UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

Programação I:

Linguagem Java, Tipos de dados, Variáveis, Operadores e Expressões

Autor: Samuel Feitosa
Contribuições: Jefferson Caramori

Breve Histórico sobre Java

- Início em 1991 pela Sun Microsystems.
- Projeto liderado por James Gosling.
- Ideia era fazer parte de plataforma para dispositivos portáteis.
- Foi liberada formalmente em 1995.

- **O Kit de desenvolvimento é liberado gratuitamente.**

- Atualmente a plataforma tem 3 segmentos:
  - **JavaME:** projetada para dispositivos embarcados e móveis
  - **JavaSE:** para desenvolvimento de aplicativos de desktop e de servidor que não requerem funcionalidades específicas de enterprise.
  - **JavaEE:** para o desenvolvimento de aplicativos corporativos de grande escala, geralmente executados em servidores de aplicativos. Oferece um conjunto extenso de APIs e serviços específicos de enterprise, como segurança, persistência de dados, mensageria, transações distribuídas e acesso a recursos.

- Sun foi comprada pela Oracle em 2009.

Principais características

- Portabilidade;
- Orientação a objetos;
- Simplicidade e Legibilidade;
- Multithreading;
- Ampla biblioteca de APIs;
- Desenvolvimento de aplicações Web;
- Grande comunidade e ecossistema;
- Sem ponteiros;
- Uma das principais linguagens para desenvolvimento em aplicações Mobile.

Tipos de dados

Tipos de dados primitivos

- Estabelecem um conjunto de valores que podem ser representados em um programa.

Tipo	Significado	Tamanho	Valores possíveis
`byte`	Números inteiros	8 bits	-128 a 127
`short`		16 bits	-65536 a 65535
`int`		32 bits	-2^{32} a $2^{32} - 1$
`long`		64 bits	-2^{64} a $2^{64} - 1$
---	---	---	---
`float`	Números real	32 bits	1.40239849^{-46} a 3.40282347^{+38}
`double`		64 bits	$4.94065645841246544^{-324}$ a $1.79776931348623157^{+308}$
---	---	---	---
`boolean`	Valor lógico	1 byte	Falso e Verdadeiro
---	---	---	---
`char`	Um caractere	16 bits	Por armazenar um caractere unicode ou um inteiro entre 0 e 65535

Tipos de dados

- Nas aulas vamos utilizar muito o tipo **String**, para armazenar conjuntos de caracteres, pois teoricamente é limitado a 2^{31} - 1, ou aproximadamente 2 bilhões de caracteres, que é limitado pelo índice que acessa esse conjunto de caracteres cujo tamanho é 32 bits.

- Uma variável do tipo String não é de um tipo primitivo, mas sim um tipo de referência, pois estes, representam objetos que armazenam referências a esses objetos na memória.

    - Como o assunto é relativamente complexo, vamos estudar sobre Strings em uma aula posterior.

Variáveis

- Variáveis são uma unidade de armazenamento que contém um valor associado a um objeto.
    - Os valores associados são pertencente a um certo tipo.
    - Nome da variável deve descrever seu propósito.
    - Informações nas variáveis ficam em memória.

- Regras para nomes de variáveis:
    - Sequência de um ou mais caracteres alfanuméricos.
    - Deve iniciar com letra, underline ou cifrão ($).
    - Não podem ter símbolos gráficos ou espaços.
    - Em Java normalmente usa-se o padrão CamelCase (Ex: "nomeCompleto", "idadeDoUsuario", "calcularSoma",).

Declaração e escopo de variáveis

- Declarar uma variável é associar um tipo, um nome e um valor.

<Tipo> <nome1> [, <nome2> [, <nomeN>]];

```
int i = 0;
double total, valorMedio;
```

- O contexto onde é declarada uma variável e onde ela é válida, é denominado de escopo.
- Uma variável local é declarada dentro de um método ou construtor, e esta vai existir dentro do bloco onde foi criada.

```
bloco {
    //variáveis locais
    // visível somente aqui
}
```

- Uma variável de Instância/Objeto é declarada dentro de uma classe, e esta pode ser acessada em qualquer lugar do dentro do objeto.

Declaração e escopo de variáveis

- Declarar uma variável é associar um tipo, um nome e um valor.

<Tipo> <nome1> [, <nome2> [, <nomeN>]];

```
int i = 0;
double total, valorMedio;
```

- Uma variável local é declarada dentro de um método ou construtor, e esta vai existir dentro do bloco onde foi criada.

```
for (int i = 1; i <= 5; i++) {
    int c= 10;
    System.out.println(i);
}
```

- O contexto onde é declarada uma variável e onde ela é válida, é denominado de escopo.
  - Uma variável de Instância/Objeto é declarada dentro de uma classe, e esta pode ser acessada em qualquer lugar do dentro do objeto.

Exemplo 2

```
public class Teste {// Declaração da Classe teste

    public static void main(String[] args) { // Declaração do método principal/Main
        int x = 10; // variável de instância/objeto

        if (x > 10) {
            int y = 50; // variável local
            System.out.println(y);
        }
        //System.out.println(y); // Variável y não é visível
        System.out.println(x);
    }
}
```

Exemplo 3

```
public class Teste {// Declaração da Classe teste

    public static void main(String[] args) { // Declaração do método principal/Main
        for (int i = 0; i <= 10; i++) {  // variável local
            System.out.println(i);
        }
        //System.out.println(i);  // Variável "i" não é visível
    }
}
```

Entrada e saída de dados

- Entrada:
  - Vamos utilizar a classe Scanner, que permite a entrada formatada de dados.
  - Principais métodos: nextInt(), nextByte(), nextInt(), nextFloat(), nextLong(), nextShort().
  - Para ler uma String, usamos next() ou nextLine().

- Saída:
  - Saída simples: System.out.println(“Mensagem”);
  - Saída formatada: System.out.printf(“Mensagem”)

Exemplo 4

```java
import java.util.Scanner;

public class Teste {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Digite um número positivo qualquer:");
        int valor = sc.nextInt();

        System.out.println("O valor lido é: " + valor);
        sc.close();
    }
}
```

- É necessário importar a classe Scanner;

Saída de dados formatada

- Utiliza-se os marcadores: %d para inteiros, %f para reais, %c para caractere e %s para String;

```
public class Teste {
    public static void main(String[] args) {

        // Adicionando marcador %n para quebra de linha
        System.out.printf("Hello world! %n");
        //Hello world!

        System.out.printf("Valor inteiro: %d é inteiro, ok?%n", 20);
        //Valor inteiro: 20 é inteiro, ok?
        double x = 10.0, y = 3.0;
        System.out.printf("%.2f / %.2f = %.2f%n", x, y, x / y); // 3/10=3.3333
        //10,00 / 3,00 = 3,3
    }
}
```

Operadores e expressões

Operadores e expressões

Java oferece um conjunto bem amplo de operadores e expressões, nos quais podemos separar em 7 grupos:

1) Operadores Aritméticos: São usados para realizar operações matemáticas básicas, como adição (+), subtração (-), multiplicação (*), divisão (/), resto (%).

```
int a = 10;
int b = 3;
// Operador aritmético
int soma = a + b;
System.out.println("Soma de A + B: " + soma);
```

2) Operadores de Atribuição: São usados para atribuir valores a variáveis e realizar operações aritméticas. Exemplos: `=`, `+=`, `-=`, `*=`, `/=`, `%=`.

```
int c = 5;
//Operador de atribuição
c += 3; // c = c + 3;
System.out.println("Valor de x após a atribuição: " + c);
```

Operadores e expressões

3) Operadores de Comparação (Relacionais): São usados para comparar valores. Eles retornam um valor booleano que indica se a comparação é verdadeira ou falsa. Exemplos: Igual (`==`), Diferente (`!=`), Menor (`<`), Maior (`>`), Menor ou igual (`<=`), Maior ou igual (`>=`).

```
int d = 10;
int e = 5;
//Operador de comparação
boolean maiorQue = (d > e);
System.out.println("D é maior que E: " + maiorQue);
```

4) Operadores Lógicos: São usados para realizar operações lógicas em valores booleanos. Exemplos: `&&` (AND lógico), `||` (OR lógico), `!` (NOT lógico).

```
boolean f = true;
boolean g = false;
//Operador lógicos
boolean resultadoOr = f || g;
System.out.println("OR lógico: " + resultadoOr);
```

5) Operadores de Incremento/Decremento: São usados para incrementar ou decrementar o valor de uma variável. Exemplos: `++` (incremento), `--` (decremento).

```
int h = 5;
h++; // Incremento
System.out.println("Valor de a após incremento: " + h);
```

Operadores e expressões

6) Operadores de Bitwise: São usados para realizar operações em nível de bits. Exemplos: & (AND bitwise), | (OR bitwise), ^ (XOR bitwise), ~ (NOT bitwise), << (shift left), >> (shift right), >>> (shift right sem sinal).

```
int a = 5; // 101
int b = 3; // 011

int bitwiseAnd = a & b; // 001
int bitwiseOr = a | b;  // 111
int bitwiseXor = a ^ b; // 110
int bitwiseComplemento = ~a; // 010

int deslocamentoEsquerda = a << 1; // 1010 (resultado: 10)
int deslocamentoDireita = a >> 1;  // 10   (resultado: 2)
```

7) Operadores Condicionais (Ternários): São utilizados para fazer operações condicionais em uma única linha. Exemplo: `expressão ? valor_se_verdadeiro : valor_se_falso`.

```
int k = 5;
int l = 10;
// Condicionais / Ternários
int maior = (k > l) ? k : l;
System.out.println("O maior número é: " + maior);
```