

# Métodos

Algoritmos e Programação III



# Métodos

Métodos são trechos de código que permitem modularizar um sistema, isto é, são pequenos blocos que, juntos, compõem um sistema maior.



## Métodos

Os principais motivos que levam a utilizar métodos se referem à redução do tamanho total de código de um sistema, à melhoria da modularização do sistema (cada trecho de código realiza uma tarefa) e à facilitação e agilização do processo de manutenção.

# Métodos

Um método pode invocar outro método, isto é, durante a execução do método1 pode ser necessária a execução do método2 que pode invocar o método3 e assim por diante. Todo método possui uma declaração e um corpo.

# Métodos

```
qualificador tipo-do-retorno nome-do-método ([lista-de-parâmetros]){  
  códigos do corpo  
}
```



# Métodos

**qualificador** : Conhecido também pelo nome de modificador, define a visibilidade do método. A Oracle o define como “nível de acesso” (accessLevel) do método. Trata-se de uma forma de especificar se o método é visível apenas à própria classe em que está declarado ou pode ser visualizado (e utilizado) por classes externas.





# Métodos

**qualificador :**

**public:** o método é visível por qualquer classe. É o qualificador mais aberto no sentido de que qualquer classe pode usar esse método.

**private:** o método é visível apenas pela própria classe. É o qualificador mais restritivo.

**protected:** o método é visível pela própria classe, por suas subclasses e pelas classes do mesmo pacote.



# Métodos

tipo-do-retorno:

refere-se ao tipo de dado retornado pelo método. Métodos que não retornam valores devem possuir nesse parâmetro a palavra **void**. Sempre que void for usada em uma declaração de método, nenhum valor é retornado após sua execução, isto é, o método atua como uma procedure de outras linguagens de programação. Um método pode ter como retorno qualquer **tipo primitivo** (int, float, double etc.), um array ou ainda um objeto qualquer.

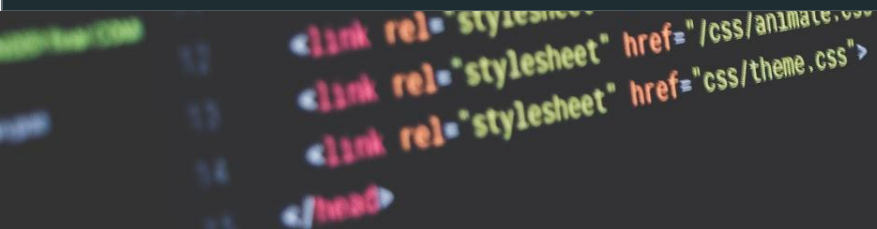




# Métodos

nome-do-método:

pode ser qualquer palavra ou frase, desde que iniciada por uma letra. Se o nome for uma frase, não podem existir espaços em branco entre as palavras. Como padrão da linguagem Java, o nome de um método sempre inicia com uma palavra com letras minúsculas. Se outras palavras forem necessárias, elas devem iniciar com maiúsculas.



# Métodos

nome-do-método:

São exemplos de nomes de métodos de acordo com o padrão da linguagem: imprimir, imprimirFrase, gravarArquivoTexto. É importante dar aos métodos nomes sugestivos, ou seja, que identificam facilmente a tarefa executada por eles.



# Métodos

`([lista-de-parâmetros])`

trata-se de uma lista de variáveis opcionais, que podem ser recebidas pelo método para tratamento interno. Quando um método é invocado (chamado), ele pode receber valores de quem o chamou.



# Métodos

`([lista-de-parâmetros])`

Esses valores podem ser manipulados internamente e devolvidos ao emissor da solicitação. Esse processo pode ser comparado ao de uma fabricação industrial: entra matéria-prima (os valores passados ao método) e sai um produto acabado (o retorno do método)



# Métodos


códigos do corpo

trata-se dos códigos implementados em Java que realizam os processos internos e retornam os valores desejados, isto é, constituem o programa do método.



# Métodos

```
qualificador tipo-do-retorno nome-do-método ([lista-de-parâmetros]){  
  códigos do corpo  
}
```



O nome do método e a lista de parâmetros formam a assinatura do método, algo que o identifica de maneira única.



```
<link rel="stylesheet" href="/css/animate.css">  
<link rel="stylesheet" href="css/theme.css">  
</head>
```



Um método pode receber  
vários parâmetros do  
mesmo tipo ou não





A chamada de um método deve corresponder exatamente à sua declaração, ou melhor, à sua assinatura. Quando invocado, o método deve possuir o mesmo nome e o mesmo número de parâmetros.

O resultado dos métodos apresentados não é atribuído a nenhuma variável, pois eles não têm retorno. Os métodos foram declarados como públicos (public), como já discutido, o que possibilita que eles sejam utilizados externamente à classe em que foram declarados

# #3



A ordem em que os métodos da classe são declarados não influencia em nada a sequência de execução.

# #4

# ##5

O uso de static é mais explorado na próxima aula; por enquanto permite acessar os métodos a partir de outras classes



# #6

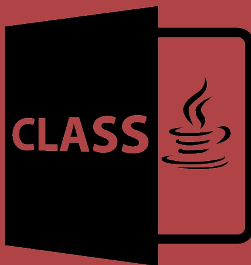
# #6

# Métodos

Sem retorno



# Métodos



ProgramaSoma.java

A=10;  
B=12;



```
metodoSoma(int y; int x){  
    System.out.println(y+x)  
}
```

22

mensagem

Funcionamento de um método sem retorno





```
qualificador void nome-do-método ([lista-de-parâmetros]){  
  
    //códigos do corpo  
  
}
```



```
public static void mostrarBomDia(){  
    System.out.println("Bom Dia");  
}
```

```
public static void main(String[] args)  
    mostrarBomDia();  
}
```



```
public static void soma(int a, int b){  
    System.out.println(a+b);  
}
```

```
public static void main(String[] args) {  
  
    int x=10;  
    int y=12;  
  
    soma(x,y);  
}
```



# Métodos

Com retorno

# Métodos



A=10;  
B=12;

22



```
metodoSoma(int y; int x){  
    return y+x;  
}
```

mensagem

Funcionamento de um método com retorno




retorno





```
qualificador tipo-de-retorno nome-do-método ([lista-de-parâmetros]){  
  
    //códigos do corpo  
  
    return(tipo-do-retorno)  
  
}
```



```
14
15 ►  public static void main(String[] args) {
16     mostrarBomDia();
17     int m= multiplicar(5,10);
18     System.out.println(m);/// ouuuu
19     System.out.println(multiplicar(3,4));
20 }
21 @  public static int multiplicar(int a, int b){
22     return a*b;
23 }
24 
```





# SOBRECARGA

A linguagem Java permite que vários métodos sejam definidos com o mesmo nome, desde que eles tenham uma assinatura diferente, ou seja, essas diferenças podem ser com base no número, nos tipos ou na ordem de parâmetros recebidos. Quando um método sobrecarregado é chamado, o compilador avalia e seleciona o método mais adequado à situação, examinando a assinatura correspondente, portanto os métodos sobrecarregados são utilizados para a realização de tarefas semelhantes sobre tipos de dados diferentes.

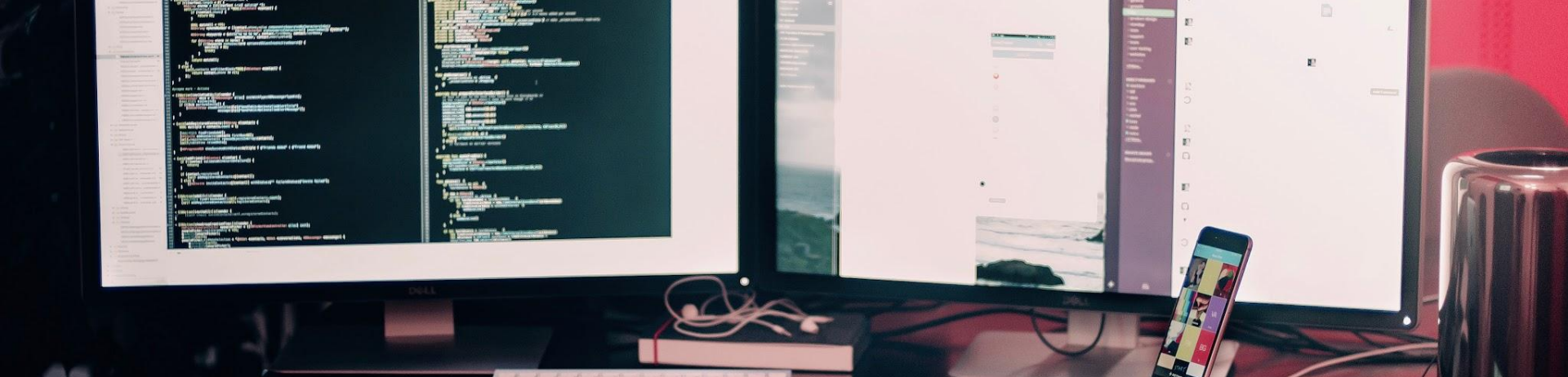
# SOBRECARGA

```
19 System.out.println(multiplicar(3,4));
20 System.out.println(multiplicar(3,4,5));
21 System.out.println(multiplicar(3,4.2));
22 }
23 @ public static int multiplicar(int a, int b){
24     return a*b;
25 }
26
27 @ public static int multiplicar(int a, int b,int c){
28     return a*b*c;
29 }
30 @ public static double multiplicar(int a, double b){
31     return a*b;
```

A photograph of a desk setup with two computer monitors, a smartphone on a stand, and headphones. The image is overlaid with a semi-transparent red filter. A white rectangular box is centered over the text.

# Acesso de outra Classe

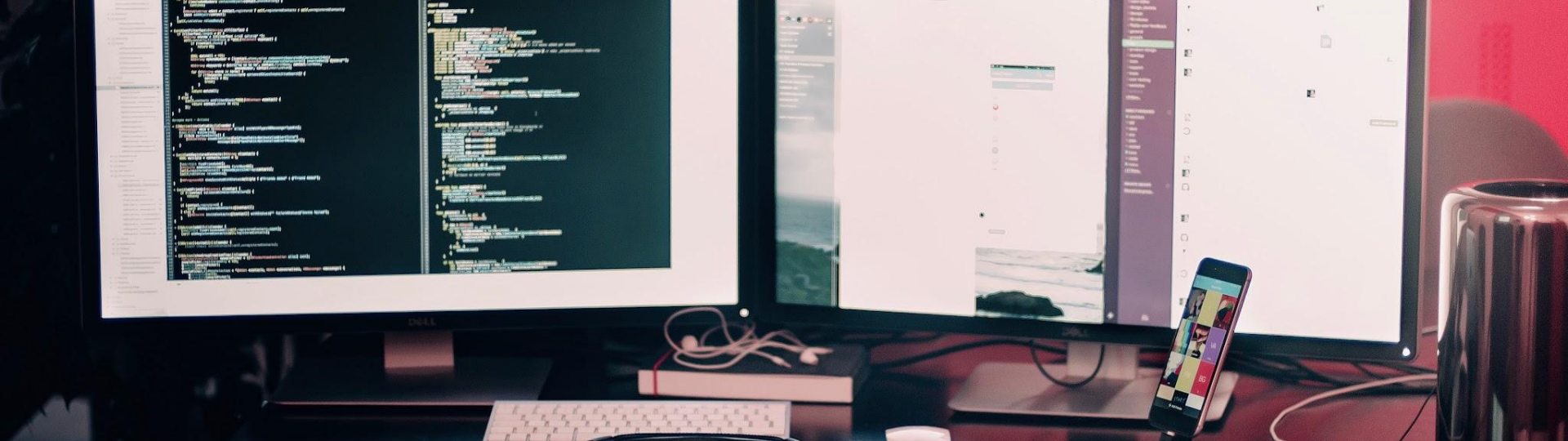




Qualquer método do tipo static criado em uma classe pode ser chamado a partir de outra classe usando o formato

**nome-da-classe.nome-do-método.**

Com isso um método necessita ser criado apenas uma vez em uma única classe. Esse recurso é muito importante, pois, uma vez que o método foi criado, ele pode ser reutilizado em outra aplicação; basta chamar a classe em que ele se encontra.

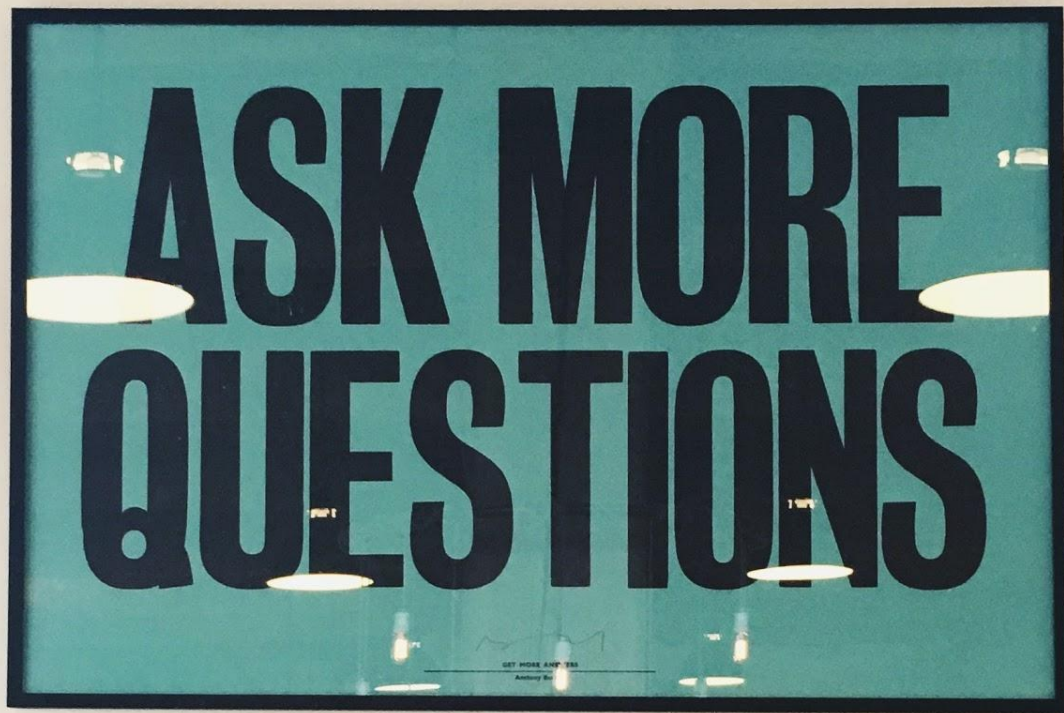


```
▶ public class ExemploDeChamada {  
▶ public static void main(String[] args) {  
    System.out.println(Metodos.multiplicar(3, 5));  
}  
}
```

Typo: In word 'Chamada' [more...](#) (⌘F1)



<http://gg.gg/4h2kq>





Furgeri, Sérgio  
Java 8 - ensino didático : desenvolvimento e implementação de aplicações / Sérgio Furgeri. -- São Paulo : Érica, 2015. 320 p.

Schildt, Herbert.  
Programação com Java [recurso eletrônico] : uma introdução abrangente / Herbert Schildt, Dale Skrien ; tradução: Aldir José Coelho Corrêa da Silva ; revisão técnica: Maria Lúcia Blanck Lisbôa. – Dados eletrônicos. – Porto Alegre : AMGH, 2013.

Exemplos  
<http://gg.gg/4go85>

Playlist dos Vídeos  
<http://gg.gg/4h2kq>

