



---

## Laboratório 02 – Java básico

Atividade individual.

Crie um novo projeto no NetBeans para este exercício contendo uma classe principal para teste (sugestão: `Teste`).

No método `main()`, crie duas variáveis de cada tipo primitivo do Java: `byte`, `short`, `int`, `long`, `float`, `double`, `char` e `boolean`:

```
1 public static void main(String[] args) {  
2     java.util.Scanner sc = new java.util.Scanner(System.in);  
3     byte byte1, byte2;  
4     short short1, short2;  
5     int int1, int2;  
6     long long1, long2;  
7     float float1, float2;  
8     double double1, double2;  
9     char char1, char2;  
10    boolean A, B;  
11    ...  
12 }
```

Adicione linhas para exibir cada uma das variáveis criadas usando o método `printf()` (imprime formatado), conforme descrito no exemplo do quadro ao início da próxima página.

Tente executar o código e veja que uma mensagem de erro é exibida dizendo que a variável pode não ter sido inicializada (repare que as variáveis estão sublinhadas pelo NetBeans). Algumas maneiras de se inicializar variáveis no Java são:

- Na própria declaração, adicionando um `=` e o valor inicial após o nome da variável;
- Antes da primeira utilização, atribuindo-se um valor qualquer com uma declaração do tipo `variável = <valor>;`
- Lendo um valor do console via um objeto `java.util.Scanner`, utilizando o método apropriado para cada tipo (`nextInt()`, `nextFloat()`, etc.);



Tabela 1: Tipos de dados em Java

Tipo	Tam. (bits)	Valores	Valor padrão	Descrição
<code>boolean</code>	1	<code>false</code> ou <code>true</code>	<code>false</code>	
<code>char</code>	16	'\u0000' a '\uFFFF' (0 a 65535)	'\u0000'	(conj. caracteres Unicode ISO)
<code>byte</code>	8	-128 a 127 ( $-2^7$ a $+2^7 - 1$ )	0	
<code>short</code>	16	-32768 a +32767 ( $-2^{15}$ a $2^{15} - 1$ )	0	
<code>int</code>	32	-2.147.483.648 a +2.147.483.647 ( $-2^{31}$ a $2^{31} - 1$ )	0	
<code>long</code>	64	$-2^{63}$ a $+2^{63} - 1$	0	
<code>float</code>	32	-3,4E+38 a -1,4E-45; +1,4E-45 a +3,4E+38; $\pm\text{inf}$ ; NaN	0.0	ponto flutuante (IEEE754)
<code>double</code>	64	-1,8E+308 a -4,9E-324; +4,9E-324 a +1,8E+308; $\pm\text{inf}$ ; NaN	0.0	ponto flutuante (IEEE754)

No mesmo programa, adicione ao final as seguintes instruções e observe os resultados ao tentar executar:

```

1 ...
2 short1 = byte1; short2 = int2;
3 int1 = short1; int2= double2;
4 float1 = int1; float2 = double2;
5 System.out.printf("Variaveis short: %d e %d\n", short1, short2);
6 System.out.printf("Variaveis inteiras: %d e %d\n", int1, int2);
7 System.out.printf("Variaveis float: %f e %f\n", float1, float2);

```

O NetBeans avisa sobre uma possível perda de precisão, sublinhando as ocorrências.

*Type casting* permite alterar o tipo do resultado da operação subsequente, antes de atribuir a outra variável. Tente novamente fazendo uso do recurso *type casting*, como dado no exemplo:

```

1 short2 = (short)int2;
2 int2 = (int)double2;
3 float2 = (float)double2;

```

Execute as seguintes operações e exiba os resultados usando `printf()`.

```

1 int resultado1 = (int1+short1+byte1)*3;
2 System.out.printf("Resultado: %d\n", resultado1);
3 int resultado2 = (int)(double1+float1)%5;
4 System.out.printf("Resultado: %d\n", resultado2);
5 double resultado3 = (int1+short1+byte1)/9;

```

```
6 System.out.printf("Resultado: %f\n", resultado3);
```

Note que houve perda de precisão no resultado3, zerando a parte fracionária (XX.00000). Como corrigir o problema?

Utilize as duas variáveis booleanas, A e B, para verificar os operadores lógicos apresentados nas tabelas abaixo.

```
1 A = false, B= true; // Alterne os valores e verifique as tabelas
2 System.out.printf("%b && %b = %b\n", A, B, A && B); // A e B
3 System.out.printf("%b || %b = %b\n", A, B, A || B); // A ou B
4 System.out.printf("%b ^ %b = %b\n", A, B, A ^ B); // A Xou B
5 System.out.printf("%b == %b = %b\n", A, B, A == B); // A = B ?
6 System.out.printf("%b != %b = %b\n", A, B, A != B); // A /= B ?
```

Finalmente vamos utilizar uma variável do tipo `String` para formatar a saída em uma tabela com largura de coluna fixa. Siga o exemplo abaixo para as demais linhas.

```
1 String tipo = "short";
2 System.out.printf("%7s: %6d %6d\n", tipo, short1, short2);
3 tipo = "int";
4 System.out.printf("%7s: %6d %6d\n", tipo, int1, int2);
5 tipo = "float";
6 System.out.printf("%7s: %6.1f %6.1f\n", tipo, float1, float2);
7 tipo = "double";
8 System.out.printf("%7s: %6.1f %6.1f\n", tipo, double1, double2);
9 tipo = "char";
10 System.out.printf("%7s: %6c %6c\n", tipo, char1, char2);
11 tipo = "boolean";
12 System.out.printf("%7s: %6b %6b\n", tipo, char1, char2);
```

Dados numéricos de diversos tipos podem adicionados a uma `String`, estes são automaticamente convertidos para texto usando o formato padrão, caso não especificado:

```
1 System.out.printf("\nVariáveis\n");
2 System.out.printf("char: %c e %c\n", char1, char2);
3 System.out.printf("int: %d e %d\n", int1, int2);
4 System.out.printf("float: %f e %f\n", float1, float2);
5 // Pode ser exibido alternativamente da forma
6 String string1 = "\nVariáveis\nchar: " + char1 + " e " + char2
7               + "\nint: " + int1 + " e " + int2
8               + "\nfloat: " + float1 + " e " + float2;
9 System.out.printf("String: %s\n", string1);
```

Para podermos definir o formato numérico a ser exibido na `String`, utilizamos o método `format()` da classe `String`, como segue:

```

1 String string2 = String.format("\nVariáveis" +
2                               "\n%7s:%6c e %6c" +
3                               "\n%7s:%6d e %6d" +
4                               "\n%7s:%6.1f e %6.1f" +
5                               "\n%7s:%6.1f e %6.1f",
6                               "char", char1, char2,
7                               "int", int1, int2,
8                               "float", float1, float2,
9                               "double", double1, double2 );
10 System.out.println(string2);

```

## Exercícios

1. Utilizando seu programa, descreva o que acontece quando atribuímos valores fora dos limites apresentados na Tabela 1 para cada tipo de variável.
2. Qual a regra geral quando atribuímos o valor de uma variável de um tipo a outro? Quando devemos usar *Type cast*?
3. Ocorre arredondamento ao passarmos uma variável de precisão maior para uma de precisão menor, como de `double` para `int`?
4. Como resolver o problema de perda de precisão no resultado?
5. O que acontece na operação `double trezQuartos = 3/4`? Por quê? Como corrigir?
6. Qual a diferença entre os operadores lógicos `^` (ou exclusivo) e `!=` (diferente) para variáveis booleanas?
7. A nota final de um aluno é composta por: duas provas com peso 3 (cada uma), um trabalho com peso 2 e duas listas de exercícios com peso 1 (cada uma). Escreva um programa para calcular a média final do aluno. Faça a entrada de dados utilizando a classe `Scanner` do Java. Adicionalmente, o programa deve apresentar na janela `Output`, as notas parciais do aluno em linhas diferentes e na última linha a seguinte mensagem "A média do aluno é Y" (onde Y é o resultado). Imprima o valor da média utilizando saída formatada.
8. Escreva um programa que leia dois números inteiros usando a classe `Scanner`, sendo os mesmos respectivamente base e altura, e calcule a área de um retângulo. No resultado, o programa deve apresentar

o título "Calculo da Área de um Retângulo", e na linha de baixo apresentar o cálculo realizado e o resultado do valor calculado, utilizando saída formatada.