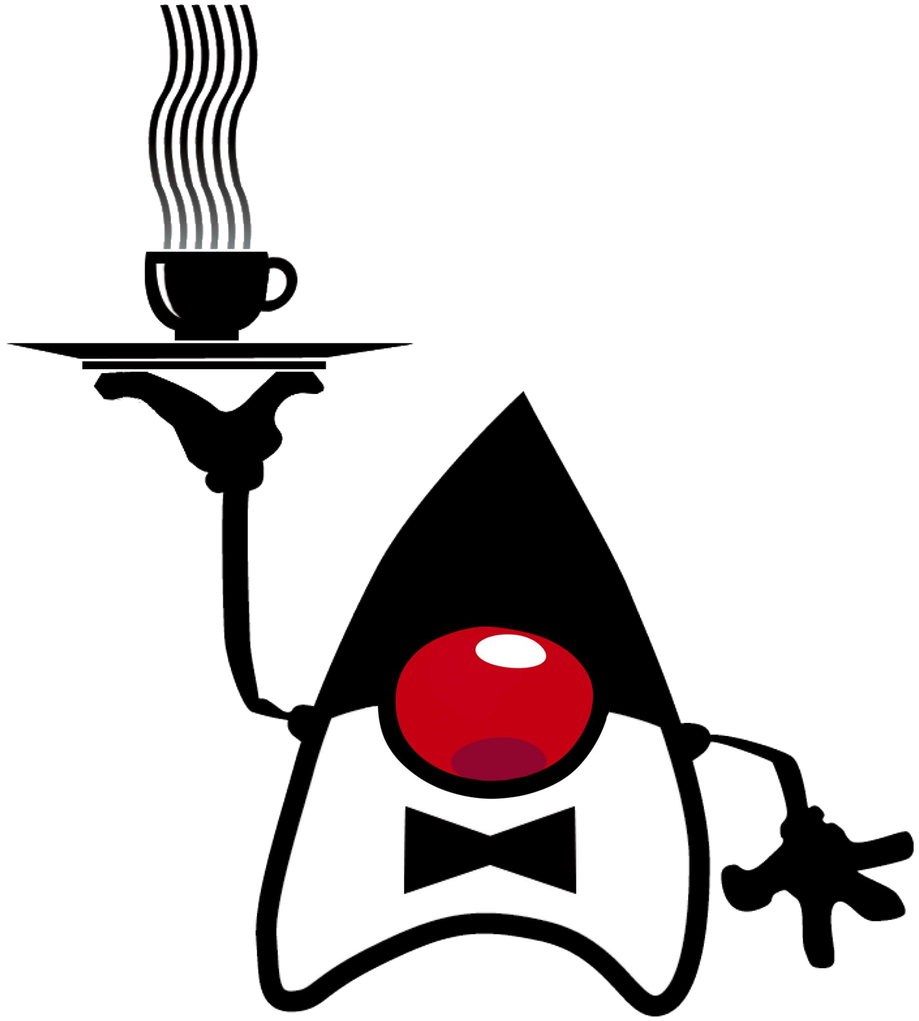
Trilha Java

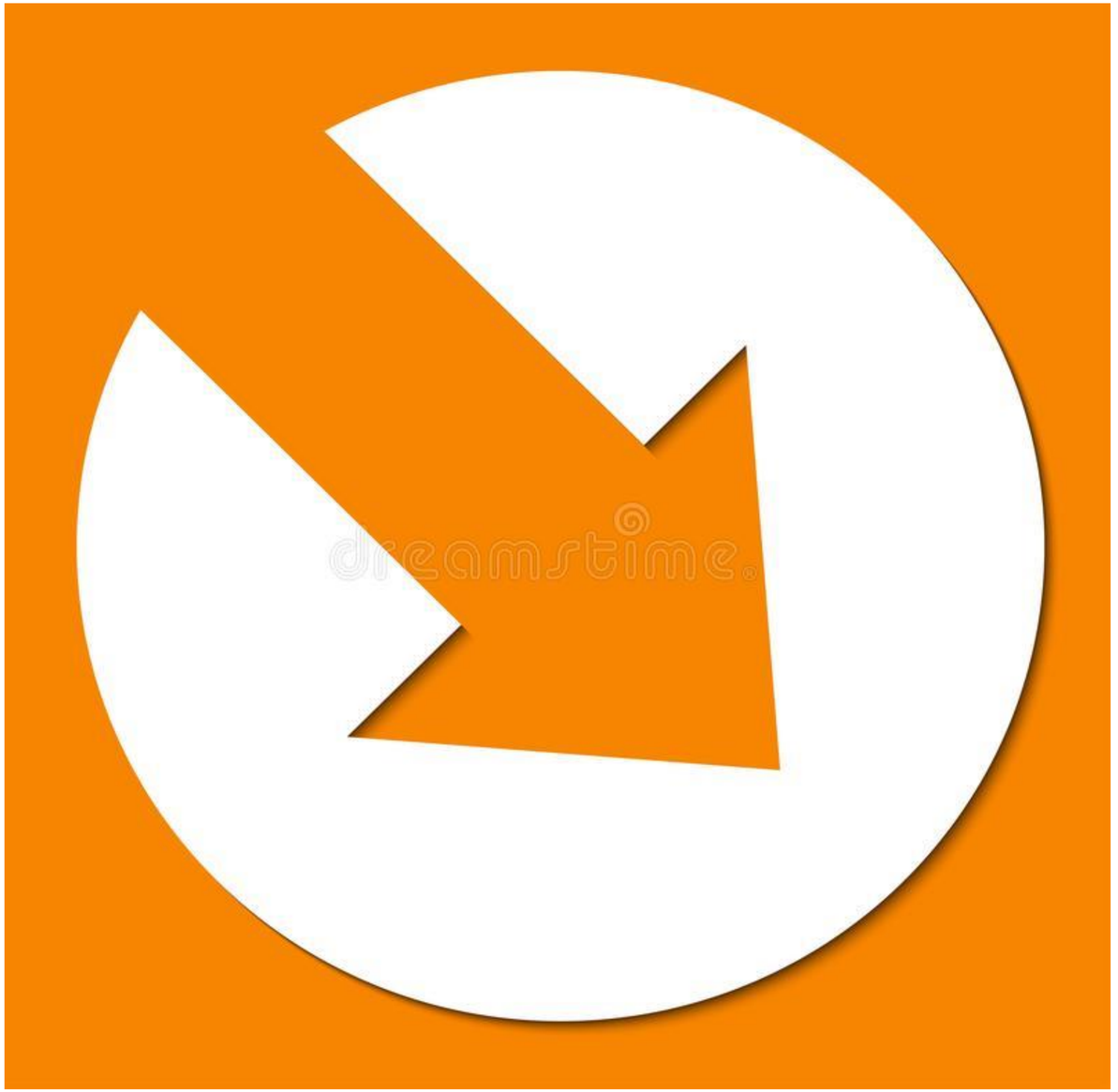
Encontro 11 – Vetores e Métodos



vnt/school
powered by venturus

Recapitulação

1. Estrutura de Repetição: While.
2. Estrutura de Repetição: For.
3. Estrutura de Repetição: Do while
4. Exemplos.
5. Atividades.



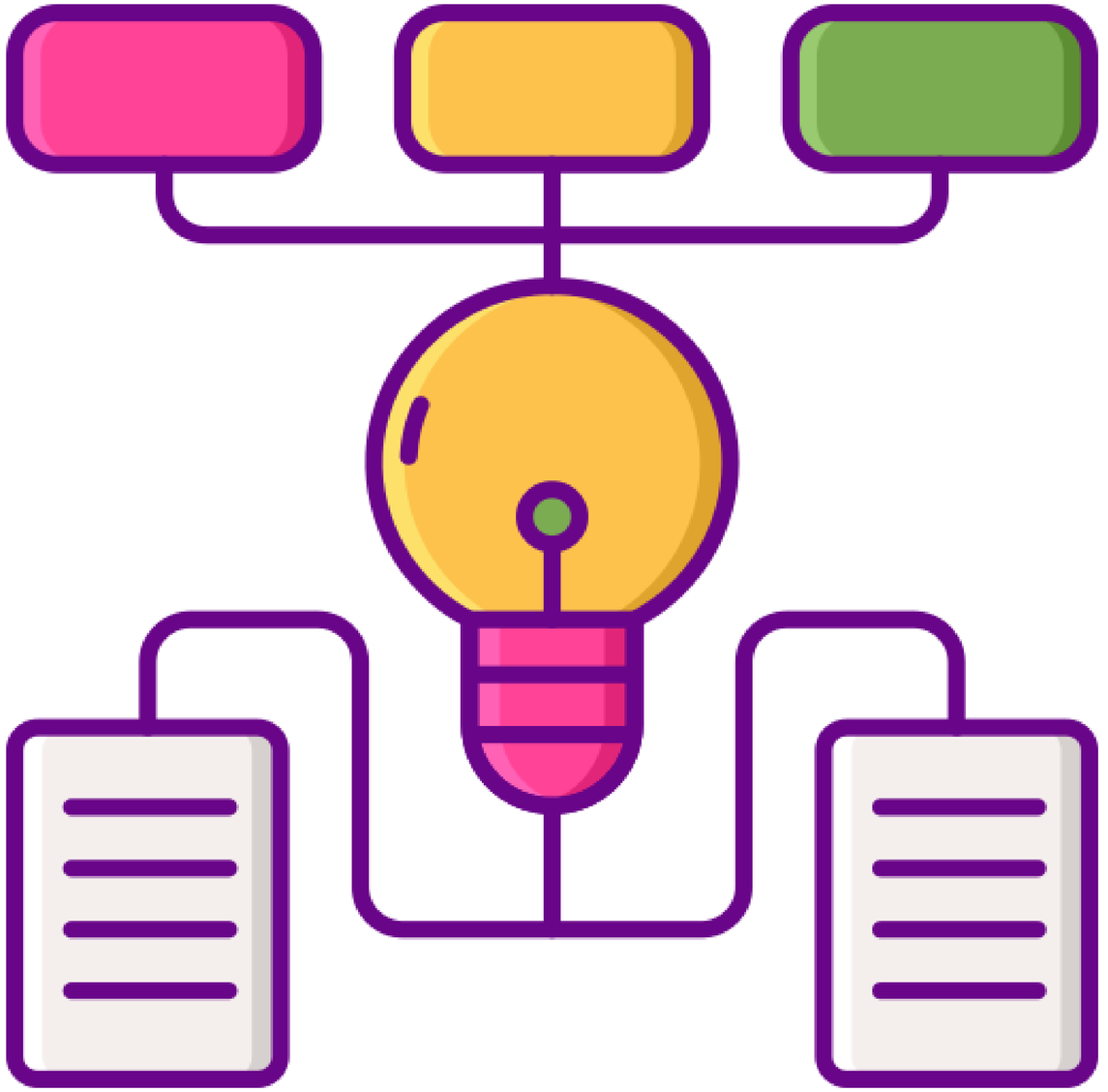
vnt/school
powered by venturus

Agenda

1. Vetores
2. Matrizes
3. Métodos(Funções)
4. Exemplos
5. Exercícios

vnt /school
powered by venturus

Introdução



vnt /school
powered by venturus

Variáveis Indexadas

A declaração de variáveis, uma a uma, é suficiente para a codificação algorítmica da solução de uma ampla gama de problemas.

Mas é insuficiente para resolver um grande número de problemas computacionais.

vnt /school
powered by venturus

Variáveis Indexadas

Exemplo:

Como construir um algoritmo que leia os nomes de 500 pessoas e imprima um relatório destes mesmos nomes, mas ordenados alfabeticamente?

vnt/school
powered by venturus

Variáveis Indexadas

Exemplo 1:

Não seria uma tarefa simples, pois teríamos que definir 500 variáveis do tipo String, como é mostrado ao lado.

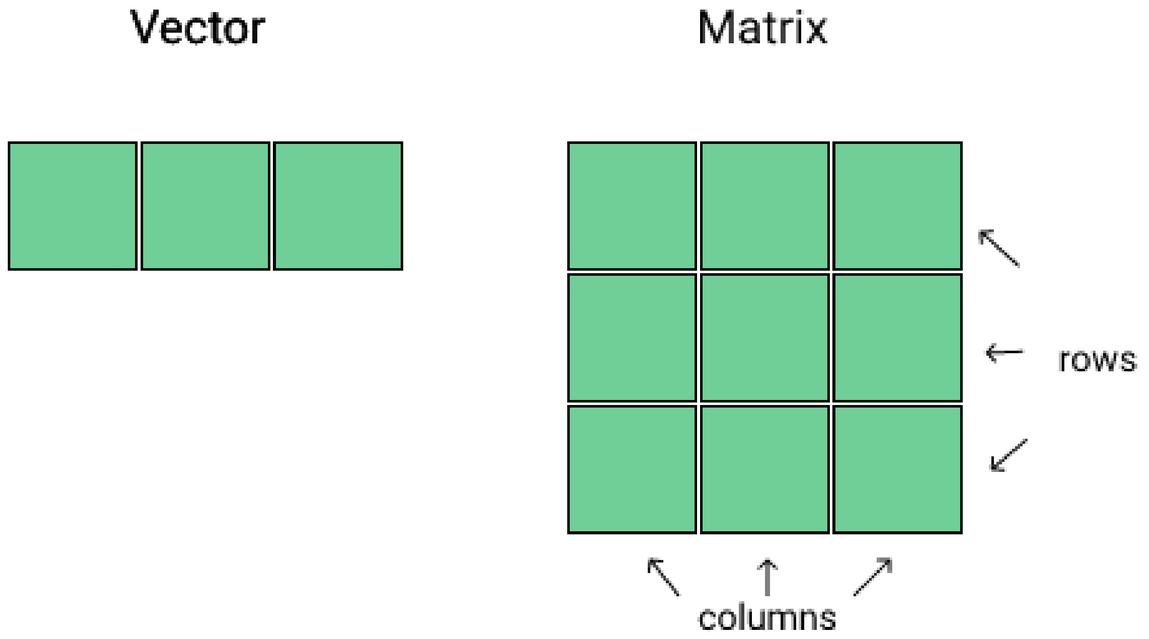
```
package aula11;
import java.util.Scanner;
public class Exemplo1 {
    public static void main(String[] args) {
        String nome1, nome2, nome499, nome500;

        Scanner entrada = new Scanner(System.in);
        nome1 = entrada.nextLine();
        nome2 = entrada.nextLine();
        nome499 = entrada.nextLine();
        nome500 = entrada.nextLine();
    }
}
```

Variáveis Indexadas

A forma mais eficiente de trabalhar com coleções de elementos em Java é através da construção de vetores (arrays).

Em Java, arrays são objetos que armazenam múltiplas variáveis do mesmo tipo.

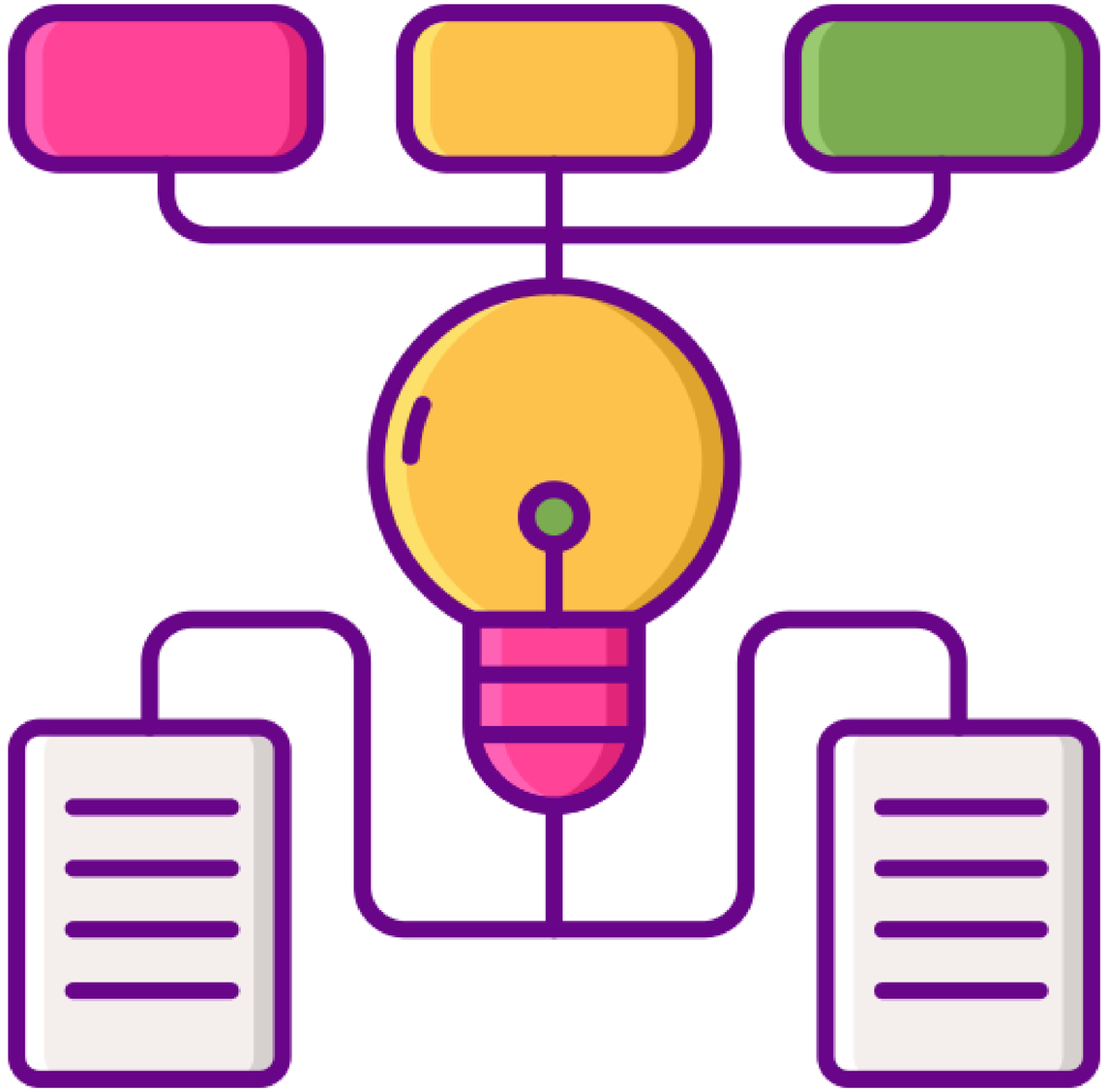
Uma dimensão = **vetor**
Duas dimensões = **matriz**



vnt/school
powered by venturus

Vetores

Array Unidimensional



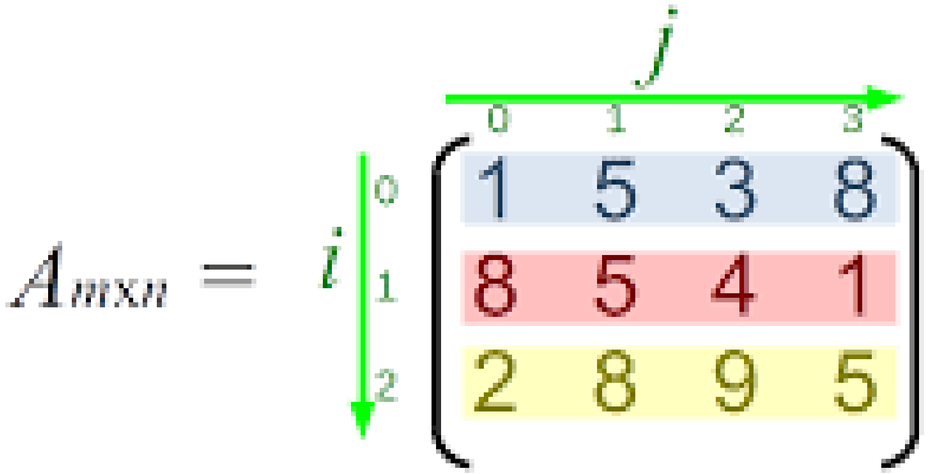
vnt/school

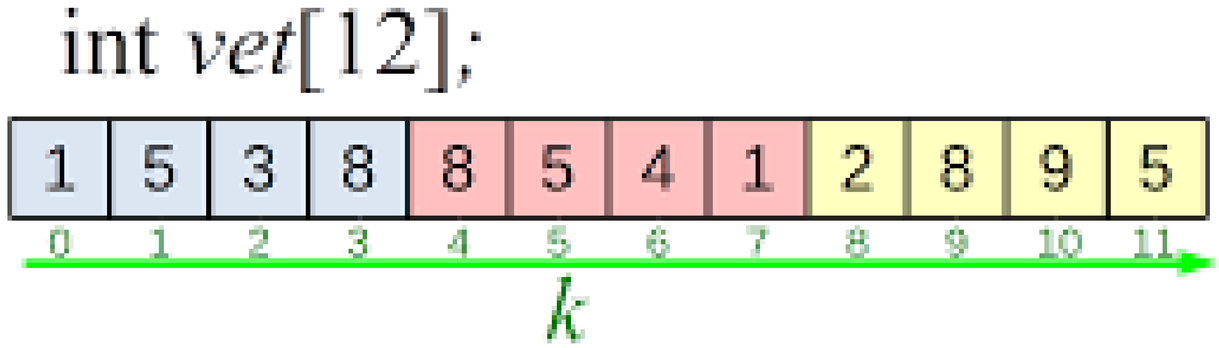
powered by venturus

Vetores

Java oferece diversos mecanismos para a criação de novos tipos de dados a partir de tipos já existentes.

Um desses mecanismos é o que permite a construção de vetores e matrizes.





vnt/school
powered by venturus

Vetores

Os vetores são definidos pelo tipo de dados que eles devem armazenar e a quantidade de posições.

Exemplo:
Vetor de 8 posições para armazenar números reais.
Vetor de 40 posições para armazenar objetos do tipo Aluno

Os vetores são estruturas homogêneas:
Um vetor de inteiros só armazena dados do tipo inteiro

Criando Vetores

Sintaxe:

```
<tipo>vetor[  ] = new  <tipo>[tamanho];

<tipo>[  ] vetor = new  <tipo>[tamanho];
```

As duas formas estão corretas.

Exemplos:

```
int meuVetor[  ] = new int[5];
boolean[  ] resultados = new boolean[30];
String nomes[  ] = new String[8];
```

vnt/school
powered by venturus

Criando Vetores

Um exemplo:

```
int v[  ] = new int [10];
```

v é declarado com um vetor de inteiros.
A expressão **new int [10]** cria efetivamente um vetor de inteiros, de tamanho 10.
o comando de atribuição associa o vetor criado ao vetor v.

Criando Vetores

Inicializando elementos:

```
V[0] = 1;
V[1] = 4;
V[2] = 7;
V[3] = 9;
V[4] = 2;
```

```
int v[  ] = new int [5];
```

Criando Vetores

Um vetor também pode ser criado a partir de uma lista de valores entre { e } e separados por vírgula. Vetor já inicializado!!

Exemplo:

```
int[  ] primos = { 2,3,5,7,11,13,17,19 };
char[  ] dias = { 'd','s','t','q','q','s','s'};
String[  ] meses = {"jan","fev","mar","abr" };
```

Acessando Vetores

Tendo criado um vetor, o acesso aos seus elementos é feito a partir da sua posição, ou índice, no vetor.

Se um vetor tem N elementos, os índices dos seus elementos vão variar entre 0 e N-1.

```
int v[ ] = new int [5];
```

Os elementos são acessados através do operador de indexação [ ].

Criando Vetores

```java
package aula11;
public class Exemplo2 {
    public static void main(String[] args) {
        int v[] = new int[5];
        v[0] = 1;
        v[1] = 4;
        v[2] = 7;
        v[3] = 9;
        v[4] = 2;
        for (int c=0; c<5; c++) {
            System.out.println("Na posicao "+ c + ", temos o valor: " + v[c]);
        }
        System.out.println("************************************************");

        int n[] = {1, 4, 7, 9, 2};
        for (int i=0; i<5; i++) {
            System.out.println("Na posicao "+ i + ", temos o valor: " + n[i]);
        }
    }
}
```

Acessando Vetores

Exemplo 3: Crie um programa que leia 4 números através de um vetor e logo em seguida imprima a soma desses números.

vnt/school
powered by venturus

Acessando Vetores

```java
package aula11;
import java.util.Scanner;
public class Exemplo3 {
    public static void main(String[] args) {
        int i;
        int num[] = new int[4];
        Scanner entrada = new Scanner(System.in);
        for (i=0; i<4; i++) {
            System.out.println("Digite o numero da posicao " + i + " :");
            num[i] = entrada.nextInt();
        }
        int total = num[0] + num[1] + num[2] + num[3];
        System.out.println("A soma dos elementos do vetor eh: " + total);
    }
}
```

Acessando Vetores

Exemplo 4: Crie um programa que inicialize os dois primeiros elementos de um vetor, como 0 e 1. Os demais elementos do vetor de comprimento 10, deve ser preenchido com a seguinte regra:

vetor[i] = vetor[i - 1] + vetor[i - 2]

Imprima o valor de cada elemento do vetor.

vnt/school
powered by venturus

Acessando Vetores

```
package aula11;
public class Exemplo4 {
    public static void main(String[] args) {
        int f[] = new int[10];
        f[0] = 0;
        f[1] = 1;
        for (int i = 2; i < 10; i++) {
            f[i] = f[i - 1] + f[i - 2];
            System.out.println("Na posicao "+ i + ", temos o valor: " + f[i]);
        }
    }
}
```

Tamanho do Vetor

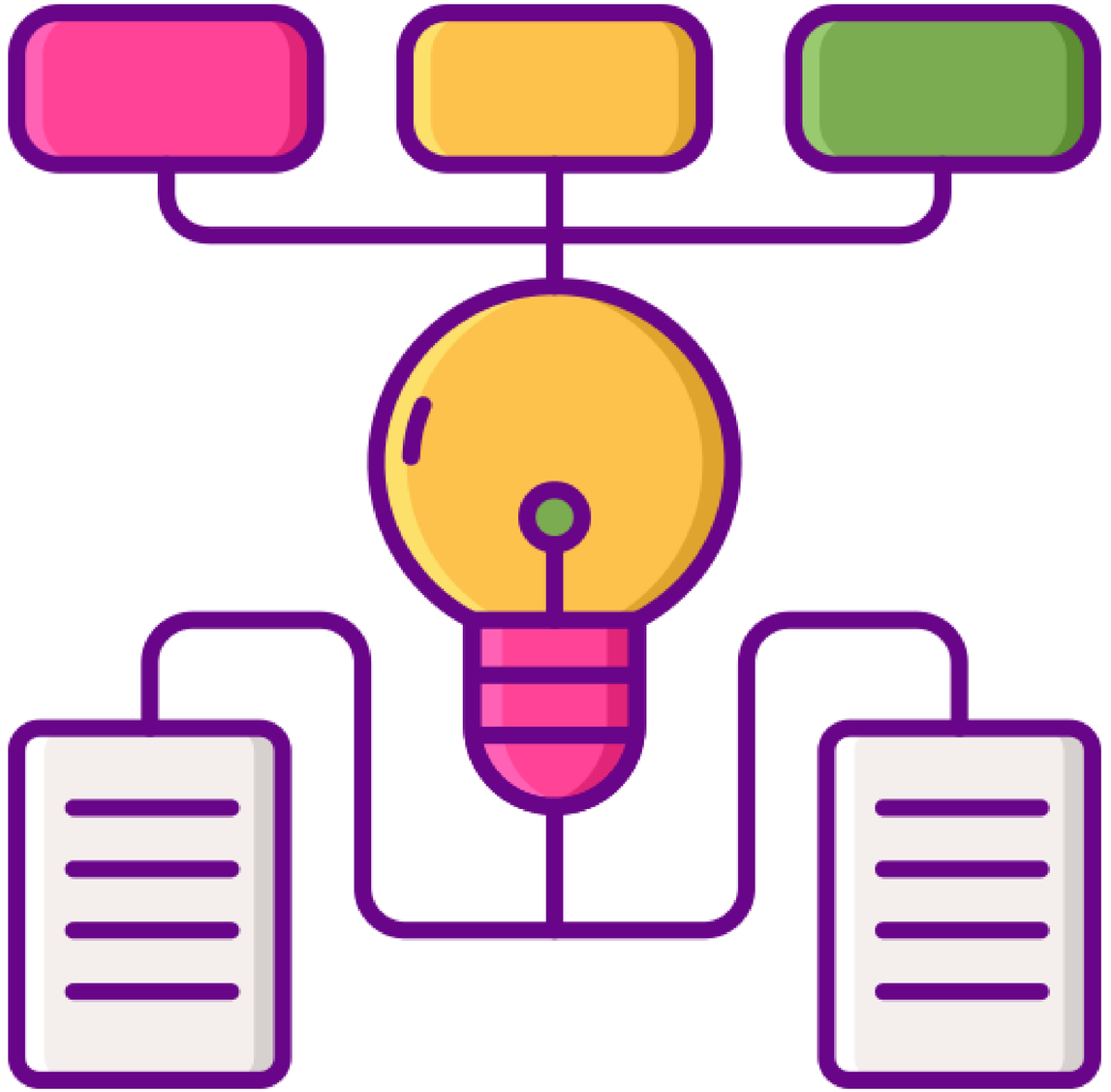
Para conhecer o tamanho total de um array basta você acessar o atributo **length**.

Este atributo retorna um valor inteiro (int) que indica qual a capacidade máxima de armazenamento deste array.

Exemplo:
```
int vetor[ ] = new int[10];
vetor.length;
```

Matrizes

Array Bidimensional

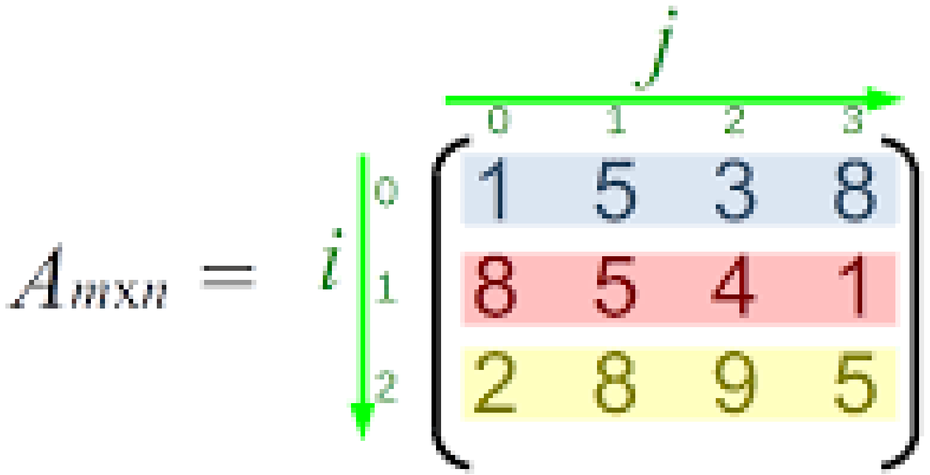


vnt /school

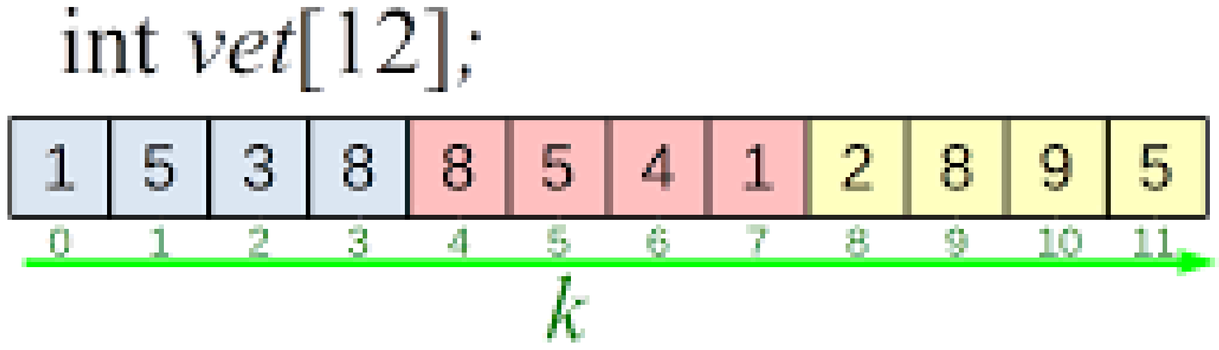
powered by venturus

Matrizes

Uma **matriz** é basicamente um vetor onde cada elemento é por sua vez um vetor.



Uma matriz pode ter **várias linhas** com vários valores, que comumente chamamos de linhas e colunas.



vnt/school
powered by venturus

Criando Matrizes

Sintaxe:

```
<tipo>matriz[  ][  ] = new  <tipo>[linhas][colunas];

<tipo>[  ][  ] matriz = new  <tipo>[linhas][colunas];
```

As duas formas estão corretas.

Exemplos:

```
int m1[  ][  ] = new int[3][3];
String nomes[  ][  ] = new String[4][3];
```

vnt/school
powered by venturus

Criando Matrizes

Inicializando elementos:

```
M[0][0] = 1;
M[0][1] = 4;
M[0][2] = 5;
M[1][0] = 6;
M[1][1] = 1;
M[1][2] = 7;
M[2][0] = 8;
M[2][1] = 9;
M[2][2] = 1;
```

```
int M[  ][  ] = new int [3][3];
```

1	4	5
6	1	7
8	9	1

Posição ?

Posição ?

vnt/school
powered by venturus

Criando Matrizes

Exemplo 5:

Neste exemplo podemos observar duas formas de criar matrizes. O primeiro caso refere-se ao formato mais simples e direto. O segundo caso, utiliza o conceito do laço de repetição para que o usuário tenha interação com o programa. **Observe na impressão!!**

<tipo>matriz[ ][ ] = **new** <tipo>[linhas][colunas];

Criando Matrizes

Exemplo 5:

```
package aula11;
public class Exemplo5 {
    public static void main(String[] args) {
        int M[][] = new int[2][2];
         M[0][0] = 1;
         M[0][1] = 4;
         M[1][0] = 7;
         M[1][1] = 9;
         System.out.println(M[0][0] + "\t" + M[0][1]);
         System.out.println(M[1][0] + "\t" + M[1][1]);
    }
}
```

Criando Matrizes

```java
package aula11;
import java.util.Scanner;
public class Exemplo5 {
    public static void main(String[] args) {
        int M[][] = new int[3][4];
        Scanner entrada = new Scanner(System.in);
        for (int i=0; i<3; i++) {
            for (int j=0; j<4; j++) {
                System.out.println("M["+ i +"]" + "M[" + j + "]:");
                M[i][j]= entrada.nextInt();
            }
        }
        for (int i=0; i<3; i++) {
            for (int j=0; j<4; j++) {
                System.out.println("Elemento: " + M[i][j]);
            }
        }
```

Tamanho da Matriz

O tamanho total da matriz é obtido através do atributo **length**.

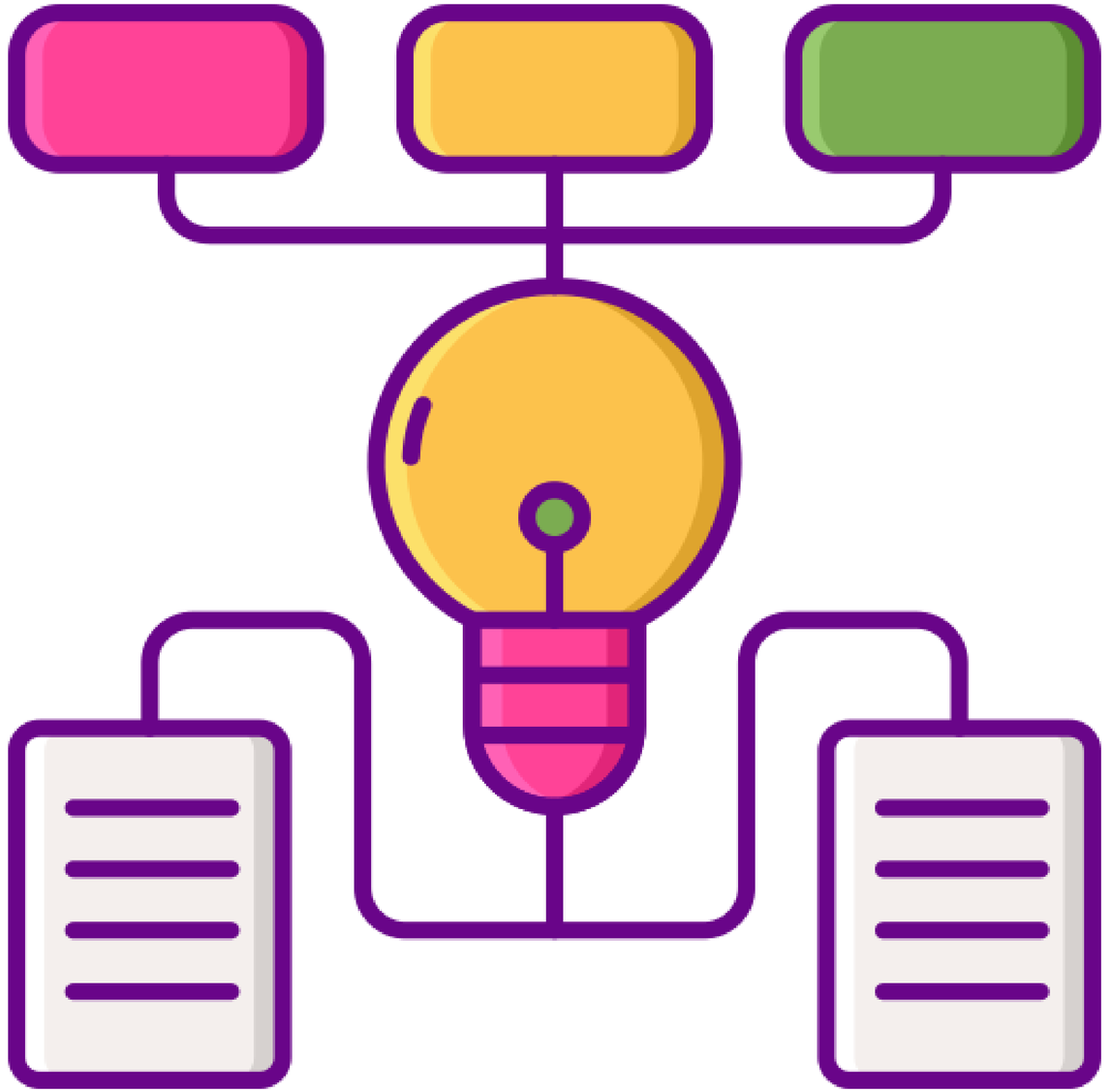
O atributo retorna um valor inteiro (int) que indica qual a capacidade máxima de armazenamento deste array.

Exemplo:
```
int[ ][ ] matriz = new int[2][3];
matriz.length;
matriz[0].length;
```

Vamos Praticar!!

Exercícios

Arrays



vnt /school

powered by venturus

Vetores e Matrizes

Atividade 1:

Como somar variáveis com vetores?

Crie um algoritmo que leia um vetor de tamanho 4. Inicialize o vetor com o usuário digitando os valores de cada elemento. Inicie a soma desse vetor com uma variável num=10.

Regra importante: para cada rodada a variável num deve assumir o seu novo valor, ou seja, o seu valor atual mais o elemento digitado na posição anterior.

Vetores e Matrizes

```java
package aula11;
public class Atividade1 {
    public static void main(String[] args) {
        int[] vetor = {1,2,3,4}; // Vetor já com valores preenchidos.
            int num = 10; // inicia variavel soma com 10.

            for(int i = 0; i < vetor.length; i++){
                num = num +  vetor[i]; // soma o valor num com o vetor.
                System.out.println("Elemento "+ i + ": " + num);
            }
    }
}
```

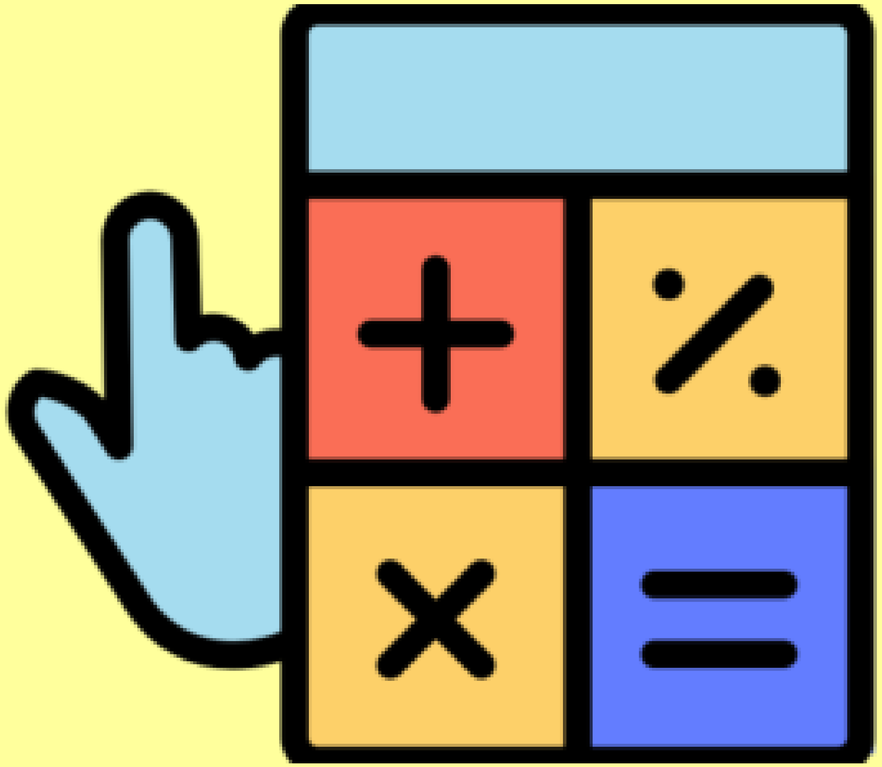
Vetores e Matrizes

Atividade 2:

Crie um algoritmo que faça a soma e a multiplicação das duas matrizes abaixo.

```
int matriza[ ][ ] = {{1, 2}, {4, 5}};
int matrizb[ ][ ] = {{7, 8}, {10, 11}};
```

Imprima o resultado da Soma e da Multiplicação.



vnt/school
powered by venturus

Vetores e Matrizes

```java
package aula11;
public class Atividade2 {
    public static void main(String[] args) {
        int matriza[][] = {{1, 2}, {4, 5}};
        int matrizb[][] = {{7, 8}, {10, 11}};

        int matrizsoma[][] = new int[2][2];
        int matrizmulti[][] = new int[2][2];

        //soma
        for (int i = 0; i < matriza.length; i++) {
            for (int j = 0; j < matriza.length; j++) {
                matrizsoma[i][j] = matriza[i][j] + matrizb
            }
        }
```

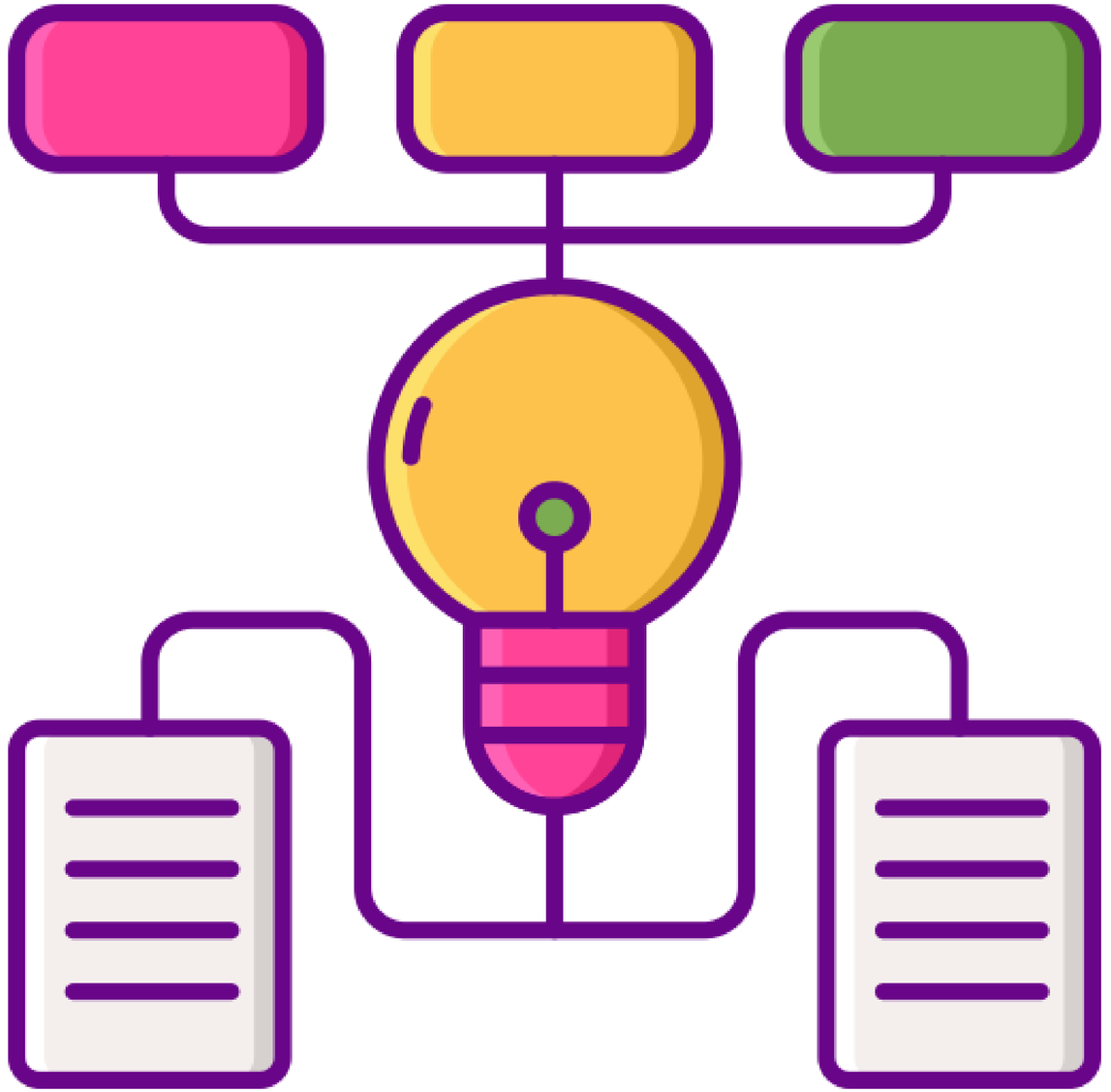
??

vnt/school
powered by venturus

Coffee time!

Métodos

Funções



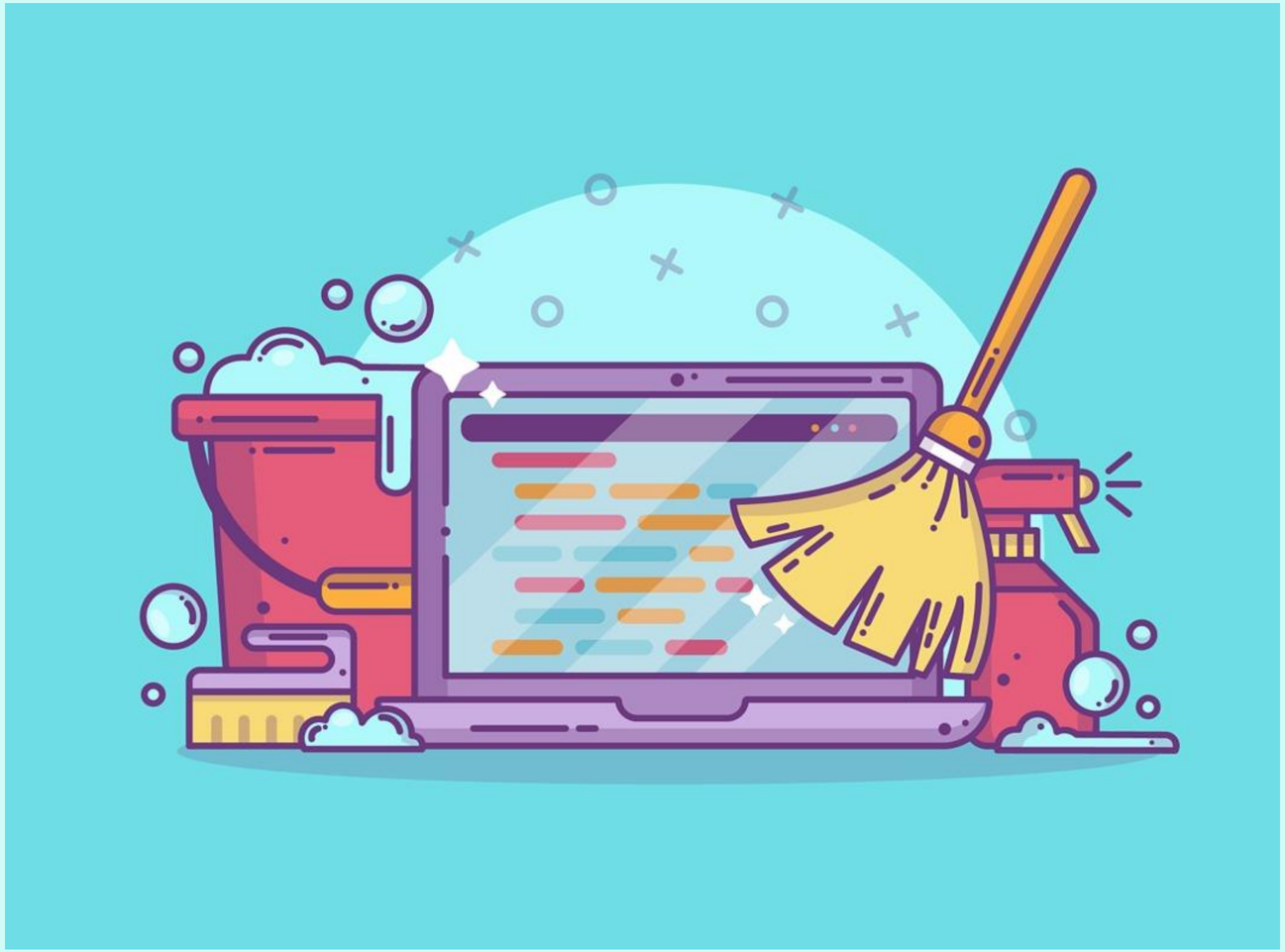
vnt /school

powered by venturus

Métodos: funções

Em geral, um **método** é uma maneira de executar alguma tarefa.

Da mesma forma,
o **método** em **Java** é
uma coleção de instruções que executam uma tarefa específica.



vnt/school
powered by venturus

Métodos: funções

Representam um processamento que possui um significado.

Math.sqrt(double)

System.out.println(string)

Principais vantagens:

modularização, delegação

e reaproveitamento

vnt/school

powered by venturus

Métodos: funções

Dados de entrada e saída:
Funções podem receber dados de entrada (parâmetros ou argumentos).
Funções podem ou não retornar uma saída.

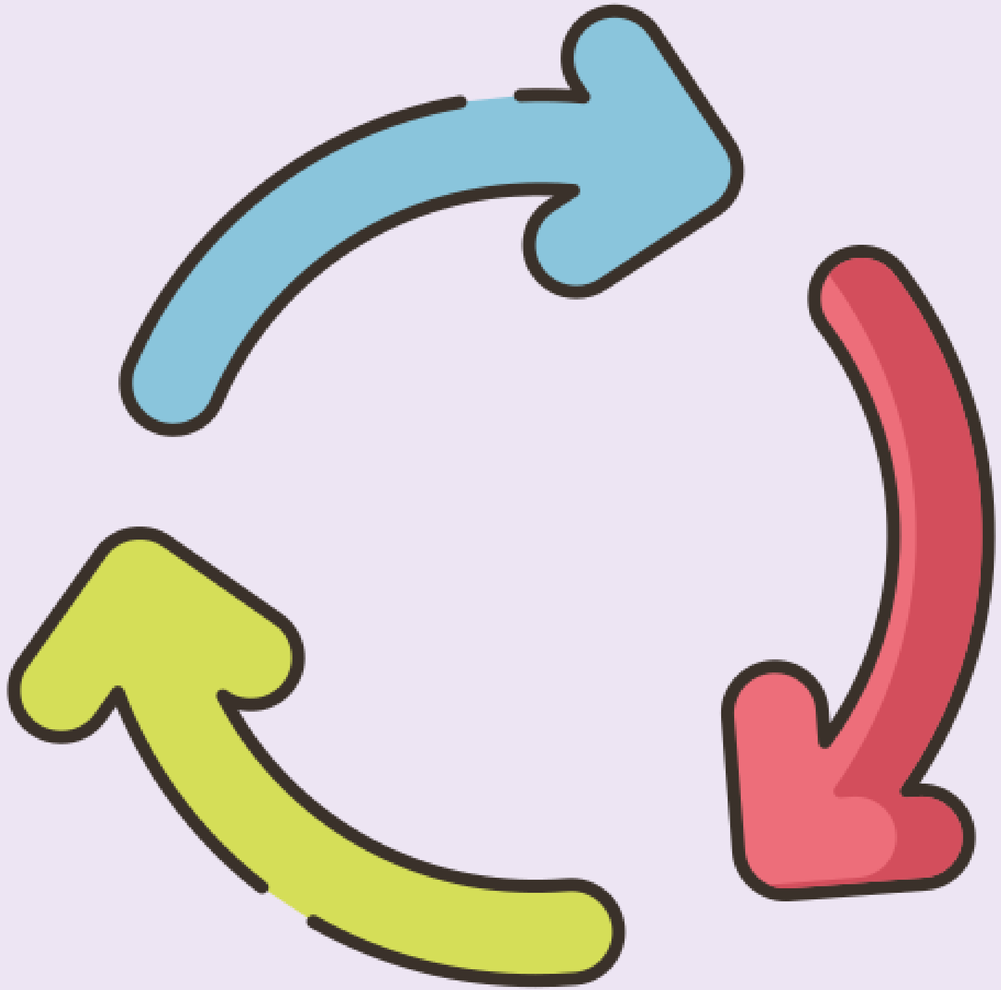
Em **orientação a objetos**, "**funções**" em classes recebem o nome de "**métodos**"

vnt/school
powered by venturus

Métodos: funções

As **funções/métodos** podem ser criadas e assim executarem determinada tarefa.

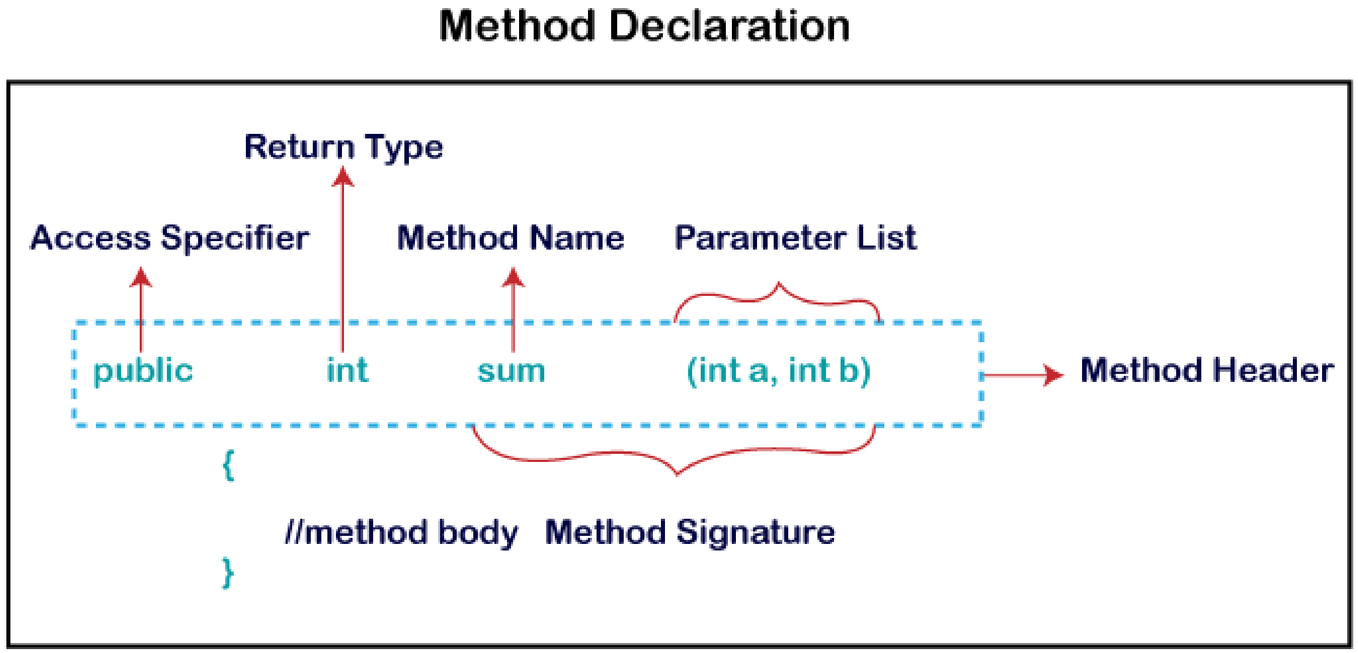
Muitas já existem dentro da biblioteca do Java e facilitam o desenvolvimento do código



vnt/school
powered by venturus

Métodos: funções

A declaração do método fornece informações sobre os atributos do método, como visibilidade, tipo de retorno, nome e argumentos.



vnt/school
powered by venturus

Métodos Predefinidos

São os métodos já definidos nas bibliotecas de classes Java.

Podem ser "chamando" no programa a qualquer momento.

Alguns métodos predefinidos são **equals(), compareTo(), sqrt()**, **random()** etc.

Métodos: funções

Exemplo 6)

Fazer um programa para ler três números inteiros e mostrar na tela o maior deles.

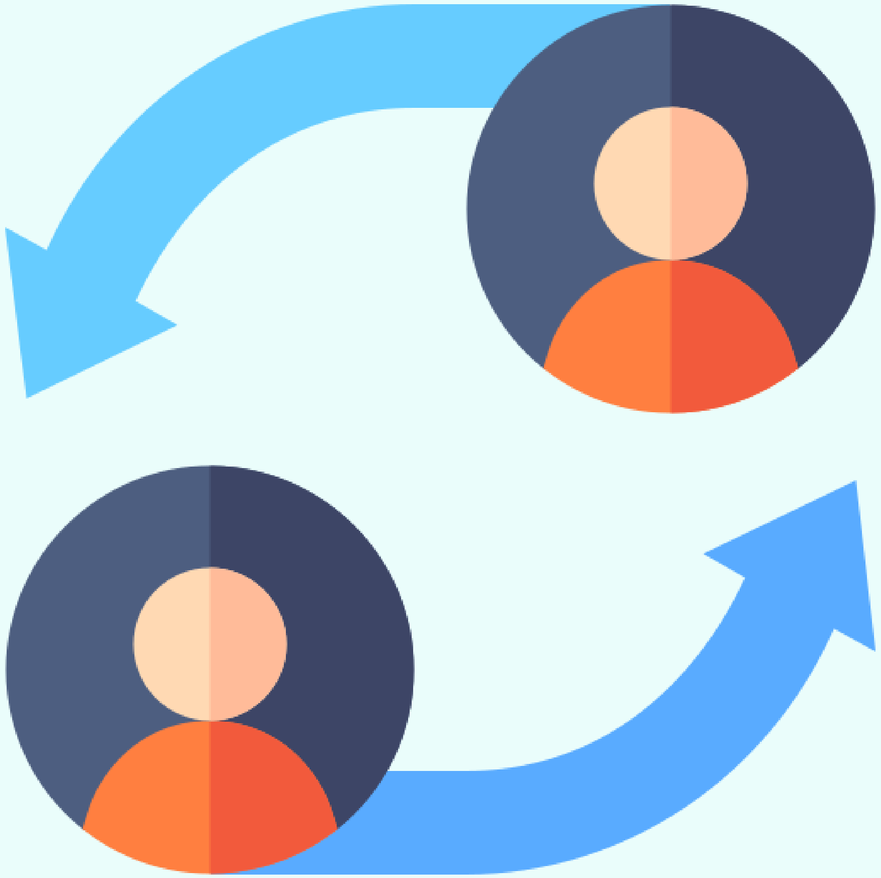
Métodos: funções

```java
package aula11;
import java.util.Scanner;
public class Exemplo6 {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Entre com 3 numeros:");
        int a = entrada.nextInt();
        int b = entrada.nextInt();
        int c = entrada.nextInt();
        if (a > b && a > c) {
            System.out.println("Maior = " + a);
        } else if (b > c) {
            System.out.println("Maior = " + b);
        } else {
            System.out.println("Maior = " + c);
        }
        entrada.close();
    }
}
```

Métodos: funções

Imagine que este procedimento de descobrir o número maior será executado várias vezes no código.

O que isso sugere? **Método**



vnt/school
powered by venturus

Métodos: funções

```java
package aula11;
import java.util.Scanner;
public class Exemplo7 {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Enter com 3 numeros:");
        int a = entrada.nextInt();
        int b = entrada.nextInt();
        int c = entrada.nextInt();
        int maior = max(a, b, c);
        mostrarResultado(maior);
        entrada.close();
    }
```

??

vnt/school
powered by venturus

Métodos: funções

No exemplo, temos três métodos predefinidos **main(), print()** e **max()**.

classe **PrintStream <- print()**

classe **Math <- max()**

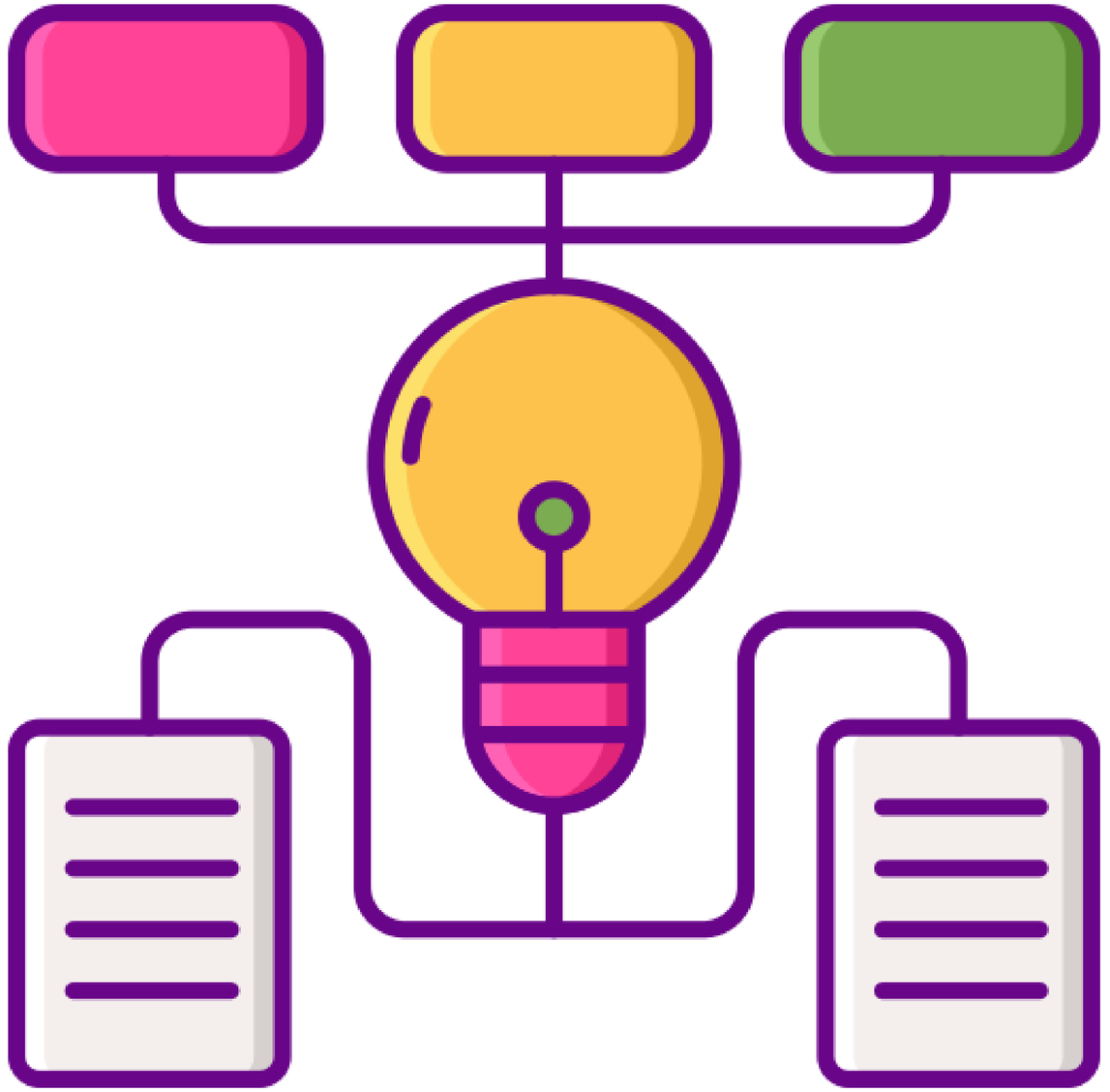
```
package aula11;
public class Exemplo8 {
    public static void main(String[] args) {
        // usando o método max() da classe Math
      System.out.print ( "O maximo eh:"  + Math.max(9 , 7));
    }
}
```

vnt/school
powered by venturus

Vamos Praticar!!

Exercícios

Métodos



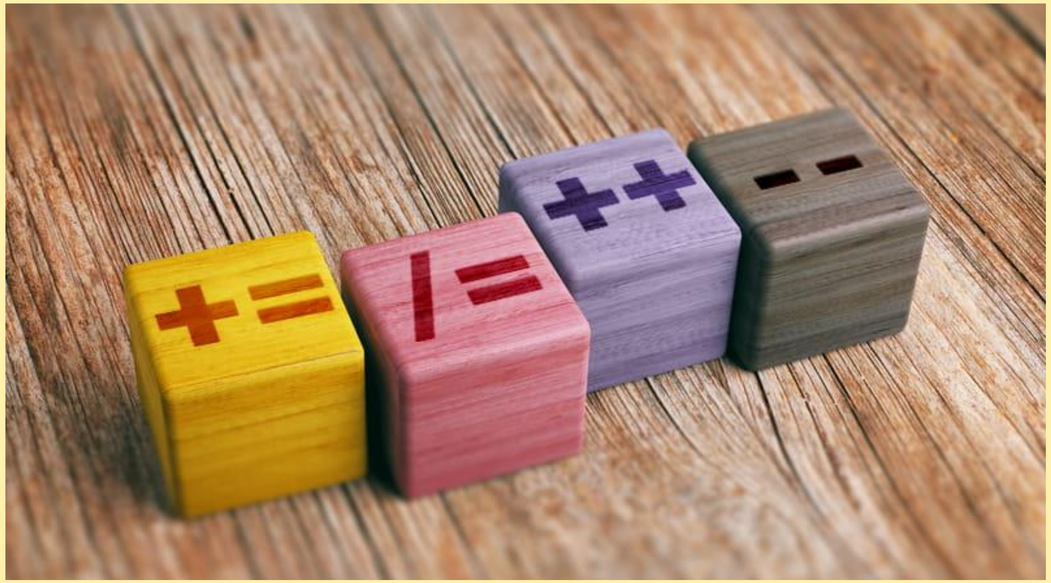
vnt /school

powered by venturus

Métodos: funções

Atividade 3:

Crie um algoritmo que leia dois números e calcule a soma, a multiplicação e o dobro. Por fim, imprima o resultado. Para a resolução do exercício, crie as funções soma, multiplicação, dobro e imprima.



vnt/school
powered by venturus

Métodos: funções

```java
package aula11;
import java.util.Scanner;
public class Atividade3 {
    static int x, y, z;
    static Scanner entrada = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("Digite o primeiro numero: ");
        x = entrada.nextInt();
        System.out.println("Digite o primeiro numero: ");
        y = entrada.nextInt();
        System.out.println("Digite o primeiro numero: ");
        z = entrada.nextInt();

        imprima("soma: " + somar(x, y));
        imprima("multiplicacao: " + multiplicar(x, y));
        imprima("dobro do primeiro: " + dobro(x));
        imprima("dobro do segundo: " + dobro(y));
        imprima("dobro do terceiro: " + dobro(z));
```

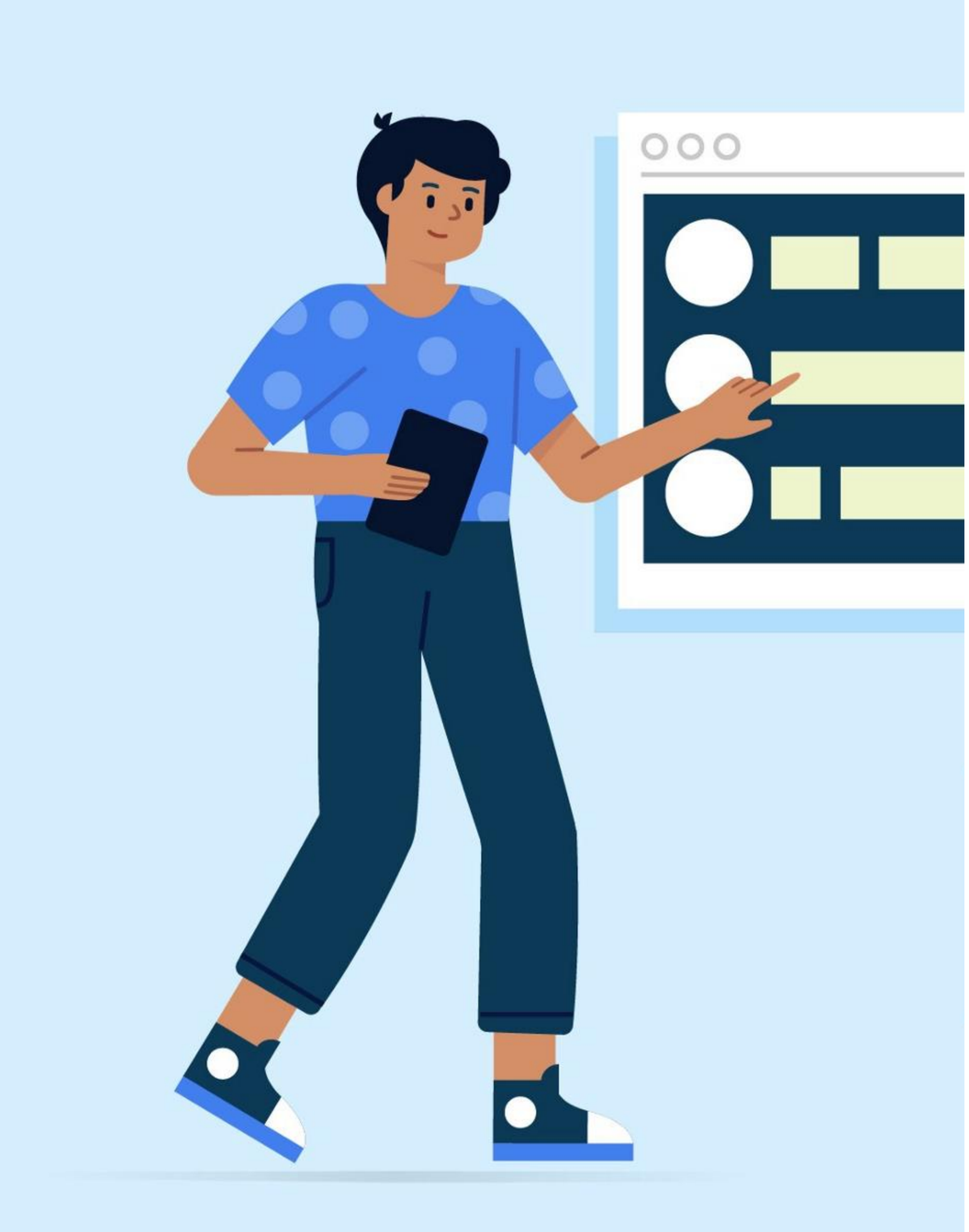
??

vnt/school
powered by venturus


Review
e
Preview

Menti

Vamos fazer teste?



Comunidade VNT

Dica de hoje

O link abaixo apresenta uma breve explicação sobre o conceito de métodos e suas variadas formas de ser aplicada. Este é um tópico muito importante abordado em programação orientada a objeto.

https://www.javatpoint.com/pt/m%C3%A9todo-em-java

Boa leitura!!

Referências

[1] A. Goldman, F. Kon, Paulo J. S. Silva; Introdução à Ciência da Computação com Java e Orientação a Objetos (USP). 2006. Ed. USP.
[2] Algoritmo e lógica de programação. Acessado julho/2022: https://visualg3.com.br/
[3] G. Silveira; Algoritmos em Java; Ed. Casa do Código.
[4] M. T. Goodrich, R. Tamassia; Estrutura de dados e algoritmos em Java. Ed Bookman. 2007.
[5] Algoritmo e lógica de programação. Acessado julho/2022: https://www.cursoemvideo.com/
[6] P. Silveira, R. Turini; Java 8 Pratico: lambdas, streams e os novos recursos da linguagem. Ed. Casa do Código.
[7] Linguagem Java: Curso acessado em agosto/2022: https://www.udemy.com/
[8] Linguagem Java: Curso acessado em setembro/2022: https://www.cursoemvideo.com/