os tipos de dados em Java são portáteis entre todas as plataformas de computadores que suportam essa linguagem.

Na maioria das linguagens, quando um dado inteiro é utilizado, pode ser que para uma determinada plataforma esse número seja armazenado com 16 *bits* e em outra 32 *bits*. Em Java isso não ocorre, uma vez que um tipo de dado terá sempre a mesma dimensão.

Os tipos primitivos da linguagem (Quadro 2.1) utilizados na criação de variáveis são:

a) *Boolean*: não é um valor numérico, só admite os valores *true* ou *false*.

b) *Char*: usa o código UNICODE e ocupa cada caractere 16 *bits*.

c) Inteiros: diferem nas precisões e podem ser positivos ou negativos.

- *Byte*: 1 *byte*.
- *Short*: 2 *bytes*.
- *Int*: 4 *bytes*.
- *Long*: 8 *bytes*.

d) Reais em ponto flutuante: igual aos inteiros, também diferem nas precisões e podem ser positivos ou negativos.

- *Float*: 4 *bytes*.
- *Double*: 8 bytes.

Tabela de Famílias, tipos e classes de variáveis

Família	Tipo Primitivo	Classe Invólucro	Tamanho	Exemplo
Lógico	boolean	Boolean	1 bit	true
Literais	char	Character	1 byte	'A'
	-	String	1 byte/cada	"JAVA"
Inteiros	byte	Byte	1 byte	127
	short	Short	2 bytes	32 767
	int	Integer	4 bytes	2 147 483
	long	Long	8 bytes	2^{63}
Reais	float	Float	4 bytes	$3.4e^{+38}$
	double	Double	8 bytes	$1.8e^{+308}$

Variáveis e Constantes

Uma variável ou constante é um tipo de **identificador** cujo nome, que é selecionado pelo programador, é associado a um valor que pertence a um tipo de dado.

Todo identificador possui um nome, um tipo e conteúdo. Os identificadores não podem utilizar palavras reservadas do Java.

A linguagem Java exige que os identificadores tenham um tipo de dado definido antes de serem utilizados no programa, ou seja, eles devem ser obrigatoriamente declarados, independentemente do ponto do programa, seja no meio, no início ou no final, desde que antes de sua utilização no programa.

Essa característica do identificador em Java difere da maioria das linguagens de programação. A linguagem Pascal, por exemplo, possui um local exclusivo para declaração de variáveis.

Uma variável precisa ser declarada para poder ser utilizada. Opcionalmente, ela pode ser inicializada já no momento de sua declaração.

Declaração de variável no Java

Inteiro

int idade = 3;

Int idade = (int)3; //Typecast

Variavél

Integer idade = new Integer (3); -> Objeto

Real

float salario = 1825.54f;

float salario = (float)1825.54;

Float salario = new Float(1825.54);

Declaração de variável no Java

Caracter

char letra = ‘G’;

char letra = (char)’G’;

Character letra = new Character (“G”);

Lógico

boolean casado = false;

boolean casado = (boolean)false;

Boolean casado = new Boolean (false);

Declaração feita no NetBeans

Exemplo02.java

```
1  public class Exemplo02 {
2      public static void main (String args[ ]) {
3      int n1; /* Declaração de um inteiro. */
4      int n2 = 4; /* Declaração e inicialização de outro inteiro. */
5      char c = 'x'; /* Declaração e inicialização de um caractere. */
6      n1 = n2 + 8; // Atribuindo valor à n1
7      System.out.println("Primeiro valor: " + n1);
8      System.out.println("Segundo valor: " + n2);
9      System.out.println("Terceiro valor: " + c);
10     }
11 }
```

A saída do programa deverá ser:

12

4

x

Convenções

1. Caso uma variável do tipo *char*, *byte*, *short*, *int*, *long*, *float* ou *double* não seja inicializada, ela é criada com o valor 0. Se ela for do tipo *boolean*, seu valor padrão será *false*.
2. Quando for necessário definir uma nova variável com um tipo de dado diferente, por convenção, utiliza-se uma nova linha. O mais comum entre os programadores Java é definir um tipo de dados e declarar uma lista com um ou mais nomes de variáveis desejadas desse tipo. Nessa lista os nomes são separados por vírgulas e a declaração terminada por ‘ ; ‘ (ponto e vírgula).
3. As variáveis também podem ter sensibilidade, isto é, ao declarar uma variável com um nome (por exemplo, **dolar**) ele deve ser utilizado sempre da mesma forma. Isto é, não pode ser usado como Dólar, DOLAR, dólar ou qualquer outra variação, apenas com todas as letras minúsculas, como realizado em sua declaração.

Convenção – cont.

4. 4. Os nomes das variáveis devem começar com letra, caractere de sublinhado ou cifrão. Não é permitido iniciar o nome da variável com número. Por convenção, a linguagem Java utiliza o seguinte padrão:
   - quando o nome da variável for composto apenas por um caractere ou palavra, os caracteres devem ser minúsculos;
   - quando o nome da variável tiver mais de uma palavra, a primeira letra da segunda palavra em diante deve ser maiúscula. Todos os outros caracteres devem ser minúsculos.
   - Exemplos: a, a1, real, nome, valorVenda, codigoFornecedor.

Declaração de constantes

Na realidade não existem constantes em Java; o que existe é um tipo de variável com comportamento semelhante a uma constante de outras linguagens.

Trata-se de um tipo de variável que não pode alterar seu conteúdo depois de ter sido inicializado, ou seja, o conteúdo permanece o mesmo durante toda execução do programa. Em Java, essa variável é chamada **final**.

Essas constantes são usadas para armazenar valores fixos, geralmente, definidos no início de uma classe. Por convenção os nomes de constantes devem ser escritos em letras maiúsculas. Exemplos: na Matemática temos a constante **PI** cujo valor é 3,1416 (isto é, p=3,1416); na Física temos o valor da aceleração da **GRAVIDADE** da Terra (g=9,81 m/s2).

Para a declaração de constantes em Java utiliza-se a palavra reservada **final** antes da definição do tipo de variável:

```
final double PI=3.14;

final double GRAVIDADE=9.81;
```

Caso um segundo valor seja atribuído a uma variável final no decorrer da classe, o compilador gera uma mensagem de erro. Não é obrigatório inicializar o conteúdo de uma variável final no momento de sua declaração.

Operadores

A linguagem Java oferece um amplo conjunto de operadores destinados à realização de operações aritméticas, lógicas e relacionais, com a possibilidade de formar expressões de qualquer tipo. Além dos operadores matemáticos, existem também operadores lógicos e relacionais.

Operadores aritméticos

Operação	Sinal	Exemplo
Adição	+	1+20
Subtração	-	35-17
Multiplicação	*	14*2
Divisão	/	14/2
Resto da divisão inteira	%	14%7
Sinal negativo	-	-4
Sinal postitivo	+	+5
Incremento unitário	++	++6 ou 6++
Decremento unitário	--	--6 ou 6--

O operador de incremento (++) aumenta o valor de uma variável qualquer em um. O mesmo vale para o operador de decremento (--), logicamente, reduzindo em um o valor da variável.

Operadores relacionais

Os operadores relacionais possibilitam comparar valores ou expressões, retornando um resultado lógico verdadeiro ou falso.

Significado	Operador	Exemplo
Igual	==	x==20
Diferente (Não igual)	!=	y!=17
Menor que	<	x<2
Maior que	>	x>2
Menor ou igual	<=	y<=7

Operadores lógicos

São operadores que permitem avaliar o resultado lógico de diferentes operações aritméticas em uma expressão.

Significado	Operador	Exemplo
Operação lógica E (AND)	&&	(x<5)&&(x>0)
Operação lógica OU (OR)	||	(y==5||y>10)
Negação	!	!true==false

Conversão de tipos

É comum que o programador precise converter um número inteiro, por exemplo, em um número real (ou vice-versa). Em Java existem basicamente dois tipos de conversão de dados:

a) conversão implícita – na qual os dados são convertidos automaticamente, sem a preocupação do programador. Ela ocorre, por exemplo, quando convertemos um número inteiro para um número real.

Nesse caso, a conversão é implícita porque é óbvio para o compilador que um número inteiro pode ser representado também como um número real.

Veja um exemplo a seguir:

```
int x = 4;

float y = x;

double z = y;
```

Conversão de Tipos

b) conversão explícita – quando o programador precisa explicitar no código que um valor será convertido de um tipo para outro. No caso de um número real para um inteiro, por exemplo, pode haver perda na precisão do número.

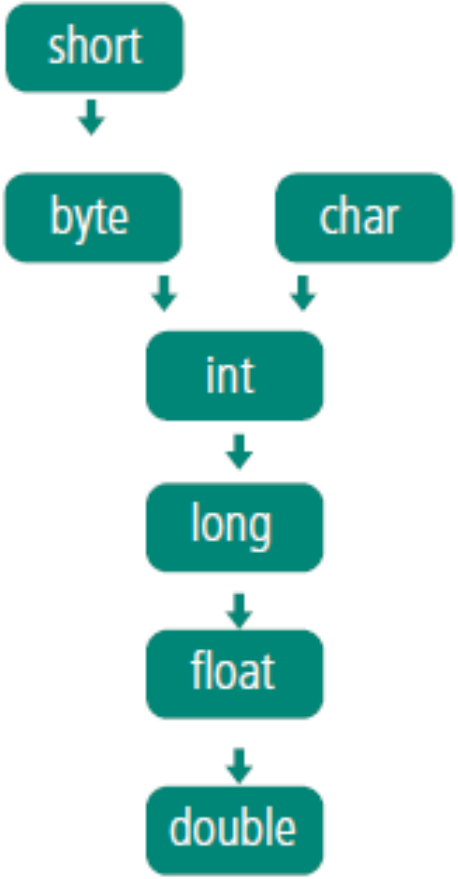
Veja o exemplo a seguir:

```
float a = 9;

float b = a/8; // b = 1.125

int c = (int)b; /* Aqui estamos forçando a conversão para um número inteiro. Nesse caso, a variável c armazenará apenas a parte inteira da variável b, ou seja, 1 */

System.out.println(b);

System.out.println(c);
```

O resultado da execução deste trecho de código é:

```
1.125

1
```

Seguindo o sentido das flechas da abaixo vemos os tipos que podem ser implicitamente convertidos em outros. Seguindo o sentido contrário, vemos os tipos que precisam ser convertidos explicitamente:



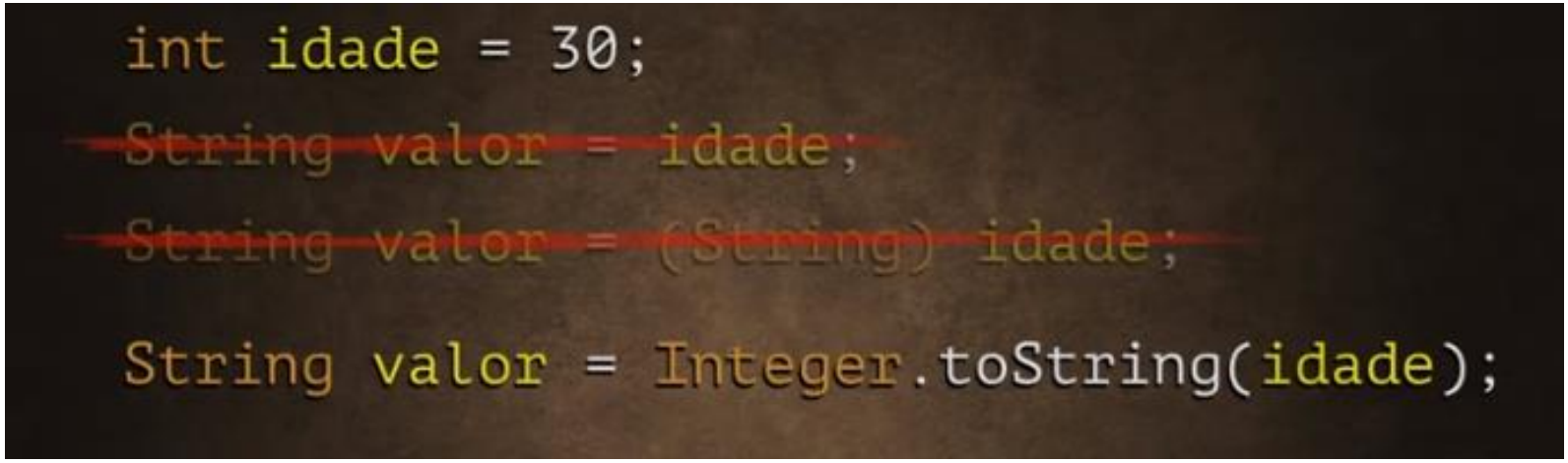
Cuidado

No caso de conversão de caracter para int o valor a ser gravado será o da tabela ASCII.

ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Conversão de tipo Inteiro para String



TesteTipos.java

Código-Fonte | Histórico

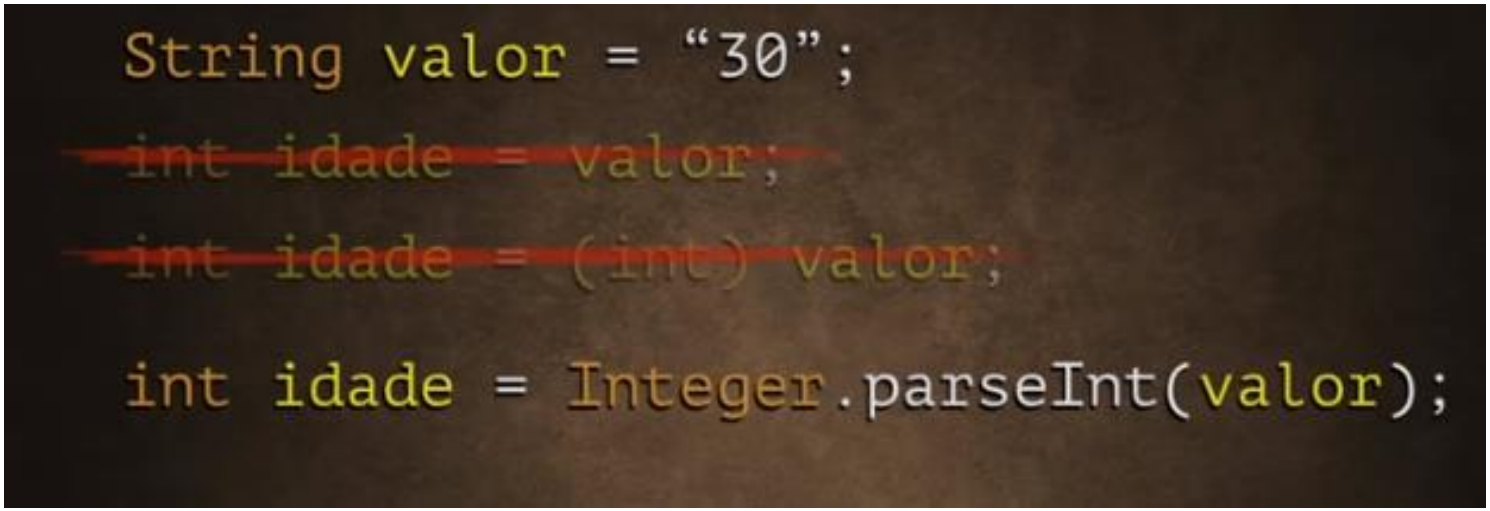
```
6   package testetipos;
7
8   /**
9    *
10   * @author PERFETTO
11   */
12  public class TesteTipos {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          int idade = 30;
19          String valor = Integer.toString(idade);
20          System.out.println(valor);
21          // TODO code application logic here
22      }
23
24  }
```

testetipos.TesteTipos > main

Saída - TesteTipos (run)

```
run:
30
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```

Conversão de tipo String para inteiro



TesteTipos.java

Código-Fonte | Histórico

```
6   package testetipos;
7
8   /**
9    *
10   * @author PERFETTO
11   */
12  public class TesteTipos {
13
14      /**
15       * @param args the command line arguments
16       */
17      public static void main(String[] args) {
18          /*int idade = 30;
19          String valor = Integer.toString(idade);
20          System.out.println(valor);*/
21          String valor ="30";
22          int idade = Integer.parseInt(valor);
23          System.out.println(idade);
24
```

testetipos.TesteTipos > main

Saída - TesteTipos (run)

```
run:
30
CONSTRUÍDO COM SUCESSO (tempo total: 0 segundos)
```


Unicesumar