LINGUAGEM DE PROGRAMAÇÃO 2
03– VARIÁVEIS E CASTING

Francisco Barretto – francisco.barretto@udf.edu.br

Roteiro

- Comentários
- Variáveis
- Operadores
- Tipos e Valores
- Casting

Comentários

- `/* texto */`

  O compilador ignora tudo entre /* e */

- `/** documentacao */`

  indica um comentário para documentação.
  Utilizado pela ferramenta *javadoc*

- `// texto`

  O compilador ignora todos os caracteres de // até o final da linha

Variáveis

- Tipos de dados
  - Inteiros: byte / short / int / long
  - Reais: float / double
  - Outros: char / boolean

- Nomes de variáveis
  - Série de caracteres Unicode
  - Não pode ser palavra chave
  - Não pode ser nome de outra variável ou classe

Variáveis

		Valores possíveis				
Tipos	***Primitivo***	***Menor***	***Maior***	***Valor Padrão***	***Tamanho***	***Exemplo***
Inteiro	byte	-128	127	0	8 bits	byte ex1 = (byte)1;
	short	-32768	32767	0	16 bits	short ex2 = (short)1;
	int	-2.147.483.648	2.147.483.647	0	32 bits	int ex3 = 1;
	long	-9.223.372.036.854.770.000	9.223.372.036.854.770.000	0	64 bits	long ex4 = 1l;
Ponto Flutuante	float	-1,4024E-37	3.40282347E + 38	0	32 bits	float ex5 = 5.50f;

	double	-4,94E-307	1.79769313486231570E + 308	0	64 bits	double ex6 = 10.20d;
						ou
						double ex6 = 10.20;

Caractere	char	0	65535	\0	16 bits	char ex7 = 194; ou char ex8 = 'a';
Booleano	boolean	false	true	false	1 bit	boolean ex9 = true;

Variáveis

- Dentro de um bloco, podemos declarar variáveis e usá-las. Em Java, toda variável tem um tipo que não pode ser mudado, uma vez que declarado:
  - `tipoDaVariavel nomeDaVariavel;`
- Por exemplo, é possível ter uma idade que guarda um número inteiro:
  - `int idade;`

Variáveis

- Com isso, você declara a variável idade, que passa a existir a partir daquela linha. Ela é do tipo int, que guarda um número inteiro. A partir daí, você pode usá-la, primeiramente atribuindo valores.
- A linha a seguir é a tradução de: "idade deve valer quinze".
  - `idade = 15;`

Variáveis

- Além de atribuir, você pode utilizar esse valor. O código a seguir declara novamente a variável idade com valor 15 e imprime seu valor na saída padrão através da chamada a System.out.println.

```
// declara a idade
int idade;
idade = 15;

// imprime a idade
System.out.println(idade);
```

Variáveis

- Por fim, podemos utilizar o valor de uma variável para algum outro propósito, como alterar ou definir uma segunda variável. O código a seguir cria uma variável chamada `idadeNoAnoQueVem` com valor de idade mais um.

```
// calcula a idade no ano seguinte
int idadeNoAnoQueVem;
idadeNoAnoQueVem = idade + 1;
```

Variáveis

- No mesmo momento que você declara uma variável, também é possível inicializá-la por praticidade:

```
int idade = 15;
```

Variáveis

- Inicialização

```
int i = 10;
char c;
c = 'X';
```

- Variáveis finais

```
final float pi = 3.14159;
```

(final também pode ser utilizado para métodos. Os métodos finais não podem ter subclasses)

Operadores

- Operadores
  - Atribuição: =
  - Aritméticos: + - * / %
  - Unários: ++ --
  - Lógicos: == != < > <= >=

Operadores

- Aritméticos

Operação	Operador	Expressão algébrica	Expressão Java
Adição	+	a + 1	a +1
Subtração	-	b -2	b -2
Multiplicação	*	cm	c * m
Divisão	/	d / e	d / e
Resto	%	f mod g	f % g

Operadores

```
int quatro = 2 + 2;
int tres = 5 - 2;

int oito = 4 * 2;
int dezesseis = 64 / 4;

int um = 5 % 2; // 5 dividido por 2 dá
2 e tem resto 1;
                // o operador % pega o
resto da divisão inteira
```

Exemplo 1

```
class TestaIdade {

  public static void main(String[] args) {
    // imprime a idade
    int idade = 20;
    System.out.println(idade);

    // gera uma idade no ano seguinte
    int idadeNoAnoQueVem;
    idadeNoAnoQueVem = idade + 1;

    // imprime a idade
    System.out.println(idadeNoAnoQueVem);
  }
}
```

Operadores

- Precedência de Operações:

Operador	Operação	Ordem de avaliação(precedência)
* / %	Multiplicação Divisão Resto	Avaliado primeiro. Se houver vários operadores desse tipo serão avaliados da esquerda para a direita
+ -	Adição Subtração	Avaliado em seguida. Se houver vários operadores desse tipo, serão avaliados da esquerda para a direita.
=	Atribuição	Avaliado por último

Operadores

- Precedência de Operações (exemplo)

```
public class Avalia_Precedencia {
    public static void main(String[] args) {
        int a = 30;
        int b = 5;
        int c = 10;
        int total = (a + b + c) / 10;
        System.out.println("O resultado = "+total);
    }
}
```

Operadores

- Operadores Relacionais

Operador de igualdade	Operador de igualdade	Exemplo de condição em Java	Significado da condição em Java
Operadores de igualdade			
=	==	x == y	x é igual a y
?	!=	x!= y	x é diferente de y
Operadores relacionais			
>	>	x > y	x é maior que y
<	<	x < y	x é menor que y
>_	>=	x >= y	x é maior que ou igual a y
<_	<=	x <= y	x é menor que ou igual a y

Variáveis

- Representar números inteiros é fácil, mas como guardar valores reais, tais como frações de números inteiros e outros?
- Outro tipo de variável muito utilizado é o double, que armazena um número com ponto flutuante (e que também pode armazenar um número inteiro).

```
double pi = 3.14;
double x = 5 * 10;
```

Variáveis

- O tipo **boolean** armazena um valor verdadeiro ou falso, e só: nada de números, palavras ou endereços, como em algumas outras linguagens.

```
boolean verdade = true;
```

- `true` e `false` são palavras reservadas do Java. É comum que um boolean seja determinado através de uma expressão booleana, isto é, um trecho de código que retorna um booleano, como o exemplo:

```
int idade = 30;
boolean menorDeIdade = idade < 18;
```

Variáveis

- O tipo char guarda um, e apenas um, caractere. Esse caractere deve estar entre aspas simples.
- Por exemplo, ela não pode guardar um código como '' pois o vazio não é um caractere!

```
char letra = 'a';
System.out.println(letra);
```

- Variáveis do tipo char são pouco usadas no dia a dia.
- Veremos mais a frente o uso das Strings, que usamos constantemente, porém estas não são definidas por um tipo primitivo.

Tipos Primitivos e Valores

- Esses tipos de variáveis são tipos primitivos do Java: o valor que elas guardam são o real conteúdo da variável. Quando você utilizar o operador de atribuição = o valor será copiado.

```
int i = 5; // i recebe uma cópia do valor 5
int j = i; // j recebe uma cópia do valor de i
i = i + 1; // i vira 6, j continua 5
```

Casting

- 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 atribuir ele 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.

```
double d = 3.1415;
int i = d; // não compila
```

Casting

- O mesmo 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
```

Casting

- Já no caso a seguir, é o contrário:

```
int i = 5;
double d2 = i;
```

- 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

Casting

- Às vezes, precisamos que um número quebrado seja arredondado e armazenado num 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 mesmo 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;
```

Casting

PARA: / DE:	byte	short	char	int	long	float	double
byte	----	*Impl.*	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
short	(byte)	----	(char)	*Impl.*	*Impl.*	*Impl.*	*Impl.*
char	(byte)	(short)	----	*Impl.*	*Impl.*	*Impl.*	*Impl.*
int	(byte)	(short)	(char)	----	*Impl.*	*Impl.*	*Impl.*
long	(byte)	(short)	(char)	(int)	----	*Impl.*	*Impl.*
float	(byte)	(short)	(char)	(int)	(long)	----	*Impl.*
double	(byte)	(short)	(char)	(int)	(long)	(float)	----

Casting

TIPO	TAMANHO
boolean	1 bit
byte	1 byte
short	2 bytes
char	2 bytes
int	4 bytes
float	4 bytes
long	8 bytes
double	8 bytes

Bibliografia

Básica:

- DEITEL, H. M.; DEITEL, P. J. Java: Como Programar. 8. ed. São Paulo: Pearson Education, 2010
- MANZANO, José Augusto N. G.; COSTA JUNIOR, Roberto Afonso da. JAVA II: programação e computadores - guia básico de introdução, orientação e desenvolvimento. 1. ed. São Paulo: Érica, 2006
- SANTOS, R. Introdução à Programação Orientada a Objetos Usando Java. 1. Ed. Rio de Janeiro: Campus, 2003.

Contato

francisco.barretto@udf.edu.br