

Sistemas de Informação

Desenvolvimento de Aplicativos I

Introdução ao Java – Parte III

Prof. Marciel de Liz Santos

Introdução ao Java

Variáveis

- ❑ Representam um espaço de memória para armazenar um valor; os programas não fariam sentido se trabalhassem sempre com os mesmos valores. Para cada área de memória associamos um nome (identificador) e o tipo de valor a ser armazenado.
 - ❑ Variáveis são classificadas em:
 - Tipo Primitivo
 - Tipo Reference
 - Arrays
-

Introdução ao Java

Variáveis

- ❑ Tipos Primitivos
 - Sintaxe para declaração e inicialização de variáveis de tipos primitivos:
`<tipoVariavel> <nomeVariavel> = valor;`
 - Tipos primitivos podem ser:
 - ❑ Numérico
 - ❑ Caracter
 - ❑ Booleanos (verdadeiro ou falso)

Exemplo:

```
int meuInt = 0;  
char meuChar = 'c';  
boolean meuBoolean = true;  
Long numeroGrande = 987654321;
```

Introdução ao Java

Variáveis

□ Números Inteiros

- Os tipos primitivos para representação de inteiros oferecidos pela linguagem são:

Tipo	Valor Mínimo	Valor Máximo	Bytes
byte	-128	127	1
short	- 32.768	32.767	2
int	- 2.147.483.648	2.147.483.647	4
long	- 922.337.203.685.475.808	922.337.203.685.475.807	8

Introdução ao Java

Variáveis

□ Números Inteiros

- Os literais numéricos podem ser expressos em **decimal**, **octal** ou **hexadecimal**. O padrão usado é decimal.

- Para indicar um número em octal, prefixe a literal com 0 (zero).
- Para indicar um número em **hexadecimal**, prefixe a literal com **0x** ou 0X. Os dígitos do hexadecimal podem ser escritos tanto em maiúsculo como em minúsculo.

Exemplos:

0 decimal

034 octal

0x1c hexadecimal

Introdução ao Java

Variáveis

□ Números Inteiros

- Todo número inteiro escrito em Java é tratado como int desde que o valor esteja na faixa de valores do int.
- Para forçar um número long utilizamos a letra L ou l no final do número: 10L. Apesar de caber em um int, o L vai forçar para que este número ocupe o espaço equivalente a um long.
- Para forçar um número byte ou short, devemos utilizar a técnica de casting (modificadores de tipo).

Exemplo: `Long x = 12; int y = (int) x;`

Introdução ao Java

Variáveis

□ Números Inteiros

Exemplo TesteTiposPrimitivosInteiros.java

```
class TesteTiposPrimitivosInteiros{
    public static void main (String[] args){
        int i = 10;
        System.out.println("int i = 10 => i = " + i);
        long l = 2566L;
        System.out.println("long l = 2566L => l = " + l);
        long l2 = 22365656561;
        System.out.println("long l2 = 22365656561 => l2 = " + l2);
        byte b = (byte) 123; // este valor cabe em um byte
        System.out.println("byte b = (byte) 123 => b = " + b);
        byte b2 = (byte) 123568545; // este valor não cabe em um byte
        System.out.println("byte b2 = (byte) 123568545 => b2 = " + b2);
        short s = (short) 12565; // este valor cabe em um short
        System.out.println("short s = (short)12565 => s = " + s);
        short s2 = (short) 1231321; // este valor não cabe em um short
        System.out.println("short s2 (short) 1231321 => s2 = " + s2);
    }
}
```

Introdução ao Java

Variáveis

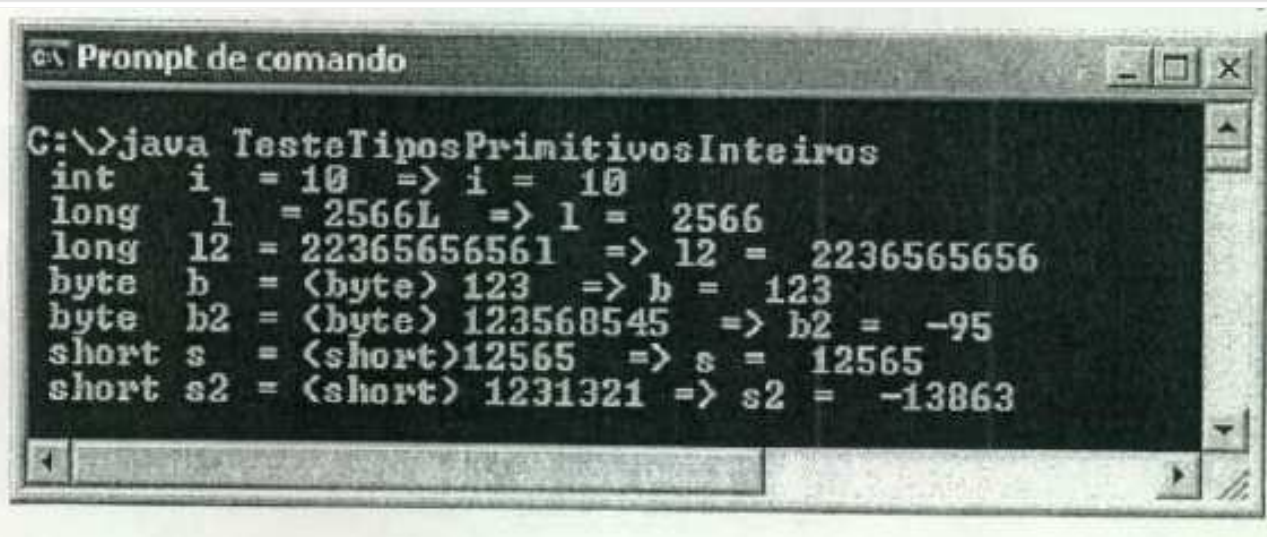
❑ Números Inteiros

Exemplo TesteTiposPrimitivosInteiros.java

Para compilar: `javac TesteTiposPrimitivosInteiros.java`

Para executar: `java TesteTiposPrimitivosInteiros`

Saída gerada pela execução da classe:



```
C:\>java TesteTiposPrimitivosInteiros
int    i    = 10    => i = 10
long   l    = 2566L => l = 2566
long   l2   = 22365656561 => l2 = 2236565656
byte   b    = (byte) 123 => b = 123
byte   b2   = (byte) 123568545 => b2 = -95
short  s    = (short) 12565 => s = 12565
short  s2   = (short) 1231321 => s2 = -13863
```


Introdução ao Java

Variáveis

- ❑ Números com ponto flutuante
 - Existem dois tipos de variáveis para números com ponto flutuante: float e double. Para reduzir o número de casas decimais e necessário formatar o número com classes da família NumberFormat.
 - Os números com ponto flutuante em Java estão implantados conforme o padrão IEEE 754.

Tipo	Mínimo	Máximo	Bytes
float	$1.4e^{-45}$	$3.4e^{38}$	4
double	$4.9e^{-324}$	$1.7e^{308}$	8

Introdução ao Java

Variáveis

- Números com ponto flutuante
 - Todo número com ponto flutuante por default é double.
 - Podemos também utilizar o caractere "d" ou "D" para indicar que um número é do tipo double.
 - Para indicarmos que um número é do tipo float devemos utilizar o caractere "F" ou "f" no final do número.
-

Introdução ao Java

Variáveis

❑ Números com ponto flutuante

Exemplo TesteTiposPrimitivosPontoFlutuante.java

```
class TesteTiposPrimitivosPontoFlutuante {  
    public static void main(String[] args){  
        float f = 10F;  
        System.out.println("float f = 10F => f = " + f);  
        // podemos utilizar o F ou f para indicar que o número é float  
        float f2 = 10.45454f;  
        System.out.println("float f2 = 10.45454f => f2 = " + f2);  
        // por default todo número é inteiro, por isto indicamos com d  
        double d = 12565484546d;  
        System.out.println("double d = 12565484546d => d = "+ d);  
        double d2 = 12565484546.0;  
        System.out.println("double d2 = 12565484546.0 => d2 = "+ d2);  
    }  
}
```

Introdução ao Java

Variáveis

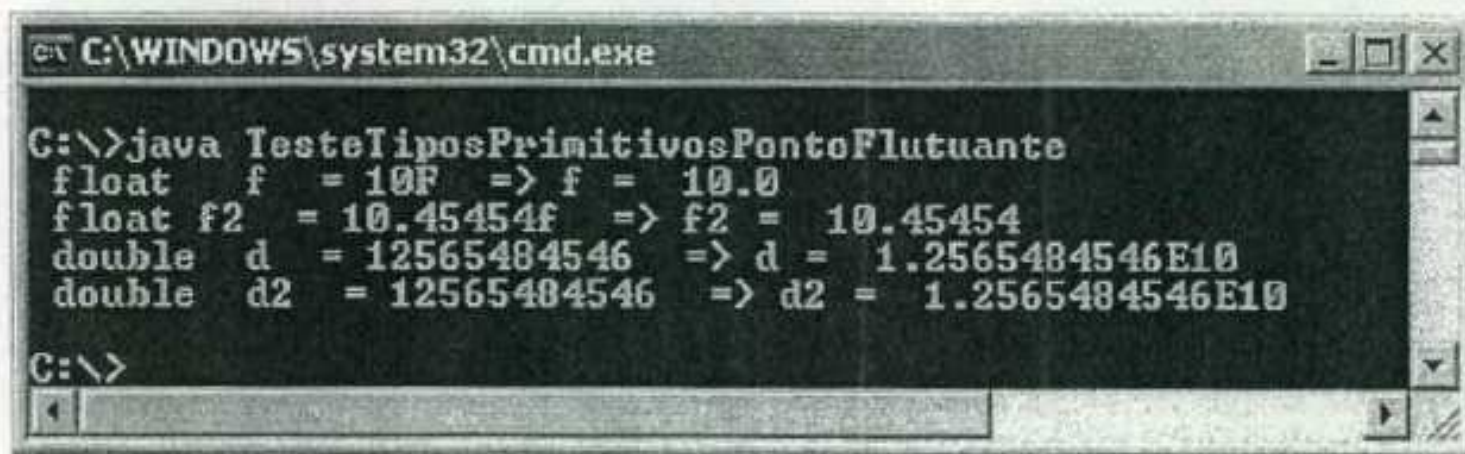
❑ Números com ponto flutuante

Exemplo TesteTiposPrimitivosPontoFlutuante.java

Para compilar: javac TesteTiposPrimitivosPontoFlutuante.java

Para executar: java TesteTiposPrimitivosPontoFlutuante

Saída gerada pela execução da classe:



```
C:\WINDOWS\system32\cmd.exe

C:\>java TesteTiposPrimitivosPontoFlutuante
float f = 10F => f = 10.0
float f2 = 10.45454f => f2 = 10.45454
double d = 12565484546 => d = 1.2565484546E10
double d2 = 12565484546 => d2 = 1.2565484546E10

C:\>
```

Introdução ao Java

Variáveis

☐ Caracteres

- Os caracteres são representados pelo tipo primitivo `char` e são utilizados para expressar uma “tecla”. Devido a utilização do Unicode, cada `char` ocupa 2 bytes na memória, o que permite expressar caracteres “double byte” como japoneses, chineses, entre outros.

■ Literais

- ☐ Literais `char` são expressas incluindo o caractere desejado entre aspas simples.

Exemplo:

```
char meuChar = 'x';
```

- ☐ Naturalmente, esta técnica funciona somente se o caractere desejado estiver disponível no teclado.
-

Introdução ao Java

Variáveis

❑ Caracteres

- Uma outra maneira de expressar uma literal de caracter é especificar o código "Unicode" usando quatro dígitos em hexadecimal precedidos por \u, com a expressão entre aspas simples.

Exemplo:

```
char meuCharUnicode = '\u45671';
```

Introdução ao Java

Variáveis

□ Caracteres

- Java suporta também uma série de sequências de escape usando a barra invertida (\), que é chamada de caracter de escape. A barra invertida indica que um caracter especial deve ser enviado para a saída quando a próximo caracter é combinado com ela, formando uma sequência de escape.

Exemplo:

\n – Nova linha

\f – Nova página

\r – Para retorno

\' – Para aspas simples

\t – Para tabulação

\” – Para aspas duplas

\b – Para backspace

\\ – Para barra invertida

Introdução ao Java

Variáveis

❑ Caracteres

Exemplo: TesteTiposPrimitivosCaracter.java

```
Class TesteTiposPrimitivosCaracter{  
    public static void main (String[] args){  
        char c = 'a';  
        System.out.println("c = 'a' => c = " + c);  
        char asc = 64;  
        System.out.println(" asc= 'a' => asc = " + asc);  
        char espaco = '\u0000';  
        System.out.println("espaco = '\u0000' => espaco = " + espaco);  
        char carinha = '\u0001';  
        System.out.println("carinha '\u0001' => carinha = " + carinha);  
    }  
}
```

Introdução ao Java

Variáveis


❑ Caracteres

Exemplo: TesteTiposPrimitivosCaracter.java

Para compilar: javac TesteTiposPrimitivosCaracter.java

Para executar: TesteTiposPrimitivosCaracter

Saída gerada pela execução:



```

C:\>java TesteTiposPrimitivosCaracter
c = 'a' ==> c = a
asc = 64 ==> asc = @
espaco = '\u0000' ==> espaco = 
carinha = '\u0001' ==> carinha =  
C:\>

```

Introdução ao Java

Variáveis

□ Booleanos

- Variáveis de tipos booleanos podem ser representadas apenas por dois valores: true ou false, elas ocupam apenas 1 bit.
 - Não existe nenhum mapeamento entre true, false e os números inteiros diferente da maioria das linguagens.
-

Introdução ao Java

Variáveis

□ Booleanos

Exemplo: TesteTiposPrimitivosBooleanos.java

```
class TesteTiposPrimitivosBooleanos{  
    public static void main (String[] args){  
        boolean ok = true;  
        System.out.println("boolean ok = true => ok = " + ok);  
        boolean acabou = false;  
        System.out.println("boolean acabou = false => acabou = " + acabou);  
    }  
}
```

Introdução ao Java

Variáveis

❑ Booleanos

Exemplo: TesteTiposPrimitivosBooleanos.java

Para compilar: javac TesteTiposPrimitivosBooleanos.java

Para executar: java TesteTiposPrimitivosBooleanos

Saída gerada pela execução:



```
Prompt de comando
C:\Eclipse\workspace\ExemplosAJ1>java TesteTiposPrimitivosBooleanos
boolean ok    = true ==> ok =  true
boolean acabou = false ==> acabou =  false
```

Introdução ao Java

Variáveis

☐ Reference

- Variáveis do tipo reference armazenam o endereço de memória estendida para um determinado objeto e a quantidade de memória estendida varia de acordo com o objeto em questão.

- Sintaxe para declaração e inicialização de uma variável reference:

`<tipoVariavel> <nomeVariavel> = new <tipoVariavel>();`

- Variáveis do tipo reference podem conter os seguintes valores:

- ☐ null
 - ☐ referência para qualquer objeto cuja classe é de um tipo compatível ao declarado.
 - ☐ referência para um array
-

Introdução ao Java

Variáveis

□ Reference

- Literais: Somente o valor null, representando que a variável não armazena nenhuma referencia.

Exemplo:

```
String s = new String();
```

```
String s2 = "Teste";
```

```
Object meuObject = new Object();
```

```
Object o = null;
```

- A classe String: Geralmente a classe String é uma das primeiras que utilizamos para representar um texto (um conjunto de caracteres) e sua inicialização pode ser feita semelhantemente a inicialização de variáveis de tipos primitivos.
-

Introdução ao Java

Variáveis

□ Variáveis locais

- Variáveis declaradas dentro de métodos ou blocos de códigos são definidas como locais. Este tipo de variável não possui valor de inicialização padrão, portanto, devemos indicar um valor, caso contrário, receberemos um erro de compilação.

Introdução ao Java

Variáveis

❑ Variáveis locais

Exemplo: TesteVariaveisLocais.java

```
Class TesteVariaveisLocais{  
    public static void main (String[] args){  
        // variavel local nome (tipo String) declarada e inicializada  
        String nome = "Sistemas";  
        // variavel local f (tipo float) declarada e inicializada  
        float f = 23.5f;  
        // variavel i (tipo int) declarada mas não inicializada  
        int i;  
        System.out.println("nome : " + nome);  
        System.out.println("f : " + f);  
        System.out.println("i : " + i);  
    }  
}
```

Introdução ao Java

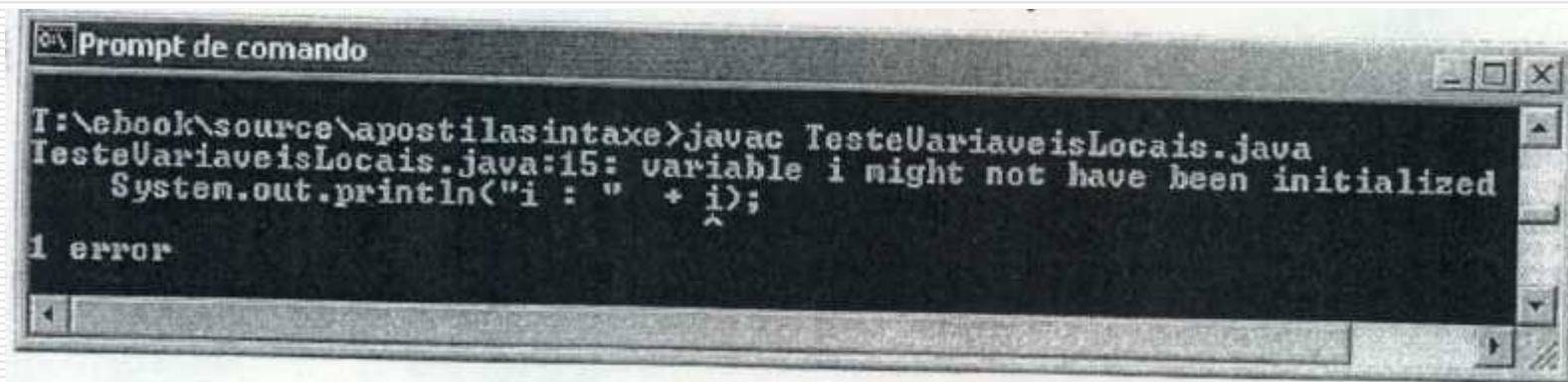
Variáveis

❑ Variáveis locais

Exemplo: TesteVariaveisLocais.java

Para compilar: javac TesteVariaveisLocais.java

Saída gerada pela compilação:



```
Prompt de comando
T:\ebook\source\apostilasintaxe>javac TesteVariaveisLocais.java
TesteVariaveisLocais.java:15: variable i might not have been initialized
    System.out.println("i : " + i);
                               ^
1 error
```

Introdução ao Java

Variáveis

□ Escopo

- O escopo define em qual parte do programa a variável estará acessível. Até agora utilizamos somente declarações de variáveis dentro de métodos (declarações no main).

Exemplo: TesteEscopo.java

```
class TesteEscopo{  
    public static void main (String[] args){  
        {  
            //as variáveis que forem declaradas dentro deste bloco  
            //não serão acessadas fora  
            int quantidade = 23;  
        }  
        System.out.println("quantidade = " + quantidade);  
    }  
}
```

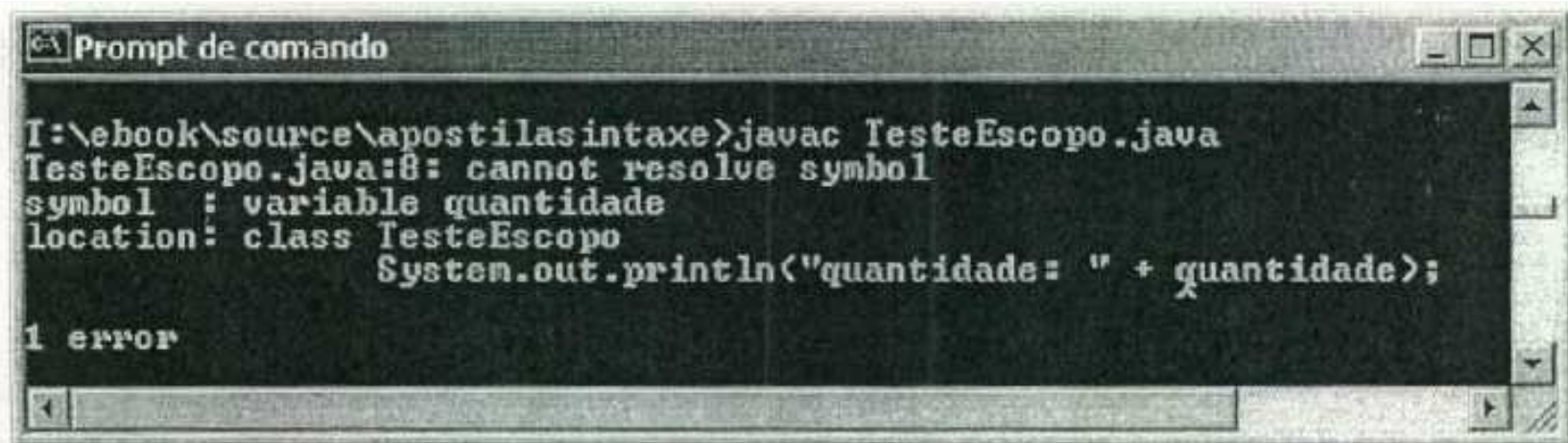
Introdução ao Java

Variáveis

❑ Escopo

- Neste exemplo declaramos uma variável do tipo int dentro de um bloco definido por { e }. Se tentarmos acessar esta variável fora do escopo onde ela foi definida ({ }) teremos o seguinte erro de compilação:

Saída gerada pela compilação:



```

C:\> Prompt de comando

I:\ebook\source\apostilasintaxe>javac TesteEscopo.java
TesteEscopo.java:8: cannot resolve symbol
symbol : variable quantidade
location: class TesteEscopo
    System.out.println("quantidade: " + quantidade);
1 error

```