O TAD fila

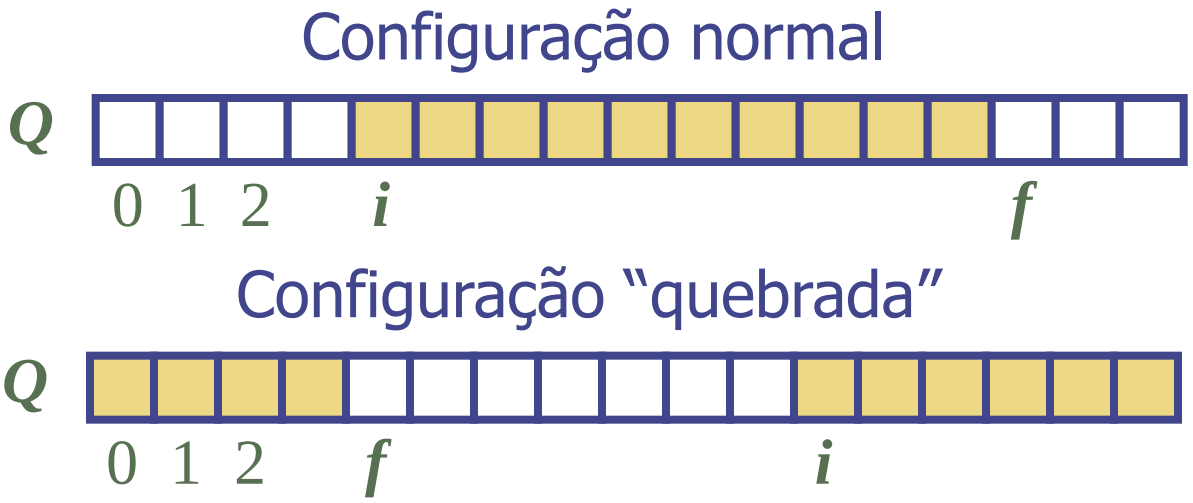
- O TAD fila armazena objetos arbitrários
- Inserções e remoções seguem o esquema FIFO
- Inserções são feitas no fim da fila e remoções no início da fila
- Operações principais:
  - enfileirar(object): insere um elemento no fim da fila
  - object desenfileirar(): remove e retorna o elemento do início da fila
- Operações auxiliares:
  - object inicio(): retorna o elemento do início sem removê-lo
  - integer tamanho(): retorna o número de elementos armazenados
  - boolean estaVazia(): indica se há elementos na fila
- Exceções
  - Tentar remover o ver um elemento do início da fila levanta a exceção EFilaVazia

Aplicações de fila

- Aplicações diretas
  - Filas de esperas
  - Acesso a recursos compartilhados (impres.)
  - Programação paralela
- Aplicações indiretas
  - Estrutura de dados auxiliar para algoritmos
  - Componentes de outras estruturas de dados

Fila baseada em *array*

- Use um *array* de tamanho N de forma circular
- Duas variáveis mantêm informações do início e fim da fila
  - i índice do elemento do início
  - f índice imediatamente após o último elemento
- A posição f no array fica vazia

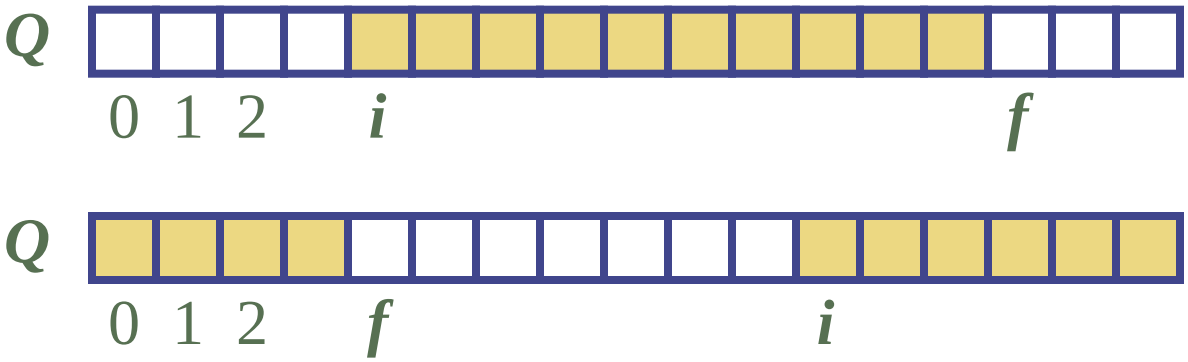


Operações sobre o TAD fila

- Usamos o operador módulo (resto da divisão)

Algoritmo *tamanho*()
  retorne $(N - i + f) \% N$

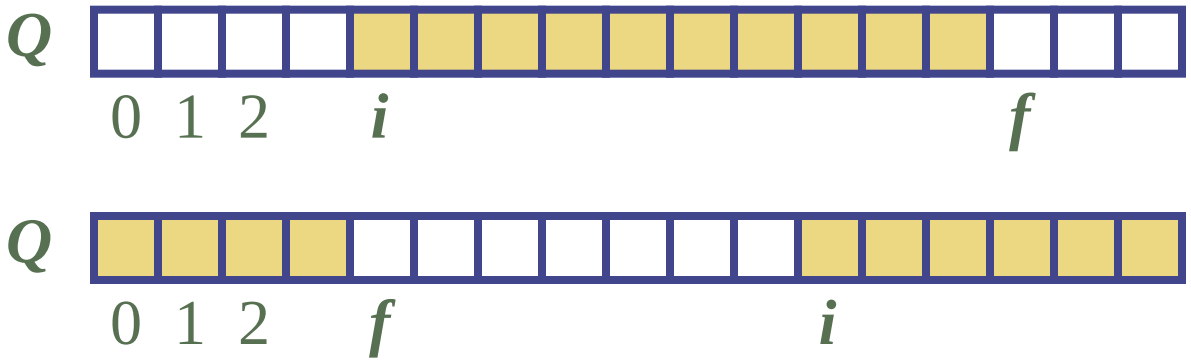
Algoritmo *estaVazia*()
  return $(i = f)$



Operações sobre o TAD fila

- Operação *enfileirar* levanta uma exceção se o *array* está cehio
- Esta exceção é dependente da implementação

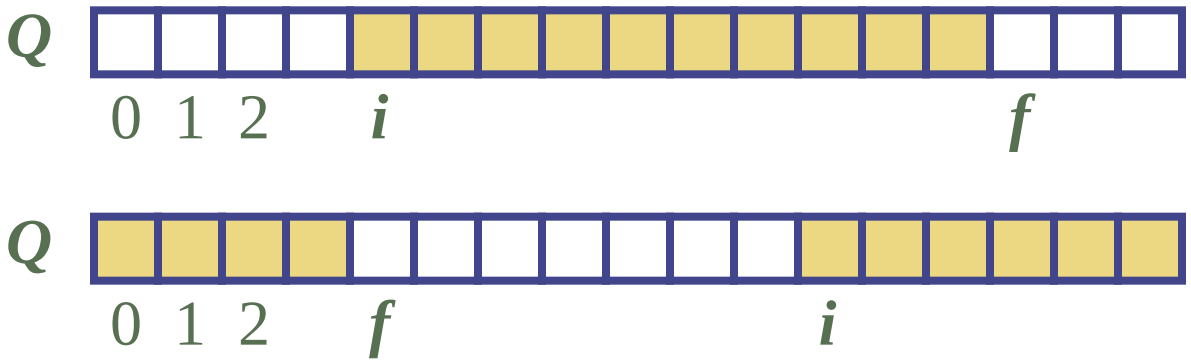
Algoritmo ***enfileirar(o)***
  Se (*tamanho*() = *N* – 1**)então**
    throw ***EFilaCheia***
  senão
    Q[*f*] ← *o*
    f ← (*f* + 1) % *N*



Operações sobre o TAD fila

- Operação *desenfileirar* levanta uma exceção se a fila está vazia
- Esta exceção é específica do TAD Fila

```
Algoritmo desenfileirar()
  Se (estaVazia()) então
    throw EFilaVazia
  senão
    o ← Q[i]
    i ← (i + 1) % N
    retorne o
```



Fila crescente baseada em *array*

- Em uma operação *desenfileirar*, quando o *array* está cheio, ao invés de levantar uma exceção, podemos substituir o *array* por um maior
- Similar ao que fizemos com Pilhas
- A operação *enfileirar* tem tempo de execução amortizado
  - $O(n)$ com estratégia incremental
  - $O(1)$ com estratégia de duplicação

Interface fila em JAVA

- Interface em JAVA que corresponde ao nosso TAD
- Requer a definição da classe EFilaVazia

```
public interface Fila {

   public int tamanho();

   public boolean estaVazia();

   public Object inicio()
         throws EFilaVazia;

   public void enfileirar(Object o);

   public Object desenfileirar()
         throws EFilaVazia;
}
```

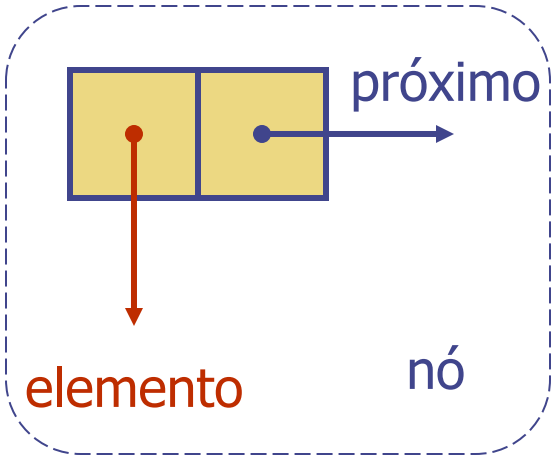
O TAD deque

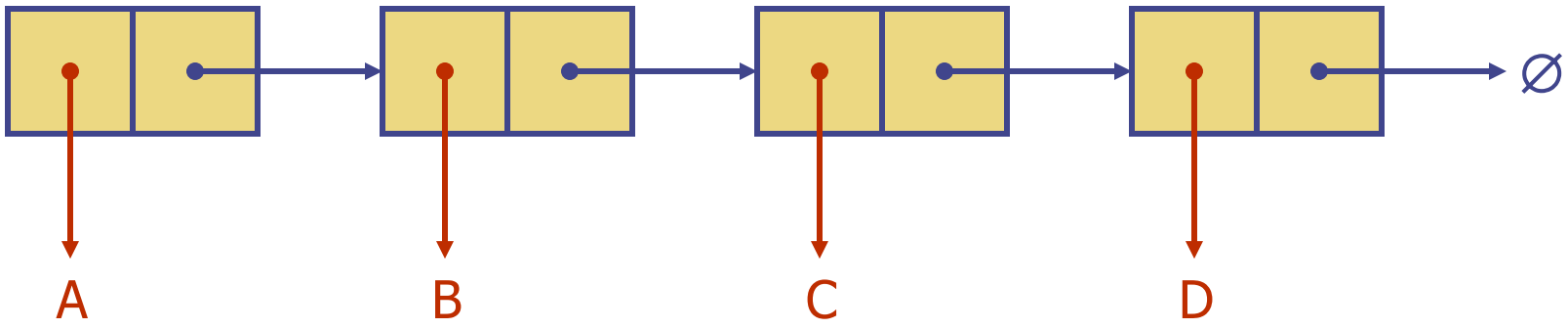
- O TAD deque armazena objetos arbitrários
- Inserções e remoções podem ser feitas no início ou no fim
  - É uma fila de duplo sentido

- Operações principais:
  - inserirInicio(object):
  - object removerInicio():
  - inserirFim(object):
  - object removerFim():
- Operações auxiliares
  - object primeiro():
  - object ultimo():
  - int tamanho():
  - boolean estaVazia():

Lista ligada

- Uma lista ligada é uma estrutura de dados concreta consistindo de uma sequência de nós
- Cada nó armazena
  - Um elemento
  - Uma *ligação* com o próximo nó



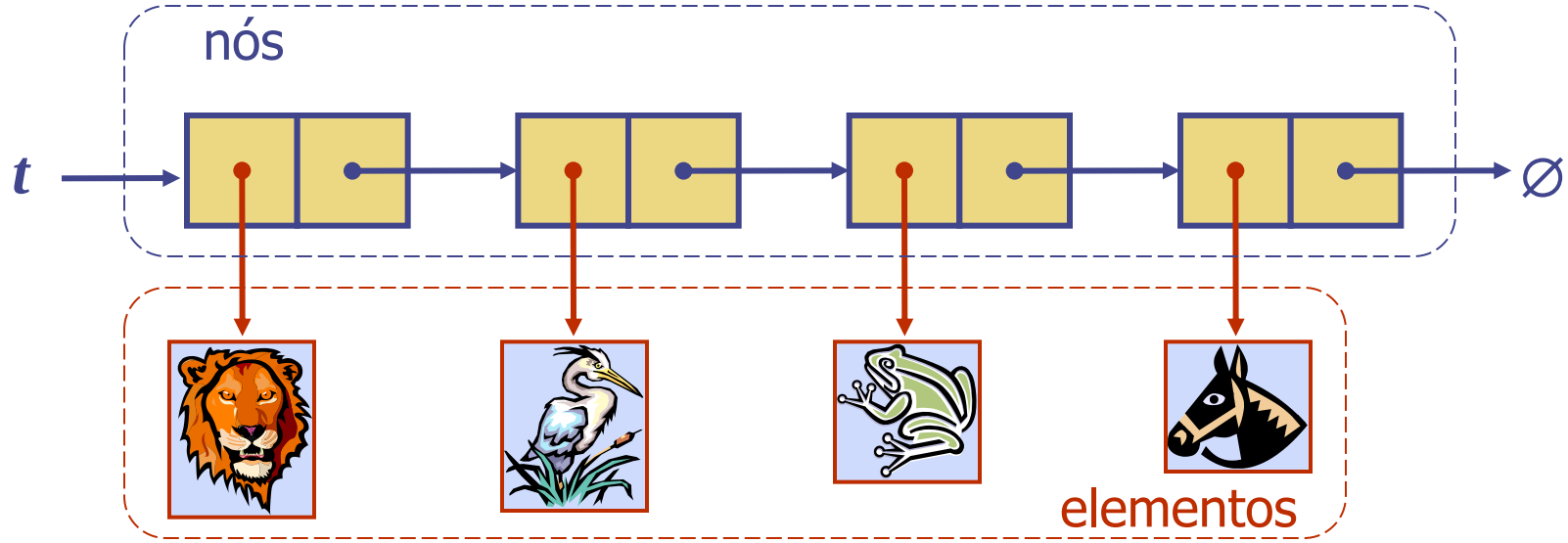


Classe No

```
public class No {
    private Object elemento;
    private No proximo;
    public Object getElemento() {
       return elemento;}
    public void setElemento(Object o){
            elemento = o;
    }
```

Pilhas com listas ligadas

- Pode-se implementar uma pilha com uma lista ligada
- O elemento do topo é armazenado no primeiro nó da lista
- O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$



Filas com listas ligadas

- Pode-se implementar uma fila com uma lista ligada
  - O elemento do início é o primeiro nó
  - O elemento do fim é o último nó
- O espaço usado é $O(n)$ e cada operação roda em tempo $O(1)$