


# Linguagem de Programação

Linguagem Java  
Considerações sobre o Uso de Métodos

# Subprogramas

- Um subprograma caracteriza-se por ser um trecho independente de programa com atribuições bem definidas.
  - Também chamados de sub-rotinas
  - Dependendo da linguagem de programação que está sendo usada, também são conhecidos como:
    - Funções (em Linguagem C)
    - Métodos (em Java)
- 

# Métodos

- Em Java são chamados de métodos
  - Obs: nessa disciplina não entraremos em detalhes como sobrecarga e sobreposição de métodos, nem em conceitos relacionados a orientação a objetos
- São blocos de instruções que realizam tarefas específicas

# Métodos

- Evitam a repetição de um conjunto de instruções que ocorre várias vezes no programa
- Modularizam o programa
  - O problema pode ser dividido em tarefas menores (dividir para conquistar)
- Reaproveitamento de código

# Métodos em Java

## ➤ Forma geral

```
[acesso] [static] tipo nome-do-método([lista-de-argumentos]) {  
    corpo do método  
}
```

- acesso: **public**, **private** ou **protected**
- static: o método existe independentemente da criação de um objeto (instância) da classe
- tipo: tipo do valor de retorno do método (void para os que não devolvem um resultado)
- lista-de-argumentos: variáveis que recebem os valores transmitidos para o método

# Métodos em Java

Assinatura do método

## ➤ Forma geral

```
[acesso] [static] tipo nome-do-método([lista-de-argumentos]) {  
    corpo do método  
}
```

- acesso: **public**, **private** ou **protected**
- static: o método existe independentemente da criação de um objeto (instância) da classe
- tipo: tipo do valor de retorno do método (void para os que não devolvem um resultado)
- lista-de-argumentos: variáveis que recebem os valores transmitidos para o método

# Métodos em Java

- Boas práticas
  - Nomes de métodos devem começar com uma letra minúscula
  - Usar letras maiúsculas apenas para facilitar o entendimento (nome compostos)
  - Nomes de métodos devem ser verbos
  - Exemplos:
    - calcularMedia()
    - ligarVeiculo()
    - pagarContaBancaria()

# Retorno de um método

## a) O método não precisa retornar nenhum valor, ou seja, sem retorno de valor

Quando um processamento é executado somente dentro do método chamado, ficando o resultado obtido “preso” dentro do método, caracterizamos esse métodos como sendo do tipo que não retorna valor.

Para esse caso é necessário apenas usar o comando de retorno dentro do método e especificar que o método é do tipo **void**, ou seja, não retorna nenhum valor.

Ao final do método usar:

```
return;           // não retorna valor
```

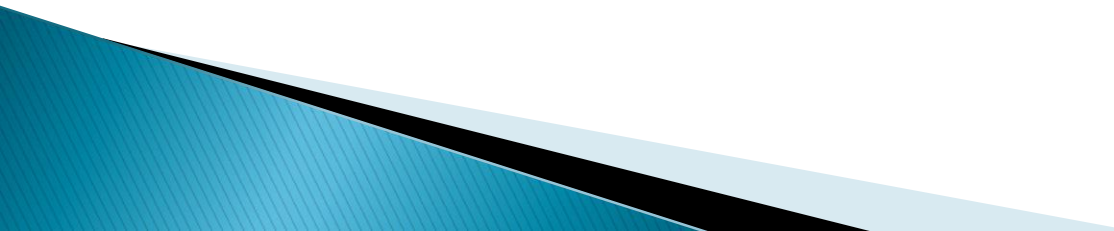


# Retorno de valor de um método

## b) Com Retorno de Valor

- ▶ Pode-se escrever métodos com retorno de valor que é efetuado pelo comando **return** e é devolvido para o ponto que chamou a função.
- ▶ Sintaxe:  
//retorna o valor da expressão ou variável  
`return (<expressão ou variável>);`

# Utilização de Parâmetros

- ▶ O uso de parâmetros permite a comunicação entre rotinas de programa.
  - ▶ Muito útil quando se trabalha apenas com variáveis locais, que são mais vantajosas por ocuparem espaço por menos tempo em memória do que as variáveis globais.
  - ▶ As variáveis locais só ocupam espaço em memória quando estão sendo utilizadas.
  - ▶ É possível passar um valor para uma função.
- 

# Passagem de parâmetro por valor

- ▶ A passagem de parâmetro por valor caracteriza-se por funcionar apenas como um mecanismo de entrada de valor para uma determinada função/método.
- ▶ Um método pode receber um ou mais parâmetros

# Exemplo de método para calcular uma média

```
import java.util.Scanner;
```

```
public class Media {
```

```
    public static void main(String[] args) {  
        Scanner entra = new Scanner(System.in);  
        System.out.print("Informe as notas de 2 provas: ");  
        float p1 = entra.nextFloat(), p2 = entra.nextFloat();  
        float media = calcularMedia(p1, p2);  
        System.out.printf("Media: %.2f\n", media);  
        entra.close();  
    }
```

Chamada  
ao método

```
    public static float calcularMedia(float n1, float n2) {  
        float calc = (n1 + n2) / 2;  
        return calc;  
    }
```

```
}
```

# Exemplo de método para calcular uma média

```
import java.util.Scanner;
```

```
public class Media {
```

```
    public static void main(String[] args) {  
        Scanner entra = new Scanner(System.in);  
        System.out.print("Informe as notas de 2 provas: ");  
        float p1 = entra.nextFloat(), p2 = entra.nextFloat();  
        float media = calcularMedia(p1, p2);  
        System.out.printf("Media: %.2f\n", media);  
        entra.close();  
    }
```

```
    public static float calcularMedia(float n1, float n2) {  
        float calc = (n1 + n2) / 2;  
        return calc;  
    }
```

```
}
```

Lista de  
argumentos

# Exemplo de método para calcular uma média

```
import java.util.Scanner;
```

```
public class Media {
```

```
    public static void main(String[] args) {  
        Scanner entra = new Scanner(System.in);  
        System.out.print("Informe as notas de 2 provas: ");  
        float p1 = entra.nextFloat(), p2 = entra.nextFloat();  
        float media = calcularMedia(p1, p2);  
        System.out.printf("Media: %.2f\n", media);  
        entra.close();  
    }
```

Tipo do valor de  
retorno do método

```
    public static float calcularMedia(float n1, float n2) {  
        float calc = (n1 + n2) / 2;  
        return calc;  
    }
```

```
}
```

# Escopo de variáveis

- O escopo de uma variável são as linhas do programa onde ela pode ser referenciada
- As variáveis definidas dentro de um método têm escopo limitado a esse método, mais especificamente ao bloco de instruções onde elas foram declaradas

# Exemplo de método para calcular uma média

```
1  import java.util.Scanner;
2
3  public class Media {
4
5      public static void main(String[] args) {
6          Scanner entra = new Scanner(System.in);
7          System.out.print("Informe as notas de 2 provas: ");
8          float p1 = entra.nextFloat(), p2 = entra.nextFloat();
9          float media = calcularMedia(p1, p2);
10         System.out.printf("Media: %.2f\n", media);
11         entra.close();
12     }
13
14     public static float calcularMedia(float n1, float n2) {
15         float calc = (n1 + n2) / 2;
16         return calc;
17     }
18
19 }
```

Escopo de variáveis



# Exemplo do escopo de variáveis

```
public class Escopo {
```

```
    public static void main(String[] args) {
```

```
        int a = 10;  
        metodo1(a);  
        int b = a / 2;  
        metodo2(b);  
        System.out.printf("a = %d\n b = %d\n", a, b);
```

```
    }
```

Escopo de a e b

```
    public static void metodo1(int x) {
```

```
        x /= 2;  
        System.out.printf("x = %d\n", x);
```

```
    }
```

Escopo de x

```
    public static void metodo2(int y) {
```

```
        if (y >= 5) {  
            int z = y * 10;  
            System.out.printf("z = %d\n", z);  
        }  
        System.out.printf("y = %d\n", y);
```

```
    }
```

```
}
```

Escopo de y e z

# Exemplo do escopo de variáveis

```
public class Escopo {
```

```
    public static void main(String[] args) {
```

```
        int a = 10;
        metodo1(a);
        int b = a / 2;
        metodo2(b);
        System.out.printf("a = %d\n b = %d\n", a, b);
```

```
    }
```

Escopo de a e b

```
    public static void metodo1(int x) {
```

```
        x /= 2;
        System.out.printf("x = %d\n", x);
```

```
    }
```

Escopo de x

```
    public static void metodo2(int y) {
```

```
        if (y >= 5) {
            int z = y * 10;
            System.out.printf("z = %d\n", z);
        }
        System.out.printf("y = %d\n", y);
```

```
    }
```

```
}
```

Escopo de y e z

```
C:\Sandra\AulasJava\Metodos>java Escopo
Metodo 1
x = 5
Metodos 2
z = 50
y = 5
Funcao main
a = 10
b = 5
```

# Passagem de parâmetros por valor e por referência

- Há 2 tipos básicos de passagem de parâmetros para subprogramas
  - Por valor: uma cópia do valor da variável é transmitida ao subprograma
  - Por referência: uma referência para a variável é transmitida ao subprograma
    - Permite alterar o conteúdo da variável que foi passada como parâmetro

# Passagem de parâmetros por valor e por referência

- Em java, tipos primitivos (int, long, float, double, etc.) são passados por valor
- Tipos referenciais (objetos, tais como StringBuffer, vetores e matrizes) são passados por referência

# Exemplo da passagem de parâmetros por valor

```
public class Trocar {  
  
    public static void main(String[] args) {  
        int a = 10, b = 20;  
        System.out.printf("a = %d\tb= %d\n", a, b);  
        trocar(a, b);  
    }  
    public static void trocar(int a, int b) {  
        int aux = a;  
        a = b;  
        b = aux;  
        System.out.printf("a = %d\tb= %d\n", a, b);  
    }  
}
```

```
C:\Sandra\AulasJava\Metodos>javac Trocar.java  
  
C:\Sandra\AulasJava\Metodos>java Trocar  
a = 10  b= 20  
a = 20  b= 10
```

# Exemplo da passagem de parâmetros por referência

```
public class PassagemPorReferencia {  
  
    public static void main(String[] args) {  
        StringBuffer entrada = new StringBuffer("Fatec");  
        System.out.println(entrada);  
        mudarValor(entrada);  
        System.out.println(entrada);  
    }  
  
    public static void mudarValor(StringBuffer entrada) {  
        entrada.append(" Carapicuiiba");  
    }  
  
}
```

```
C:\Sandra\AulasJava\Metodos>javac PassagemPorReferencia.java  
  
C:\Sandra\AulasJava\Metodos>java PassagemPorReferencia  
Fatec  
Fatec Carapicuiiba
```

# Exemplo da passagem de um vetor como parâmetro (1/2)

```
1  import java.util.Random;
2  import java.util.Scanner;
3
4  public class VetorComoParametro {
5
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8
9          System.out.print("Informe o tamanho do vetor: ");
10         int n = entrada.nextInt();
11         int[] vet = new int[n];
12
13         System.out.print("Informe o valor limite para os elementos do vetor: ");
14         int limite = entrada.nextInt();
15         preencherVetor(vet, limite);
16         imprimirVetor(vet);
17         entrada.close();
18     }
```

O vetor é passado como parâmetro por referência

# Exemplo da passagem de um vetor como parâmetro (1/2)

```
19 public static void preencherVetor(int[] vet, int limite) {  
20     Random rand = new Random();  
21     for (int i = 0; i < vet.length; i++) {  
22         vet[i] = rand.nextInt(limite);  
23     }  
24 }  
  
26 public static void imprimirVetor(int[] vet) {  
27     int i;  
28     System.out.print("Conteudo do vetor: [ ");  
29     for (i = 0; i < vet.length - 1; i++) {  
30         System.out.printf("%d, ", vet[i]);  
31     }  
32     System.out.printf("%d ]\n", vet[i]);  
33 }  
34 }  
35 }
```

```
C:\Sandra\AulasJava\Metodos>javac VetorComoParametro.java
```

```
C:\Sandra\AulasJava\Metodos>java VetorComoParametro
```

```
Informe o tamanho do vetor: 5
```

```
Informe o valor limite para os elementos do vetor: 50
```

```
Conteudo do vetor: [ 9, 27, 34, 6, 44 ]
```



# Exemplo da passagem de uma matriz como parâmetro (1/2)

```
1  import java.util.Random;
2  import java.util.Scanner;
3
4  public class MatrizComoParametro {
5
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          System.out.print("Informe o numero de linha e coluna da matriz: ");
9          int linhas = entrada.nextInt(), colunas = entrada.nextInt();
10         int[][] mat = new int[linhas][colunas];
11         System.out.print(
12             "Informe o valor limite para os elementos da matriz: ");
13         int limite = entrada.nextInt();
14         preencherMatriz(mat, limite);
15         imprimirMatriz(mat);
16         entrada.close();
17     }
18 }
```

A matriz é passada como parâmetro por referência

# Exemplo da passagem de uma matriz como parâmetro (1/2)

```
18
19 public static void preencherMatriz(int[][] mat, int limite) {
20     Random rand = new Random();
21     for (int i = 0; i < mat.length; i++) {
22         for (int j = 0; j < mat[i].length; j++) {
23             mat[i][j] = rand.nextInt(limite);
24         }
25     }
26 }
27 public static void imprimirMatriz(int[][] mat) {
28     System.out.println("Conteúdo da matriz:");
29     for (int i = 0; i < mat.length; i++) {
30         for (int j = 0; j < mat[i].length; j++) {
31             System.out.printf("%d\t", mat[i][j]);
32         }
33         System.out.println();
34     }
35 }
36 }
37 }
```

```
C:\Sandra\AulasJava\Metodos>javac MatrizComoParametro.java
```

```
C:\Sandra\AulasJava\Metodos>java MatrizComoParametro
```

```
Informe o numero de linha e coluna da matriz: 2
```

```
3
```

```
Informe o valor limite para os elementos da matriz: 20
```

```
Conteúdo da matriz:
```

```
4      17      11
```

```
19     16      19
```

# Vetor e matriz como valor de retorno

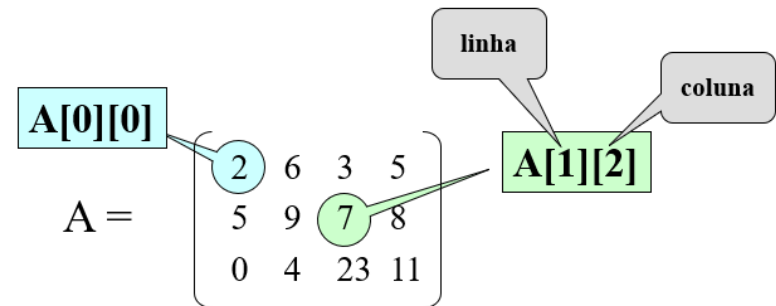
- Em Java é possível que um método devolva um vetor ou uma matriz como valor de retorno

## Vetor

NUMEROS

	34	23	54	43	5
Índice	0	1	2	3	4

## Matriz



# Vetor como valor de retorno (1/2)

```
1  import java.util.Random;
2  import java.util.Scanner;
3
4  public class VetorComoRetorno {
5
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          System.out.print("Informe o tamanho: ");
9          int n = entrada.nextInt();
10         System.out.print("Informe o valor limite para os elementos do vetor: ");
11         int limite = entrada.nextInt();
12         int[] vet = criarVetor(n, limite);
13         imprimirVetor(vet);
14         entrada.close();
15     }
16 }
```

# Vetor como valor de retorno (2/2)

```
17 public static int[] criarVetor(int n, int limite) {  
18     Random rand = new Random();  
19     int[] v = new int[n];  
20     for (int i = 0; i < v.length; i++) {  
21         v[i] = rand.nextInt(limite);  
22     }  
23     return v;  
24 }  
25 public static void imprimirVetor(int[] vet) {  
26     int i;  
27     System.out.print("Conteudo do vetor: [ ");  
28     for (i = 0; i < vet.length - 1; i++) {  
29         System.out.printf("%d, ", vet[i]);  
30     }  
31     System.out.printf("%d ]\n", vet[i]);  
32 }  
33  
34 }
```

```
C:\Sandra\AulasJava\Metodos>javac VetorComoRetorno.java
```

```
C:\Sandra\AulasJava\Metodos>java VetorComoRetorno
```

```
Informe o tamanho: 10
```

```
Informe o valor limite para os elementos do vetor: 20
```

```
Conteudo do vetor: [ 11, 10, 13, 6, 0, 14, 10, 12, 14, 3 ]
```

# Matriz como valor de retorno (1/2)

```
1  import java.util.Random;
2  import java.util.Scanner;
3
4  public class MatrizComoRetorno {
5
6      public static void main(String[] args) {
7          Scanner entrada = new Scanner(System.in);
8          System.out.print("Informe o numero de linha e coluna da matriz: ");
9          int linhas = entrada.nextInt(), colunas = entrada.nextInt();
10         System.out.print(
11             "Informe o valor limite para os elementos da matriz: ");
12         int limite = entrada.nextInt();
13         int[][] mat = criarMatriz(linhas, colunas, limite);
14         imprimirMatriz(mat);
15         entrada.close();
16     }
```

# Matriz como valor de retorno (2/2)

```
17
18 public static int[][] criarMatriz(int linhas, int colunas, int limite) {
19     Random rand = new Random();
20     int[][] m = new int[linhas][colunas];
21     for (int i = 0; i < m.length; i++) {
22         for (int j = 0; j < m[i].length; j++) {
23             m[i][j] = rand.nextInt(limite);
24         }
25     }
26     return m;
27 }
28 public static void imprimirMatriz(int[][] mat) {
29     System.out.println("Conteudo da matriz:");
30     for (int i = 0; i < mat.length; i++) {
31         for (int j = 0; j < mat[i].length; j++) {
32             System.out.printf("%d\t", mat[i][j]);
33         }
34         System.out.println();
35     }
36 }
37 }
```

```
C:\Sandra\AulasJava\Metodos>javac MatrizComoRetorno.java
```

```
C:\Sandra\AulasJava\Metodos>javac MatrizComoRetorno.java
```

```
C:\Sandra\AulasJava\Metodos>java MatrizComoRetorno
```

```
Informe o numero de linha e coluna da matriz: 3
```

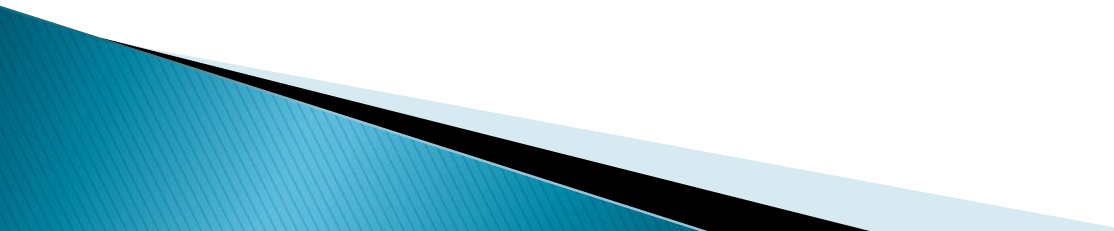
```
4
```

```
Informe o valor limite para os elementos da matriz: 20
```

```
Conteudo da matriz:
```

18	0	1	14
18	1	19	8
1	2	6	8

# Exercícios

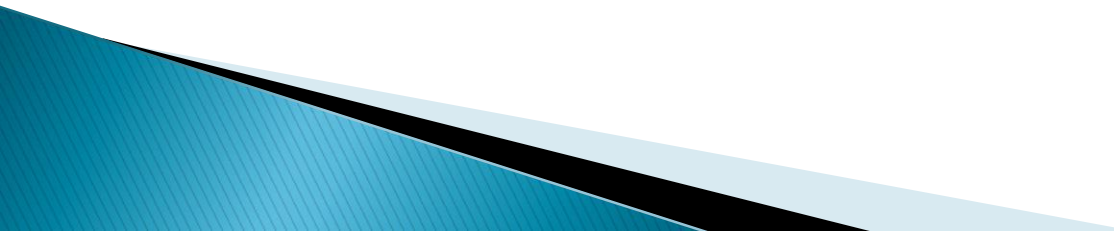
1. Crie um programa que contenha um método denominado Verifica. O método Verifica recebe um parâmetro que é um valor inteiro e verifica se esse valor é par ou ímpar. Imprimir a mensagem dentro do método.
  2. Crie um programa que lê dois números e envia esses números para um método chamado calculaSoma. O método calculaSoma, após receber esses valores, deverá realizar uma soma e imprimir o resultado obtido dentro do método.
- 



# Exercícios

3. Crie um programa que possua um método chamado calculaMedia e que recebe as 2 notas de um aluno e uma letra por parâmetro.

Se a letra recebida for a letra 'A' o método deve calcular a média aritmética das notas do aluno, porém se a letra recebida for a 'P', o método deve calcular a média ponderada com pesos 7 e 3 para a primeira nota e a segunda nota, respectivamente. A média calculada deve retornar à função principal (main), sendo o valor da média exibido na main.



# Exercícios

4. Faça um programa que possua um método chamada Tempo que recebe por parâmetro o tempo de duração de uma fábrica expressa em segundos. O método deve efetuar cálculos a fim de imprimir a equivalência do tempo recebido em horas, minutos e segundos.

exemplo:

se valor recebido pela função for 7265 segundos então a mesma deve imprimir:

7265 segundo(s) equivalem a 2 hora(s) , 1 minuto(s) e 5 segundo(s)

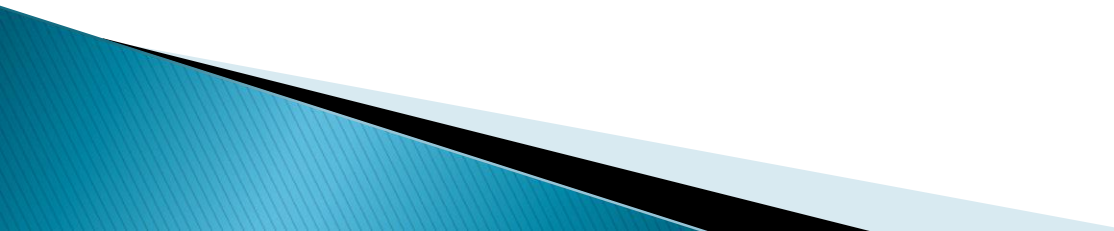
# Exercícios

5. Faça um programa que contenha um método chamado Conceito e que recebe a média final de um aluno por parâmetro e retorna o seu conceito, conforme a tabela:

Nota	Conceito
de 0,0 a 4,9	D
de 5,0 a 6,9	C
de 7,0 a 8,9	B
de 9,0 a 10,0	A

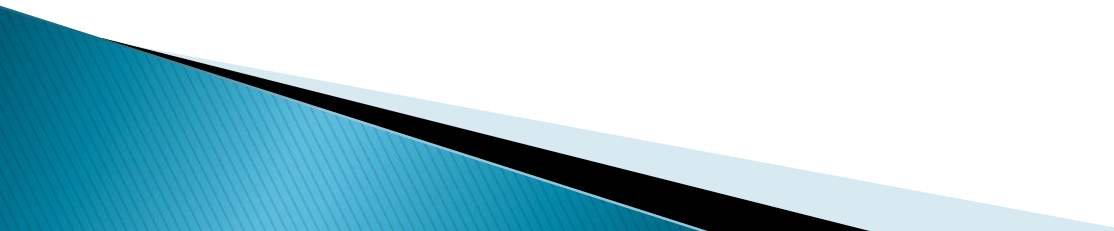
Obs: imprimir o conceito na função main.

# Exercícios

6. Faça um programa que contenha um método que verifique se um valor é perfeito ou não. Um valor é dito perfeito quando ele é igual a soma dos seus divisores. (Ex: 6 é perfeito,  $6 = 1 + 2 + 3$ , que são seus divisores).
  7. Faça um programa que contenha um método chamado LeVetor. Dentro do método permita a leitura de 10 valores inteiros, armazene-os em um vetor, e imprime o maior e o menor deles.
- 

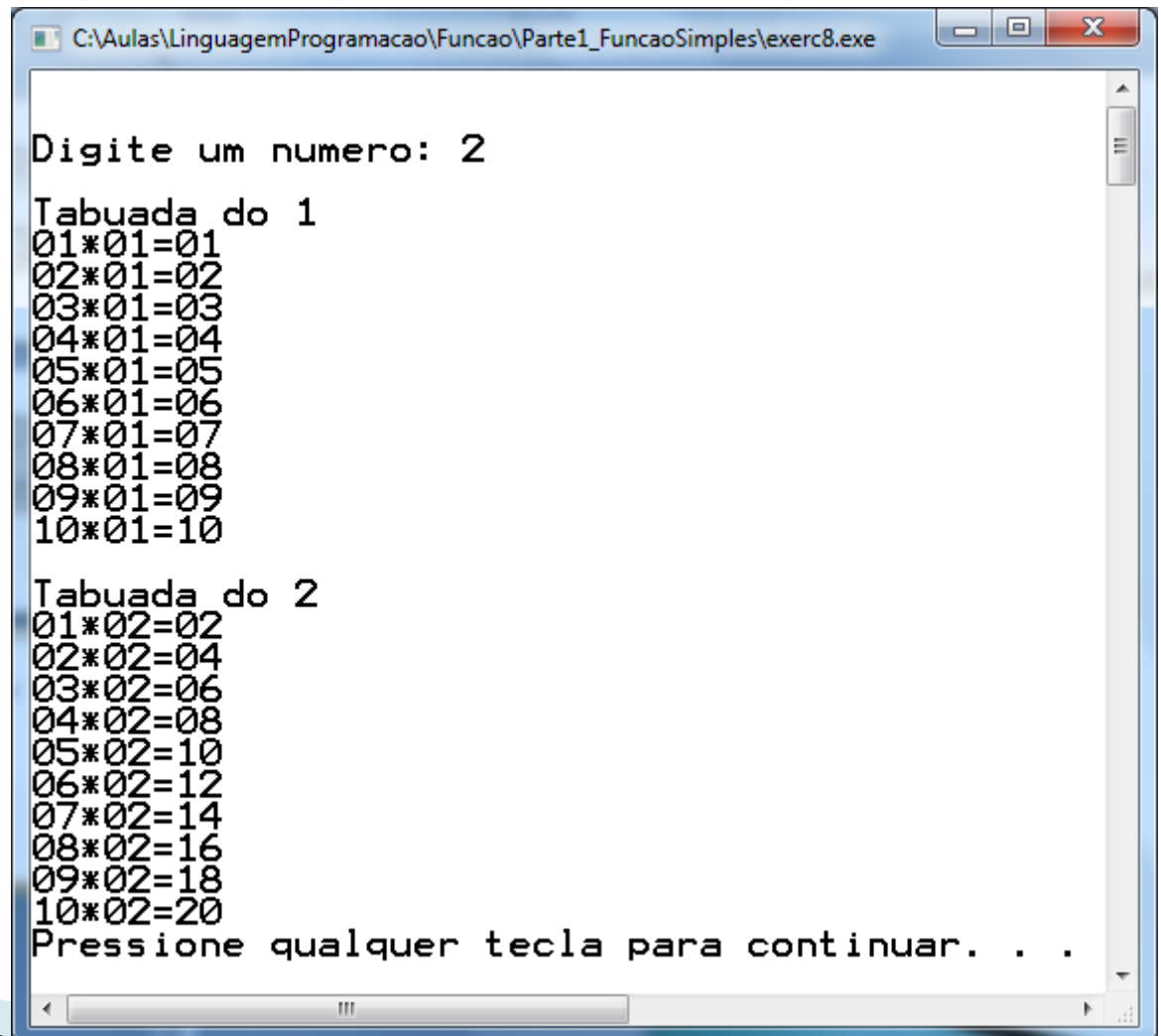
# Exercícios

8. Faça um método chamado MostraTabuadas e que recebe, por parâmetro, um valor inteiro N. Calcular e imprimir as tabuadas dos números de 1 até N, ou seja, se o usuário digitou o valor 3 deve imprimir a tabuada do 1, a tabuada do 2 e a tabuada do 3. Exibir as tabuadas, dentro do método, considerando os cálculos de 1 a 10. Dessa forma, método não deve retornar nada.



# Exercícios

## 8. Exemplo de execução para N=2



```
C:\Aulas\LinguagemProgramacao\Funcao\Parte1_FuncaoSimples\exerc8.exe

Digite um numero: 2

Tabuada do 1
01*01=01
02*01=02
03*01=03
04*01=04
05*01=05
06*01=06
07*01=07
08*01=08
09*01=09
10*01=10

Tabuada do 2
01*02=02
02*02=04
03*02=06
04*02=08
05*02=10
06*02=12
07*02=14
08*02=16
09*02=18
10*02=20

Pressione qualquer tecla para continuar. . .
```

# Referências

- DEITEL, H.M.; DEITEL, P.J.. *Java Como Programar*. 4. ed., Porto Alegre: Bookman, 2002.
- ECKEL, B.. *Thinking in Java*. 3. ed., Prentice Hall, 2002.