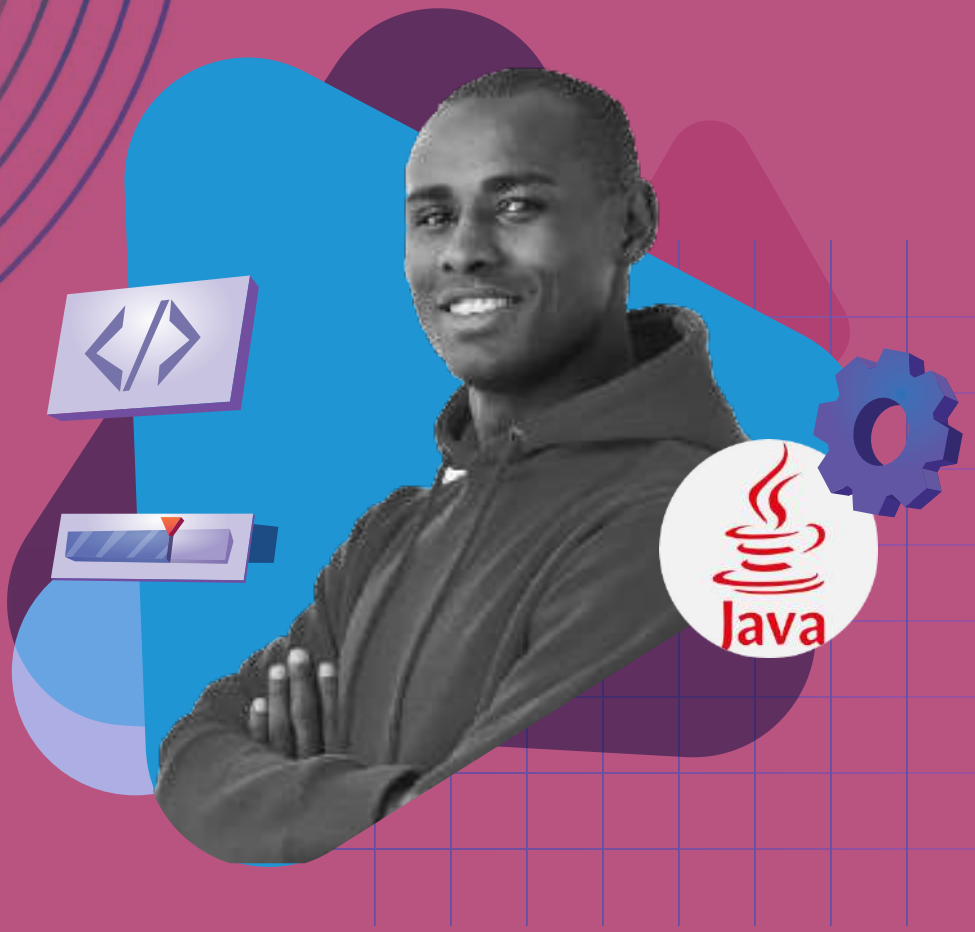
Google olabi

<ESTRUTURADEDADOS>



Estrutura de Dados

Com a estrutura de dados, encontramos maneiras de tornar o acesso aos dados mais eficiente. Ao lidar com estas estruturas, não nos concentramos apenas em um dado, mas em diferentes conjuntos de dados e em como eles podem se relacionar uns com os outros de maneira organizada.

Nos tópicos de hoje, falaremos sobre estruturas de dados lineares dinâmicas, nas quais, ao contrário de estruturas baseadas em arrays, os objetos são criados e removidos sob demanda.

Array List

A classe ArrayList é um array redimensionável que pode ser encontrado no pacote java.util.

A diferença entre um array e um ArrayList é que o tamanho de um array não pode ser modificado (se você quiser adicionar ou remover elementos de/para um array, você tem que criar um novo), enquanto os elementos podem ser adicionados e removidos de uma ArrayList sempre que você quiser. A sintaxe também é ligeiramente diferente:

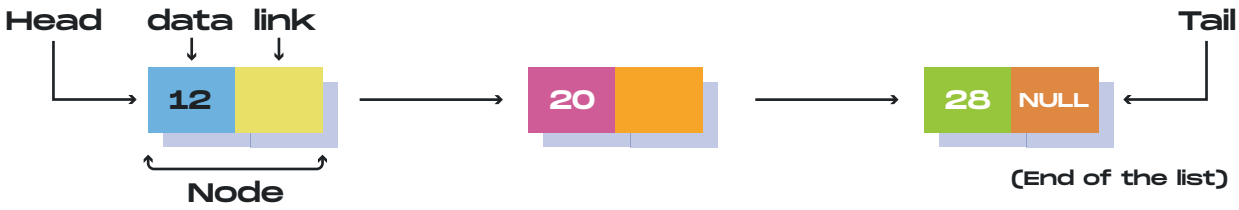
```
import java.util.ArrayList;

ArrayList<String> nome = new ArrayList<String>();
```

Lista vinculada

Lista vinculada, ou LinkedList, é uma estrutura de dados linear dinâmica, na qual os elementos não são armazenados em locais de memória contíguos. Apresentam uma coleção de objetos encadeados chamados “nós” que possuem referências para seus vizinhos, sendo armazenados aleatoriamente na memória.

Os nós guardam a informação que queremos manipular e as referências para seus vizinhos, que são do mesmo tipo. Estes apresentam dois campos, ou seja, dados armazenados naquele endereço em particular e o ponteiro que contém o endereço do próximo nó na memória, sendo que o último nó da lista contém o ponteiro para o nulo.



A sintaxe de listas vinculadas está apresentada a seguir:

```
import java.util.LinkedList;

LinkedList<String> nome = new LinkedList<String>();
```

{ ArrayList vs. LinkedList

A classe LinkedList é uma coleção que pode conter muitos objetos do mesmo tipo, assim como a ArrayList. Ambas implementam a interface da lista que permite adicionar itens, alterar itens, remover itens e limpar a lista. Entretanto, enquanto a classe ArrayList e a LinkedList podem ser usadas da mesma forma, elas são construídas de maneira muito diferente.

ArrayList	LinkedList
A classe ArrayList tem uma matriz regular dentro dela. Quando um elemento é adicionado, ele é colocado dentro da matriz. Se o array não for suficientemente grande, um novo array maior é criado para substituir o antigo e o antigo é removido.	A LinkedList armazena seus itens em “caixas". A lista tem um link para a primeira caixa e cada caixa tem um link para a próxima caixa da lista. Para adicionar um elemento à lista, o elemento é colocado em uma nova caixa e essa caixa é ligada a uma das outras caixas da lista.

{ Métodos

Listas possuem vários métodos para fazer determinadas operações de forma mais eficiente:

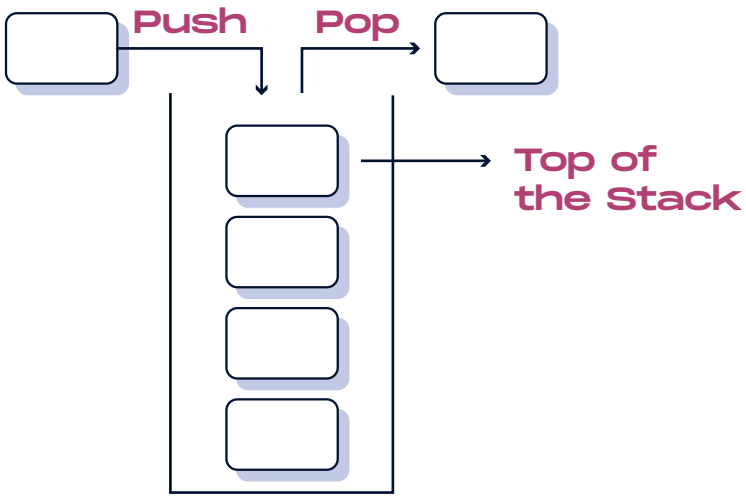
addFirst()	Adiciona um item ao início da lista
addLast()	Acrescenta um item ao final da lista
removeFirst()	Remove um item do início da lista
removeLast()	Remover um item do final da lista
getFirst()	Obtém o item do início da lista
getLast()	Obtém o item do final da lista

{ Stack

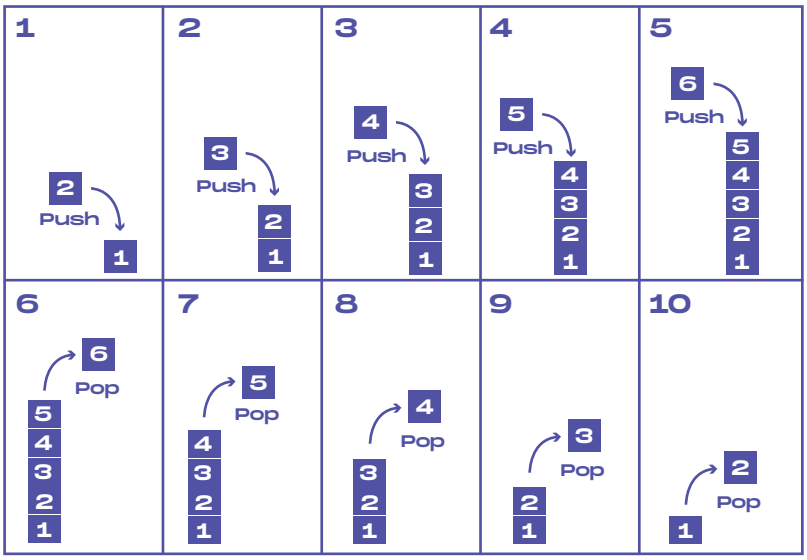
Stack, ou pilha, é uma estrutura de dados linear que é usada para armazenar a coleção de objetos. Ela é baseada no Last-In-First-Out (LIFO), ou seja, o último elemento que é armazenado é obrigatoriamente o primeiro elemento a ser retirado.

Uma maneira fácil para explicar como essa estrutura de dados funciona é compará-la à uma pilha de pratos, onde o primeiro prato a entrar na pilha será o último a sair (LIFO: Last In First Out).

Os dois métodos fundamentais associados a uma estrutura de pilha devem permitir adicionar um elemento no topo da pilha (**push()**) e recuperar/remover o elemento no topo da pilha (**pop()**).



Outras funcionalidades associadas a esta classe inclui os métodos:
- **peek()** - qual o objeto que está no topo da pilha?
- **search()** - a que profundidade encontra-se o objeto especificado?
- **empty()** - pilha está vazia?



{ Empty Stack

Empty Stack ou pilha vazia pode ser definida quando a pilha não apresenta nenhum elemento. Nesse caso, o valor da variável superior é -1.

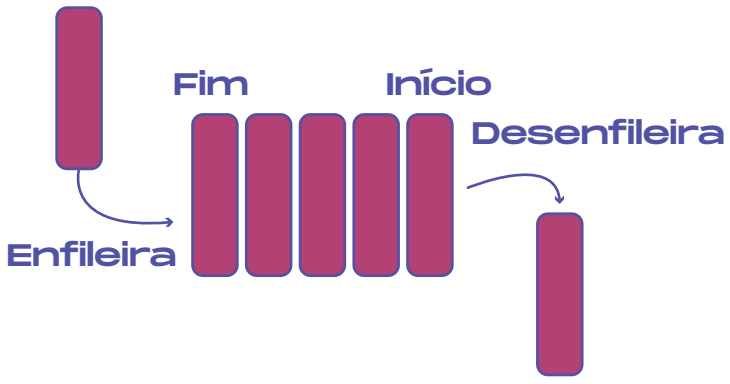
Diferentes valores do topo (top value):

A classe Number vem com vários métodos que são úteis para realizar operações em números.

Top value	Significado
−1	Mostra que a pilha está vazia
0	A pilha tem apenas um elemento
N−1	A pilha está cheia
N	A pilha está transbordando

Queue

Queue, ou fila, é uma estrutura de dados que ordena o elemento de forma FIFO (First-In-First-Out). No FIFO, o primeiro item da fila será o primeiro a ser consumido. Ao ser consumido, é removido do grupo de itens, passando então o segundo item para primeiro, terceiro para segundo e assim por diante, até o final da fila.
Uma analogia que podemos fazer com esse algoritmo, por exemplo, é uma fila de supermercado, onde as pessoas são atendidas por ordem de chegada (FIFO: First In First Out).
Filas normalmente tem apenas dois métodos, sendo um para adição de novos itens chamado de enqueue, e outro para remoção de um item chamado de dequeue.



Método	Descrição
add()	Insere um elemento na fila; retorna uma exceção quando não há mais adições possíveis na fila
offer()	Insere um elemento na fila; retorna true ou false se a adição for feita.
remove()	recuperar e remover o primeiro elemento da fila.
poll()	retorna e remove o primeiro elemento da fila, ou retorna nulo se estiver vazia.
element()	retorna o primeiro elemento da fila sem removê–lo, ou retorna uma exceção se estiver vazia.
peek()	retorna o primeiro elemento da fila, sem removê–lo, ou retorna nulo se estiver vazia.

Referências:

LinkedList
https://www.javatpoint.com/singly-linked-list
https://www.w3schools.com/java/java_linkedlist.asp
https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html
https://www.geeksforgeeks.org/data-structures/linked-list/

Stack
https://docs.oracle.com/javase/7/docs/api/java/util/Stack.html4
https://www.javatpoint.com/java-stack
https://www.geeksforgeeks.org/stack-class-in-java/

Queue
https://docs.oracle.com/javase/7/docs/api/java/util/Queue.html
https://www.javatpoint.com/java-priorityqueue
https://www.geeksforgeeks.org/queue-interface-java/

Google olabi

<ESTRUTURADEDADOS>

/exercícios/

1 ➔ Crie um objeto chamado Cliente com os atributos: id, nome, idade, telefone. Faça um programa para solicitar os dados de vários clientes e armazenar em um ArrayList até que se digite um número de ID negativo. Em seguida, exiba os dados de todos os clientes via SystemOut, formatando cada objeto separado por linhas.

2 ➔ Fazer um programa que trate o seguinte menu:

1 - Empilhar elemento
2 - Desempilhar elemento
3 - Mostrar o topo
4 - Imprimir tudo zerando a pilha
5 - Sair

3 ➔ Adicione os números 100, 20, 200, 30, 80, 40, 100, 200 a um LinkedList, percorra todos os elementos utilizando um Iterator e calcule a média.

4 ➔ Escreva um programa com Queue que simule o controle de uma pista de decolagem de aviões em um aeroporto. Neste programa, o usuário deve ser capaz de realizar as seguintes tarefas:
a) Listar o número de aviões aguardando na fila de decolagem;
b) Autorizar a decolagem do primeiro avião da fila;
c) Adicionar um avião à fila de espera;
d) Listar todos os aviões na fila de espera;
e) Listar as características do primeiro avião da fila.
Considere que os aviões possuem um nome e um número inteiro como identificador. Adicione outras características conforme achar necessário.