

02 - Modularização

BCM0505-15 - Processamento da Informação - Turma A9
(Teoria Gordana)

Emilio Francesquini
e.francesquini@ufabc.edu.br

2020.Q1

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC



- Estes slides foram preparados para o curso de **Processamento da Informação** na UFABC.
- Este material pode ser usado livremente desde que sejam mantidos, além deste aviso, os créditos aos autores e instituições.



- Já usamos algumas funções sem nos darmos conta:
 - ▶ `System.out.println`
 - ▶ `Math.random`, `Math.sqrt`
 - ▶ `scanner.nextInt()`, `nextFloat()`
 - ▶ ...

- Há algumas rotinas de código que podem ser utilizadas repetidamente e em diversos contextos diferentes. Frequentemente é conveniente as **encapsular** em funções.
- Funções/Métodos servem para reunir o código responsável por desempenhar um papel e o deixar disponível através de um nome coerente (ex. seno, cosseno, raiz quadrada, ...).
- Isso nos foi conveniente até agora para nos importarmos apenas com **o que** elas fazem e não **como** elas fazem (pense em como implementaria `Math.sqrt`, por exemplo).

- Funções, assim como em matemática, podem receber **parâmetros** (ocasionalmente também chamados de **argumentos**)
- Considere a função $f(a, b) = a^b$. Ela recebe dois parâmetros, **a** e **b** e **devolve** o valor de **a** elevado a **b**.
- A função equivalente já existe em Java e é chamada **Math.pow**.

```
1 double a = scanner.nextDouble();  
2 double b = scanner.nextDouble();  
3 double pot = Math.pow(a, b);
```

- Considere a fórmula de Heron para o cálculo da área do triângulo dados os seus três lados, a, b e c.

```
1 public static double area(double a, double b, double c){
2     double s = (a + b + c) / 2;
3     double area = Math.sqrt(s * (s-a) * (s-b) * (s-c));
4     return area;
5 }
```

- A primeira linha é chamada de **assinatura** da função. Ela define o tipo da função (**double**), e os parâmetros (seus tipos e seus nomes).
- **return** devolve o valor calculado pela função. O tipo do valor devolvido **deve** ser igual ao tipo da função

- Funções podem chamar (e frequentemente o fazem) outras funções.
- Se você está copiando e colando o mesmo código frequentemente e com apenas pequenas adaptações, considere encapsulá-lo em uma função.
- Ao contrário de funções matemáticas, métodos em Java podem não devolver valor algum. Neste caso esses métodos executam algum procedimento que é útil para o utilizador. Ex.: `println`.

- O tipo `void` é um tipo especial.
- Ele representa "nada", ou seja, uma variável deste tipo não faz sentido. Contudo é útil para indicar o tipo de uma função que não tem devolve valor algum. Por exemplo, a função abaixo:

```
1 public static void imprime (int numero) {  
2     System.out.printf("Número %d\n", numero);  
3 }
```

- A função `main` é do tipo `void`

Escreva uma função que calcule a área de um círculo de raio fornecido r .

Escreva uma função que calcule o perímetro de um círculo de raio fornecido **r**.

Escreva uma função que calcule a área da superfície de um cilindro com o raio da base r e altura h .

- Um outro tipo de variável presente na linguagem de programação Java é o tipo **boolean**.
- Ele serve para guardar os valores *verdadeiro* ou *falso*.

```
1 boolean maior = x > y;  
2 System.out.printf("%d é maior que %d? %b\n", x, y,  
  ↪ maior);
```

- Escreva uma função que receba r , o raio de uma circunferência com centro em $(0,0)$ e uma coordenada (x_0, y_0) e imprima **true** caso a coordenada esteja na parte interna da circunferência e **false** caso contrário.

- 1005
- 1007
- 1012
- 1015
- 1016
- 1017
- 1018