SBVESDD: Estruturas de Dados

Aula 03: Estruturas de Dados Lineares - Filas

Bacharelado em Ciência da Computação
Prof. Dr. David Buzatto

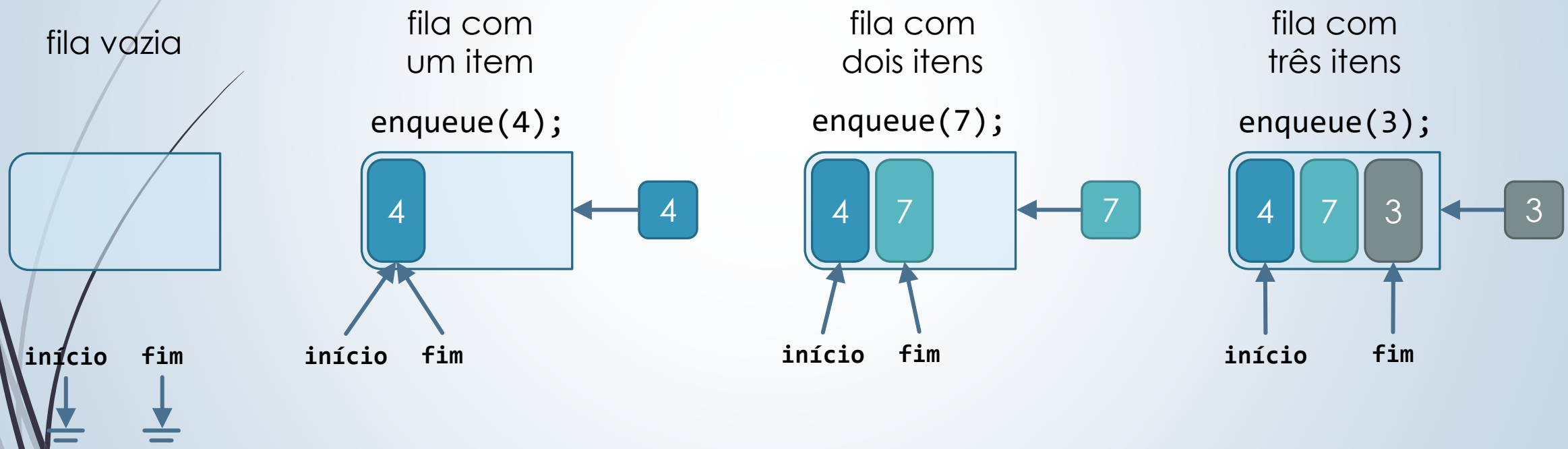
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista

Fila
Contextualização

- A Fila é definida como uma estrutura de dados do tipo FIFO (*First In First Out*), pois os elementos que são inseridos nela, processo denominado enfileirar (***enqueue***), são sempre inseridos no seu final, comumente chamado de **fim ou cauda** e processo inverso do enfileiramento é o desenfileirar (***dequeue***), sendo que o elemento que é desenfileirado é sempre aquele que está no **início ou cabeça** da fila. A seguir é apresentado um esquema onde é simulada a execução de uma fila.

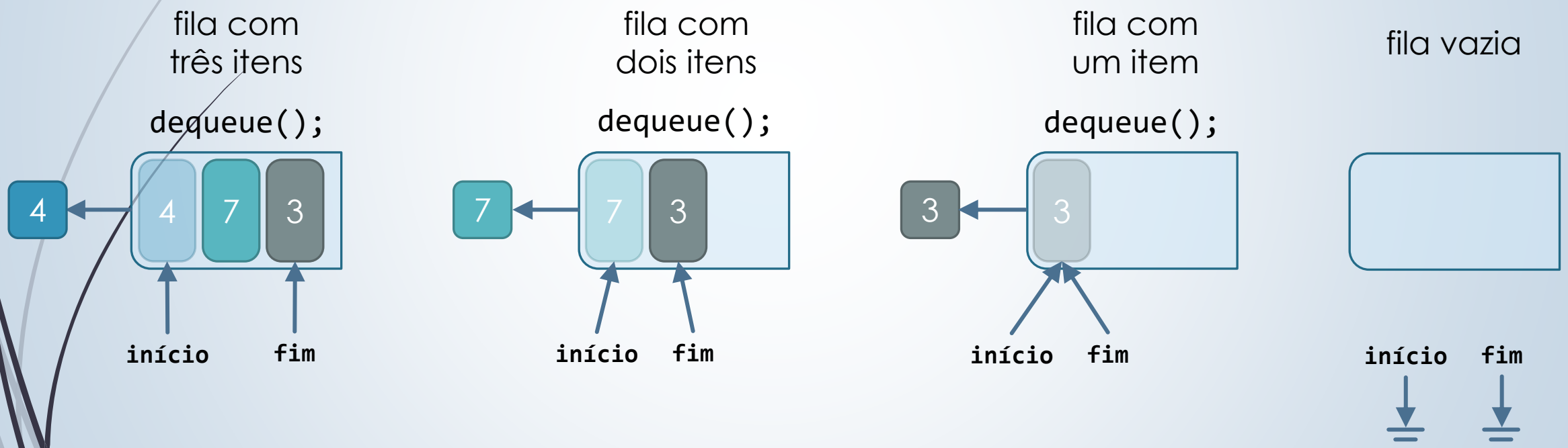
Fila
Operações

- Operação enfileirar (**enqueue**):



Fila
Operações

- Operação desenfileirar (**dequeue**):

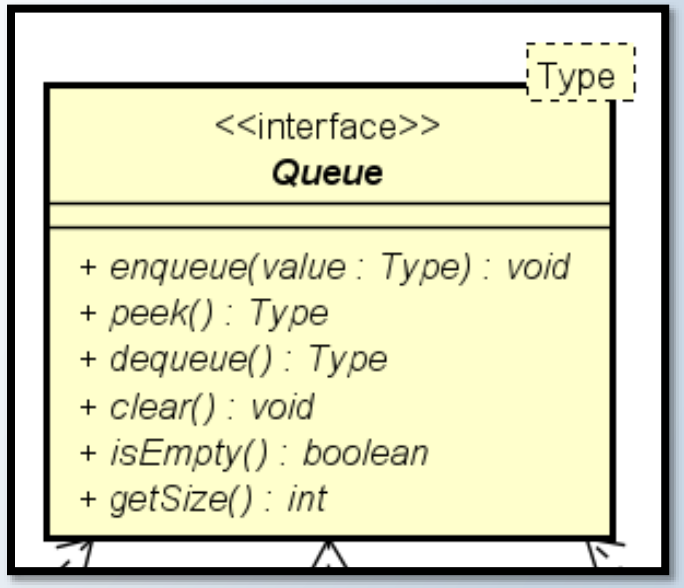


Fila
Aplicações

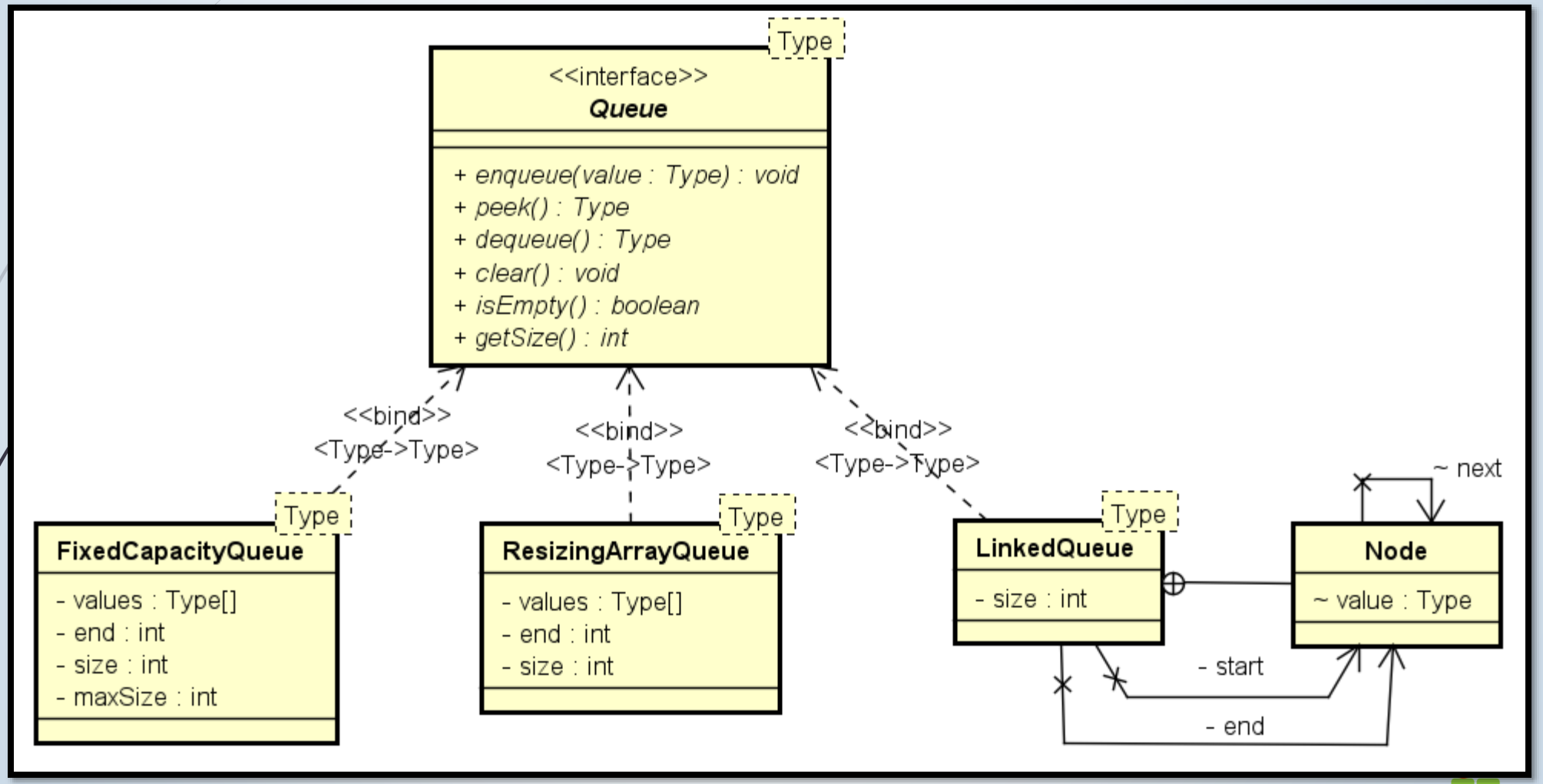
- Dentre as aplicações das filas, pode-se citar:
  - Teoria das filas (matemática);
  - Fila de processos de um sistema operacional;
  - Fila de pacotes que chegam a um roteador;
  - Ordem em que os itens de um estoque são consumidos;
  - Fila de downloads de um navegador;
  - Fila de músicas de um mp3 player;
  - Buffers;
  - Busca em largura em grafos.

Fila
Implementação

- Iremos estudar três formas de implementar Filas:
  1. Fila de capacidade fixa, também chamada de fila estática;
  2. Fila redimensionável;
  3. Fila com estrutura encadeada, também chamada de fila dinâmica.
- Seguiremos uma API padrão (em inglês):
  - **`enqueue:`** enfileira um item/elemento;
  - **`peek:`** consulta qual o item/elemento está no início;
  - **`dequeue:`** desenfileira um item/emento;
  - **`clear:`** limpa a fila, removendo todos os elementos;
  - **`isEmpty:`** verifica se a fila está vazia;
  - **`getSize:`** obtém a quantidade de itens/elementos enfileirados.

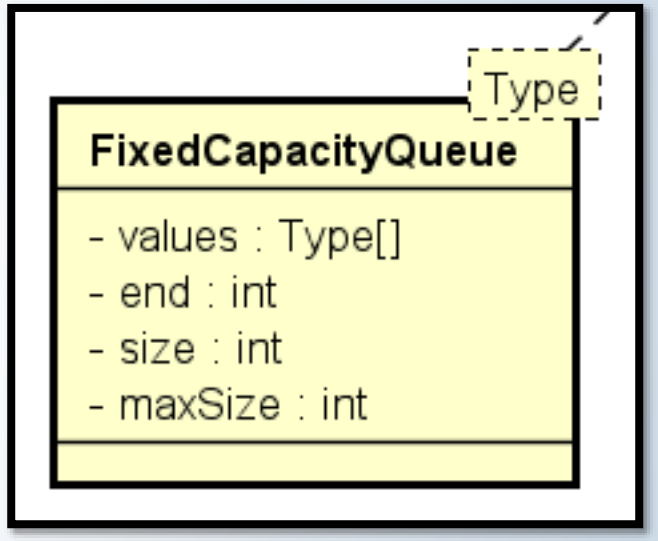


Fila
Implementação



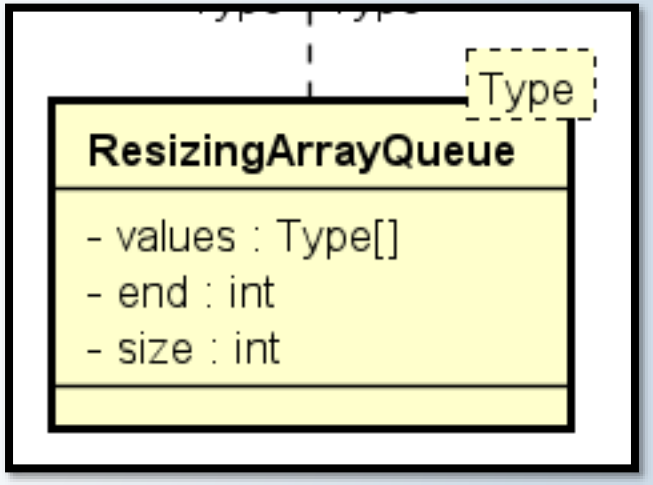
Fila
Implementação: **FixedCapacityQueue**

- Implementação usando um array como estrutura de armazenamento;
- Implementação com a marcação do início/cabeça para a esquerda e o fim/cauda para direita (fim do array);
- A quantidade de elementos que podem ser armazenados, após a instanciação da fila, é fixa;
- **Para pensar:**
  - Qual a ordem de crescimento das operações **enqueue** e **dequeue**?
  - Há como melhorar?
  - Mudar a topologia resolve?



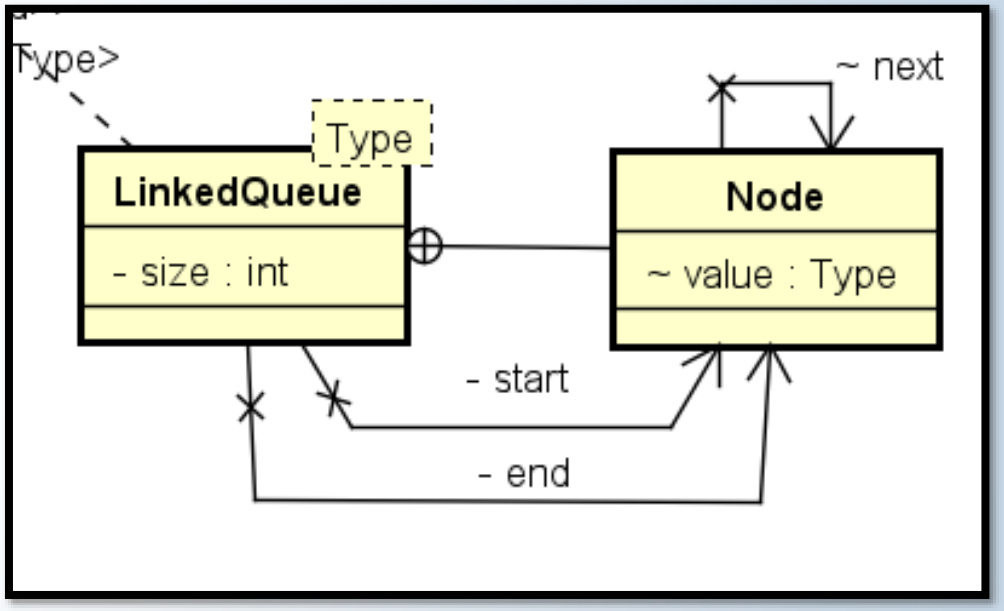
Fila
Implementação: **ResizingArrayQueue**

- Implementação usando um array como estrutura de armazenamento;
- Implementação com a marcação do início/cabeça para a esquerda e o fim/cauda para direita (fim do array);
- A quantidade de elementos que podem ser armazenados, após a instanciação da fila, é variável, pois ela crescerá ou diminuirá em função de inserções e remoções;
- **Para pensar:**
  - Qual a ordem de crescimento das operações **enqueue** e **dequeue**?
  - Há como melhorar?
  - Mudar a topologia resolve?



Fila
Implementação: **LinkedQueue**

- Implementação usando uma estrutura baseada em nós (**nodes**) com encadeamento simples;
- O início e o fim da fila são indicados por membros do tipo dos nós;
- A alocação e liberação de espaço para o armazenamento dos elementos tem característica dinâmica;
- **Para pensar:**
  - Qual a ordem de crescimento das operações **enqueue** e **dequeue**?



Fila
Exercícios de Implementação

- **Exercício i3.1:** No problema de Josefo da antiguidade, n pessoas estão passando por necessidades e concordam com a seguinte estratégia para reduzir a população. Eles se arranjam em um círculo, nas posições numeradas de 0 a $n-1$, e seguem ao redor do círculo, eliminando (sim, matando) cada m-ésima pessoa até restar apenas uma. Diz a lenda que Josefo conseguia descobrir aonde se sentar para evitar ser eliminado. Seria ele um cientista da computação? Infelizmente não sabemos... Sendo assim, no projeto **ESDC4Aula03**, implemente, na classe **Exercicioi3p1**, o método:

```
public static int[] josephus( int n, int m ) throws IllegalArgumentException
```

- Esse método, a partir da quantidade n de pessoas e de uma posição m, deve gerar um array de inteiros com a ordem em que as pessoas serão ~~assassi...~~ eliminadas, contendo assim, em sua última posição, o lugar aonde Josefo deveria se sentar. Por exemplo, após a execução da linha abaixo:

```
int[] ordem = josephus( 7, 2 );
```

- Os valores contidos no array **ordem** serão:

1	3	5	0	4	2	6

💀 💀 💀 💀 💀 💀 😰

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus São João da Boa Vista

Bibliografia

SEDGEWICK, R.; WAYNE, K. **Algorithms**. 4. ed. Boston: Pearson Education, 2011. 955 p.

GOODRICHM M. T.; TAMASSIA, R. **Estruturas de Dados & Algoritmos em Java**. Porto Alegre: Bookman, 2013. 700 p.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos – Teoria e Prática**. 3. ed. São Paulo: GEN LTC, 2012. 1292 p.