Análise Orientada a objetos

Aula 2

Prof. Me. Juliana Costa-Silva

Na aula de hoje...

1. Strings
2. Caracteres Especiais
3. Tipos de dados
4. Identificadores
5. Operadores
6. Estruturas de controle
7. Atividades

Concatenação de String

A concatenação de String é realizada com o uso do operador (+)

Sintaxe:

<String1> + <String2> = <String1String2>

<String> + <DadoPrimitivo> = <StringDadoPrimitivo>

Exemplo Concatenação

```java
public class ExemploConcatenacao {
    public static void main(String[] args) {
        String texto1 = ">> O operador de concatenação (+) ";
        String texto2 = "é muito prático ";
        String texto3 = texto1 + texto2 + "!";
        System.out.println(texto3 + "<<");
        System.out.println(">> 2 + 5 = " + 2 + 5); //Incorreto
        System.out.println(">> 2 + 5 = " + (2 + 5)); //Correto
    }
}
```

Saída

```
>> O operador de concatenação (+) é muito prático !<<
>> 2 + 5 = 25
>> 2 + 5 = 7
```

Caracteres especiais

Seqüência de escape	Descrição da ação
\n	**Nova linha**. Posiciona o cursor de tela no início da próxima linha.
\t	**Tabulação horizontal**. Move o cursor de tele para a próximo parada de tabulação
\r	**Retorno de Carro**. Retorna o curso para o início da linha.
\\	**Barra Invertida**. Utilizado para imprimir um caractere de barra invertida.
\"	**Aspas Duplas**. Utilizado para imprimir um caractere de aspas duplas.
\uXXXX	**Unicode**. Insere o caractere de código Hexadecimal XXXX.

Tipos de dados

Java é uma linguagem **fortemente tipada**, ou seja, só podemos usar uma variável depois de definirmos um tipo pra ela.

As variáveis podem ser de dois tipos:

Tipos de Dados Primitivos

São semelhantes as variáveis comuns em linguagens estruturadas, simplesmente armazenam um valor.

Tipos de dados Construídos

São as classes construídas, que podem estar definidas no Java ou serem criadas pelo programador.

Tipos de dados Primitivos

	Tipo	Tamanho	Valores
Lógico	**boolean**	1 bit	true ou false
Inteiro	**char**	16 bits	'\u0000' até '\uFFFF' (0 até 65535)
	byte	8 bits	-128 até +127
	short	16 bits	-32.768 até +32.767
	int	32 bits	-2.147.483.648 até +2.147.483.647
	long	64 bits	-9.223.372.036.854.775.808 até +9.223.372.036.854.775.807
Ponto-Flutuante	**float**	32 bits	-3,4E-38 até 3,4E+38
	double	64 bits	-1,7E-308 até 1,7E+308

Identificadores

- São nomes de variáveis, métodos, classes, interfaces e pacotes, definidos pelo usuário dentro do código
  - Os nomes são **case-sensitive** (diferenciam maiúscula de minúscula).
  - podem conter letras, números (**0 a 9**), sublinhado (**_**) e cifrão (**$**).
  - NÃO podem começar com números.
  - NÃO podem possuir espaços em branco
  - NÃO podem ser iguais as palavras reservadas pela linguagem.
- Extras
  - Usar nomes sugestivos e seguir as recomendações da Oracle.

Exemplo

Identificadores válidos

Idade e idade: **São variáveis diferentes, pois Java diferencia minúscula de maiúscula.**

- int **Idade**;
- int **idade**;
- String **nomeCliente**;
- int **_A$_09**;

Identificadores inválidos

- int **9Numeros**; // iniciando com número
- String **Nome Cliente**; // possui espaço em branco
- public class **%exemplo%**; // possui caractere inválido
- int **return**; // uso de palavra reservada

Palavras Reservadas

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

Padrão da linguagem

Variáveis e métodos

- Iniciam o nome com letra minúscula;
- Nomes compostos separar por meio de letras maiúsculas
- **Ex:** nomeDeVariavel, nomeDeMetodo.

Padrão da linguagem

Classes

- Iniciam o nome com letra Maiúscula;
- Nomes compostos separar por meio de letras maiúsculas
- **Ex:** NomeDeClasse.

Constantes

- Todas as letras Maiúsculas, separadas por (_)
- Ex: TOTAL_ALUNOS.

Operadores

- (=) Copia o valor da variável da direita para a variável da esquerda.
- int idade = 57;
- double salario = 1300.00, abono = 600.00;

Operadores aritméticos

Adição	+	num1 + num2	num1 somado com num2
Subtração	-	num1 - num2	num2 subtraído de num1
Multiplicação	*	num1 * num2	num1 multiplicado por num2
Divisão	/	num1 / num2	num1 dividido por num2
Resto da divisão	%	num1 % num2	o resto da divisão de num1 por num2

Operadores relacionais

Avaliam uma expressão e retornam **true** ou **false**;

Maior que	>	n1 > n2	n1 é maior que n2?
Maior ou igual	>=	n1 >= n2	n1 é maior ou igual a n2?
Menor que	<	n1 < n2	n1 é menor que n2?
Menor ou igual	<=	n1 <= n2	n1 é menor ou igual a n2?
Igual a	==	n1 == n2	n1 é igual a n2?
Diferente de	!=	n1 != n2	n1 é diferente de n2?
Instancia de	**instanceof**	n1 **instanceof** n2	n1 é um objeto instanciado da classe n2?

Operadores lógicos

Avaliam uma expressão lógica e retornam **true** ou **false**;

Não Lógico (NOT)	**!**	**!** n1	Retorna o valor invertido de n1. Se n1 é true retorna false, se é false retorna true
Operação E (AND)	**&&**	n1 **&&** n2	Retorna true somente se n1 e n2 forem true, caso contrário retorna false. Se n1 for false n2 não é avaliada
E (binário)	**&**	n1 **&** n2	Retorna true somente se n1 e n2 forem ambos true, caso contrário retorna false. **Ambas expressões são sempre avaliadas**
Operação OU (OR)	**||**	n1 **||** n2	Retorna true se n1 for true, se n2 for true ou se ambos forem true. Só retorna false se n1 e n2 forem ambos false. Se n1 for true n2 não é avaliada
OU (binário)	**|**	n1 **|** n2	Retorna true se n1 for true, se n2 for true ou se ambos forem true. Só retorna false se n1 e n2 forem ambos false. **Ambas expressões são sempre avaliadas**
OU Exclusivo (XOR)	**^**	n1 **^** n2	Retorna true somente se n1 for true ou se somente n2 for true. Caso os dois sejam ao mesmo tempo true ou false retorna false.

Operadores de atribuição composta

Op.	Exemplo	Equivalente	Descrição
+=	n1 **+=** n2	n1 = n1 + n2	É armazenado em n1 o resultado da soma de n1 com n2
-=	n1 **-=** n2	n1 = n1 - n2	É armazenado em n1 o resultado da subtração de n2 de n1
***=**	n1 ***=** n2	n1 = n1 * n2	É armazenado em n1 o resultado da multiplicação de n1 por n2
/=	n1 **/=** n2	n1 = n1 / n2	É armazenado em n1 o resultado da divisão de n1 por n2
%=	n1 **%=** n2	n1 = n1 % n2	É armazenado em n1 o resultado do resto da divisão de n1 por n2

Operadores de incremento e decremento

Op.	Exemplo	Equivalente	Descrição
++	n1++	n1 = n1 + 1	Retorna o valor antigo de n1 e depois aumenta em 1 o valor de n1
	++n1	n1 = n1 + 1	Aumenta em 1 o valor de n1 e depois retorna o valor novo de n1.
−−	n1−−	n1 = n1 − 1	Retorna o valor antigo de n1 e depois Diminui em 1 o valor de n1.
	−−n1	n1 = n1 − 1	Diminui em 1 o valor de n1 e depois retorna o valor novo de n1.

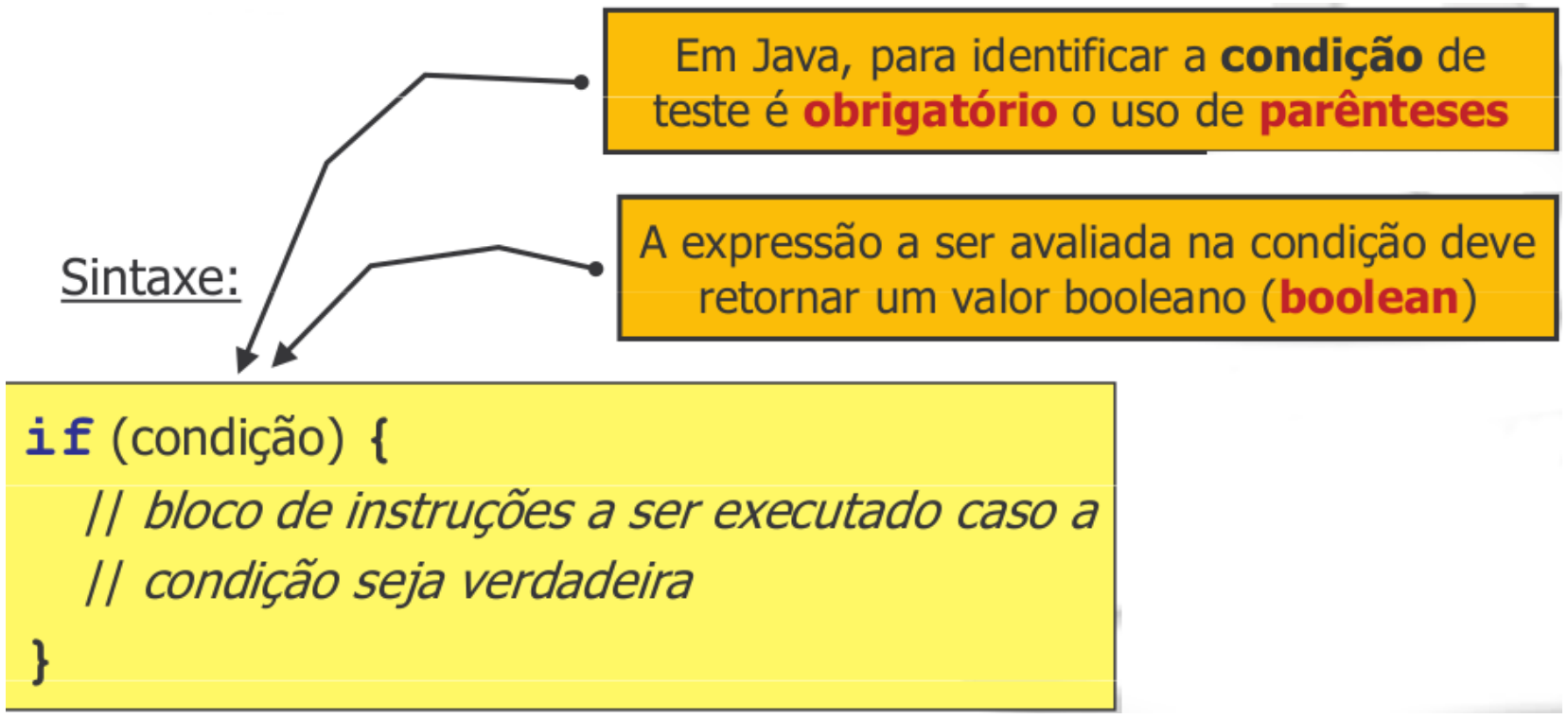
Precedência de Operadores

Ordem na qual as operações serão executadas.

Ordem	Operador
1	( ) parênteses
2	++ pós-incremento e -- pós-decremento
3	++ pré-incremento e -- pré-decremento
4	! Negação
5	* Multiplicação e / Divisão
6	% Resto da divisão
7	+ Soma e – Subtração
8	< menor que, <= menor ou igual, > maior que, >= maior ou igual
9	== igual e != diferente
10	& (e binário)
11	| (ou binário)
12	^ (ou exclusivo binário)
13	&& (e lógico)
14	|| (ou lógico)
15	?: condicional
16	= atribuição

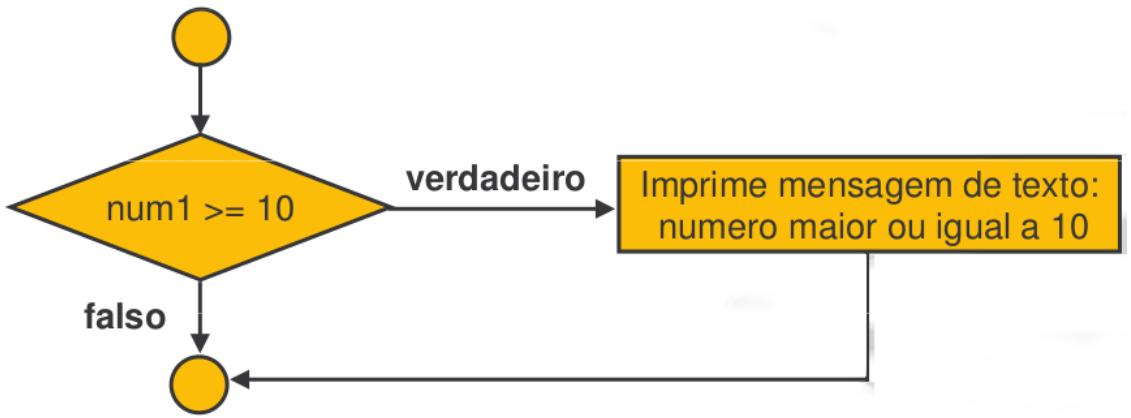
Estruturas de seleção - IF

A partir do teste de uma condição executa ou não um conjunto de instruções.



Estruturas de seleção - IF

```
1  int num1 = 8;
2  if (num1 >= 10){
3      System.out.println("Numero maior ou igual a 10");
4  }else{
```



Estruturas de seleção - IF/ ELSE

```
1  int num1 = 8;
2  if (num1 >= 10){
3      System.out.println("Numero maior ou igual a 10");
4  }else{
5      System.out.println("Numero menor que 10");
6  }
```

Estruturas aninhadas

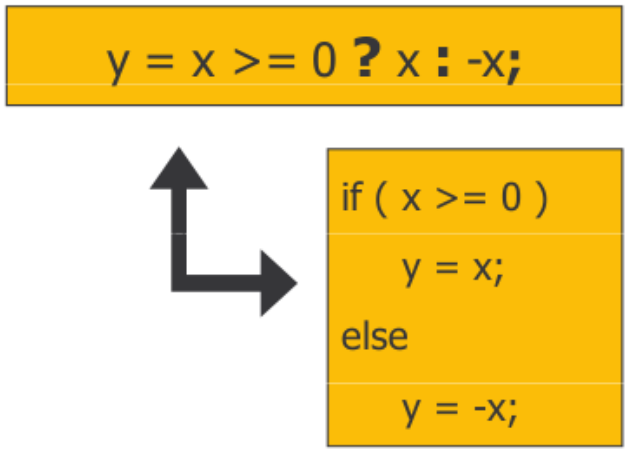
```
1  int nota = 70;
2  if(nota >= 90){
3      System.out.println("Conceito A");
4  }else{
5      if(nota >= 80){
6          System.out.println("Conceito B");
7      }else{
8          if(nota >= 70){
9              System.out.println("Conceito C");
10         }else{
11             System.out.println("Insuficente");
12         }
13     }
14 }
```

Estruturas aninhadas

```
1  int nt = 70;
2  if(nt > 90){
3      System.out.println("Conceito A");
4  }else if(nt > 80){
5      System.out.println("Conceito B");
6  }else if(nt >= 70){
7      System.out.println("Conceito C");
8  }else{
9      System.out.println("Insuficente");
10 }
```

Operador Ternário

teste lógico ? valorSeVerdadeiro : valorSeFalso



Exemplo:

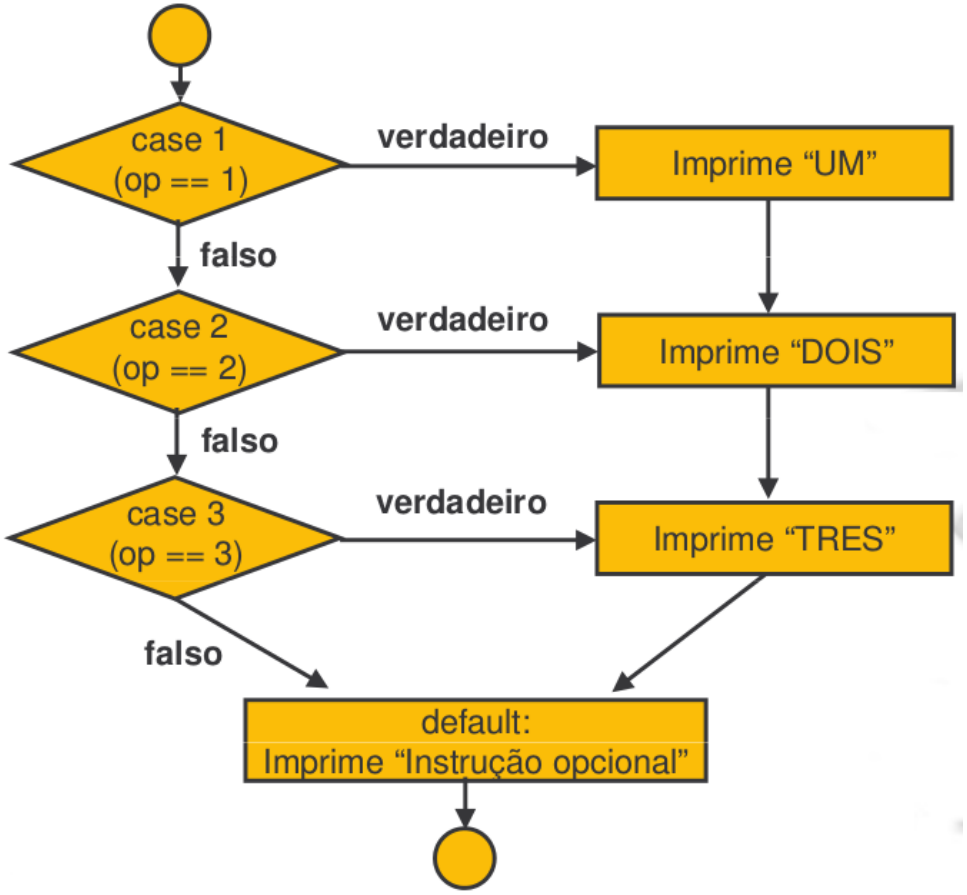
```
1  System.out.println(nt >=7 ? "Aprovado":"Reprovado");
```

Estruturas de seleção - **Switch**

Chamada de **estrutura de seleção múltipla**, pois a partir da comparação de um valor passado com os outros valores definidos, seleciona um ponto do qual iniciará a execução de um conjunto de instruções.

```
1  int opcao = 2;
2  switch(opcao){
3      case 1:
4          System.out.println("UM");
5      case 2:
6          System.out.println("DOIS");
7      default:
8          System.out.println("Opcional");
9  }
```

Estruturas de seleção - **Switch**



Estruturas de seleção - **Switch**

```
1  int opc = 2;
2  switch(opc){
3      case 1:
4          System.out.println("UM");
5          break;
6      case 2:
7          System.out.println("DOIS");
8          break;
9      default:
10         System.out.println("Opcional");
11 }
```

Atividade

1. Faça um programa usando os operadores relacionais que escreva no console, seguinte saída:

```
Os valores das variáveis...
    i = 37
    j = 42
    k = 42
MAIOR QUE...
    i > j = false
    j > i = true
    k > j = false
MAIOR OU IGUAL A...
    i >= j = false
    j >= i = true
    k >= j = true
MENOR QUE....
    i < j = true
    j < i = false
    k < j = false
MENOR OU IGUAL A...
    i <= j = true
    j <= i = false
```

Atividade

2. Examine o código abaixo e determine qual o valor das variáveis i, j, x e y depois desses passos.

```
int i = 10;
int j = 2;
int x = 0;
int y = 0;
j++;
++i;
x = i++ + j;
y = ++j + ++i;
```

Atividade

3. Crie um programa que obtenha a média de 3 números. Considere o valor para os três números como sendo 10, 20 e 45. O resultado esperado do exercício é:

número1 com o valor 10
número2 com o valor 20
número3 com o valor 45
A média é 25

4. Dada as expressões abaixo, reescreva-as utilizando parênteses de acordo com a forma como elas são interpretadas pelo compilador.

```
1  int a = 10, b =5, c =2, d=3, e=1, f=5, g=7, h=2, i=2;
2
3  System.out.println(" - " + (a/b^c^d-e+f-g*h+i));
```

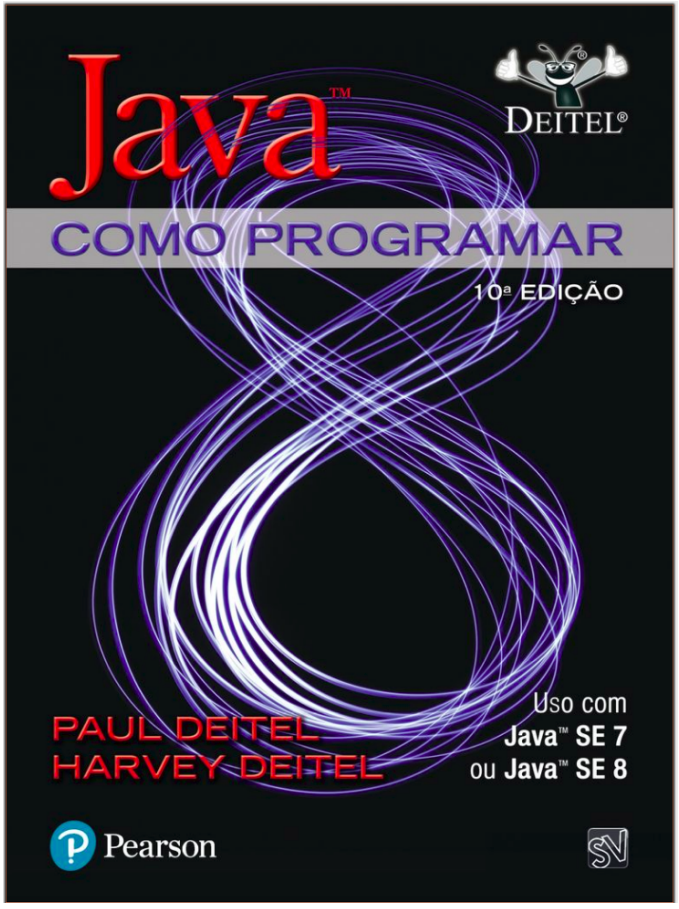
Atividade

5. Faça um programa que leia três números reais representando os lados de um triângulo. Primeiramente verifique se os três lados formam um triângulo. Caso não formam imprima uma mensagem de erro para o usuário. Caso os lados formem um triângulo imprima qual é o tipo dele. Lembre-se que:

- Só é triangulo se: *um de seus lados deve ser maior que o valor absoluto (módulo) da diferença dos outros dois lados e menor que a soma dos outros dois lados*.
- **Equilátero:** todos os lados congruentes;
- **Isósceles:** dois lados congruentes;
- **Escaleno:** todos os lados com medidas distintas.

Leitura complementar

Para mais informações sobre JAVA, leia:



Capítulo 1 e 2: [Deitel, 2016]

Referências

Deitel, Paul J.; Deitel, H. M. (2016).
Java 8: Como programar. 10ª Edição.
Pearson.