

# N\_EST DAD\_A5 – Texto de Apoio

Site: [EAD Mackenzie](#)

Tema: ESTRUTURA DE DADOS {TURMA 03A} 2023/1

Livro: N\_EST DAD\_A5 – Texto de Apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: quarta, 26 abr 2023, 01:55

# Índice

## TAD FILA

Simulando uma fila

Implementação em vetores

Conceito de Fila Circular

Classe Queue do Java Collections

# TAD FILA

## Conceitos

A fila é uma lista linear **com disciplina de acesso**: todas as inserções são feitas no final da fila, e as remoções de elementos ocorrem em apenas uma extremidade, que é o início da fila. Os elementos são colocados uns após os outros. O inserido mais recentemente está no final da fila. O menos recente está no início da fila. A esta regra, atribui-se o nome **FIFO** (*First in First Out*).

Em nossa vida, encontramos filas diariamente. A fila de um banco é um exemplo comum que pode ser citado, na qual a ordem de chegada reflete na ordem de atendimento dos clientes (fila sem prioridade de atendimento).



As únicas operações que podemos realizar em uma fila são: a operação de inserção (ou enfileiramento), que é chamada **enqueue**, a remoção (desenfileiramento), que é denominada **dequeue** e, ainda, é possível consultar o elemento que está na frente da fila.

**DESENFILAR**



**FRENTE**

**FINAL**

**ENFILAR**

## Simulando uma fila

Acompanhe a sequência abaixo para uma fila de números inteiros:

Operação	Retorno	Status da Fila
enqueue (5)		<div><div>5</div><div>Frente</div><div>Final</div></div>
enqueue (3)		<div><div>5</div><div>3</div><div>Frente</div><div>Final</div></div>
enqueue (7)		<div><div>5</div><div>3</div><div>7</div><div>Frente</div><div>Final</div><div></div></div>
dequeue ()		<div><div>3</div><div>7</div><div>Frente</div><div>Final</div></div>
enqueue (9)		<div><div>3</div><div>7</div><div>9</div><div>Frente</div><div>Final</div></div>
dequeue ()		<div><div>7</div><div>9</div><div>Frente</div><div>Final</div></div>
front ()	7	<div><div>7</div><div>9</div><div>Frente</div><div>Final</div></div>

<b>dequeue ()</b>		<div>9</div> <b>Frente</b> <b>Final</b>
<b>dequeue ()</b>		
<b>dequeue ()</b>	<b>Erro</b>	
<b>isEmpty ()</b>	<b>True</b>	

Repare que a operação dequeue() apenas remove (desenfileira) um elemento da fila. Em nossa abordagem, para ter acesso ao elemento da frente, devemos usar a operação front().

# Implementação em vetores

Faremos a implementação da Fila em um vetor. Implementaremos, inicialmente, uma fila de dados do tipo String, da mesma forma que fizemos com a lista linear. Para essa implementação, utilizaremos a classe Vetor que vimos na Aula 2 para Listas Lineares e que tem as seguintes operações:

```
public class Vetor {  
  
    private String[ ] A;           // armazena os elementos do vetor  
    private int capacity;         // capacidade do vetor  
    private int size;             // elementos no vetor  
  
    public Vetor(int capacity) {  
        A = new String[capacity];  
        this.size = 0;  
        this.capacity = capacity;  
    }  
  
    public boolean isEmpty() {...}  
    public int size() {...}  
    public String get(int i) throws Exception {...}  
    public void set(int i, String n) throws Exception {...}  
    public void add(int i, String n) throws Exception {...}  
    public void remove(int i) throws Exception {...}  
    public int search(String n) {...}  
}
```

Na verdade, não precisaremos de todos os métodos, porém, como a classe já está implementada, manteremos todos eles.

**A classe Fila estenderá a classe Vetor e, portanto, terá acesso aos métodos públicos dessa classe. É muito importante que você retome os conceitos de herança que já estudou no semestre passado!**

Ao implementar uma Fila em um vetor, utilizamos a seguinte premissa: o início da Fila sempre será a posição 0 (zero) do vetor, e o final da Fila está associado ao último elemento armazenado no vetor.

0	1	2	3	4	...	n
"PAULO"	"PEDRO"	"ANA"	"SANDRA"			

Início da fila – “PAULO” – posição 0 do vetor

Final da fila – “SANDRA” – posição 3 do vetor

Faça sempre a implementação dos códigos apresentados, para que você possa consolidar os conceitos.

### PASSO 1 – IMPLEMENTAR A CLASSE FILA, DE FORMA A ESTENDER A CLASSE VETOR



[Podcast](#)

EST DAD - AULA 5 - PODCAST 1 TEXTO DE APOIO.m4a

[Privacy policy](#)

Veja como fica a implementação dos métodos, uma vez que a classe vetor já está pronta.



```

public class Fila extends Vetor {

    public Fila(int capacity) {
        super(capacity);
    }

    public int size() {
        // Devolve o número de elementos da fila
        return super.size();
    }

    public void enqueue(String n) throws Exception {
        // Enfileira, caso a fila não esteja cheia, o elemento n
        add (size(),n);
    }

    public void dequeue() throws Exception {
        // Desenfileira, caso a fila não esteja vazia, o primeiro elemento
        remove(0);
    }

    public String front() throws Exception{
        // Devolve, sem desenfileirar, o primeiro elemento da fila
        return get(0);
    }
}

```

## PASSO 2 – IMPLEMENTAR A CLASSE DE TESTE

Façamos uma fila de Strings que represente uma fila de espera:

```

public class ProjFila {
    public static void main(String[] args) {

        Fila f1 = new Fila(10);
        try {
            f1.enqueue("Marcos Paulo");
            f1.enqueue("Ana Luiza");
            f1.enqueue("José Almeida");
            System.out.println("Primeiro da fila: " + f1.front());
            f1.dequeue();
            f1.enqueue("Julia Muniz");
            System.out.println("Primeiro da fila: " + f1.front());
            f1.enqueue("Maria Marta");
            System.out.println("Pessoas na fila: " + f1.size());
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

```

Experimente modificar a classe de teste, de forma que você represente novas situações para nossa fila.

## Conceito de Fila Circular

A principal ideia de uma fila circular é o armazenamento dos elementos na fila como se ela fosse um círculo. Ao se usar este conceito, o início da Fila não fica fixo na posição 0 do vetor, como no exemplo anterior. A cada elemento adicionado, é atualizada a posição de início e, a cada elemento removido, é atualizada a posição de Fim da fila. Acompanhe a sequência abaixo, que simula uma **Fila Circular** em um vetor de nove posições:

	0	1	2	3	4	5	6	7	8	Início	Fim
enfileira A	A									0	0
enfileira B	A	B								0	1
enfileira C	A	B	C							0	2
desenfileira		B	C							1	2
enfileira D		B	C	D						1	3
enfileira E		B	C	D	E					1	4
desenfileira			C	D	E					2	4
enfileira F			C	D	E	F				2	5
enfileira G			C	D	E	F	G			2	6
desenfileira				D	E	F	G			3	6

enfileira H				D	E	F	G	H		3	7
enfileira I				D	E	F	G	H	I	3	8
enfileira J	J			D	E	F	G	H	I	3	0
desenfileira	J				E	F	G	H	I	4	0
enfileira K	J	K			E	F	G	H	I	4	1
enfileira L	J	K	L		E	F	G	H	I	4	2
enfileira M	J	K	L	M	E	F	G	H	I	4	3
enfileira N	Fila cheia!										

Um novo elemento só não será inserido na fila se, de fato, não houver espaço. Para a implementação, é necessário fazer algumas adaptações na classe nos métodos add e remove da classe Vetor e, também, na classe Fila.

**Métodos add e remove da classe vetor (os demais métodos da classe vetor são iguais):**

```
public void add(int pos, String n) throws Exception {  
    if (qtde == tamanho) {  
        throw new Exception("A Fila está cheia! Impossível inserir!");  
    }  
    vetor[pos] = n;  
    qtde++;  
}  
  
public void remove(int pos) throws Exception {  
    if (isEmpty()) {  
        throw new Exception("Fila vazia - não há elemento para remover!");  
    }  
    qtde--;  
}
```

Classe Fila (adaptada para Fila Circular):



[Podcast](#)

EST DAD - AULA 5 - PODCAST 2 TEXTO DE APOIO.m4a

[Privacy policy](#)

```
public class Fila extends Vetor {
    int inicio;
    int fim;

    public Fila(int capacity) {
        super(capacity);
        inicio = 0;
        fim = -1;
    }

    public int size() {
        // Devolve o número de elementos da fila
        return super.size();
    }

    public void enqueue(String n) throws Exception {
        // Enfileira, caso a fila não esteja cheia, o elemento n
        if (fim == tamanho-1){
            fim = 0;
        }
        else{
            fim++;
        }
        add (fim,n);
    }

    public void dequeue() throws Exception {
        // Desenfileira, caso a fila não esteja vazia, o primeiro elemento
        remove(inicio);
        if (inicio == tamanho-1){
            inicio = 0;
        }
        else{
            inicio++;
        }
    }
}
```

Este projeto está disponível na sessão Explorando Conceitos.

```
public String front() throws Exception{  
    // Devolve, sem desenfileirar, o primeiro elemento da fila  
    return get(inicio);  
}  
}
```

# Classe Queue do Java Collections

O framework Java Collections implementa a estrutura de fila por meio da interface Queue, conforme mostrado no diagrama de classes abaixo.

Um objeto do tipo **Queue <T>** representa uma sequência de objetos (qualquer objeto) do tipo T armazenados em um vetor que aumentará seu tamanho, quando necessário, para a inserção de novos elementos. O método de acesso aos dados é também FIFO (*First In First Out*). Assim:

```
Queue <String> lista = new LinkedList<String>();
```

Se existisse um objeto chamado Aluno e se desejasse criar uma Queue de objetos do tipo Aluno, a instrução ficaria:

```
Queue <Aluno> lista = new LinkedList <Aluno>();
```

Repare que Queue é uma interface e sua implementação é feita, neste caso, em uma lista ligada. Os principais métodos para manipulação de Queue são:

- `nomeQueue.offer(objeto)` - adiciona um elemento no final da fila.
- `nomeQueue.peek()` - retorna o objeto posicionado na cabeça da fila; ou retorna null se a pilha estiver vazia.
- `nomeQueue.poll()` - remove o objeto posicionado na cabeça da fila, retornando-o; retorna null se a fila estiver vazia.
- `nomeQueue.isEmpty()` - retorna true se a fila estiver vazia ou false, caso contrário.
- `nomeQueue.size()` - retorna a quantidade de elementos enfileirados.
- `nomeQueue.clear()` - remove todos os elementos da fila.
- `nomeQueue.contains (objeto)` - retorna true se o objeto indicado existir na fila ou false, caso contrário.

Exemplo:



```
import java.util.*;

public class Teste {

    public static void main(String[] args) {
        Queue<String> nomes = new LinkedList<String>();
        nomes.offer("Solange");
        nomes.offer("Pedro");
        nomes.offer("Álvaro");
        System.out.println("A fila tem " + nomes.size() + " elementos!");

        if (nomes.isEmpty()) {
            System.out.println("A fila está vazia!");
        } else {
            System.out.println("A fila não está vazia!");
        }

        System.out.println("O nome da cabeça da fila é " + nomes.peek());
        nomes.poll();
        System.out.println("O nome da cabeça de fila é " + nomes.peek());

        nomes.offer("Cleide");
        nomes.offer("Tania");

        if (nomes.contains("Pedro")) {
            System.out.println("Pedro está na fila ");
        } else {
            System.out.println("Pedro não está na fila ");
        }

        nomes.clear();
        System.out.println("A fila tem " + nomes.size() + " elementos!");
    }
}
```

Tela de execução:

```
A fila tem 3 elementos!  
A fila não está vazia!  
O nome da cabeça da fila é Solange  
O nome da cabeça de fila é Pedro  
Pedro está na fila  
A fila tem 0 elementos!  
BUILD SUCCESSFUL (total time: 1 second)
```