

Conceitos base da linguagem JAVA

Profª. Heloisa Moura

Conceitos base da linguagem JAVA

O que vamos ver:

- Variáveis
- Variáveis finais
- Constantes
- Tipos Primitivos
- Conversões de Tipos Primitivos

Iniciando no Java - Conceitos base da linguagem

Variáveis

Uma variável é uma estrutura que permite armazenar dados na memória durante a execução do programa, para processamento de informações.

Todas as variáveis devem ser declaradas antes que possam ser usadas. Declarar uma variável significa criá-la em qualquer ponto do programa.

A linguagem Java é fortemente tipada. Isso significa que cada variável deve ter um tipo declarado.

Em Java, podemos declarar variáveis, variáveis finais, variáveis de classe e constantes. As variáveis podem ter seu valor modificado a qualquer momento, enquanto as variáveis finais e constantes recebem um valor e não podem ser alteradas.

Iniciando no Java - Conceitos base da linguagem

Regras para criação de identificadores

Toda variável possui um identificador, que é o nome da variável. Para criar um identificador(nome da variável) em Java, precisamos seguir algumas regras, listadas a seguir:

- Deve conter apenas letras, _(underline), \$ ou os números de 0 a 9
- Deve obrigatoriamente se iniciar por uma letra, _ ou \$
- Não podemos usar palavras-chave da linguagem
- O nome deve ser único dentro de um escopo

Além disso, o Java é case sensitive, o que significa que os nomes de variáveis diferenciam maiúsculas e minúsculas.

Iniciando no Java - Conceitos base da linguagem

Palavras reservadas da linguagem

Não se deve usar nenhuma dessas palavras como nome de variável

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

Palavras não usadas atualmente

const	goto
-------	------

Iniciando no Java - Conceitos base da linguagem

Exemplos de nomes de variáveis

Válidos:

NomeCliente

Telefone_1

Preco\$

Inválidos

1Telefone

Nome Cliente

#Preco

Iniciando no Java - Conceitos base da linguagem

Declaração de Variáveis

Sintaxe

tipo identificador = valor;

Exemplos:

byte a;

char t;

float x,y;

Int dia ;

char sexo ;

Iniciando no Java - Conceitos base da linguagem

Inicialização de variáveis

A inicialização de uma variável durante sua declaração é opcional. Caso a variável não seja inicializada, ela pode vir a ter um valor padrão, que depende do seu tipo declarado:

- boolean = false
- byte, short, int, long, float, double: 0
- char: \u0000

Importante inicializar a variável sempre que possível, mesmo que durante a execução do programa ela vá receber dados vindo de outras fontes alterando o seu valor inicial.

Iniciando no Java - Conceitos base da linguagem

Atribuição de valores a variáveis

Para atribuir um

Valor a uma variável, devemos utilizar o operador de atribuição =

Veja os exemplos de atribuição:

```
byte a = 45
```

```
char t = 'T'
```

```
int valor = 200
```

```
float x = 98.80f
```

```
int dia ; // variável declarada e não inicializada
```

```
dia = 20 // variável atribuída agora
```


Iniciando no Java - Conceitos base da linguagem

Variáveis finais

No Java quando precisamos lidar com dados que não devem ser alterados durante a execução do programa criamos variáveis finais ou constantes.

Declaramos uma variável final utilizando a palavra-chave **final** antes do tipo da variável. Veja abaixo um exemplo de declaração de uma variável final dentro do método main:

Iniciando no Java - Conceitos base da linguagem

Variáveis finais

```
1 package br.com.devmedia;
2
3 public class Main {
4     public static void main(String[] args) {
5         final String msgPadrao = "Olá!";
6     }
7 }
```

A variável final msgPadrao recebeu o valor "Olá!" e não pode ser alterado.

Iniciando no Java - Conceitos base da linguagem

Variáveis finais

Uma observação quanto a criação de uma variável final, é que o seu valor não precisa ser atribuído no momento da sua criação. Podemos primeiro criar a variável final e atribuir o seu valor posteriormente. Veja um exemplo abaixo:

```
1 package br.com.devmedia;
2
3 public class Main {
4     public static void main(String[] args) {
5         final String msgPadrao;
6         msgPadrao = "Olá!";
7     }
8 }
```


Iniciando no Java - Conceitos base da linguagem

Variáveis finais

Mas cuidado, lembre-se de que só podemos atribuir o valor a uma variável final uma única vez. Do contrário o compilador vai gerar erro.

Veja outros exemplos de declaração de variáveis finais:

```
1 package br.com.devmedia;
2
3 class Main {
4     public static void main(String[] args) {
5         final String nomePagina = "home";
6         final double e = 1.234e2;
7
8         System.out.println(nomePagina); // vai imprimir "home"
9         System.out.println(e); // vai imprimir "1.234e2"
10    }
11 }
```


Iniciando no Java - Conceitos base da linguagem

Constantes

As constantes são melhor entendidas junto com o conceito de orientação a objetos e classes, porém vamos apresentar a sua definição mas não se preocupe, pois elas serão melhor abordadas mais a frente.

Assim como uma variável final, uma constante é declarada quando precisamos lidar com dados que não devem ser alterados durante a execução do programa. No Java declaramos uma constante utilizando as palavras-chave `static final` antes do tipo da variável.

Iniciando no Java - Conceitos base da linguagem

Constantes

Veja abaixo um exemplo de declaração de uma constante:

```
1 package br.com.devmedia;
2
3 public class Main {
4     public static final float PI = 3.1416F;
5
6     public static void main(String[] args) {
7         System.out.println(PI);
8     }
9 }
```


Iniciando no Java - Conceitos base da linguagem

Constantes

Uma observação é que uma constante é criada na classe, ou seja, fora do método. Por esse motivo utilizamos o modificador de acesso 'public' (linha 4) que veremos mais a frente quando estudarmos sobre classes.

Diferente de uma variável final, uma constante precisa receber o seu valor no momento em que ela for declarada.

Por convenção, usamos letras maiúsculas para declarar constantes e assim distingui-las das variáveis.

Nota: Constantes são indicadas para substituir “números/valores mágicos”, isto é, aqueles números que aparecem no meio do código criado para realizar algum cálculo, agregando mais significado e facilitando a compreensão.

Iniciando no Java - Conceitos base da linguagem

```
1 package br.com.devmedia;
2
3 public class Main {
4     public static void main(String[] args) {
5         final float pi = 3.146f;
6         final double e;
7
8         e = 1.234e2;
9
10        String nome = "Estevao Dias";
11        String dataAniversario = "23/05/2000";
12        int lote = 35456;
13        boolean aprovado;
14
15        aprovado = true;
16
17        System.out.println(pi); // vai imprimir "3.146f"
18        System.out.println(e); // vai imprimir "1.234e2"
19        System.out.println(nome); // vai imprimir "Estevao Dias"
20        System.out.println(dataAniversario); // vai imprimir "23/05/2000"
21        System.out.println(lote); // vai imprimir "35456"
22        System.out.println(aprovado); // vai imprimir "true"
23    }
24 }
```

Mais alguns
exemplos de
variáveis

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos

Os tipos de dados em Java são divididos em duas categorias: **Tipos primitivos** e **Tipos por Referência** (tem haver com os objetos)

Os Tipos primitivos são o **boolean, byte, char, short, int, long, float e double**. São tipos especiais, que são internos da linguagem. Não são objetos criados a partir de uma classe como os tipos de referência.

Uma variável do tipo primitivo armazena exatamente um valor do seu tipo declarado por vez. Quando um outro valor é atribuído a uma dessas variáveis, seu valor anterior é substituído.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos: Números inteiros

São Tipos Primitivos que são usados para representar números inteiros. São: **byte, short, int e long**

Tipo	Tamanho (bits)	Faixa	Valor padrão
byte	8	-128 a 127	0
short	16	-32.768 a 32.767	0
int	32	-2^{31} a $2^{31}-1$	0
long	64	-2^{63} a $2^{63}-1$	0L

Tamanho em bit: Espaço que é ocupado na memória quando se declara uma variável com um desses tipos.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos: Números inteiros

Tipo	Tamanho (bits)	Faixa	Valor padrão
byte	8	-128 a 127	0
short	16	-32.768 a 32.767	0
int	32	-2^{31} a $2^{31}-1$	0
long	64	-2^{63} a $2^{63}-1$	0L

Tamanho em bit: Espaço que é ocupado na memória quando se declara uma variável com um desses tipos.

Faixa: São os valores numéricos que podem ser representados com cada um desses tipos.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos: Números de ponto flutuante

Um número de ponto flutuante é aquele que tem a parte decimal. Ex.: 4.56, 12.30 . Os Tipos Primitivos de ponto flutuante em Java são dois: **float e double**. Esses dois tipos diferem em tamanho e precisão.

Tipo	Tamanho	Faixa	Valor padrão
float	32 bits	IEEE 754 $\pm 1,40129846432481707e-45$ a $3,40282346638528860e+38$	0.0f
double	64 bits	IEEE 754 $\pm 4,94065645841246544e-324$ a $1,79769313486231570e+308$	0.0d

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos: Números de ponto flutuante

As variáveis **double** armazenam valores com maior magnitude e precisão, do que as do tipo **float**, e devem ser preferivelmente empregados, quando a precisão do valor for um fator importante.

Ex.: Programas de aplicações científicas ou financeiras é recomendado usar o **double**.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos primitivos: char

O tipo char permite armazenar um caractere Unicode, utilizando para isso 16 bits. Seu valor mínimo é '\u0000' (ou 0), e seu valor máximo é '\uffff' (ou 65535).

O Unicode é um padrão da indústria para representar dados relacionados a texto, incluindo letras, símbolos e caracteres especiais.

Valor padrão para o tipo char: '\u0000'

Lembrando que o char é utilizado para armazenar 1 caractere. Se precisar armazenar em uma variável um conjunto de caracteres. Ex.: Um nome ou uma frase será utilizado um outro tipo de dados que iremos ver depois.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Tipos Primitivos: boolean

O tipo boolean permite armazenar um valor lógico nos estados **True** ou **False** (verdadeiro ou falso), ocupando apenas 1 bit de espaço.

Valor padrão para o tipo boolean: false

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Literais

Um literal é uma representação no código-fonte de um valor fixo. São representados diretamente no código, sem a necessidade de processamento.

Podemos atribuir valores literais a variáveis de tipo primitivo.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Exemplos de tipo Primitivo e Literais

Literais de caracter:

```
char c = 'a';
```

```
char z = '\u0041'; // em Unicode
```

Literais inteiros

```
int i = 10; short s = 15; byte b = 1;
```

```
long hexa = 0x9afoL; int octal = 0633;
```

Literais de ponto-flutuante

```
float f = 123.0f;
```

```
double d = 12.3;
```

```
double g = .1e-23 em notação científica;
```

Literais booleanos

```
boolean v = true;
```

```
boolean f = false;
```

**Literais de string (não é tipo primitivo
- s é uma referência)**

```
String s = "abcde";
```


Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Conversão de Tipos Primitivos

- As conversões de tipos primitivos são tratadas em tempo de compilação. Toda a informação necessária para efetuar a mudança de tipo já está disponível para o compilador Java no momento em que ele efetua o seu trabalho.
- A conversão de primitivos ocorrem em três situações:
- Atribuições;
- Chamada de Métodos; Veremos no assunto de Orientação a Objetos
- Operações Aritméticas.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Conversões de Tipos Primitivos: Atribuição

- A conversão durante a atribuição ocorre quando atribui-se a uma variável de um determinado tipo um valor diferente do valor original.
- Alguns valores são incompatíveis se você tentar fazer uma atribuição direta. Enquanto um número real costuma ser representado em uma variável do tipo double, tentar atribuí-lo a uma variável int não funciona, porque é um código que diz: "i deve valer d", mas não se sabe se d realmente é um número inteiro ou não.

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Conversões de Tipos Primitivos: Atribuição

- `double d = 3.1415;`
- `int i = d; // não compila.`
- A mesma coisa ocorre no seguinte trecho:
- `int i = 3.14;`
- O mais interessante é que nem mesmo o seguinte código compila:
- `double d = 5; // ok, o double pode conter um número inteiro.`
- `int i = d; // não compila.`

Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

Conversões de Tipos Primitivos: Atribuição

- Apesar de 5 ser um bom valor para um **int**, o compilador não tem como saber qual valor estará dentro desse **double**, no momento da execução. Esse valor pode ter sido digitado pelo usuário, e ninguém garantirá que essa conversão ocorra sem perda de valores.
- Já no caso a seguir é o contrário:

```
int i = 5;
```
- ```
double d2 = i;
```
- O código acima compila sem problemas, uma vez que um **double** pode guardar um número com ou sem ponto flutuante. Todos os inteiros representados por uma variável do tipo **int** podem ser guardados em uma variável **double**, então não existem problemas no código acima.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Atribuição

As regras para conversão com tipos primitivos podem ser resumidas na lista abaixo:

- O tipo boolean não pode ser convertido para nenhum outro;
- Os tipos 'não-booleans' podem ser convertidos para outros tipos 'não-booleans', contanto que não haja perda de precisão. Ou seja, podem ser convertidos apenas para tipos que suportem um limite igual ou maior do que o seu.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Atribuição

As conversões podem ser resumidas da seguinte forma:

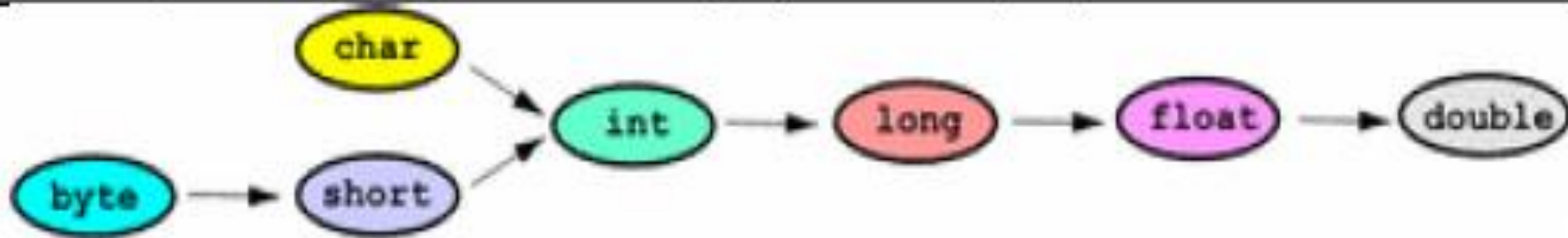
- Um **byte** pode ser convertido em um short, int, long, float ou double
- Um **short** pode ser convertido em um int, long, float ou double
- Um **char** pode ser convertido em um int, long, float ou double
- Um **int** pode ser convertido em um long, float ou double
- Um **long** pode ser convertido em um float ou double
- Um **float** pode ser convertido em um double



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

## Conversões de Tipos Primitivos: Atribuição

A figura ilustra as possíveis conversões de tipo primitivo que podem ocorrer em Java.





# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Atribuição

### Valores Literais

- Existem alguns detalhes peculiares na conversão de tipos primitivos, sobretudo quando a conversão é feita a partir de valores literais.

Veja o exemplo abaixo:

```
float f = 1.2345
```

- O código acima não compila, isso porque os números literais em Java ou são **double** ou **int**. Dessa forma, o programa tenta atribuir um valor **double** (1.2345) a uma variável **float**. Todos os literais com ponto flutuante são considerados **double** pelo Java. O tipo **float** não pode receber um **double** sem perda de informação. Para fazê-lo funcionar devemos escrever:

```
float f = 1.2345f;
```

- A letra f, que pode ser maiúscula ou minúscula, indica que aquele literal deve ser tratado como um **float**.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Atribuição

Outro exemplo:

```
byte b = 2;
```

```
short s = 4;
```

```
char c = 3;
```

É comum achar que esse trecho de código não compila, por estar atribuindo a variáveis byte, short e char o valor de um literal int. Porém, ele executa sem nenhum problema. A razão é que Java resolve a conversão durante a atribuição apenas quando um literal do tipo int é atribuído a uma variável "menor" (byte, short ou char). É importante saber ainda, que para a conversão ser efetuada é necessário que o valor do literal esteja dentro da capacidade da variável.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Atribuição

### Mais um exemplo:

1. `byte b = 987654321;`

2. `int i = 10;`

3. `byte b2 = i;`


- O código acima tem dois erros. O primeiro deles é o da linha 1: o valor que está sendo atribuído a variável do tipo `byte` é maior do que sua capacidade. O segundo erro aparece na linha 3: a conversão do tipo `int` para tipos menores só funciona para valores literais.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Casting

- Às vezes, precisamos que um número quebrado seja arredondado e armazenado em um número inteiro. Para fazer isso sem que haja o erro de compilação, é preciso ordenar que o número quebrado seja moldado (casted) como um número inteiro. Esse processo recebe o nome de casting.
- `double d3 = 3.14;`
- `int i = (int) d3;`  Casting
- O casting foi feito para moldar a variável `d3` como um `int`. O valor de `i` agora é 3.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Casting

- O mesmo caso ocorre entre valores int e long.

```
long x = 10000;
```

```
int i = x; // não compila, pois pode estar perdendo informação.
```

- E se quisermos realmente fazer isso, fazemos o casting:

```
long x = 10000;
```

```
int i = (int) x;
```



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Casting

Outro caso que é mais comum:

```
double d = 5;
```

```
float f = 3;
```

```
float x = f + (float) d;
```

Precisa-se do casting porque o Java faz as contas e vai armazenando sempre no maior tipo que apareceu durante as operações, neste caso, o **double**.

Até casting com variáveis do tipo **byte, short e char** podem ocorrer. O único tipo primitivo que não pode ser atribuído a nenhum outro tipo é o **boolean**.



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

---

## Conversões de Tipos Primitivos: Casting

- Castings possíveis
- A tabela exibe todos os casts possíveis na linguagem Java, mostrando a conversão **de** um valor **em** outro.
- A indicação **Impl.** quer dizer que aquele cast é implícito e automático, ou seja, você não precisa indicar o cast explicitamente (lembrando que o tipo boolean não pode ser convertido em nenhum outro tipo).



# Iniciando no Java - Conceitos base da linguagem - Tipos de Dados

## Conversões de Tipos Primitivos: Casting

| PARA:         | byte   | short        | char   | int          | long         | float        | double       |
|---------------|--------|--------------|--------|--------------|--------------|--------------|--------------|
| DE:           | byte   | short        | char   | int          | long         | float        | double       |
| <b>byte</b>   | ----   | <i>Impl.</i> | (char) | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>short</b>  | (byte) | ----         | (char) | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>char</b>   | (byte) | (short)      | ----   | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>int</b>    | (byte) | (short)      | (char) | ----         | <i>Impl.</i> | <i>Impl.</i> | <i>Impl.</i> |
| <b>long</b>   | (byte) | (short)      | (char) | (int)        | ----         | <i>Impl.</i> | <i>Impl.</i> |
| <b>float</b>  | (byte) | (short)      | (char) | (int)        | (long)       | ----         | <i>Impl.</i> |
| <b>double</b> | (byte) | (short)      | (char) | (int)        | (long)       | (float)      | ----         |