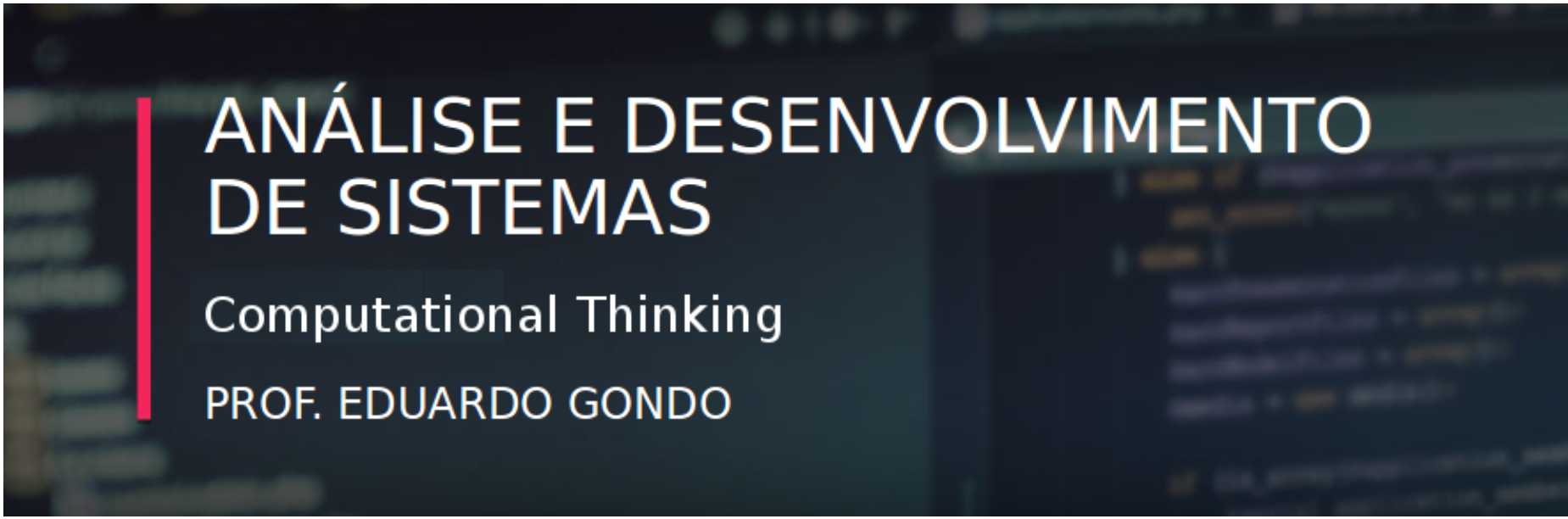
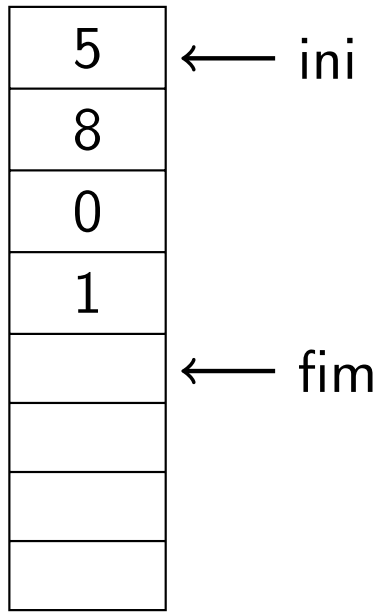

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
Computational Thinking
PROF. EDUARDO GONDO

Fila (Queue) — Introdução

- FIFO - First In First Out
- inserções são feitas no fim da lista e remoções são feitas no início da lista
- podemos imaginar que a fila dentro da programação funciona igual a uma fila de banco ou a fila do ônibus da FIAP
- na computação temos a fila de processos e a fila de impressão
- usaremos um vetor para representar uma fila
- ini aponta para o 1º elemento da fila e fim para a 1ª posição livre da fila



Fila — Operações

Uma fila possui as seguintes operações (métodos e construtor):

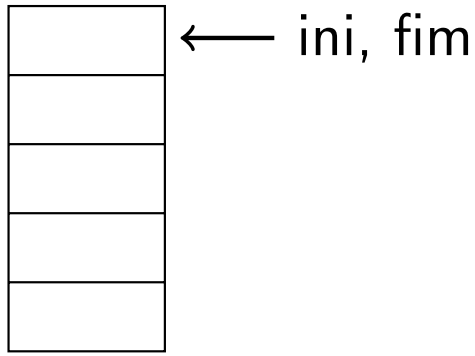
- `Fila(n)`: instancia uma fila com capacidade para armazenar *n* informações
- `boolean isCheia()`: retorna true se a fila está cheia
- `boolean isVazia()`: retorna true se a fila está vazia
- `void enfila(<info>)`: coloca info na fila
- `<info> desenfila()`: devolve info da fila removendo-a da fila
- `<info> primeiroDaFila()`: devolve info da fila sem removê-la

, onde `<info>` representa o **tipo** de dado armazenado na fila.

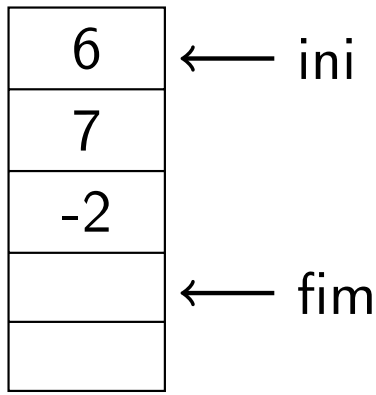
Antes de mostrarmos a implementação de uma Fila, vamos mostrar como fica o desenho dela após a execução de algumas instruções:

Fila — Teste de Mesa

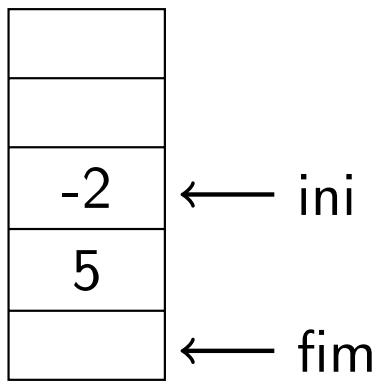
```
1  Fila<Integer> f = new Fila<>(5);
```




```
1  f.enfila(6);
2  f.enfila(7);
3  f.enfila(-2)
```

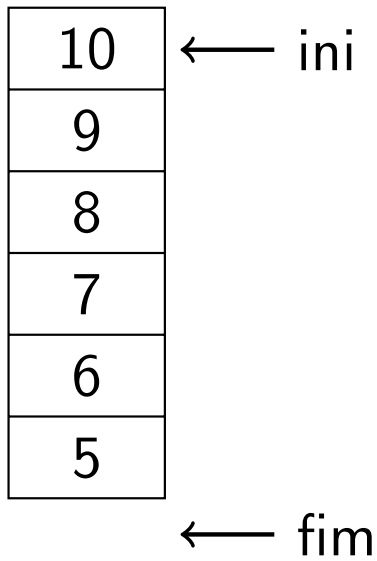



```
1  f.desenfila();
2  f.desenfila();
3  f.enfila(5);
```



Fila — Teste de Mesa 2

```
1  Fila<Integer> f = new Fila<>(6);
2  int i = 10;
3  while (!f.isCheia()) {
4     f.enfila(i);
5     i--;
6  }
```



Fila — Genérica

Abaixo segue a implementação da classe Fila:

```
1  public class Fila<T> {
2
3     private int ini;
4     private int fim;
5     private Object[] lista;
6
7     public Fila(int tamanho) {
8        lista = new Object[tamanho];
9        ini = 0;
10       fim = 0;
11    }
12
13    public T primeiroDaFila() {
14       return (T)lista[ini];
15    }
```

Fila —Genérica

```
16      public boolean isCheia() {
17          if (ini == 0 && fim == lista.length)
18              return true;
19          else
20              return false;
21      }
22
23      public boolean isVazia() {
24          if (ini == 0 && fim == 0)
25              return true;
26          else
27              return false;
28      }
29
30      public T desenfila() {
31          int aux = ini;
32          ini++;
33          if (ini == fim) {
34              ini = 0;  fim = 0;
35          }
36          return (T)lista[aux];
37      }
```

Fila —Genérica

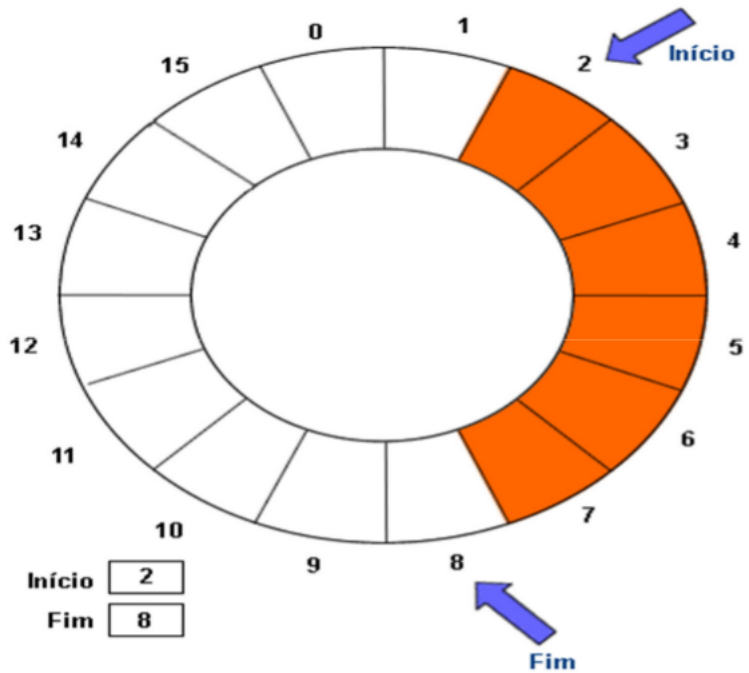
```
38      public void enfila(T info) {
39          if (fim == lista.length) {
40              arrumaFila();
41          }
42          lista[fim] = info;
43          fim++;
44      }
45
46      private void arrumaFila() {
47          //TODO
48      }
49  }
```

- Quando chamamos o desenfila a fila pode ficar vazia, nesse caso, posicionamos os apontadores *ini* e *fim* para a posição 0
- No enfila pode acontecer do fim apontar para *lista.length* mas a fila ainda possui elementos disponíveis (ini != 0), nessa situação devemos mover os dados para o início do vetor através do arrumaFila.

Considerações

- o custo do método desenfila é pequeno, pois quando a fila fica vazia basta atribuirmos 0 aos ponteiros início e fim tornando o custo do método constante
- no método enfila precisamos checar quando fim está posicionado além da capacidade do vetor. Nesta situação devemos mover todos os elementos da fila para o início do vetor.
- note que mover os elementos da fila para o início do vetor pode ser uma ação que leva algum tempo pois devemos mover todos os elementos armazenados na fila
- assim, o método enfila pode tornar-se uma operação demorada, será que existe algum modo de eliminar a movimentação dos elementos na fila?

Fila Circular



Na fila circular resolvemos o problema de chegar ao fim do vetor adicionando um atributo inteiro quantidade que indica a quantidade de elementos que temos na fila. Com esse atributo são alterados todos os métodos e eliminamos a necessidade do `arrumaFila()`

Exercícios

1) Instancie um objeto fila com capacidade para armazenar 20 elementos e preencha totalmente a fila com números aleatórios. Para este exercício está proibido a utilização do comando for. Retire na ordem todos os elementos da fila imprimindo na tela os valores armazenados.
2) Escreva um método que recebe uma sequência de 100 números inteiros e separa os 100 números em duas outras sequências: uma contendo os números pares e outra os ímpares. Use, obrigatoriamente, duas filas dentro desse método.
3) Implemente o método arrumaFila
4) Implemente a Fila Circular.

Referência Bibliográfica

- **Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java**, Ascencio e Campos - 2ª ed., Pearson 2007
- **Lógica de Programação e Estrutura de Dados**, Puga e Rissetti - 2ª ed., Pearson Prentice Hall, 2008.
- **Algoritmos em linguagem C**, Feofiloff - Campus/Elsevier, 2009 (`http://www.ime.usp.br/~pf/algoritmos-livro/index.html`)
- **Construção de Algoritmos e Estruturas de Dados**, Forbellone e Eberspacher - Pearson Prentice Hall, 2010.
- **Projeto de Algoritmos com Implementações com Java e C++**, Ziviani - Thompson, 2006
- **Java como Programar**, Deitel e Deitel - 8ª ed., Pearson, 2010
- **Algoritmos**, Cormen, Leiserson, Rivest e Stein - Campus

Copyleft