

IOS – Instituto de
Oportunidade Social

Java 07 - Teste de Mesa e Vetor



- > Teste de Mesa
- > Ferramentas de debug
- > Vetores

IOS – Instituto de
Oportunidade Social

Teste de Mesa



Teste de Mesa

O teste de mesa é uma análise para descobrir se um programa funciona logicamente. Basicamente, o Teste de Mesa (Trace Table) é um processo manual que é utilizado para validar a lógica de um determinado algoritmo. Esse processo era realizado com papel e caneta em cima de uma mesa e o programador ia seguindo o fluxo do algoritmo e atualizando os valores das variáveis do programa manualmente a cada instrução do algoritmo que era analisada.

IOS – Instituto de
Oportunidade Social

Vamos Praticar I, II e III



Vamos Praticar I, II e III



Apostila de Java:

Aula_07_Testes_de_mesa

Páginas 1 a 4

OBS: Acompanhar o passo a passo com o instrutor

Teste de Mesa

```
public class Teste {  
    public static void main(String[] args) {  
        int controle, acumulador = 0;  
        boolean condicao;  
        System.out.println("Numero da  
iteração\tCondição\tControle\tAcumulador");  
        for(controle = 1; controle <= 10; controle++){  
            acumulador += controle;  
            condicao = controle <= 10;  
            System.out.println("Iteração "+ controle+ "\t\t\t"+ condicao +  
"\t\t\t\t" + controle + "\t\t\t\t" + acumulador);  
        }  
    } // fim do método main  
}
```

Teste de Mesa

Run: Teste x

[C:\Users\Bigode\.jdk\openjdk-16.0.2\bin\java.exe -java:](#)

Numero da iteração	Condição	Controle	Acumulador
Iteração 1	true	1	1
Iteração 2	true	2	3
Iteração 3	true	3	6
Iteração 4	true	4	10
Iteração 5	true	5	15
Iteração 6	true	6	21
Iteração 7	true	7	28
Iteração 8	true	8	36
Iteração 9	true	9	45
Iteração 10	true	10	55

Vamos praticar II

```
public class Teste2 {  
    public static void main(String[] args) {  
        int controle = 10;  
        long acumulador = 1;  
        boolean condicao;  
        System.out.println("Numero da  
        iteração\tCondição\tControle\tAcumulador");  
        while(controle > 0){  
            int iteracao = 10 - controle + 1;  
            acumulador *= controle;  
            condicao = controle > 0;  
            System.out.println("Iteração "+ iteracao+ "\t\t\t"+ condicao +  
            "\t\t\t" + controle + "\t\t\t" + acumulador);  
            controle--;  
        }  
    } // fim do método main
```

Vamos Praticar I, II e III

Vamos praticar II

```
Run: Teste2 x
C:\Users\Bigode\.jdk\openjdk-16.0.2\bin\java.exe -javaager
```

Numero da iteração	Condição	Controle	Acumulador
Iteração 1	true	10	10
Iteração 2	true	9	90
Iteração 3	true	8	720
Iteração 4	true	7	5040
Iteração 5	true	6	30240
Iteração 6	true	5	151200
Iteração 7	true	4	604800
Iteração 8	true	3	1814400
Iteração 9	true	2	3628800
Iteração 10	true	1	3628800

```
Process finished with exit code 0
```

Vamos Praticar I, II e III

Vamos praticar III

```
public class Teste3 {  
    public static void main(String[] args) {  
        int controle = 1;  
        long acumulador = 100;  
        boolean condicao;  
        System.out.println("Numero da iteração\tCondição\tControle\tAcumulador");  
        do{  
            acumulador -= controle;  
            condicao = controle <=10;  
            System.out.println("Iteração "+ controle+ "\t\t\t"+ condicao + "\t\t\t" +  
controle + "\t\t\t" + acumulador);  
            controle++;  
        }while(controle <=10);  
    } // fim do método main  
}
```

Vamos Praticar I, II e III

Vamos praticar III

```
Run: Teste3 x
C:\Users\Bigode\.jdk\openjdk-16.0.2\bin\java.exe -java
Numero da iteração  Condição  Controle  Acumulador
Iteração 1          true      1         99
Iteração 2          true      2         97
Iteração 3          true      3         94
Iteração 4          true      4         90
Iteração 5          true      5         85
Iteração 6          true      6         79
Iteração 7          true      7         72
Iteração 8          true      8         64
Iteração 9          true      9         55
Iteração 10         true     10         45

Process finished with exit code 0
```

IOS – Instituto de
Oportunidade Social

Ferramentas de debug

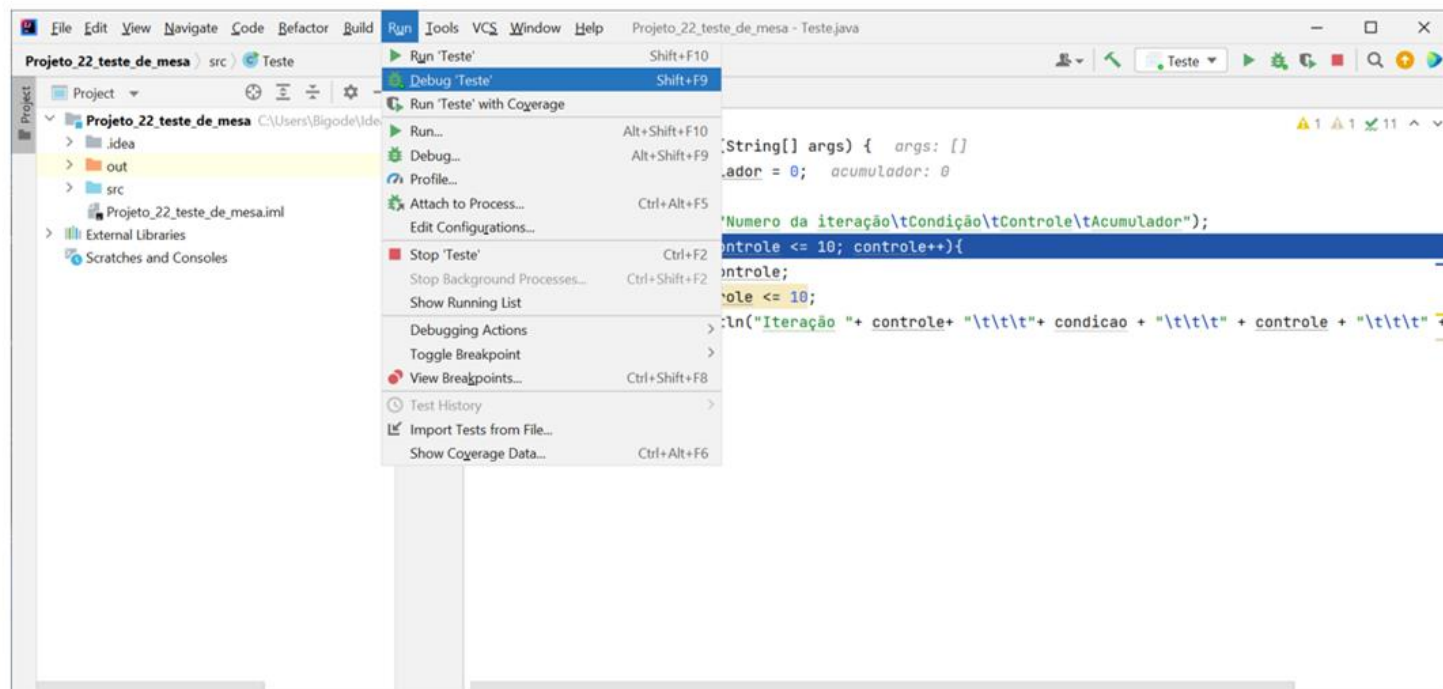


Ferramentas de debug

brincadeira para testarmos instruções e formatar saídas. Isso porque, é muito mais útil você aprender a utilizar as ferramentas de depuração (debug) disponíveis na IDE que você está utilizando, do que inserir instruções no código que não são necessárias para o funcionamento principal do programa.

Ferramentas de debug

Agora, você pode acessar a ferramenta de debug no Menu Run → Debug



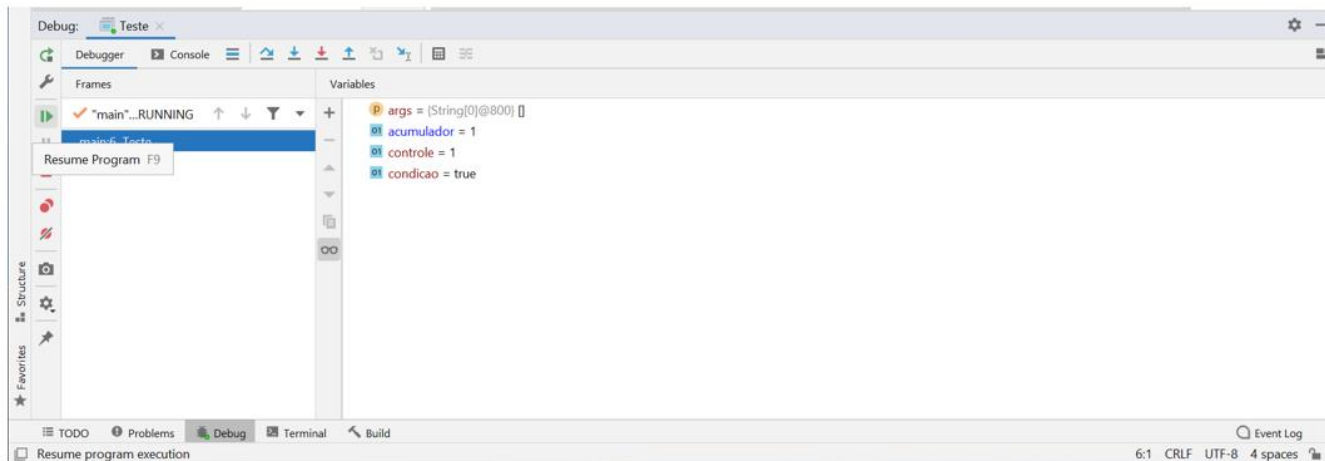
Ferramentas de debug

Agora, você pode acessar a ferramenta de debug no Menu Run → Debug

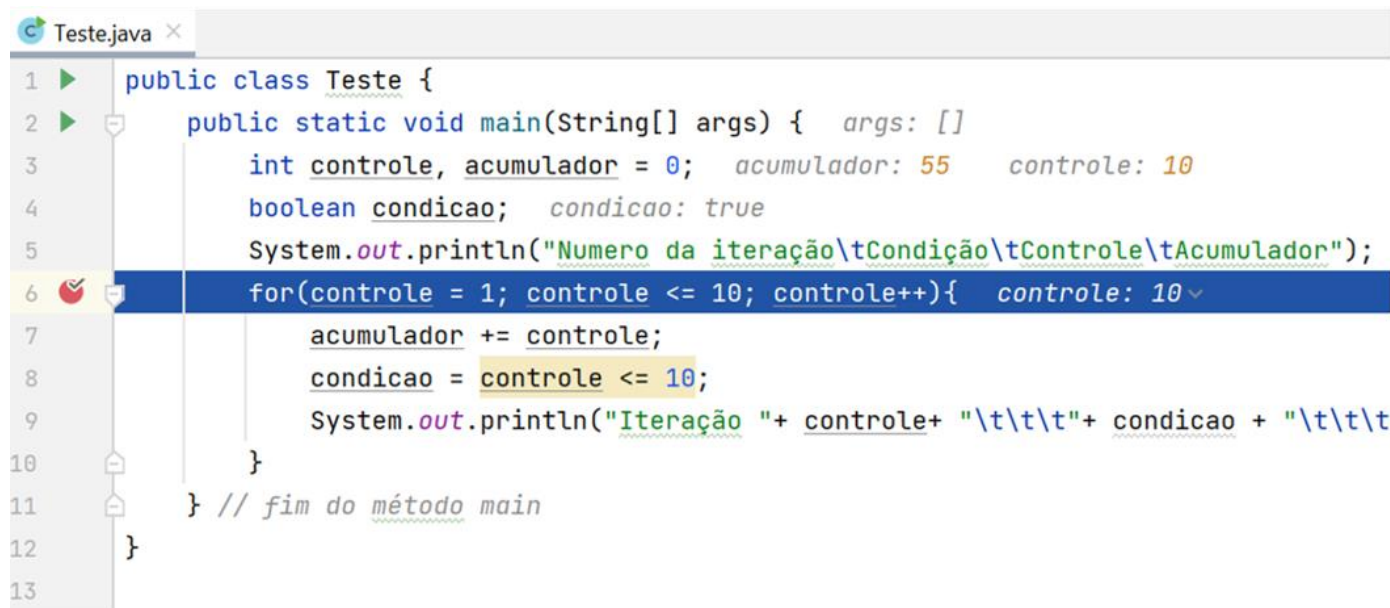


Ferramentas de debug

Observe que ao acessar o Debug do programa, as variáveis que já foram inicializadas com algum valor já aparecem na janela do depurador. Se você clicar uma vez no botão Resume Program (tecla de atalho F9), o programa irá executar uma iteração do for e parar na linha novamente.



Você pode clicar várias vezes até acabar as iterações do comando for:



```
1 public class Teste {
2     public static void main(String[] args) {    args: []
3         int controle, acumulador = 0;    acumulador: 55    controle: 10
4         boolean condicao;    condicao: true
5         System.out.println("Numero da iteração\tCondição\tControle\tAcumulador");
6         for(controle = 1; controle <= 10; controle++){    controle: 10
7             acumulador += controle;
8             condicao = controle <= 10;
9             System.out.println("Iteração "+ controle+ "\t\t\t"+ condicao + "\t\t\t"
10         }
11     } // fim do método main
12 }
13 }
```

IOS – Instituto de
Oportunidade Social

Vetores



Vetor é um tipo especial de variável homogênea, que possui **posições contínuas na memória**, que são acessadas pelo **mesmo nome**. Eles armazenam “dados” do **mesmo tipo** (int, char, double, etc). Exemplo: um vetor do tipo double para armazenar cinco notas dos alunos.

4.5	6.5	8.0	3.5	6.0
0	1	2	3	4
notas				

Sintaxe do vetor:

```
tipo identificador[] = new tipo[tamanho];
```

```
double notas[] = new double[5];
```

declarando um vetor de
double com 5 elementos

equivalente a 5 variáveis do tipo double

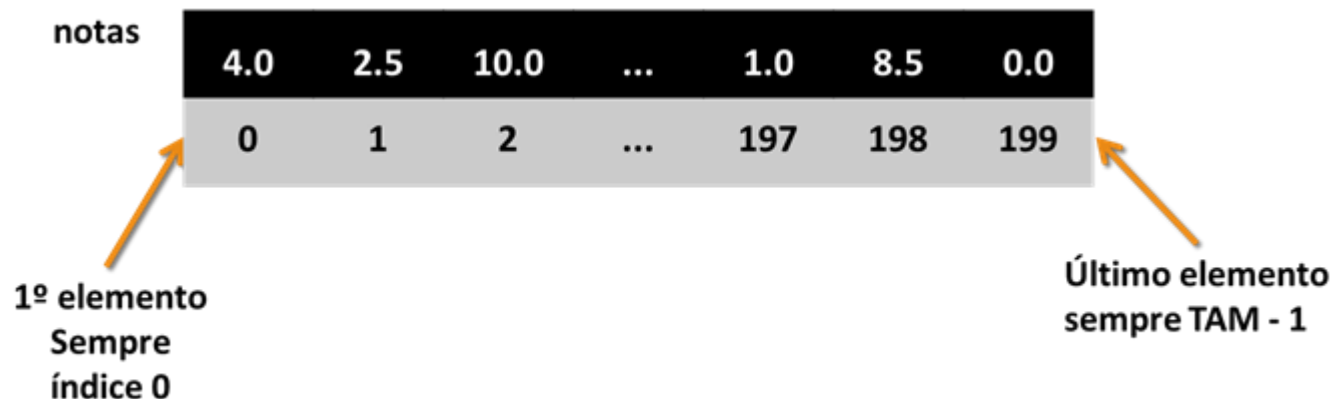
Para acessar um elemento de um vetor:

valores

4.0	2.5	1.2	4.8	1.0	8.5	0.0
0	1	2	3	4	5	6

índices

```
double notas[] = new double[200]
```



Inicialização de vetores:

```
double notas[] = new double[]{1.5, 4.5, 1.2, 9.8, 9.9};  
int primos[] = new int[]{2, 3, 5, 7, 11, 13};  
char dias[] = new char[]{'d', 's', 't', 'q', 'q', 's', 's'};
```

Acessando o vetor:

```
double notas[] = {4.5, 6.5, 8.0, 3.5, 6.0};
```

Atribuindo no primeiro elemento:

```
notas[0] = 9.0;
```

Imprimir último elemento no console:

```
System.out.println(notas[4]);
```

Vetores com estrutura de repetição:

```
for(int i = 0; i < 5; i++) {  
    System.out.println("Digite uma nota:");  
    nota[i] = entrada.nextDouble();  
}
```

length:

```
Scanner entrada = new Scanner(System.in);  
System.out.println("Digite o tamanho do vetor");  
int tamanho = entrada.nextInt();  
  
double notas[] = new double[tamanho];  
int tamanhoVetor = notas.length; // lendo o tamanho do vetor  
  
System.out.println("O tamanho do vetor é " + tamanhoVetor);  
entrada.close();
```


IOS – Instituto de
Oportunidade Social

Vamos Praticar



Vamos Praticar I, II e III



Apostila de Java:

Aula_08_Vetor

Páginas 4 a 8

OBS: Acompanhar o passo a passo com o instrutor

IOS – Instituto de
Oportunidade Social

Exercícios



Fazer Exercícios da apostila da Aula 07 de Java (Teste de Mesa) do Exercício 2 ao 6, e da Aula 08 (Vetores) do Exercício 1 ao 5 e subir no GitHub.