



**INSTITUTO FEDERAL**  
Sertão Pernambucano  
Campus Salgueiro

# Estrutura de Dados e Algoritmos com Java

**Prof. Heraldo Gonçalves Lima Junior**

[heraldo.junior@ifsertao-pe.edu.br](mailto:heraldo.junior@ifsertao-pe.edu.br)

[heraldolimajr.com.br](http://heraldolimajr.com.br)

A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines. The nodes are represented by small circles, some of which are larger and have concentric circles, suggesting different levels or types of nodes. The lines are thin and gray, connecting the nodes in a non-linear fashion.

# 1. Apresentação

## 1.1. Ementa (60h)

- Contextualização
- Tipos de Dados
- Arranjos
- Algoritmos de ordenação e busca
- Listas Simples, Encadeadas e Circulares
- Filas
- Pilhas
- Recursão
- Árvores Binárias
- Tabela hash
- Grafos

## 1.2. Bibliografia Básica

- © GOODRICH, M. T.; TAMASSIA, R. **Estruturas de Dados e Algoritmos em Java**. 5. ed. Porto Alegre: Bookman, 2013.
- © SCHILDT, Herbert. **C completo e total**. 3. ed. São Paulo: Pearson Makron Books, 2010. 827p.
- © FERRARI, R.; RIBEIRO, M. X; DIAS, R. L.; FALVO, M. **Estruturas de Dados com Jogos**. Rio de Janeiro: Elsevier, 2014.

## 1.3. Bibliografia Complementar

- ◎ BORGES, L. E. Python para desenvolvedores. São Paulo, SP: Novatec, 2014.
- ◎ DEITEL, Paul J; DEITEL, Paul J. Java: como programar. 8. ed São Paulo: Prentice Hall, 2010. 1144p.
- ◎ DROZDEK, A. Estrutura de Dados e Algoritmos em C++. São Paulo: Cengage, 2002.

## 1.4. Avaliações

- ◎ **AV1:** Listas Lineares (xx/xx/xxxx) - 2,0
- ◎ **AV2:** Listas Lineares (xx/xx/xxxx) - 2,0
- ◎ **AV3:** Listas Lineares (xx/xx/xxxx) - 2,0
- ◎ **AV4:** Listas Lineares (xx/xx/xxxx) - 2,0
- ◎ **Listas de Exercícios:** 4 listas valendo 0,5 cada.

## 1.5. Pré-requisitos Básicos

- ◎ Lógica de Programação
- ◎ Java Básico



A decorative network diagram in the top-left corner, featuring a complex web of interconnected nodes and lines, with some nodes highlighted in blue.

# 2. Conceitos Básicos



## 2.1. Tipos de Dados

- ◎ Um **TIPO DE DADO** consiste da definição do conjunto de valores que uma variável pode assumir durante a execução de um programa e das operações que podem ser aplicadas sobre ele.
- ◎ Ex: Tipo de dado Inteiro.
  - **Valores:**  $\langle \dots, -2, -1, 0, 1, 2, \dots \rangle$
  - **Operações possíveis:** soma, subtração, multiplicação, divisão, entre outras

## 2.1. Tipos Abstratos de Dados - TAD

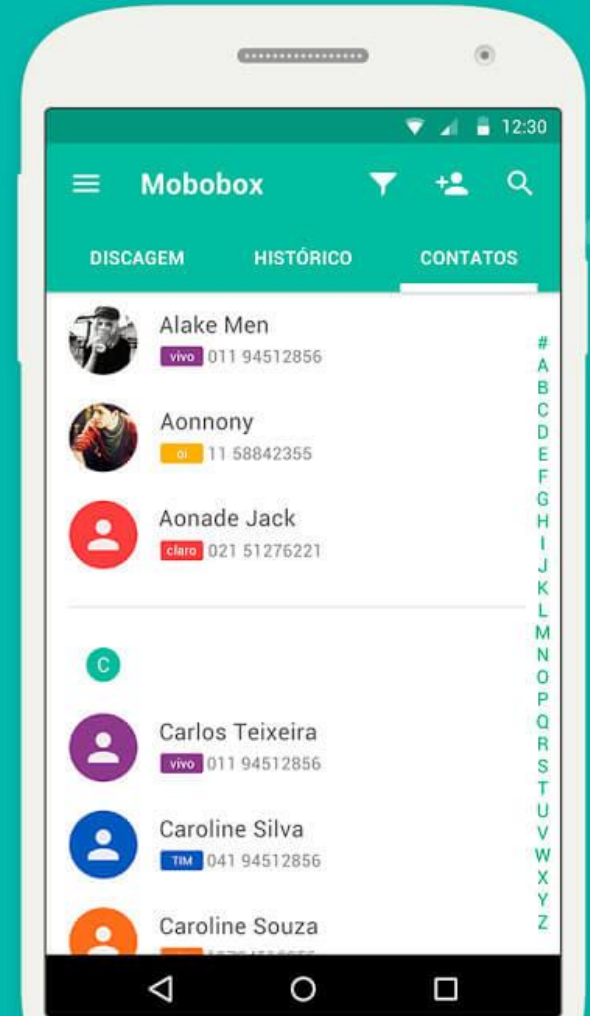
- Os tipos abstratos de dados (TADs) são estruturas de dados capazes de representar os tipos de dados que não foram previstos no núcleo das linguagens de programação e que, normalmente, são necessários para aplicações específicas. Essas estruturas são divididas em duas partes: os **dados** e as **operações**.

## 2.1. O que é uma Estrutura de Dados?

- © É uma estrutura que armazena e organiza dados/informações de modo que os dados possam ser acessados e manipulados de forma **eficiente**.
- © O ideal é que a estrutura de dados seja o mais independente possível dos dados que ela vai armazenar.

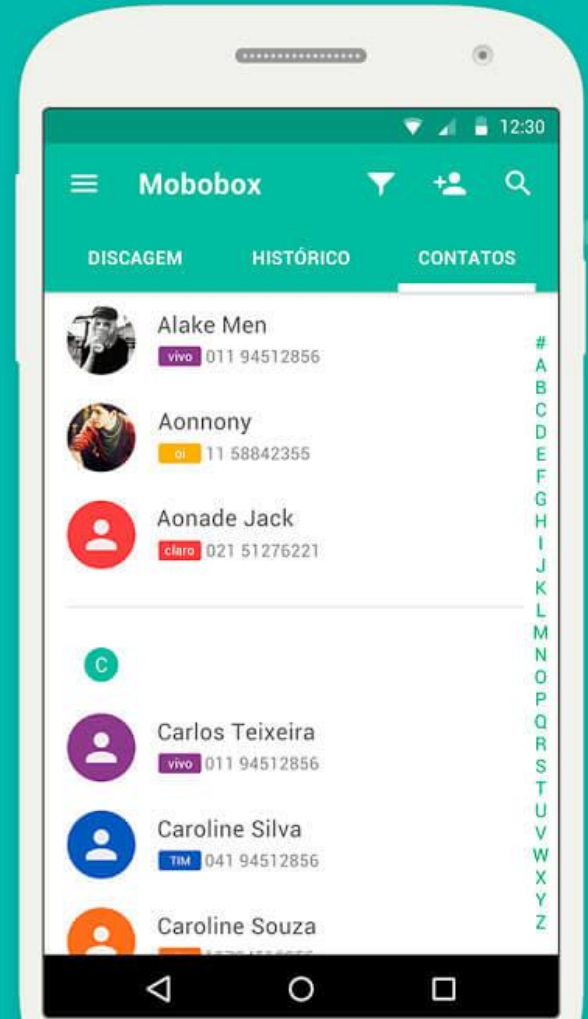
## 2.2. Exemplo

- ◎ **Agenda de contatos do celular:**
  - Quais as principais tarefas dessa aplicação?



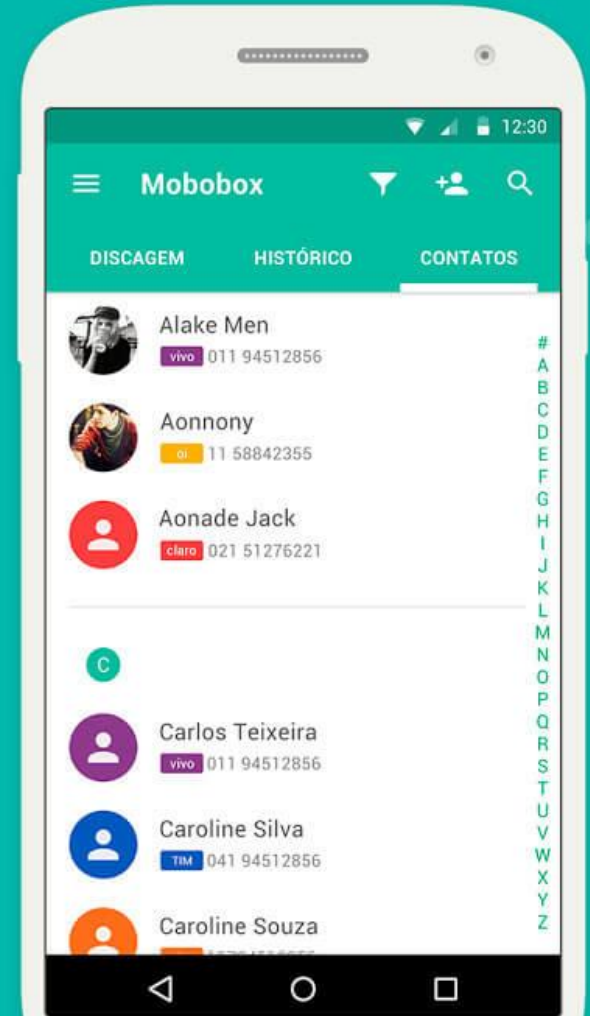
## 2.2. Exemplo

**1.** Definir **como** as **informações** dos contatos serão armazenadas.



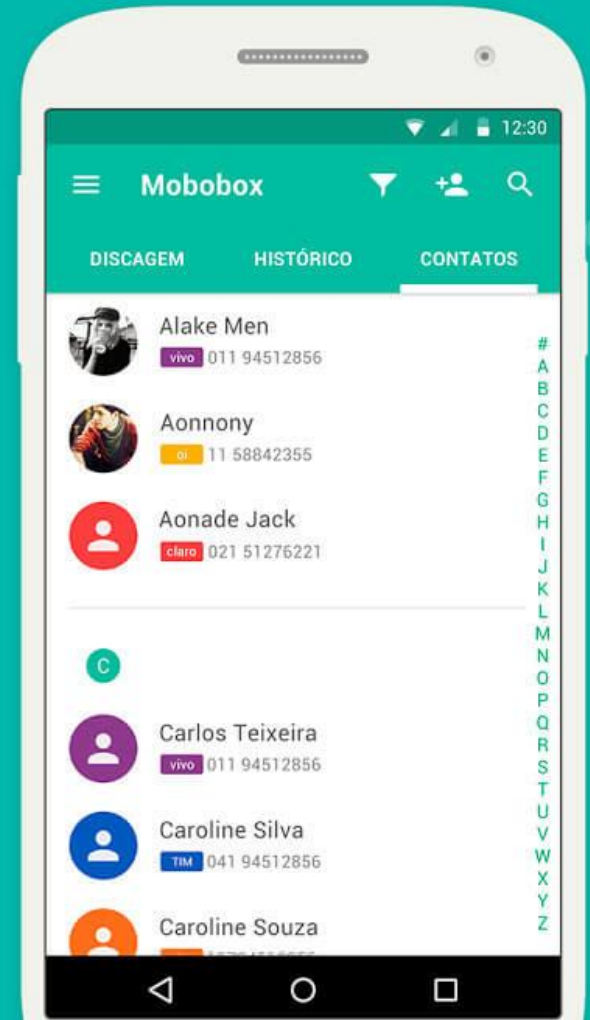
## 2.2. Exemplo

**2.** Disponibilizar **operações** para criar, recuperar, ordenar, atualizar e remover contatos.



## 2.3. Exemplo

**3.** A única coisa que precisa ser mostrada para o usuário são as operações que ele pode fazer na agenda (**interface**).



## 2.4. Alternativas de representação física

- Um aspecto que deve ser considerado na análise das estruturas de dados a serem estudadas é a forma utilizada para representar fisicamente os relacionamentos lógicos entre os dados.

**CONTIGUIDADE  
FÍSICA**

**ENCADEAMENTO**



## 2.5. Contiguidade Física - Vantagens

◎ Os dados são armazenados em posições contíguas na memória.

◎ **Vantagens:**

- **Proteção de memória** – a alocação é feita antes do início da execução do programa, garantindo a proteção da memória;
- **Transferência de dados** – como todos os dados estão alocados em bloco, a transferência de dados entre memória principal e secundária fica facilitada;

## 2.5. Contiguidade Física - Vantagens

- **Estruturas simples** – é apropriada para armazenar estrutura simples, principalmente aquelas que utilizam a ordem física em sua representação;
- **Acesso** – qualquer nodo pode ser diretamente acessado a qualquer momento, através de um índice associado à sua posição.

## 2.5. Contiguidade Física - Desvantagens

### ⦿ Desvantagens:

- **Previsão de espaço físico** – é necessário definir, antes da execução da aplicação, o número máximo de nodos a serem alocados. Isto pode constituir um grave problema quando não for possível estimar, a priori, o número de nodos que a estrutura vai apresentar;

## 2.5. Contiguidade Física - Desvantagens

- **Estruturas complexas** – não é apropriado para estruturas complexas, devido à natureza sequencial;
- **Inserção e exclusão de componentes** – estas duas operações geralmente implicam no deslocamento de um número considerável de informações.

## 2.6. Encadeamento

- ◎ O espaço necessário para a representação dos dados pode ser alocado à medida que se torne necessário, através de **alocação dinâmica**. Sempre que um espaço de memória para armazenar um dado seja necessário, este é solicitado pela aplicação, e alocado pelo gerenciador de memória em algum espaço livre da memória, sendo seu endereço devolvido em uma variável especial (**ponteiros**).

## 2.6. Encadeamento

- ◎ Uma estrutura armazenada através de encadeamento apresenta seus **nodos alocados em posições aleatórias na memória**, e não lado a lado, como na representação por contiguidade física.
- ◎ Para percorrer a estrutura é necessário que os nodos, além do(s) campo(s) de informação, apresentem algum campo adicional, onde é colocado o **endereço físico do próximo nodo**.

## 2.6. Encadeamento - Vantagens

- ◎ **Maleabilidade** – a alocação e a liberação de memória feita de forma dinâmica favorece a maleabilidade dos programas;
- ◎ **Facilidade para inserção e remoção de componente** – a inserção e a remoção de nodos é facilmente realizada, sendo somente necessário ajustar os campos de elo dos nodos envolvidos na operação, sem a necessidade de deslocamento de informações.

## 2.6. Encadeamento - Desvantagens

- ◎ **Gerência de memória mais onerosa** – toda a manipulação da estrutura é feita através de alocação e/ou liberação de memória, o que deve ser realizado pelo gerenciador de memória;
- ◎ **Procedimentos menos intuitivos** – a utilização de alocação dinâmica faz com que os procedimentos escritos para as operações sobre os dados sejam mais complexos;



## 2.6. Encadeamento - Desvantagens

- ◎ **Acesso** – o processamento dos dados encadeados deve ser feito de forma serial, isto é, um nodo deve ser sempre acessado a partir de outro acessado anteriormente. Não é possível, como nos casos dos arranjos, acessar qualquer nodo a qualquer momento, a menos que todos os endereços sejam armazenados individualmente.

**CALMA,  
RESPIRA!**



# Obrigado!

## Perguntas?



heraldo.junior@ifsertao-pe.edu.br



heraldolimajr.com.br