iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

Módulo 1: Programação em Linguagem Java

Aula 2
Variáveis, Funções, Tipos de Dados e Operadores

UPskill

Exemplo de algoritmo em Java

```java
int ano_atual = 2022;
int ano_nascimento = 1996;

int idade = ano_atual - ano_nascimento;

System.out.println(idade);
```

Tipos de Dados

- Definem as **operações** que podem ser feitas nos dados, o **significado** dos dados e a maneira como os valores desse tipo são **armazenados**.

```
int numeroInteiro = 8;
double numDecimal = 3.2;
boolean booleano = true;
char caractere = ‘c’;
String texto = "João";
```

Tipos de Dados - **Numérico**

- Valores numéricos inteiros, decimais, positivos ou negativos

```
int numeroInteiro = 8;
double numeroDecimal = 8.5;
int numeroNegativo = -2;
```

Tipos de Dados - **Texto**

- Também conhecidos por “string”, guardam uma sequência de caracteres. Devem ser delimitadas por aspas, plicas, ou acentos graves

```
String texto = "João";
String texto = 'Maria';
String texto = `José`;
```

Tipos de Dados - **Booleano**

- Pode ter dois valores: verdadeiro ou falso.

```
boolean booleano = true;
boolean booleano = false;
```

Operadores Aritméticos

- Os operadores permitem-nos indicar ao computador o cálculo que queremos que faça entre dois valores.
- Alguns dos operadores aritméticos:

+	**soma / adição**	-	**subtração**
*	**multipllicação**	/	**divisão**
%	**resto da divisão inteira** (eg: 21%7 = 0 ou 22%7 = 1)		

Operadores Aritméticos - **Exemplo**

- `2 + 3 → 5`
- `5 - 3 → 2`
- `5 * 3 → 15`
- `15 / 3 → 5`
- `10 % 3 → 1`

Operadores Relacionais

- Se, no entanto, pretendermos saber como dois valores se comparam, devemos utilizar operadores relacionais:

==	**é igual?**	!=	**é diferente?**
<	**é menor?**	>	**é maior?**
<=	**é menor ou igual?**	>=	**é maior ou igual?**

Operadores Relacionais - **Exemplo**

- `2 == 2` → **`true`**
- `2 == 3` → **`false`**
- `2 != 3` → **`true`**
- `2 != 2` → **`false`**
- `2 < 3` → **`true`**
- `2 < 2` → **`false`**
- `2 > 3` → **`false`**
- `2 > 2` → **`false`**
- `2 <= 3` → **`true`**
- `2 <= 2` → **`true`**
- `2 >= 3` → **`false`**
- `2 <= 2` → **`true`**

Operadores Lógicos

- Podemos também aplicar operações lógicas sobre dois valores.

&& **conjunção (e / and)**

|| **disjunção (ou / or)**

! **negação**

Operadores Lógicos - **Exemplo**

`true && true` → **true**

`true && false` → **false**

`false && true` → **false**

`false && false` → **false**

`true || true` → **true**

`true || false` → **true**

`false || true` → **true**

`false || false` → **false**

`!true` → **false**

`!false` → **true**

`!!true` → **true**

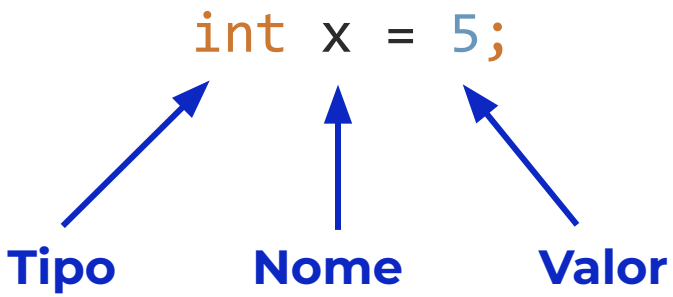
`!!false` → **false**

Operadores Lógicos - **Combinações**

- Tendo em conta os valores de a, b, c e d, qual será o resultado da operação da última linha?

```
int a = 1
int b = 2
int c = 3
int d = 4

a > b || c < d
```

Variáveis



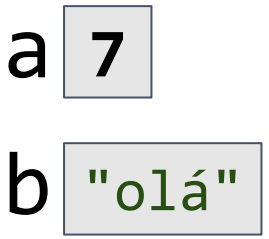
Variáveis

- Uma variável pode ser vista como um espaço em memória onde um valor de determinado tipo pode ser guardado.
- As variáveis têm três características
    - **Tipo**
    - **Nome**
    - **Valor**
- Em Java, ao definir uma variável estamos a indicar o seu **tipo**, **nome** e o seu **valor** inicial.

Variáveis em Java

- Em Java, o tipo de cada variável é **fixo**, o que nos obriga a definir o tipo de cada variavel à partida, o qual não pode ser alterado mais tarde.

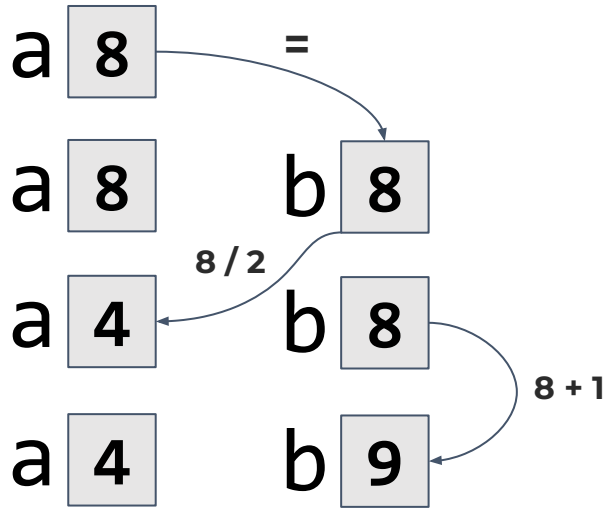
```java
int a = 7;
String b = "olá";
```



Variáveis - Atribuição

- Usamos o operador de atribuição (**=**) para **definir/alterar o valor** das variáveis;

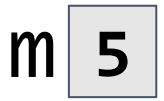
```
int a = 8;
int b = a;

a = b / 2;
b = b + 1;
```



Expressões Matemáticas

- Na programação, tal como na matemática, podemos declarar expressões que representam algoritmos e contas:

```
int m = 60 / 10 - 1;
```



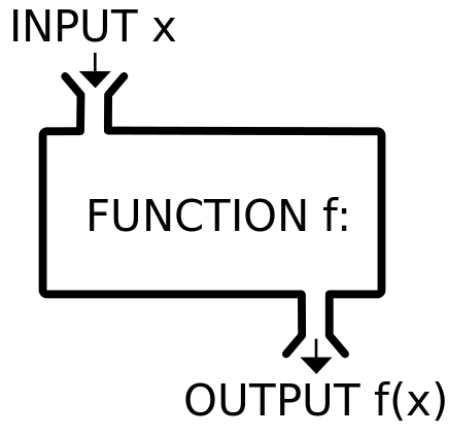
Funções

- Em Matemática, uma função **f(x)** associa a um argumento **x**, um valor **y**. Ou seja, **y = f(x)**.

  Se **f(x) = x²**
  então **f(3) = 9**

- Em Programação, o conceito de função é similar:

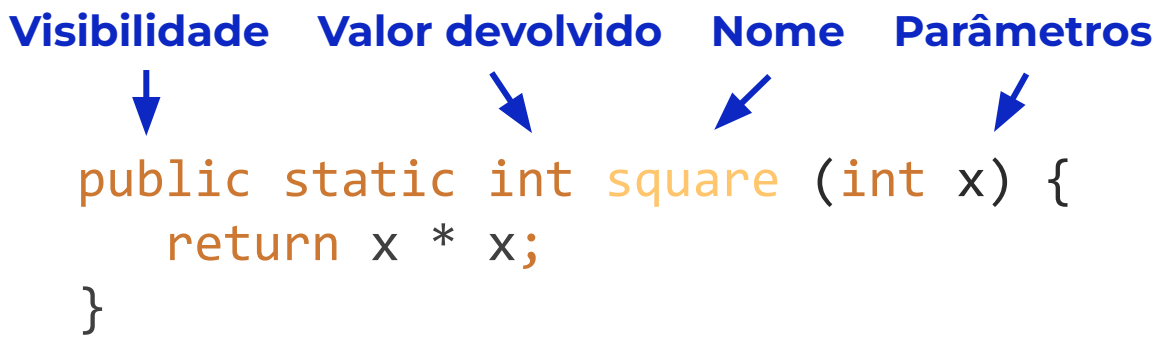
```
public static int square (int x) {
    return x * x;
}
```



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA
emprego digital

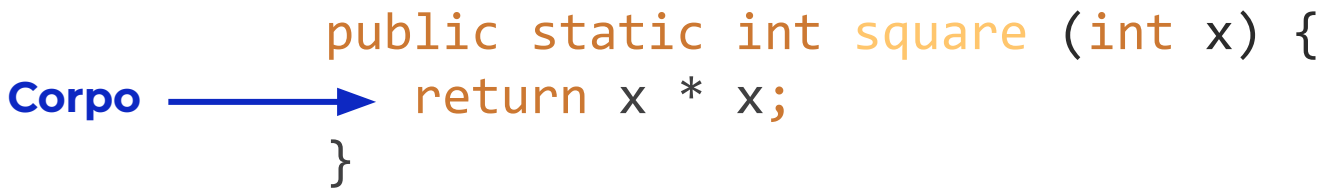
Funções - **Assinatura**

- Para que seja possível utilizar uma função, tal como na Matemática, esta tem de ter um nome. O nome da função deve indicar aquilo que a função calcula.
- Uma função tem uma assinatura, onde se define o **nome**, o **tipo do valor** devolvido (o que sai), os **parâmetros** da função (o que entra) e a **visibilidade**



Funções - **Corpo**

- O corpo de uma função é a **explicação ao computador** do método de resolução de um problema. Esta definição aparece a seguir à assinatura da função, entre chavetas.



Funções - **Vários parâmetros**

- Uma função pode ter vários parâmetros, por exemplo:

```
public static int mult (int x, int y, int z) {
   return x * y * z;
}
```

Funções - **Retorno**

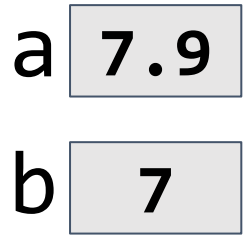
- A instrução **return** indica qual o valor devolvido pela função.
- O valor devolvido tem de ser compatível com o que é indicado na assinatura da função (neste caso, um número inteiro).
- A expressão `a + b` calcula a soma de **a** com **b** e a instrução **`return`** devolve o valor calculado.

```
public static int soma (int a, int b) {
    return a + b;
}
```

Conversões entre tipos numéricos

- Em Java, precisamos de ter atenção às conversões entre tipos numéricos.
- Por exemplo, ao converter de **double** para **int**:

```
double a = 7.9;
int b = (int) a;
```



- O valor é truncado, **conversão de tipo não é um arredondamento**!

Funções - **Exemplo**

- Esta função verifica se um dado número **`n`** é par.
- O tipo de retorno é booleano (**`boolean`**).
  - A função devolve verdadeiro (**`true`**) se **`n`** é par, e falso (**`false`**) caso contrário.
- O resultado da expressão **`n % 2`** irá ser **`0`** ou **`1`**. Caso seja igual a zero (**`== 0`**), então a expressão **`n % 2 == 0`** é verdadeira, caso contrário é falsa. Esta avaliação determinará o valor a devolver.

```
public static boolean isEven (int n) {
    return n % 2 == 0;
}
```

Funções - **Invocação**

- As funções na programação podem ser utilizadas numa determinada instrução pelo computador através de uma **invocação**.
- Uma invocação é composta pelo **nome** da função que se pretende invocar, seguida dos **argumentos** a passar nessa função:

```
int n = min(2,8);
```

- Os argumentos das funções têm de ser compatíveis com a assinatura da mesma. Por exemplo, se uma função tem como parâmetros dois inteiros, os argumentos terão de ser dois inteiros.

Variáveis como Argumentos

- O valor de um argumento pode ser dado usando uma variável
- O argumento será o **valor guardado na variável** no momento em que a função é invocada.

```
int a = 2;
int n = min(a,8);
```

- O valor de um argumento também pode ser dado usando uma expressão
- O argumento será o **valor da expressão** no momento em que a função é invocada.

```
int n = min(5-2,8);
```

Funções como Argumentos

- O valor de um argumento pode ser dado usando o valor devolvido por uma função.

```
int a = 7;
int m = min(max(a,9),8);
```

- As **invocações usadas como argumento são executadas primeiro**, de modo a que o valor devolvido possa ser utilizado como argumento.

Incrementar e Decrementar

- Algumas expressões podem alterar imediatamente o valor da variável em que são implementadas, como é o caso dos operadores de incrementar e decrementar:

```
int n = 6;
n++;

int p = 9;
p--;
```

ou

```
int n = 6;
n = n + 1;

int p = 9;
p = p - 1;
```

n 7

p 8

Funções Úteis

- Java, tal como quase todas as outras linguagens de programação, inclui por defeito diversos pacotes de funções de utilidade.
- Estes contêm instruções para se efetuarem determinadas contas ou retornarem valores que podem ser utilizados versatilmente.

```
int n = (int) Math.random() * 10;
```

- Por exemplo, cada vez que é corrido este código, a função invocada vai retornar um valor diferente entre `0` e `1` que de seguida é multiplicado por `10` e convertido em inteiro. A variável `n` vai conter um valor diferente todas as vezes.
- O pacote `Math` inclui muitas outras funções úteis!

Exercício 1

Crie uma função que tenha 3 argumentos e retorne a soma dos primeiros dois, multiplicando pelo terceiro argumento.

```
public static int sumMult (int a, int b, int c) {
    //...
}
```

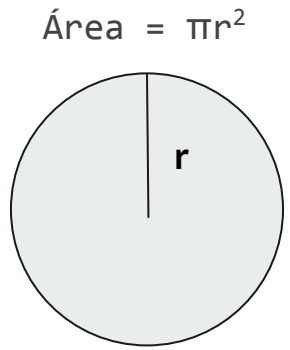
E se quisermos multiplicar os primeiros dois e depois somar o terceiro?

Exercício 2

Crie uma função que retorne a área de um círculo com raio **r**

- Dica: a expressão **`Math.PI`** devolve o valor de pi

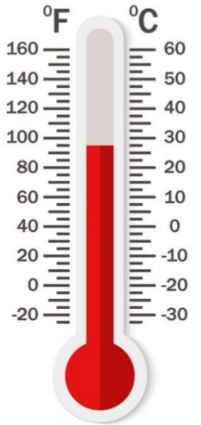
```
public static double circleArea(int r) {
   //..
}
```



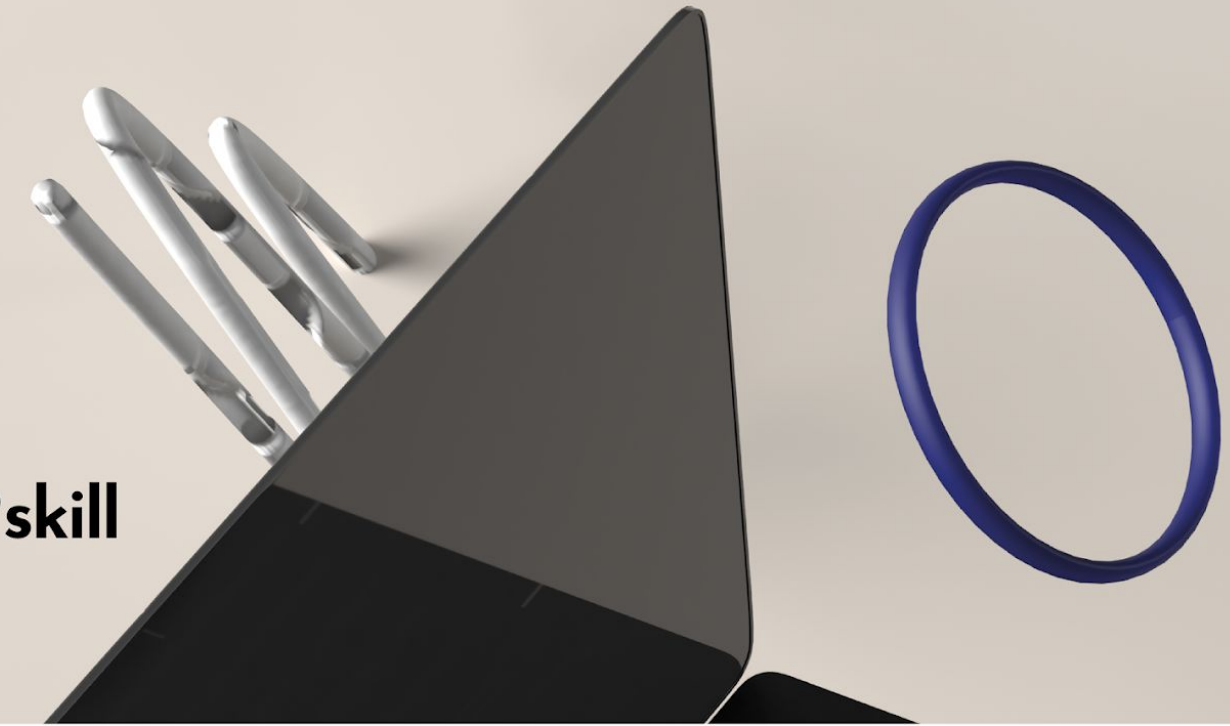
Exercício 3

Crie uma função que permita **converter uma temperatura** em graus Celsius para Fahrenheit

- Dica: quando em dúvida, o Google é vosso amigo ☺️



O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill