Algoritmos e Estruturas de Dados

Caros alunos, as videoaulas desta disciplina encontram-se no AVA (Ambiente Virtual de Aprendizagem).

UniFil

Unidade 3

Estruturas de Fila e Pilha

UniFil

Introdução da Unidade

Olá aluno, nesta unidade, no sentido de evoluir ainda mais o nosso conteúdo, vamos aprofundar um pouco mais o desenvolvimento de algoritmos estruturados utilizando como recurso o objeto *Array*.

Vamos aprender e identificar novas estruturas de dados e assim detectar novas formas de propor soluções de problemas.

Será apresentado nesta Unidade de Ensino o conceito de Listas, que a partir desse entendimento macro, prosseguirá com o estudo de estruturas de dados clássicas (tais como: (FILA E PILHA).

Os algoritmos de fila e pilha são rotineiramente utilizados em todo cenário organizacional, por exemplo, em rotinas de processos bancários, atendimentos na área de saúde ou área de vendas, presenciais ou não.

Também podemos ver exemplos de execuções dos processos de fila e pilha no meio corporativo e no meio doméstico. No meio corporativo, podemos ver algoritmos implementados nos processos logísticos de um produto, já em tarefas comuns no meio doméstico, pode-se observar rotinas realizadas por cada pessoa, seja em casa, no trabalho ou na escola.

O que vamos estudar realmente, é como implementar a ideia desses algoritmos para solucionar os problemas em sistemas computacionais. Como exemplo, é possível desenvolver um algoritmo para ser aplicado no controle de contêineres para o porto de Paranaguá. Outra aplicação, seria o simples controle de uma fila para atendimentos médicos, porém considerando as especialidades procuradas e não somente uma fila geral. Isso é o que chamados de tabela de espalhamento, também conhecida como tabela *hash*.

Os conhecimentos prévios para esta unidade são os conteúdos que foram abordados nas Unidades 1 e 2 dessa disciplina. Os conteúdos sobre *Array*s, algoritmos de busca e ordenação de elementos são fundamentais para o estudo das estruturas de fila e de pilha da Unidade 3.

Este conteúdo, objetivando maximizar o processo de ensino e de aprendizagem, também será apresentado por meio de uma linguagem dialógica, facilitando assim a sua compreensão. Para esta unidade, a linguagem de programação Java continuará sendo utilizada no desenvolvimento do conteúdo e nos exercícios das videoaulas, mas sempre é bom lembrar, que os conteúdos estudados são aplicáveis a qualquer outra linguagem que você tenha familiaridade e queira utilizar.

Objetivos

- Conhecer e compreender a estrutura de dado Fila;
- Conhecer e compreender a estrutura de dados Pilha;
- Implementar algoritmos manipulando informações por meio da estrutura Fila;
- Implementar algoritmos manipulando informações por meio da estrutura Pilha.

Conteúdo programático

Aula 01 – Algoritmos de Fila

Aula 02 – Algoritmos de Pilha

Você poderá também assistir às videoaulas em seu celular! Basta apontar a câmera para os **QRCodes** distribuídos neste conteúdo.

Pode ser necessário instalar um aplicativo de leitura QRcode no celular e efetuar login na sua conta Gmail.

Aula 1 – Algoritmos de Fila

Introdução à estrutura de Fila

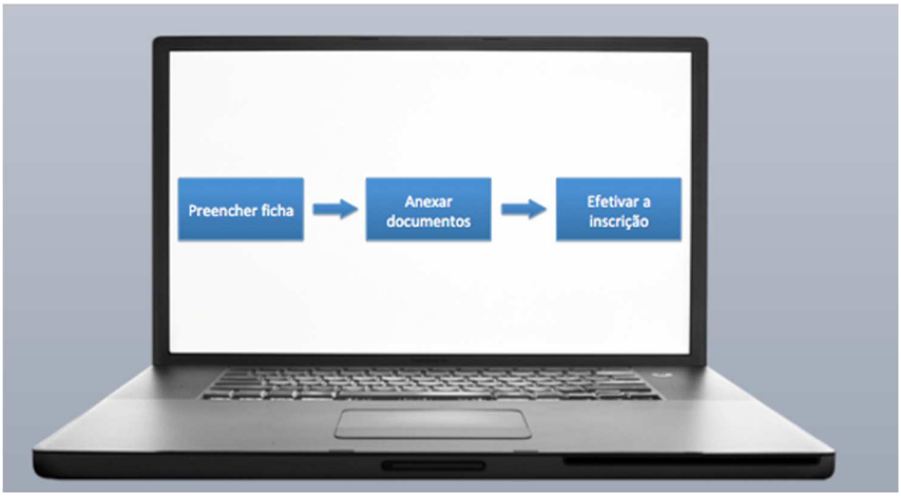
A estrutura de dados Fila, antes mesmo de ter suas características de construção e de funcionamento, é considerada uma lista. Para ilustrar o exemplo, vamos imaginar que você precisa realizar a inscrição para um curso de estrutura de dados em sua universidade. Para se inscrever no curso você precisa cumprir uma lista com alguns itens. O primeiro item é preencher uma ficha de inscrição solicitando sua participação no curso, o segundo item é anexar todos os documentos solicitados junto à ficha de inscrição e para finalizar, o terceiro item é efetivar a inscrição junto à coordenação de cursos da universidade. Observe que já estamos falando de uma lista de coisas ou itens que deverão ser cumpridas. Vamos colocar esses itens em forma de lista:

1. Preencher a ficha de inscrição;
2. Anexar os documentos necessários;
3. Efetivar a inscrição.

Existe também uma informação implícita em cada item de como cumprir o próximo, ou seja, no item 1 preencher a ficha de inscrição, já consta informações que juntamente com a ficha você precisa anexar alguns documentos necessários. Portanto, os procedimentos do próximo item estão ligados. Este tipo de lista recebe o nome de Lista Encadeada.

Para um melhor detalhamento, vamos apresentar uma imagem fazendo a ligação desses passos.

Figura 1 – Site de Pesquisa *Google*



Fonte: elaborada pelo próprio autor (2020).

A Figura 1, exibe exatamente o fluxo dos itens na ordem em que deverão acontecer, da esquerda para a direita. As setas de ligação entre os itens na realidade não existem no cenário da programação, são meramente ilustrativas, visto que foi possível representar o mesmo encadeamento lógico sem elas. Na codificação elas não existem.

Basicamente, o princípio da Fila ou da Pilha é o mesmo, pois são estruturas derivadas de uma Lista. Existem características de funcionamento e manipulação dos valores que são diferentes entre uma estrutura e outra, porém elas são iguais se analisarmos essas estruturas unicamente como: estrutura para armazenamento de valores.

Indicação de Vídeo

Veja este vídeo sobre a construção de uma Lista Encadeada. No vídeo, é implementada uma lista na prática:

Disponível em: https://www.youtube.com/watch?v=jIM87UqOq3g. Acesso em: 10 dez. 2020.

Lista Ordenada

Como o próprio nome diz são listas que estão ordenadas a partir de um critério estabelecido. Este tipo de lista possui uma particularidade importante a ser destacada, ou seja, se houver a inclusão de um novo elemento na lista, ele deverá ocupar uma posição específica de acordo com o critério estabelecido (FORBELLONE, 2005). Vamos observar alguns exemplos.

Exemplos

Exemplo 1:

Lista de números decimais em ordem crescente: {2,4,6,8,10}.

Se desejarmos incluir o número 5, ele tem obrigatoriamente que posicionar-se após o número quatro e antes do número seis, devido ao critério de números decimais em ordem crescente, ficando da seguinte forma: {2,4,**5,**6,8,10}.

Exemplo 2:

Lista de nomes em ordem alfabética: {“Ana Paula”,”Cibeli”,”Joana”}

Se desejarmos incluir o nome “Solange”, este deverá posicionar-se ao final da lista. Devido ao critério alfabético, ficando assim: {“Ana Paula”,”Cibeli”,”Joana”, **“Solange”**}.

Lista Desordenada

Uma lista desordenada, se caracteriza em um conjunto de elementos que ainda não foram ordenados, ou que estão fora de ordem propositalmente.

Curiosidades

De acordo com o Dicionário on-line de Português, a palavra **"desordenado"** significa: em que há desordem (falta de organização); que se conseguiu desordenar; que se encontra fora de ordem; desarrumado ou desorganizado; que se opõe ao que é normativo; que **não aceita regras**; de modo tumultuado. O termo aceitável para Listas Desordenadas caracteriza-se pela inexistência de regras, sendo seus elementos distribuídos de maneira desarmônica.

Fonte: disponível em: http://www.dicio.com.br. Acesso em: 10 dez. 2020.

Exemplos

Exemplo 3:

Lista de números decimais: {33, 18, 56, -1, 50}.

O Exemplo 3, apresenta uma lista numérica fora de ordem. Neste caso, a ordenação não se aplica, pois, o objetivo é exatamente classificar a lista como desordenada.

Exemplo 4:

Lista de nomes: {"Joana", "Ana Paula", "Cibeli" }.

O Exemplo 4, apresenta uma lista textual e tem as mesmas características da lista do Exemplo 3, ou seja, também está fora de ordem, pois o objetivo é classificar a lista como desordenada.

Estrutura da Fila

Nesta aula, vamos estudar a estrutura de Fila que faz parte das Listas Disciplinadas. A Fila é considerada uma lista linear disciplinada por possuir regras de manipulação de seus elementos. Essas regras podem ser de Inclusão de um novo elemento na lista, exclusão de um elemento da lista ou pesquisa e/ou modificação de um, ou vários elementos da lista (FORBELLONE, 2005).

Basicamente, a Fila adota a política FIFO (*First In First Out*), ou em português, o primeiro que entra é o primeiro que sai. Esse tipo de estrutura é bem conhecida e vivenciada por todos no cotidiano. Podemos observar uma Fila fisicamente em um balcão de atendimento, por exemplo, ou uma fila no caixa de um banco. Também precisamos

aguardar em uma fila para abastecimento em um posto de combustível e tantos outros exemplos do nosso dia a dia (FORBELLONE, 2005).

Quanto as regras de manipulação em uma Fila, temos:

- As inserções são feitas no final da fila;
- As remoções são feitas no início da fila;
- A consulta na fila é feita elemento por elemento até que ele seja ou não encontrado.

Um exemplo de uma Fila de pessoas pode ser observado na Figura 2, repare que as pessoas estão viradas para a direita, indicando o sentido da fila. Conclui-se que o senhor de bengala é o primeiro da fila e o homem de pasta o último da fila.

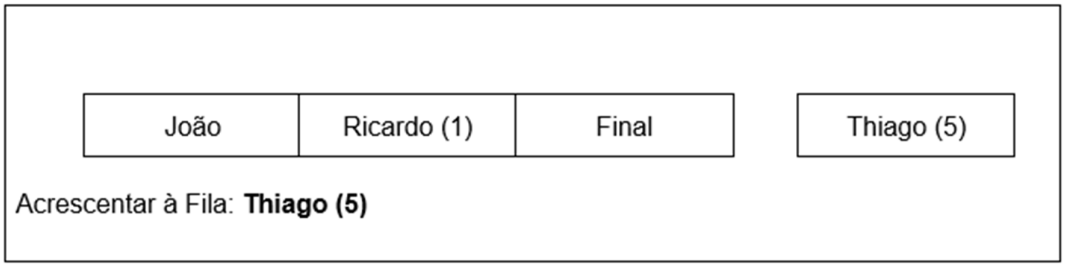
Figura 2 – Fila de pessoas

A Figura 2, ajuda a exemplificar o que acontece na regra dos algoritmos de fila. Esses algoritmos, se comportam da mesma maneira que essa fila de pessoas. Vamos detalhar um pouco mais sobre as regras de manipulação, iniciando pela inserção de elementos na fila.

Inclusão de um novo elemento na fila

Como a fila trabalha com a política FIFO (*First In First Out*), já citada nesta aula, sabemos que para inserir um novo elemento na fila, é sempre no final, o que pode ser realizado por uma variável de controle, indicando a posição do último elemento da fila (FORBELLONE, 2005).

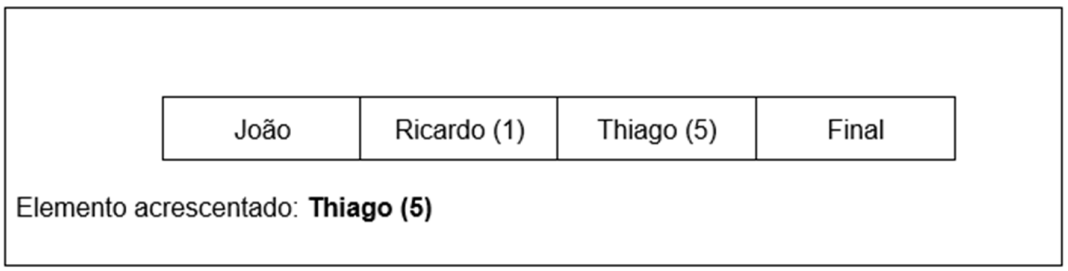
Figura 3 – Inserção



Fonte: elaborada pelo próprio autor (2020).

Na Figura 3, os elementos João e Ricardo estão na fila: {"João", "Ricardo"}, o elemento Thiago é o elemento que está aguardando para entrar na fila. Sendo assim, o Final da fila aponta para Ricardo. Temos que realizar dois processos para incluir o Thiago. O primeiro é o Ricardo apontar para Thiago, e por sua vez o próximo passo é fazer o Final, que apontava para Ricardo, apontar para Thiago.

Figura 4 – Inserção



Fonte: elaborada pelo próprio autor (2020).

Para realizar a inserção, é necessário identificar qual elemento será inserido. Vale lembrar que é preciso realizar sempre esses dois processos para entrada de elementos na fila. Em uma visão algorítmica em linha de comando observe o que foi realizado nos dois processos (FORBELLONE, 2005):

1º Passo: fila[1].prox = 5;

2º Passo: final = 5;

Videoaula 1

Nesta videoaula, o professor vai apresentar o conceito de fila, bem como sua criação e a inserção de elementos nessa estrutura. No vídeo, você verá a explicação na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Videoaula 1

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai apresentar o conceito de fila, bem como sua criação e a inserção de elementos nessa estrutura. No vídeo, você verá a explicação na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Agora que você já sabe como funciona a criação e a inserção de elementos dentro de uma fila, vamos continuar nosso estudo abordando a exclusão ou remoção de elementos na estrutura Fila.

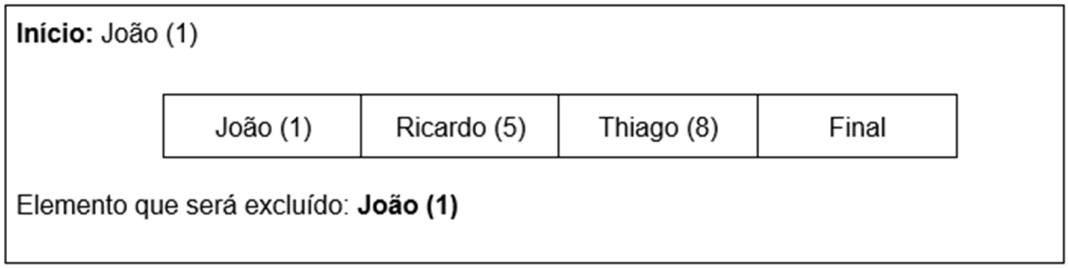
Exclusão de um novo elemento na fila

O processo de exclusão de um elemento na lista se dá de forma invertida à inserção. Enquanto para inserir um elemento na lista é necessário saber qual é o final da lista, para exclusão é preciso localizar o elemento que se encontra no início da fila (FORBELLONE, 2005).

O tipo de comportamento FIFO, como já observado, faz com essa regra seja verdadeira. Portanto a exclusão é no início da fila, da mesma maneira que uma fila convencional de pessoas. Se a pessoa chegou primeiro, ela será atendida e retirada da fila antes que todas as outras pessoas (FORBELLONE, 2005).

Vamos apresentar um exemplo de exclusão de um elemento da lista. A partir desse exemplo, será possível desenvolver um algoritmo que o coloque em prática.

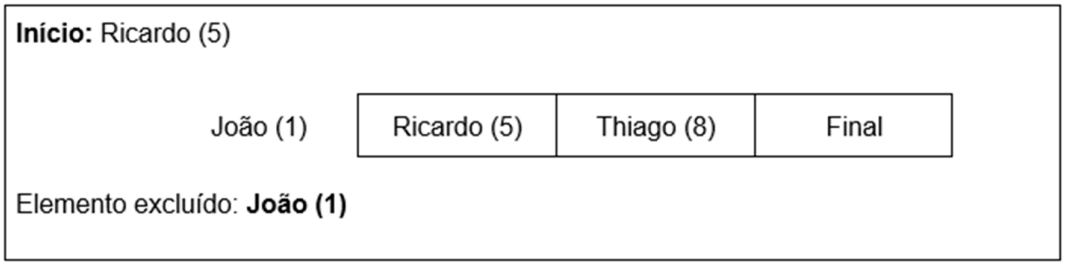
Figura 5 – Exclusão



Fonte: elaborada pelo próprio autor (2020).

Na Figura 5, os elementos João, Ricardo e Thiago estão na fila: {"João", "Ricardo", "Thiago"}. O primeiro da fila, João, será retirado, supondo que já tenha sido atendido. Após a sua retirada, o início da fila passará a ser o elemento "Ricardo". Para realizar esse processo, o apontador do início da fila deverá apontar agora para o segundo elemento da fila, Ricardo, dessa forma, retiramos João da fila. Observe a Figura 6.

Figura 6 – Exclusão



Fonte: elaborada pelo próprio autor (2020).

No algoritmo de fila não se exclui o elemento, ou seja, não se apaga o "João" da lista, simplesmente muda-se o apontador para o próximo elemento. Novamente, em uma visão algorítmica em linha de comando observe o que foi realizado na exclusão do elemento da fila (FORBELLONE, 2005):

1º Passo: inicio = fila[5].prox;

Videoaula 2

Nesta videoaula, o professor vai abordar o conceito e a prática para exclusão de elementos nessa estrutura. No vídeo, você verá a explicação na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Já abordamos a inserção e exclusão de elementos dentro de uma fila. Portanto, vamos continuar nosso estudo abordando a consulta de um elemento na estrutura.

Videoaula 2

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai abordar o conceito e a prática para exclusão de elementos nessa estrutura.

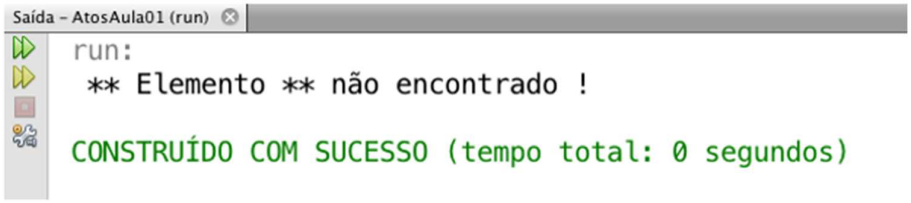
Já abordamos a inserção e exclusão de elementos dentro de uma fila. Portanto, vamos continuar nosso estudo abordando a consulta de um elemento na estrutura.

Pesquisa de um novo elemento na fila

A pesquisa de elementos na fila pode ser realizada com os algoritmos de busca abordados na unidade 2. Uma busca sequencial seria uma opção, pois as filas geralmente não são ordenadas. A ordem de uma fila genericamente é a ordem de entrada dos seus elementos.

Para uma possível busca em elementos na fila, basta varrer a fila comparando o valor de busca com os elementos da fila. Se encontrado, exibe-se o valor no programa, caso o valor não for encontrado em nenhum elemento na fila, significa que ele não existe. Logo, uma mensagem pode ser exibida ao usuário informando essa situação (FORBELLONE, 2005).

Figura 7 – Pesquisa



Fonte: elaborada pelo próprio autor (2020).

A Figura 7, exibe a mensagem quando o elemento de busca não foi encontrado em nenhum elemento da fila.

Fila vazia

Na exclusão de elementos da fila, não podemos continuar excluindo se a fila já está vazia. Para essa situação ser evitada, temos obrigatoriamente que testar se a fila tem elementos antes de tentar excluir. Observe a Figura 8.

Figura 8 – Criação da Fila

```
public class Fila {

    private List<Fila> alunos = new LinkedList<Fila>();

    public void insere(Aluno aluno) {
        // implementação
    }

    public Aluno remove() {
        // implementação
        return null;
    }

    public boolean vazia() {

        return this.alunos.size() == 0;
    }

}
```

Fonte: elaborada pelo próprio autor (2020).

A Figura 8, cria a classe Fila com os métodos: *insere()*, *remove()* e *vazia().* O método *vazia(),* retorna VERDADEIRO se a quantidade de elementos for igual a zero(0). Isso quer dizer que, enquanto o método *vazia()* retornar FALSO, pode-se remover elementos da fila, pois ela não está vazia.

Para exemplificar, uma execução desse método pode ser observada na Figura 9.

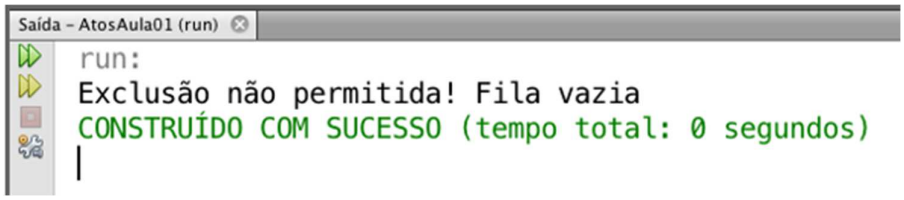
Figura 9 – Fila Vazia

```
12  class Aluno {
13
14      public static void main(String[] args) {
15
16          Fila f = new Fila();
17          boolean ret = f.vazia();
18
19
20          if(f.vazia()){
21
22              System.out.println("Exclusão não permitida! Fila vazia ");
23
24          }
25
26
27      }
28
29  }
```

Fonte: elaborada pelo próprio autor (2020).

A classe Aluno na Figura 9, instancia a classe Fila para executar a chamada do método *vazia()* presente na classe Fila. O retorno do método sendo verdadeiro, exibirá a mensagem dentro do comando de saída na linha 22, figura 10.

Figura 10 – Execução do algoritmo



Fonte: elaborada pelo próprio autor (2020).

Videoaula 3

Nesta videoaula, o professor vai abordar o conceito e a prática para os testes de fila vazia. No vídeo, você terá uma explicação na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados, além de dicas do professor para executar o código.

Videoaula 3

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai abordar o conceito e a prática para os testes de fila vazia.

Na próxima aula, será apresentada a estrutura de Pilha. Vamos abordar a criação e manipulação de elementos na Pilha, dentro de um *Array* Unidimensional.

Aula 2 – Algoritmos de Pilha

Introdução à estrutura de Pilha

Vamos iniciar nossa segunda aula falando um pouco sobre a estrutura de pilha. Assim como as filas, as pilhas são listas com uma disciplina de acesso oposto denominada LIFO (*Last In, First Out*). Isso quer dizer que quando um elemento entra na pilha, ele pode ser removido somente quando todos os outros que entraram depois dele serem removidos.

Para ilustrar uma pilha, também podemos abordar vários exemplos do cotidiano como: uma pilha de pratos, uma pilha de livros, uma pilha de roupas ou então exemplos do mercado de trabalho como: um produto que é composto por várias peças em uma linha de montagem. Um veículo, por exemplo. O processo de montagem de um veículo é constituído pelo encaixe de várias peças empilhadas.

A ordem desse empilhamento é importante, imagine fazer toda a montagem do motor e no final descobrir que ficou faltando uma peça interna. Por isso o conceito do empilhamento e desempilhamento é fundamental em algoritmos, visto que muitas etapas da montagem de um veículo são automatizadas.

Figura 1 – Execução do algoritmo

Fonte: disponível em: https://www.alamy.com/search.html?CreativeOn=1&adv=1&ag=0&all=1&creative=&et=0x00000000000000000000&vp=0&loc=0&qt=CXRFFX&qn=&lic=6&lic=1&imgt=0&archive=1&dtfr=&dtto=&hc=&selectdate=&size=0xFF&aqt=&epqt=&oqt=&nqt=&gtype=0. Acesso em: 10 dez. 2020.

A Figura 1 retrata esse cenário da linha de montagem de um veículo. Podemos também abordar a estrutura de pilha, em tarefas como: empilhar e desempilhar pratos. Uma tarefa muito comum, é armazenar vários pratos em armários de cozinha um acima do outro. Dependendo do peso do prato, só conseguimos ter acesso aos pratos que foram colocados primeiro se retirarmos os pratos que estão por cima. É exatamente esses tipos de controles que nosso algoritmo deve fazer com essa estrutura.

Na realidade, não importa o tipo de elemento que está na pilha, ou seja, se são peças da montagem de um veículo ou uma pilha de pratos. O que interessa é a maneira de manipular esses elementos na pilha.

No destaque na Figura 2, uma pilha de pratos e uma pilha de livros. É importante saber, que dentro da construção do algoritmo, não se pode simplesmente erguer os livros que estão por cima, todos de uma vez para ter acesso ao último livro da pilha. Essa lógica também vale para a pilha de pratos e para qualquer outro objeto.

Figura 2 – Pilhas de Objetos

Fonte: disponível em: https://pixabay.com/. Acesso em: 10 dez. 2020.

Todo processo de inclusão e exclusão de elementos, agora na estrutura de pilha se dará por algoritmos de empilhamento e desempilhamento dos seus elementos respectivos.

Indicação de Leitura

Para melhor compreensão deste tópico, você pode estudar o livro de FORBELLONE. O capítulo 7 “Estrutura de Dados Avançadas”, especificamente, a partir da página 155, apresenta a estrutura de Pilha abordando o processo de criação e manipulação dos elementos. Vale a pena conferir!

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. **Lógica de programação.** 2. ed. São Paulo: Makron Books, 2000, 195 p.

Na prática, vamos abordar como se dá a codificação inicial para o controle da pilha neste primeiro momento. Para tanto, devemos também aprender e compreender quais são as regras de manipulação de elementos nessa estrutura.

Indicação de Vídeo

Veja este vídeo bem interessante sobre as principais Estruturas de Dados (EDs). No vídeo, é apresentada a estrutura de pilha, contendo história, curiosidades e funcionamento. Por meio de um exemplo prático no cenário de pilha de pratos, o vídeo traz um exemplo com clareza, facilitando assim a compreensão da estrutura de dados. Pause o vídeo quantas vezes for necessário e faça anotações para que você compreenda o processo realizado.

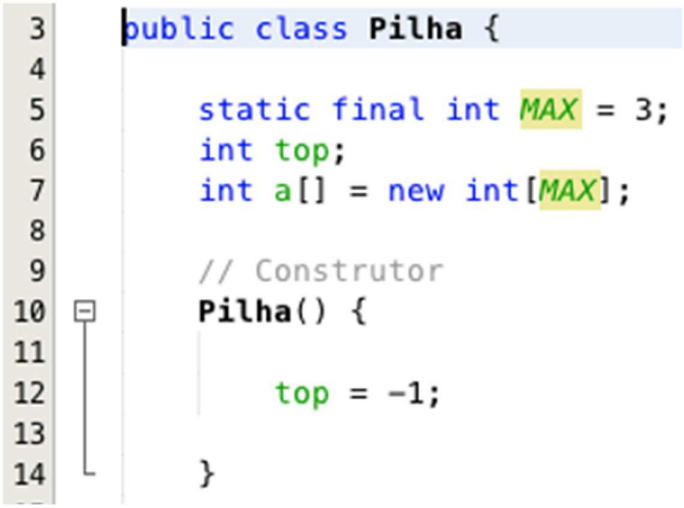
Disponível em: https://www.youtube.com/watch?v=EfF1M7myAyY. Acesso em: 30 nov. 2020.

Tal qual a estrutura de fila, a pilha também trabalha com a inserção e exclusão de elementos. Dentro da pilha damos um outro nome para este processo. Para inserir um novo elemento na pilha, se diz empilhar, e para remover um elemento existente na pilha, se diz desempilhar. Esses termos são comuns dentro da estrutura de dados.

Algumas linguagens possuem classes nativas para trabalhar com a estrutura de pilha, porém se entendermos o conceito, é possível criar as nossas próprias classes e assim, sermos proprietários do nosso código.

Em primeiro lugar, devemos criar a classe Pilha. Nela, devemos ter o método *push()*, para inserção de elementos, o método *pop()* para exclusão de elementos na fila, além da verificação quando a pilha está cheia ou vazia.

Figura 3 – Classe Pilha – variáveis e construtor

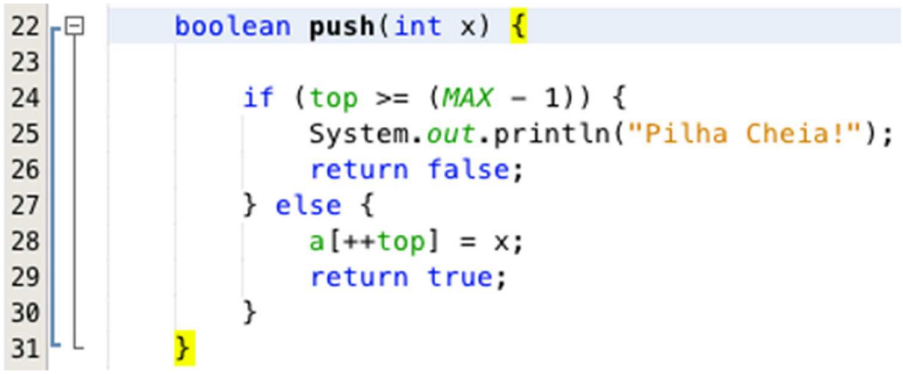


```
3   public class Pilha {
4
5       static final int MAX = 3;
6       int top;
7       int a[] = new int[MAX];
8
9       // Construtor
10      Pilha() {
11
12          top = -1;
13
14      }
```

Fonte: elaborada pelo próprio autor (2020).

A Figura 3, exibe a classe Pilha criada na sintaxe do Java. As linhas 5, 6 e 7, faz a criação de um *Array* Unidimensional, utilizado para controlar a pilha e as variáveis de controle de tamanho da pilha.

Figura 4 – Classe Pilha – método de inserção



```
22  boolean push(int x) {
23
24      if (top >= (MAX - 1)) {
25          System.out.println("Pilha Cheia!");
26          return false;
27      } else {
28          a[++top] = x;
29          return true;
30      }
31  }
```

Fonte: elaborada pelo próprio autor (2020).

A Figura 4, apresenta o método *push(),* responsável pela inserção de elementos na pilha. O método pede um valor inteiro por parâmetro e após verificar se a pilha não está cheia, o elemento é inserido no vetor a[ ], linha 28. Observe que o método é boleano, retornando verdadeiro para valor incluído e falso quando a pilha está cheia. Nessa última situação, o método apresenta uma mensagem de "**Pilha Cheia!**", linha 25.

Videoaula 1

Nesta videoaula, o professor vai abordar a criação da classe Pilha. No vídeo, será abordado a criação da classe, a criação das variáveis de controle e do *Array* Unidimensional. Também será abordado a criação do método *push()*. O professor vai explicar o código, sua funcionalidade e fará alguns testes modificando alguns parâmetros no programa. Toda a explicação será na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Vale a pena conferir mais essa Videoaula!

Videoaula 1

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai abordar a criação da classe Pilha. No vídeo, será abordado a criação da classe, a criação das variáveis de controle e do Array Unidimensional.

Dando continuidade ao nosso exemplo de Pilha, vamos agora apresentar o método *pop()* da classe Pilha, responsável por excluir elementos na mesma. Observe a Figura 5.

Figura 5 – Classe Pilha – método de exclusão

```
33    int pop() {
34        if (top < 0) {
35            System.out.println("Pilha Vazia!");
36            return 0;
37        } else {
38            int x = a[top--];
39            return x;
40        }
41    }
```

Fonte: elaborada pelo próprio autor (2020).

O método *pop()* retorna um inteiro, sendo 0 (zero) quando a pilha já estiver vazia e o valor excluído quando ele estiver no topo da pilha. A condição responsável para validar se o valor da variável “top” é menor que 0 (zero) na linha 34, faz a verificação se existe elemento a ser removido, pois a cada exclusão é decrementado o valor da variável “top”,

variável que controla a pilha. No método de inserção, já abordado na Figura 3, a mesma variável é incrementada. Essa é a lógica para controle tanto da "**Pilha Cheia**" quanto da "**Pilha Vazia**".

Na próxima videoaula, abordaremos com detalhes o método *pop()*, bem como, a verificação de pilha vazia.

Videoaula 2

Nesta videoaula, o professor vai abordar o método *pop()* da classe Pilha. No vídeo, será abordado a criação do método *pop()*. O professor vai explicar o código continuando o exemplo da videoaula anterior. Sendo assim, no final desta videoaula teremos a classe pilha com os métodos de inserção e exclusão de elementos.

A explicação será na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Vamos continuar nosso exemplo agregando informações importantes em mais esta videoaula!

Videoaula 2

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai abordar o método pop() da classe Pilha. No vídeo, será abordado a criação do método pop(). O professor vai explicar o código continuando o exemplo da videoaula anterior.

Depois de estudarmos a criação da classe Pilha e os métodos de inserção e exclusão de elementos, vamos agora focar no teste dessas funcionalidades.

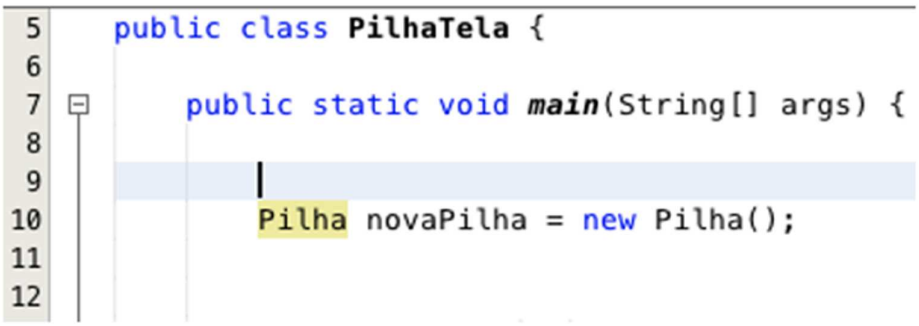
Para o teste, será necessário a criação de outra classe, ou se você preferir, a utilização da classe principal do seu projeto em Java. Vamos optar por utilizar a classe principal do projeto nesta aula. No exemplo apresentado, o nome da classe será **PilhaTela**{ }. Não é necessário criar métodos adicionais nesta classe, utilizaremos o método principal "*main()*".

Na classe "PilhaTela", vamos optar por fornecer valores fixos aleatórios para nossos testes, portanto, a leitura dos elementos via teclado por meio da classe "*Scanner*" não será utilizada.

Enfim, vamos a construção da classe "PilhaTela".

O primeiro passo é fazer a instância da classe Pilha que já está criada e onde se encontram os métodos *push()* e *pop()*.

Figura 6 – Classe "PilhaTela" – instância da classe Pilha

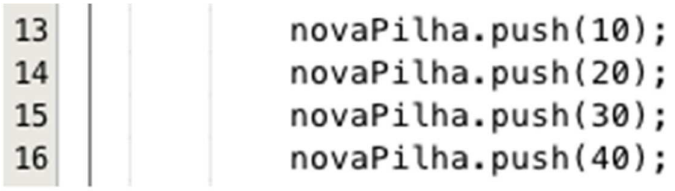


Fonte: elaborada pelo próprio autor (2020).

A Figura 6, apresenta o código da instância da classe Pilha, criando o objeto "novaPilha". A partir desta linha de código, podemos obter acesso aos métodos públicos da classe Pilha por meio dos paradigmas da Programação Orientada a Objetos.

Em nosso exemplo de teste para o método *push()*, vamos inserir quatro elementos na pilha.

Figura 6 – Classe PilhaTela – instância da classe Pilha



Fonte: elaborada pelo próprio autor (2020).

Após a execução das linhas 13 a 16, o que acontecerá com nosso exemplo? Tente fazer o teste de mesa para responder essa questão. No entanto, se você observar novamente a Figura 2, perceberá que o total de espaço em nossa pilha é de 3 elementos. Portanto, devemos tratar esse procedimento para que nosso programa não apresente uma falha na execução.

Uma das maneiras de se fazer esse tratamento é construir um procedimento que teste se a Pilha está cheia, ou seja, se ela aceita mais elementos ou não.

Figura 7 – Teste de pilha cheia

```
22      boolean push(int x) {
23
24          if (top >= (MAX - 1)) {
25              System.out.println("Pilha Cheia!");
26              return false;
27          } else {
28              a[++top] = x;
29              return true;
30          }
31      }
```

Fonte: elaborada pelo próprio autor (2020).

A Figura 7, apresenta novamente o método *push()*, utilizado para inserir elementos na pilha, porém o teste de “Pilha Cheia”, está dentro desse método. Na linha 24, existe um teste condicional que retornará “VERDADEIRO” quando a variável “top” for maior que a variável (“MAX” - 1), pois isso indica que o valor do índice será maior que o último índice existente. Por exemplo, se a Pilha tiver o tamanho de 3, o maior índice é o 2. Portanto qualquer valor maior que 2, será considerado “Pilha Cheia”, pois ele não poderá ser utilizado.

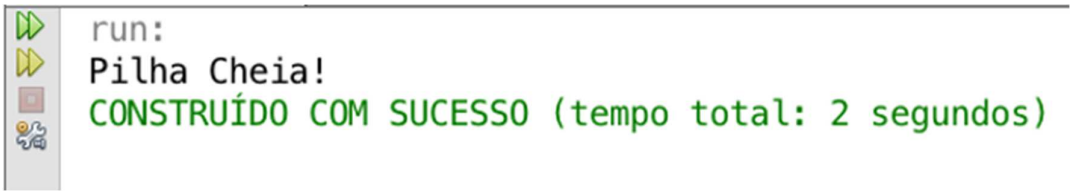
Voltando à Figura 6, estamos adicionando por meio do método *push()*, os elementos 10, 20, 30 e 40 em nossa Pilha. Lembrando que ele será inserido dentro do vetor de inteiros, a[ ], criado na classe Pilha. No entanto, como podemos perceber o elemento {40} não fará parte da pilha, pois ela estava cheia no momento de sua inserção.

Como o primeiro elemento que entra na pilha ficará na base, temos na ordem inversa, da base ao topo: {10,20,30}. O elemento {40} será rejeitado, conforme explicado: “Pilha Cheia”!

Para concluir, na Figura 7, quando a pilha estiver cheia, além de retornar o valor boleano “FALSO”, o algoritmo avisa por meio de uma mensagem que a pilha está cheia, linha 25.

Vamos testar o nosso exemplo para observar o que está acontecendo, quando colocamos mais elementos do que a pilha pode suportar. Visto que nossa pilha tem espaço para 3 elementos, vamos executar o programa tentando inserir os 4 elementos já citados anteriormente: {10, 20, 30, 40}.

Figura 8 – Executando o programa



Fonte: elaborada pelo próprio autor (2020).

Na Figura 8, temos exatamente o reflexo da execução quando o algoritmo tenta adicionar o último elemento, no caso o valor "40". Conforme explicado, o método *push()* vai retornar o valor lógico "FALSO" e apresenta a mensagem "Pilha Cheia!" por meio do método *println(),* comando de exibição de dados na tela do programa.

Uma variação da execução do método *push()*, seria armazenar o seu retorno e por meio desse fator, apresentar essa mensagem dentro da classe **PilhaTela** e não dentro do método *push()* da classe **Pilha**.

Vamos à explicação. O método *push()* sofreria uma alteração, retirando dele, a linha de comando que apresenta a mensagem de "Pilha Cheia!". Ficaria dessa maneira:

Figura 9 – Modificação no método *push()*

```
if (top >= (MAX - 1)) {
    //System.out.println("Pilha Cheia!");
    return false;
```

Fonte: elaborada pelo próprio autor (2020).

Observe na Figura 9, o comando *println()* está como linha de comentário, ou seja, esse comando não será mais executado. Ele exibia a mensagem dentro do método *push().*

Agora, temos que alterar a classe PilhaTela para que exiba a mensagem de acordo com o retorno do método *push().* Vale lembrar que se o retorno for do tipo lógico "FALSO" ou "*false*", como está programado no método, a pilha estará cheia!

Figura 10 – Classe PilhaTela modificada

```
10          Pilha novaPilha = new Pilha();
11
12          int valor = 0;
            int cont  = 0;
14
15          while (novaPilha.push(10)){
16
17              valor = valor + 10;
18              cont++;
19          }
20          System.out.println("Pilha Cheia!");
21          System.out.println(cont);
```

Fonte: elaborada pelo próprio autor (2020).

A codificação da Figura 10 é simples. A variável "valor" é utilizada para que os elementos inseridos na pilha não sejam todos iguais. A variável "*cont*", será utilizada toda a vez que existir espaço na pilha. Sendo assim, ela será incrementada de um em um, até que a condição seja falsa. Quando a condição for falsa, significa que a pilha já estará cheia, pois o método *push()* retornou um valor lógico "FALSO".

Para concluir o nosso exemplo, a execução do código da Figura 10, exibirá na tela a mensagem de "Pilha Cheia!", após inserir três elementos. Em seguida é exibida a variável "*cont*" mostrando o valor "3", tamanho máximo da nossa pilha criada anteriormente.

Figura 11 – Execução do algoritmo

```
run:
Pilha Cheia!
3
CONSTRUÍDO COM SUCESSO
```

Fonte: elaborada pelo próprio autor (2020).

A Figura 11, exibe as mensagens do algoritmo ao usuário. Lembrando que essa é somente uma das formas para resolver o algoritmo, ou seja, não deixar que simplesmente o sistema apresente erro quando inserir mais elementos do que a pilha tem de espaço. Por exemplo, é possível construir um método somente para testar quando a Pilha estará **cheia** e outra quando ela estará **vazia**, no entanto, neste exemplo, foi optado por fazer os tratamentos dentro das respectivas funções.

Videoaula 3

Nesta videoaula, o professor vai abordar a execução do código apresentado como exemplo para controle das inserções de elementos na Pilha. O professor vai explicar o código e realizar possíveis alterações de execução, visando apresentar outras maneiras de resolver o problema do algoritmo.

A explicação continua sendo na sintaxe do Java, Linguagem de Programação de apoio à disciplina de Algoritmos e Estrutura de Dados.

Vamos continuar nosso exemplo agregando informações importantes em mais esta videoaula!

Videoaula 3

Utilize o QRcode para assistir!

Nesta videoaula, o professor vai abordar a execução do código apresentado como exemplo para controle das inserções de elementos na Pilha.

Espero que esses exemplos apresentados no contexto desta Unidade de Ensino, tenham auxiliado você com o objetivo de agregar em seu aprendizado, especificamente nos conteúdos de Fila e Pilha, que foram os principais assuntos aqui abordados.

Encerramento da Unidade

Chegamos ao fim de mais uma Unidade de Ensino após caminhar por conteúdos fundamentais para o seu conhecimento e desenvolvimento na disciplina de Algoritmos e Estrutura de Dados. A Unidade 3, apresentou as estruturas de fila e pilha, exemplificando o seu funcionamento com videoaulas, exemplos de implementação dessas estruturas e uma abordagem clara e objetiva. Foram apresentados os processos de criação e manipulação dos elementos, utilizando como recurso um *Array* Unidimensional.

A Unidade 3 trouxe na prática uma visão geral sobre as estruturas de fila e pilha, bem como, apresentou variações no controle dos elementos dentro das estruturas de dados.

Após a assimilação dos conteúdos abordados nesta Unidade, você estará apto a prosseguir estudando as estruturas abordadas na Unidade de Ensino 4, última unidade da disciplina de Algoritmos e Estrutura de Dados.

Tenha uma ótima leitura e não se esqueça de verificar e estudar a Bibliografia utilizada e as aulas gravadas pelo professor.

Obrigado pela oportunidade de ser o mediador entre os conteúdos específicos abordados e o conhecimento adquirido por você por meio da didática e abordagem desta Unidade.

Referências

CHEN, P. P. S. The entity-relationship model—toward a unified view of data. **ACM Transactions on Database System**, v. 1, p. 9–36, 1976.

DEITEL, H. M.; DEITEL, P. J. **Java**: como programar. 8. ed. São Paulo: Pearson, 2010.

FORBELLONE, A. L. V.; EBERSPÄCHER, H. F. **Lógica de programação.** 2. ed. São Paulo: Makron Books, 2000, 195 p.

LAUREANO, M. **Estrutura de dados com algoritmos e C**. São Paulo: Brasport, 2012.

MANZANO, J. A. N. G. **Estudo Dirigido**: algoritmos. São Paulo: Érica, 1997, 265 p.

Esperamos que este guia o tenha ajudado compreender a organização e o funcionamento de seu curso. Outras questões importantes relacionadas ao curso serão disponibilizadas pela coordenação.

Grande abraço e sucesso!

UniFil