Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Linguagem de Programação I

Aspectos Fundamentais de Java (Tipos, Operadores e E/S)

- Tiago Maritan
- tiago@ci.ufpb.br

Vamos começar a por a mão na massa!!!

Tipos em Java

- Java é uma **linguagem fortemente tipada**;

- É necessário definir o tipo da variável antes de usá-la;
  - Qual o tipo de dado que ela vai armazenar;
  - Ex: Quantidade de peças em estoque: variável inteira;

```
int quantEstoque;
```

  - Ex: Saldo da conta bancária: variável de ponto-flutuante;

```
float saldo;
```

Tipos de Dados

- Na maioria das linguagens, os tipos de dados não são portáveis
  - Tamanhos diferentes dependendo da plataforma;
  - Ex: `int`: 16 ou 32 bits dependendo da plataforma (C/C++);

- Em Java, os tipos de dados são portáveis
  - Tamanho fixo independente de plataforma;
  - Ex: `int`: 32 bits em todas as plataformas;

Tipos em Java

- Podem ser:
  - **Tipos primitivos (embutidos)**
    - Incorporados na própria linguagem
    - Representados por palavras chaves
    - Ex: `int, float, char, boolean`, etc;
  - **Tipos derivados**
    - Criado pelo programador ou provido pela biblioteca de Java;
    - Ex: Classes (estudaremos mais adiante no curso)

Tipos Primitivos de Java

- Têm tamanho fixo e valores *default*;

Tipo	Qde. de bits	Valor default	Faixa de Valores
`boolean`	8	false	true ou false
`char`	16	\u0000	'\u0000' a '\uFFFF'
`byte`	8	0	-128 a 127
`short`	16	0	-32.768 a 32.767
`int`	32	0	-2.147.483.648 a +2.147.483.647
`long`	64	0	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
`float`	32	0.0	-3.40292347E+38 a +3.40292347E+38
`double`	64	0.0	-1.79769313486231570E+308 a -1.79769313486231570E+308

Declaração de variáveis em Java

- Variáveis em Java devem ser declaradas antes de serem usadas
  - Caso contrário, teremos um **erro de compilação**;

- Formato:

```
<tipo> <nomeDaVariavel>;
```

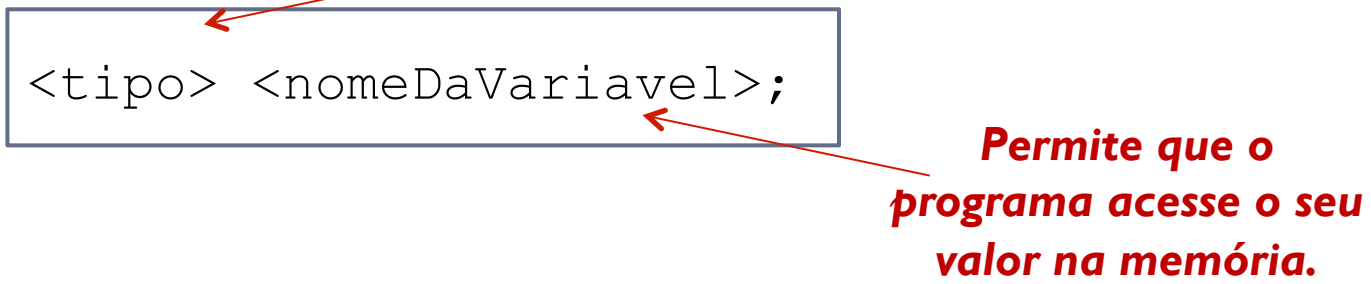
- Exs:

```
int num1;
double dolar;
char inicialNome;
boolean foiComprado;
```

Declaração de variáveis em Java

- Variáveis em Java devem ser declaradas antes de serem usadas
  - Caso contrário, teremos um **erro de compilação**;

- Formato:



- Exs:

```
int num1;
double dolar;
char inicialNome;
boolean foiComprado;
```

Declaração de variáveis

- Variáveis do mesmo tipo podem ser declaradas na mesma linha e separadas por vírgula;

```
int x, y;
double dolar, media;
char;
```

- Também podem ser inicializadas:

```
int x = 10, y = 20;
double dolar = 1.61;
char c = 'a';
```

Declarações de variáveis

- Nomes das variáveis são **identificadores**;
  - Formados por: letras, dígitos, '_' e '$';
  - Não podem começar com dígitos;
  - Não podem ser palavras reservadas de Java (ex: `for, if`)

- Ex: **`botao7, $valor, _real, m_campo, Teste`**
- Ex: **`7button, t.d@ce, tiago maritan, while`**

- **Case Sensitive:** Distinção entre maiúsculas e minúsculas:
  - ex: `MinhaVar` e `minhaVar` são diferentes;

Palavras reservadas do Java 7 SE

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

*** não usadas;**

Declaração de Constantes

- Em Java, usa-se a palavra reservada **`final`** para denotar constantes.

- Ex:

```
final double pi = 3.14;
final int milimetrosPorCentrimento = 10;
```

- Uma vez inicializada, a constante não pode mais ser alterada;
  - A tentativa de alteração implica em um **erro de compilação**;

Operadores em Java

- Todos os operadores possuem:
  - **Resultado**
  - **Aridade**
  - **Precedência**
  - **Associatividade**

Operadores em Java

- Resultado
  - Valor resultante da aplicação do operador sobre seus operandos
  - Ex: aplicação do operador "+" na expressão "2 + 3" resulta em 5;

- Aridade
  - Número de operandos sobre os quais o operador atua
    - ✓ Operadores unários – requerem um operando
    - ✓ Operadores binários – requerem dois operandos
    - ✓ Operador ternário – requer três operandos
  - Ex: operador + tem aridade 2 (i.e., um operador binário)

Operadores em Java

- Precedência
  - Ex: Qual o resultado de "2+3*4"? Por quê?
    - Operador '*' tem maior precedência que '+'
  - Convenção sobre a ordem em que as operações são executadas
  - Operador com maior precedência é aplicado antes

- Associatividade
  - Qual é o resultado de "8/2/2"? Por quê?
    - Operador "/" tem **associatividade à esquerda**;
  - **À esquerda**: operador mais a esquerda é aplicado 1°
  - **À direita:** operador mais a direita é aplicado 1°

Operadores em Java

- Uso de Parênteses
  - Alteram a ordem de aplicação de operadores numa expressão
  - Tornam as expressões complexas mais legíveis
  - Exemplos:

```
int a = 2+(3*4);
int b = ((8/2)/2);
int c = (2 + 2) * (3-9) / 3;
```

Operadores em Java

- Podem ser de 4 tipos:
  - Atribuição;
  - Aritméticos;
  - Relacionais;
  - Lógicos;

Operador de Atribuição

- Operador =
  - Copia o valor de uma variável (expressão ou constante) do lado direito para a variável do lado esquerdo;

```
x = 13; // copia a constante 13 para x
y = x;  // copia o valor de x para y
```

Operadores Aritméticos em Java

Operador	Significado	Exemplo
+	mais unário	+x
–	menos unário (inversão de sinal)	-x
+	adição	x + y
–	subtração	x - y
*	multiplicação	x * y
/	divisão	x / y
%	resto da divisão inteira	x % y
++	incremento unitário	++x ou x++
--	decremento unitário	--x ou x--

Operadores Aritméticos

- Operador divisão (/)
  - Serve para ambas as divisões: inteira e ponto flutuante;
    - Tipos dos operandos é que definem o tipo de divisão;
  - Ou seja:
    - Inteiro / Inteiro => Inteiro
    - Ponto flutuante / Ponto flutuante => Ponto flutuante
    - Ponto flutuante / Inteiro => Ponto flutuante

Operadores Aritméticos

- Operadores de Incremento (++) e Decremento (--)
  - Adicionam ou subtraem 1 ao valor da variável;

```
int n = 12, t = 2;
n++; // mudou n para 13
t--; // mudou t para 1
```

- Podem ser:
  - **Pré-fixos**: `++x` ou `--x;`
  - **Pós-fixos:** `x++` ou `x--;`

- Qual é a diferença?

Operadores Aritméticos

- Operadores de Incremento e Decremento
  - A diferença é quando são usados em expressões:

  - **Pré-fixo**: faz a adição primeiro;
    - Tem maior precedência que os outros operadores aritméticos;
  - **Pós-fixo**: faz a adição depois;
    - Tem a menor precedência de todos os operadores;

```
int m = 7;
int n = 7;
int a = 2 * ++n; // a = 16 e m é 8
int b = 2 * n++; // b = 14 e n é 8
```

Operadores Relacionais

- Produzem sempre valores booleanos (**true** ou **false**)

Operador	Significado	Exemplo	Resultado
==	Igual a	`x == y`	true se x é igual a y; false caso contrário
!=	Diferente de	`x != y`	true se x é diferente de y; false caso contrário
<	Menor do que	`x < y`	true se x é menor do que y; false caso contrário
<=	Menor ou igual a	`x <= y`	true se x é menor ou igual do que y; false caso contrário
>	Maior do que	`x > y`	true se x é menor do que y; false caso contrário
>=	Maior ou igual a	`x >= y`	true se x é maior ou igual do que y; false caso

Operadores Lógicos

- Produzem sempre valores booleanos (true ou false)

Operador	Significado	Exemplo
!	Operador Negação (ou not)	!x
&&	Operador E lógico (ou and)	x && y
||	Operador Ou lógico (ou or)	x || y

Operadores de Atribuição com Operação

Operador	Exemplo	Significado
+=	x += 2	x = x + 2
-=	x -= 1	x = x - 1
*=	x *= 1	x = x * 1
/=	x /= 3	x = x / 3
%=	x %= 5	x = x % 5

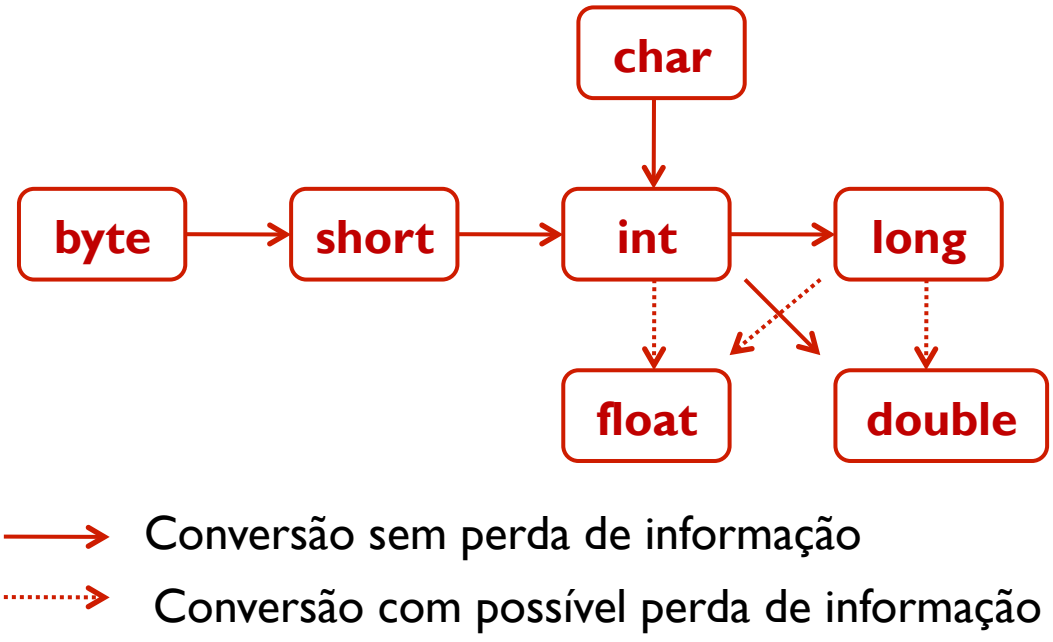
Precedência e Associatividade dos Operadores em Java

GRUPO DE OPERADORES	PRECED.	ASSOC.
!, +, - (unários)	Mais alta	Esquerda
++, -- (pré-fixados)	↓	Esquerda
*, /, %	↓	Esquerda
+, - (binários)	↓	Esquerda
>, >=, <, <=	↓	Esquerda
==, !=	↓	Esquerda
&&	↓	Esquerda
||	↓	Esquerda
=, +=, -=, *=, /=, %= (atribuição)	↓	Direita
++, -- (pós-fixados)	Mais baixa	Esquerda

Conversões entre tipos

- Muitas vezes é necessário converter um tipo numérico para outro.
  - Ex: converter um nº `int` para `float`;

- Conversões válidas em Java:



Conversões entre tipos

- Exemplo:

```
byte b  = 127;
short s = b; // s é 127

int n = 123456789;
float f = n; // f é 1.2345679E8
// houve perda de precisão
```

Conversão entre tipos

- Quando 2 valores de tipos diferentes são combinados, eles são convertidos em tipo comum antes da operação. Ex:

```
int n = 10;
float f = 12.3;
float x = n + f; // n é convertido p/ float
```

- Regras:
  1. Se um operando for `double` => ambos viram `double`;
  2. Caso contrário, se um for `float` => ambos viram `float`;
  3. Caso contrário, se um for `long` => ambos viram `long`;
  4. Caso contrário => todos viram `int`;

Coerção (*casting*) entre tipos

- Também é possível fazer conversões forçadas entre tipos
  - **Casting**: Envolve, em geral, perda de informação;
  - Ex: converter um nº `double` para `int`;

- Coloca o tipo que se deseja converter entre parênteses

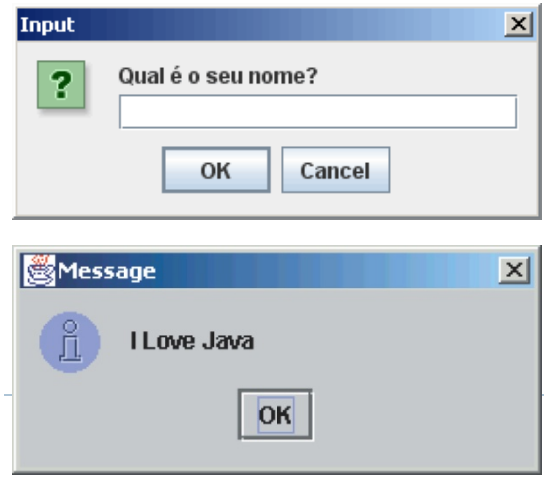
```
double x = 9.997;
int nx = (int) x; // nx é 9.
// (int) - conversão (casting) de double para int

int t = 100;
byte b;
b = (byte) t; // casting de int para byte
```

Entrada e Saída em Java

- Java possui uma extensa biblioteca para manipular E/S
  - **Entrada**: teclado (console ou GUI), mouse, arquivos, rede, etc;
  - **Saída:** console, GUI, arquivo, rede, etc;

- Manipular GUI, arquivos, rede, exige um conjunto de ferramentas e técnicas;

- Então, por enquanto, vamos trabalhar E/S apenas com:
  - Console (Scanner e System.out);
  - Caixa de Diálogo (JOptionPane);



Saída do Usuário - `System.out`

- `System.out` possui alguns métodos para exibir dados no console:
  - `println()` – exibe os dados e depois muda de linha (\n);
  - `print()` – exibe os dados e continua na mesma linha;
  - `printf()` – exibe os dados formatados (similar ao `printf` de C);

```
String nome = "Tiago";
System.out.println("Ola " + nome); // muda de linha
System.out.print("Seja"); // continua na mesma linha
System.out.println("bem vindo"); // muda de linha

double x = 10000/3;
// 8 caracteres e 2 casas de precisão
System.out.printf(" %8.2f", x); // 3333.33
```

Entrada do usuário – Classe Scanner

- Classe `Scanner` (introduzida no Java 5.0)
  - Lê a entrada do teclado e converte-a para tipos primitivos;
  - Separa a entrada do usuário (String) em ***tokens;***
  - **Tokens:** seqüências de caracteres separados por delimitadores
    - Ex: espaço, tabulação, mudança de linha;

Entrada do usuário – Classe Scanner

- Classe `Scanner` (introduzida no Java 5.0)
  - Lê a entrada do teclado e converte-a para tipos primitivos;

- Primeiro, é preciso criar um **Scanner** sobre o canal **System.in**

```
Scanner entrada = new Scanner(System.in);
```

- Então, utiliza-se os métodos da classe Scanner para ler a entrada:
  - `next()` – lê uma palavra (como String);
  - `nextInt()` – lê um int;
  - `nextLine()` – lê uma linha;
  - `nextByte()`, `nextShort()`, `nextDouble()`...

- Maiores detalhes em:
  - http://download.oracle.com/javase/1,5.0/docs/api/java/util/Scanner.html

Entrada do Usuário – Classe Scanner

```
import java.util.Scanner; /* diretiva para usar classe de
                          * outro pacote */
public class TestaEntrada{

    public static void main(String args[]){
        String nome;
        int idade;

        Scanner entr = new Scanner(System.in);
        System.out.print("Qual o seu nome: ");
        nome = entr.next(); // obtem 1ª entrada

        System.out.print("Qual é a sua idade: ");
        idade = entr.nextInt(); // obtem 2ª entrada

         System.out.println("Olá " + nome +
              " próximo ano você terá " + (idade+1) );
       }
}
```

Entrada e Saída – Caixa de Diálogo

- Classe **JOptionPane**
  - Forma gráfica simples para E/S de dados;
  - Necessário importar: `javax.swing.JOptionPane`;
  - Métodos mais usados:
    - `showMessageDialog()` exibe caixa de diálogo de saída de dados;
    - `showInputDialog()` exibe caixa de diálogo para entrada de dados;

- `showInputDialog()`: sempre retorna entrada como String
  - Deve-se converter se entrada for um tipo primitivo
  - Ex: `Integer.parseInt(String x)`: converte x para inteiro;
  - `Double.parseDouble(String x)`: converte x para double;

Entrada e Saída – Caixa de Diálogo

- Exemplo:

```
import javax.swing.JOptionPane;
...
String aux = "";
float nota1 = 0.0, nota2 = 0.0, media = 0.0;

aux = JOptionPane.showInputDialog(null, "Digite a nota1");
nota1 = Float.parseFloat(aux); // converte str para float

aux = JOptionPane.showInputDialog(null, "Digite a nota2");
nota2 = Float.parseFloat(aux); // converte str para float

media = (nota1 + nota2)/2;
JOptionPane.showMessageDialog(null, "Média = " + media);
```


~ programando
Imports System
Public Class Employee
Private _name As St
Private _salary As
Public ReadOnly Pro
Get
Return _nam
End Get
End Property
Public ReadOnly Pro
Get
Return _sal
End Get
End Property
Public Sub Increase
Por hoje é só, hora de salvar tudo!
Sempre que possível, comente seu código
Vendo o código 6 semanas depois ...

Comentários

- Linhas adicionadas para facilitar o entendimento do programa;
- São totalmente ignorados pelo compilador
- Em Java, podem ser de 3 tipos:
  - **Comentário de Linha**: Uma única linha (//)
  - **Comentário de bloco**: Múltiplas linhas (/* */)
  - **Comentário de documentação:** Múltiplas linhas (/** */)
    - Gera documentação automaticamente por meio do javadoc

Comentários

```
/** Exemplo02:
    Essa classe demonstra o uso de variáveis em Java.
    São declaradas variáveis int e double...
*/
public class Exemplo02{
   public static void main(String args[]){
       int x = 10; // declaração variavel int
       double dolar = 2.62;
       /* As linhas seguintes enviam o conteudo
       das variaveis para a tela */
       System.out.println(x);
       System.out.prinln(dolar);
   }
}
```

Universidade Federal da Paraíba

Centro de Informática

Departamento de Informática

Linguagem de Programação I

Aspectos Fundamentais de Java (Tipos, Operadores e E/S)

- Tiago Maritan
- tiago@ci.ufpb.br