

# Aula 14: Variáveis primitivas e controle de fluxo

Professor(a): Virgínia Fernandes Mota

<http://www.dcc.ufmg.br/~virginiaferm>

ALGORITMOS E ESTRUTURAS DE DADOS - SETOR DE INFORMÁTICA



# Variáveis primitivas e controle de fluxo

- Nesta aula veremos os seguintes recursos da linguagem Java:
  - declaração, atribuição de valores, casting e comparação de variáveis;
  - controle de fluxo através de if e else;
  - instruções de laço for e while, controle de fluxo com break e continue.

# Declarando e usando variáveis

- Dentro de um bloco.

```
1 // declara a idade
2 int idade;
3 idade = 15;
4
5 // imprime a idade
6 System.out.println(idade);
7
8 // calcula a idade no ano seguinte
9 int idadeNoAnoQueVem;
10 idadeNoAnoQueVem = idade + 1;
```

# Declarando e usando variáveis

- Operações básicas:

```
1  int quatro = 2 + 2;  
2  int tres = 5 - 2;  
3  
4  int oito = 4 * 2;  
5  int dezesseis = 64 / 4;  
6  
7  int um = 5 % 2;
```

# Declarando e usando variáveis

- Exemplo:

```
1  class TesteIdade {
2
3      public static void main(String[] args) {
4          // imprime a idade
5          int idade = 20;
6          System.out.println(idade);
7
8          // gera uma idade no ano seguinte
9          int idadeNoAnoQueVem;
10         idadeNoAnoQueVem = idade + 1;
11
12         // imprime a idade
13         System.out.println(idadeNoAnoQueVem);
14     }
15 }
```

# Declarando e usando variáveis

- Outros tipos de dados:

```
1 double pi = 3.14;  
2 double x = 5 * 10;  
3  
4 int idade = 30;  
5 boolean menorDeIdade = idade < 18;  
6  
7 char letra = 'a'; //Em Java variáveis do tipo char são  
    pouco usadas no dia a dia. Veremos mais a frente o  
    uso das Strings,
```

# 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.
- Tipos primitivos: int, double, boolean, char, byte, short, long e float.

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

- Alguns valores são incompatíveis se você tentar fazer uma atribuição direta.

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

- ou ainda:

```
1 int i = 3.14; //não  
2  
3 double d = 5; // ok, o double pode conter um número  
   inteiro  
4 int i = d; // não compila
```

- E o contrário?

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



- 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**.

```
1 double d3 = 3.14;  
2 int i = (int) d3;
```

- E esse caso?

```
1 float x = 0.0; //Não compila!
```

- Todos os literais com ponto flutuante são considerados double pelo Java. Como resolver?

```
1 float x = 0.0f;
```

- O único tipo primitivo que não pode ser atribuído a nenhum outro tipo é o boolean.

## Exercícios: Variáveis e tipos primitivos

- Na empresa onde trabalhamos, há tabelas com o quanto foi gasto em cada mês. Para fechar o balanço do primeiro trimestre, precisamos somar o gasto total. Sabendo que, em Janeiro, foram gastos 15000 reais, em Fevereiro, 23000 reais e em Março, 17000 reais, faça um programa que calcule e imprima o gasto total no trimestre. Siga esses passos:
  - Crie uma classe chamada BalancoTrimestral com um bloco main, como nos exemplos anteriores;
  - Dentro do main, declare uma variável inteira chamada gastosJaneiro e inicialize-a com 15000;
  - Crie também as variáveis gastosFevereiro e gastosMarco, inicializando-as com 23000 e 17000, respectivamente, utilize uma linha para cada declaração;
  - Crie uma variável chamada gastosTrimestre e inicialize-a com a soma das outras 3 variáveis:  
`int gastosTrimestre = gastosJaneiro + gastosFevereiro + gastosMarco;`
  - Imprima a variável gastosTrimestre.

- Adicione código (sem alterar as linhas que já existem) na classe anterior para imprimir a média mensal de gasto, criando uma variável `mediaMensal` junto com uma mensagem. Para isso, concatene a String com o valor, usando "Valor da média mensal = "+ `mediaMensal`.

- A sintaxe do if no Java:

```
1  if (condicaoBooleana) {  
2      codigo;  
3  }
```

- Uma condição booleana é qualquer expressão que retorne true ou false.
- Para isso, você pode usar os operadores <, >, <=, >= e outros.

```
1  int idade = 15;  
2  if (idade < 18) {  
3      System.out.println("Não pode entrar");  
4  }
```

- A cláusula else:

```
1  int idade = 15;
2  if (idade < 18) {
3      System.out.println("Não pode entrar");
4  } else {
5      System.out.println("Pode entrar");
6  }
```

- Você pode concatenar expressões booleanas através dos operadores lógicos "E" e "OU". O "E" é representado pelo && e o "OU" é representado pelo ||.

```
1  int idade = 15;
2  boolean amigoDoDono = true;
3  if (idade < 18 && amigoDoDono == false) {
4      System.out.println("Não pode entrar");
5  }
6  else {
7      System.out.println("Pode entrar");
8  }
```

- O operador de negação !

```
1  int idade = 15;
2  boolean amigoDoDono = true;
3  if (idade < 18 && !amigoDoDono) {
4      System.out.println("Não pode entrar");
5  }
6  else {
7      System.out.println("Pode entrar");
8  }
```



- Para comparar se uma variável tem o mesmo valor que outra variável ou valor, utilizamos o operador ==

```
1  int mes = 1;
2  if (mes == 1) {
3      System.out.println("Você deveria estar de férias");
4  }
```

- Teste: Utilize o operador = dentro de um if.

# O while

- O while é um comando usado para fazer um laço (loop), isto é, repetir um trecho de código algumas vezes.

```
1  int idade = 15;
2  while (idade < 18) {
3      System.out.println(idade);
4      idade = idade + 1;
5  }
```

- Outro comando de loop extremamente utilizado é o for.

```
1  for (int i = 0; i < 10; i = i + 1) {  
2      System.out.println("olá!");  
3  }
```

# ○ Controlando os loops

- Parar um loop:

```
1  for (int i = x; i < y; i++) {  
2      if (i % 19 == 0) {  
3          System.out.println("Achei um número divisível por  
4              19 entre x e y");  
5          break;  
6      }  
}
```

- Pular uma iteração:

```
1  for (int i = 0; i < 100; i++) {  
2      if (i > 50 && i < 60) {  
3          continue;  
4      }  
5      System.out.println(i);  
6  }
```

- O escopo da variável é o nome dado ao trecho de código em que aquela variável existe e onde é possível acessá-la.

```
1 // aqui a variável i não existe
2 int i = 5;
3 // a partir daqui ela existe
4 while (condicao) {
5     // o i ainda vale aqui
6     int j = 7;
7     // o j passa a existir
8 }
9 // aqui o j não existe mais, mas o i continua dentro do
   escopo
```

# Um bloco dentro do outro

- Um bloco também pode ser declarado dentro de outro.

```
1 while (condicao) {  
2     for (int i = 0; i < 10; i++) {  
3         // código  
4     }  
5 }
```

- E o comando switch?

Para cada exercício, crie um novo arquivo com extensão .java:

```
1 class ExercicioX {  
2     public static void main(String[] args) {  
3         // seu exercício vai aqui  
4     }  
5 }
```

- 1- Imprima todos os números de 150 a 300.
- 2- Imprima a soma de 1 até 1000.
- 3- Imprima todos os múltiplos de 3, entre 1 e 100.
- 4- Imprima os fatoriais de 1 a 10.
- 5- No código do exercício anterior, aumente a quantidade de números que terão os fatoriais impressos, até 20, 30, 40. Em um determinado momento, além desse cálculo demorar, vai começar a mostrar respostas completamente erradas. Por quê? Mude de int para long para ver alguma mudança.



- 6- Imprima os primeiros números da série de Fibonacci até passar de 100.
- 7- Escreva um programa que, dada uma variável  $x$  com algum valor inteiro, temos um novo  $x$  de acordo com a seguinte regra:  
se  $x$  é par,  $x = x / 2$   
se  $x$  é ímpar,  $x = 3 * x + 1$   
imprime  $x$   
O programa deve parar quando  $x$  tiver o valor final de 1. Por exemplo, para  $x = 13$ , a saída será: 40 -> 20 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1

- 8- Imprima a seguinte tabela, usando fors encadeados:  
1  
2 4  
3 6 9  
4 8 12 16  
 $n$   $n*2$   $n*3$  ....  $n*n$
- 9- Desafio: Faça o exercício da série de Fibonacci usando apenas duas variáveis.

Orientação a objetos com java