



INSTITUTO FEDERAL
Sertão Pernambucano
Campus Salgueiro

Estrutura de Dados e Algoritmos com Java

Prof. Heraldo Gonçalves Lima Junior
heraldo.junior@ifsertao-pe.edu.br

1. A linguagem JAVA

1.1. Linguagem JAVA

- ◎ Orientada a objetos;
- ◎ Grande comunidade;
- ◎ Fácil aprendizado;
- ◎ Desenvolva em qualquer sistema operacional para qualquer sistema operacional.

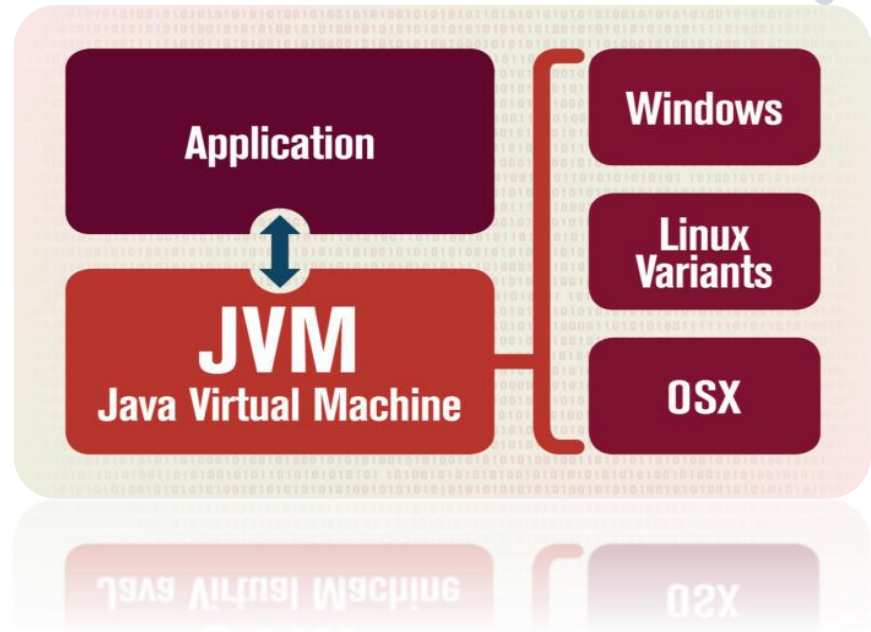


1.2. JVM - JAVA VIRTUAL MACHINE

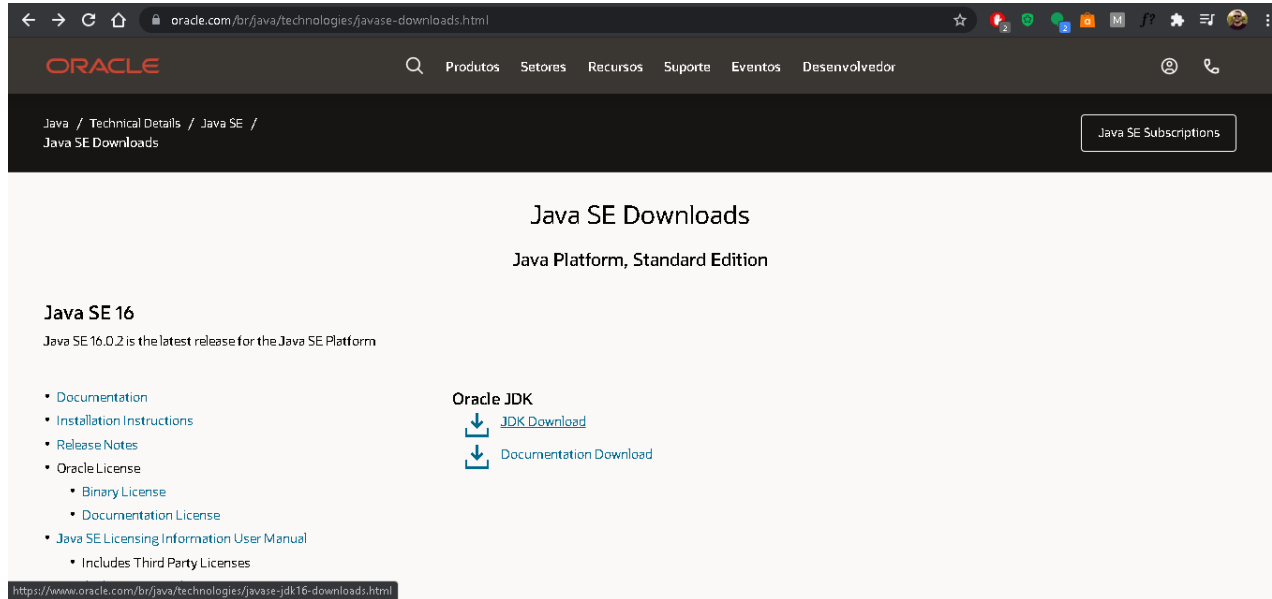
- © O Java utiliza o conceito de máquina virtual, no qual existe, entre o sistema operacional e a aplicação, uma camada extra responsável por traduzir o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional em que ela está rodando no momento

1.2. JVM - Java virtual machine

- ◎ Sua aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a Java Virtual Machine (JVM).



1.3. Ambiente de Desenvolvimento: JDK (Java Development Kit)



The screenshot shows the Oracle Java SE Downloads page in a web browser. The browser's address bar displays the URL `oracle.com/br/java/technologies/javase-downloads.html`. The Oracle logo is in the top left, and a navigation menu with links like 'Produtos', 'Setores', 'Recursos', 'Suporte', 'Eventos', and 'Desenvolvedor' is at the top. Below the navigation bar, the page title is 'Java SE Downloads' with the subtitle 'Java Platform, Standard Edition'. The main content area features 'Java SE 16' as the latest release. A list of links on the left includes 'Documentation', 'Installation Instructions', 'Release Notes', 'Oracle License' (with sub-links for 'Binary License' and 'Documentation License'), and 'Java SE Licensing Information User Manual' (with a sub-link for 'Includes Third Party Licenses'). On the right, under 'Oracle JDK', there are two download links: 'JDK Download' and 'Documentation Download', each with a download icon. The browser's status bar at the bottom shows the full URL: `http://www.oracle.com/br/java/technologies/javase-jdk16-downloads.html`.

ORACLE

Produtos Setores Recursos Suporte Eventos Desenvolvedor

Java / Technical Details / Java SE / Java SE Downloads

Java SE Subscriptions

Java SE Downloads

Java Platform, Standard Edition

Java SE 16

Java SE 16.0.2 is the latest release for the Java SE Platform

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
 - [Binary License](#)
 - [Documentation License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)

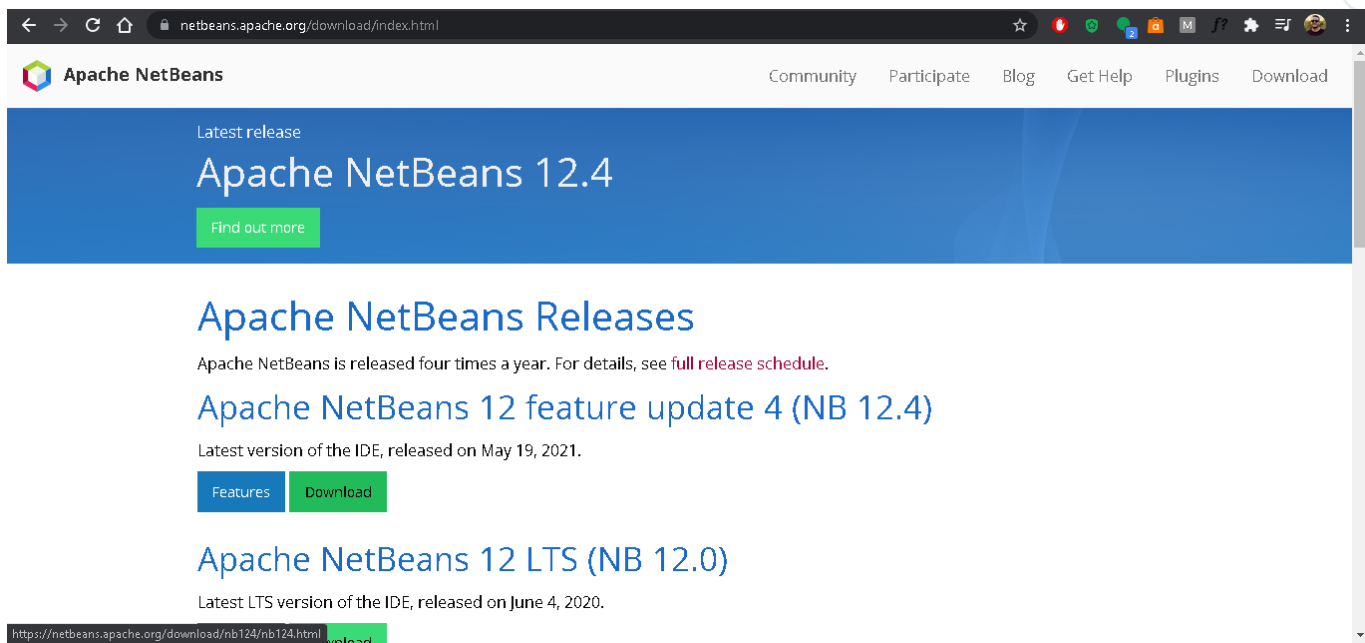
Oracle JDK

- [JDK Download](#)
- [Documentation Download](#)

`http://www.oracle.com/br/java/technologies/javase-jdk16-downloads.html`

Download: <https://www.oracle.com/br/java/technologies/javase-downloads.html>

1.3. Ambiente de Desenvolvimento: Apache Netbeans



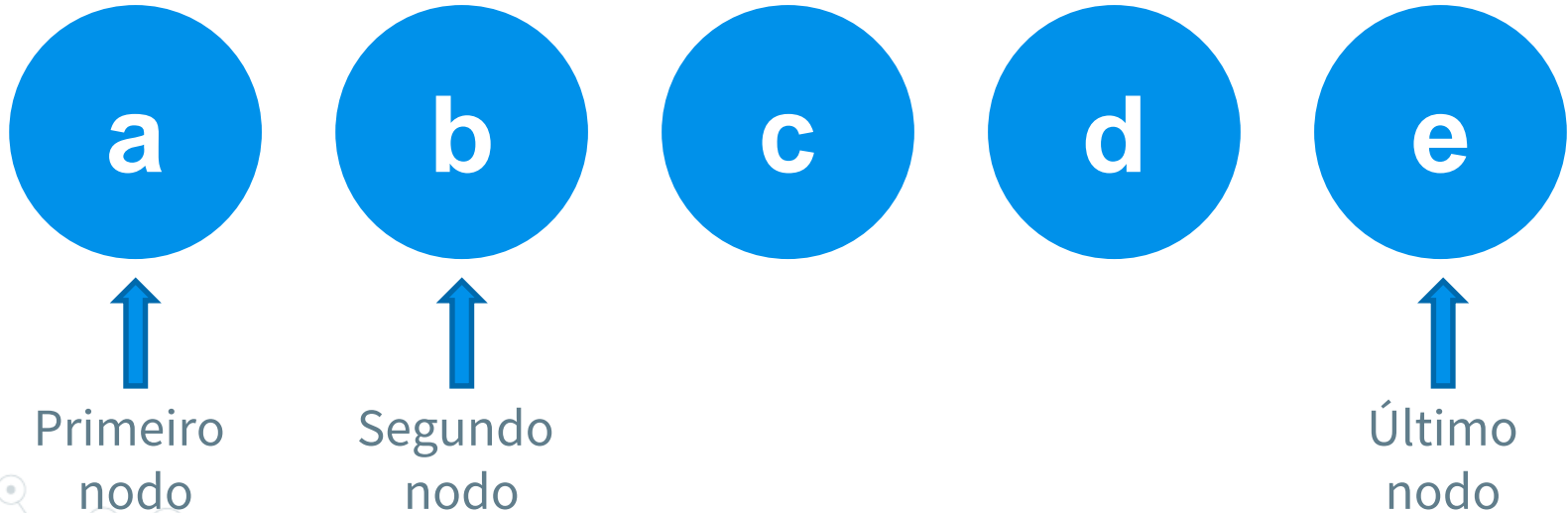
Download: <https://netbeans.apache.org/download/index.html>

2. Listas Lineares

2.1. Listas Lineares: Conceito

- ◎ Lista Linear é um conjunto de **elementos de mesmo tipo**, denominados nodos, entre os quais existe uma relação de ordem linear (ou total).
- ◎ Toda lista linear apresenta um nodo que encabeça a lista, que é o primeiro nodo da lista. A partir deste segue uma sequência de nodos, conforme a ordem definida entre eles, até o último nodo.

2.1. Listas Lineares: Conceito



2.2. Listas Lineares: Operações básicas

- ◎ **Criação de uma lista** – a primeira operação a ser executada, através da qual são alocadas as variáveis necessárias para a definição da lista e inicializadas suas variáveis de controle. As demais operações ficam habilitadas somente depois da execução desta operação;

2.2. Listas Lineares: Operações básicas

- ◎ **Inserção de um nodo** – é a maneira de formar a lista, inserindo os nodos um a um. A inserção de um nodo pode ser feita no início da lista, no final da lista, ou em alguma posição dentro da lista;
- ◎ **Exclusão de um nodo** – é a forma de excluir um nodo da lista. A exclusão também pode ser feita no início da lista, no final da lista, ou em alguma posição no meio da lista;

2.2. Listas Lineares: Operações básicas

- ◎ **Acesso a um nodo** – para consulta ou alteração de seus valores internos. O nodo pode ser identificado por sua posição na lista ou através de alguma informação que ele contém;
- ◎ **Destruição de uma lista** – operação executada quando uma lista existente não é mais necessária. A lista não poderá mais ser utilizada depois da execução desta operação.

3. Listas Lineares Sequenciais

3.1. Listas Lineares Sequenciais

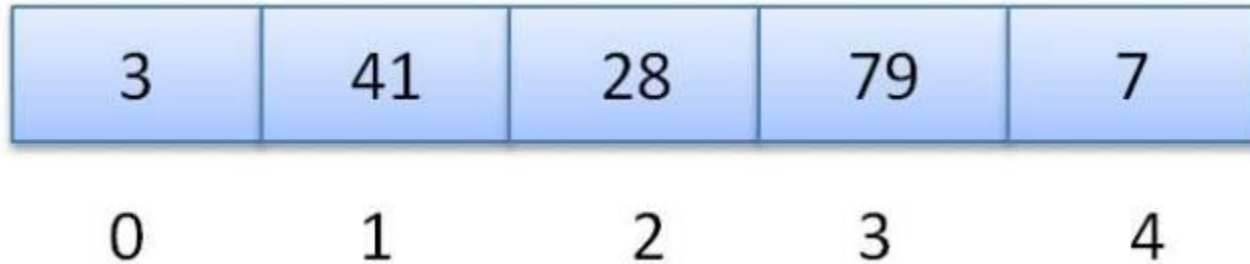
- ◎ Listas lineares implementadas através de contiguidade física utilizam a sequencialidade da memória do computador para representar a ordem dos nodos na lista.
- ◎ A forma mais usual de implementar uma lista linear através de contiguidade física é através da utilização de um arranjo de uma dimensão (**vetor**).

3.1. Listas Lineares Sequenciais

- ◎ A possibilidade de acessar diretamente um nodo é uma característica importante deste tipo de implementação, pois não é necessário percorrer toda a lista, a partir de seu início, para que um determinado nodo seja acessado.

3.2. Vetores: Conceito

- ◎ Um vetor (ou array) é a estrutura de dados mais simples que existe.
- ◎ Um vetor armazena uma sequência de valores onde todos são do mesmo tipo.



3.2. Vetores: Exemplo de aplicação

◎ Armazenamento de temperaturas:

- Declarando uma variável para cada dia:

```
double tempDia001 = 31.3;  
double tempDia002 = 32;  
double tempDia003 = 30;  
double tempDia004 = 28;  
double tempDia005 = 33;
```

3.2. Vetores: Exemplo de aplicação

◎ Armazenamento de temperaturas:

- Utilizando vetores:

```
double[] temperaturas = new double[365];  
temperaturas[0] = 31.3;  
temperaturas[1] = 30.2;  
temperaturas[2] = 28.5;  
temperaturas[3] = 34.8;  
temperaturas[4] = 27.4;
```

3.2. Vetores: Exemplo de aplicação

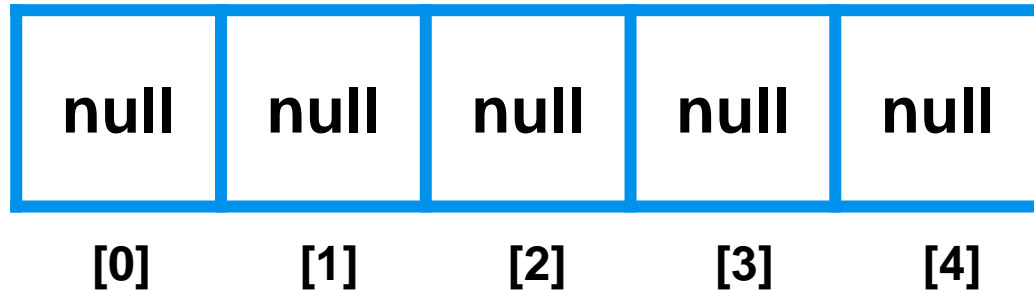
```
double[] temperaturas = new double[365];  
temperaturas[0] = 31.3;  
temperaturas[1] = 30.2;  
temperaturas[2] = 28.5;  
temperaturas[3] = 34.8;  
temperaturas[4] = 27.4;
```

31.3	30.2	28.5	34.8	27.4			
[0]	[1]	[2]	[3]	[4]	[5]	[...]	[364]

3.2. Vetores: Declaração

- ⦿ Na declaração de um array, cada elemento recebe um valor padrão, sendo 0 (zero) para números de tipo primitivo, falso (false) para elementos booleanos e nulo (null) para referências.

3.3. Criando uma Lista Linear vazia



3.3. Criando uma Lista Linear vazia

```
public class Vetor {  
    //CRIANDO O ATRIBUTO ELEMENTOS - Elementos do vetor  
    private String[] elementos;  
  
    //CRIANDO O CONSTRUTOR  
    public Vetor(int capacidade){  
        this.elementos = new String[capacidade];  
    }  
}
```

3.4. Inserindo um elemento

- ⦿ Verificar qual é o primeiro elemento nulo do vetor e inserir o novo elemento.

É NULO?

1	3	null	null	null
[0]	[1]	[2]	[3]	[4]

3.4. Inserindo um elemento

- ⦿ Verificar qual é o primeiro elemento nulo do vetor e inserir o novo elemento.

É NULO?

1	3	null	null	null
[0]	[1]	[2]	[3]	[4]

3.4. Inserindo um elemento

- ⦿ Verificar qual é o primeiro elemento nulo do vetor e inserir o novo elemento.

É NULO?

1	3	null	null	null
[0]	[1]	[2]	[3]	[4]

3.4. Inserindo um elemento

- ⦿ Verificar qual é o primeiro elemento nulo do vetor e inserir o novo elemento.

INSERE

1	3	10	null	null
[0]	[1]	[2]	[3]	[4]

3.5. Propriedade length

- ◎ Cada array conhece seu próprio tamanho;
- ◎ O tamanho é armazenado na variável de instância length;

3.6. Inserindo um elemento

```
//ADICIONAR ELEMENTO NO VETOR
public void adiciona(String elemento){
    //verificando quais elementos estão nulos,
    for (int i=0; i<this.elementos.length; i++){
        //se estiver nulo, pode adicionar nessa posição
        if(this.elementos[i] == null){
            //atribui o valor do parametro na posição do vetor
            this.elementos[i] = elemento;
            //para após encontrar o primeiro elemento nulo
            break;
        }
    }
}
```

3.6. Inserindo um elemento

- ⦿ E se o vetor for muito grande? Imagine um vetor de tamanho 500.
- ⦿ Teríamos que verificar todas as posições?
- ⦿ Esse algoritmo é eficiente?



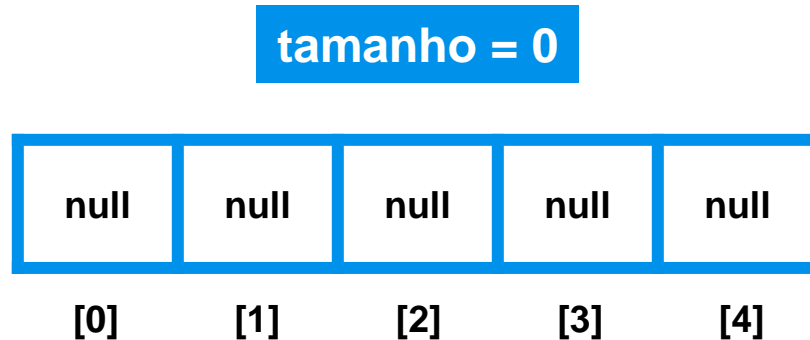
3.6. Inserindo um elemento

- © E se tivéssemos um atributo para guardar o **tamanho** do nosso vetor? Isso ajudaria nas novas inserções?



3.7. Inserindo um elemento - OTIMIZADO

- Quando o vetor está vazio, tamanho = 0. Pois todos os índices estão vazios.



3.7. Inserindo um elemento - OTIMIZADO

- Após a inserção de um elemento(**3**), o atributo tamanho é incrementado: **tamanho++**

3	null	null	null	null
[0]	[1]	[2]	[3]	[4]

tamanho = 1

3.7. Inserindo um elemento - OTIMIZADO

- © A partir de agora, antes de inserirmos um elemento(**100**), devemos verificar o valor do atributo **tamanho** e inserir o novo elemento do vetor no índice que tem o valor igual a **tamanho**.

3	null	null	null	null
[0]	[1]	[2]	[3]	[4]

tamanho = 1

Onde será inserido o novo elemento de valor 100?

3.7. Inserindo um elemento - OTIMIZADO

- © E assim por diante. A inserção é feita na posição correspondente ao valor do atributo tamanho. Vamos inserir o valor **25** agora.

1	100	null	null	null
[0]	[1]	[2]	[3]	[4]

tamanho = 2

3.7. Inserindo um elemento - OTIMIZADO

- Assim não precisamos percorrer todo o vetor para encontrar a primeira posição cujo valor é vazio. Basta ir direto na posição indicada pelo atributo **tamanho**.

1	100	25	null	null
[0]	[1]	[2]	[3]	[4]

tamanho = 3

3.7. Inserindo um elemento - OTIMIZADO

```
//ADICIONAR ELEMENTO NO VETOR - M E L H O R A D A
public boolean adiciona(String elemento){
    //verificar se o tamanho é menor que a capacidade do vetor
    if (this.tamanho < this.elementos.length){
        //adicionando o elemento novo na posição TAMANHO
        this.elementos[this.tamanho] = elemento;
        this.tamanho++;
        return true;
    }
    return false;
}
```

**CALMA,
RESPIRA!**



Obrigado!

Perguntas?



heraldo.junior@ifsertao-pe.edu.br



heraldolimajr.com.br