ESTRUTURA DE DADOS

FILA

Prof. Adeamar Schmitz, M.Sc.

BIBLIOGRAFIA

- GOODRICH, Michael. T., TAMASSIA, Roberto. **Estruturas de Dados e Algoritmos em Java.** São Paulo: Bookman, 2002.
  - Capítulo 4: Pilhas, Filas e Deques.
- WEISS, Mark A. **Data Structures & Algorithm Analisis in Java**. Addison Wesley Longman, 1999.
  - Chapter 3: Lists, Stacks, and Queues

DEFINIÇÃO

- Estrutura de dados na qual o primeiro elemento a entrar é o primeiro elemento a sair.
- Conhecida como FIFO (first-in first-out).
- As filas (queues) são conjuntos de elementos cujas operações de inserção são feitas por uma extremidade e de remoção, por outra extremidade.

EXEMPLOS: FILA

- Mundo Real:
  - Lojas, teatros, locais públicos.
  - Agências de reservas de passagens.
  - Outros serviços similares.
- Computação:
  - Processos compartilhando um recurso.
  - Alocação de memória em Java.
  - Programação com threads em Java.

TAD FILA: QUEUE

- *size():* retorna o número de objetos na fila.
  - entrada: nenhuma
  - saída: inteiro
- *isEmpty():* retorna um booleano indicando se a fila está vazia ou não.
  - entrada: nenhuma
  - saída: booleano
- *front():* retorna o objeto da frente da fila sem retirá-lo; ocorre um erro se a fila estiver vazia.
  - entrada: nenhuma
  - saída: objeto

TAD FILA: QUEUE

- *enqueue(o):* insere o objeto *o* no fim da fila.
  - entrada: objeto
  - saída: nenhuma
- *dequeue():* retira e retorna o objeto da frente da fila; ocorre um erro se a fila estiver vazia.
  - entrada: nenhuma
  - saída: objeto

TAD FILA: QUEUE

- Implementar um TAD envolve dois passos:
  - Definir uma interface que descreve os nomes dos métodos que o TAD suporta e como eles são declarados e usados.
  - Fornecer uma classe concreta que implemente os métodos descritos na interface associada com o TAD.

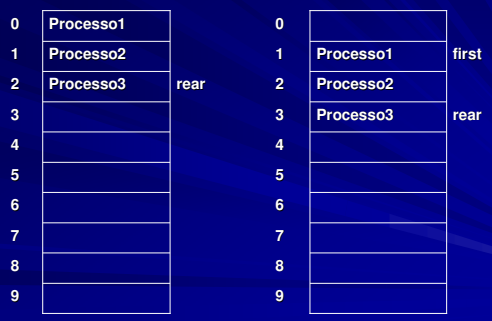
INTERFACE QUEUE

```
public interface Queue {

   public int size();

   public boolean isEmpty();

   public Object front() throws
   EmptyQueueException;

   public void enqueue(Object element) throws
       FullQueueException;

   public Object dequeue() throws
       EmptyQueueException;
}
```


IMPLEMENTAÇÃO COM VETOR

- Q[0] pode ser a frente da fila, deixando a fila crescer a partir dai.
  - Exige que movamos todos os elementos para a frente em uma posição cada vez que efetuarmos uma operação dequeue.
- Definir duas variáveis *first* e *rear* que possuem os seguintes significados:
  - *first* é o índice para a posição de Q que guarda o primeiro elemento da fila.
  - *rear* é o índice para a próxima posição livre em Q.

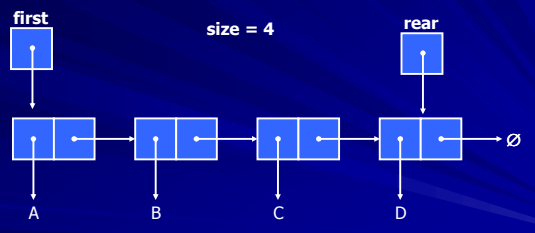
IMPLEMENTAÇÃO COM VETOR



IMPLEMENTAÇÃO COM VETOR

- **Desvantagem:** definir a capacidade da fila em um certo número N.
- Se temos uma boa estimativa do número de elementos que terão na fila em um dado momento, então a implementação baseada em vetores é bastante eficiente.

IMPLEMENTAÇÃO COM LISTA ENCADEADA



ANALISE ASSINTOTICA DAS OPERAÇÕES

OPERAÇÃO	ARRAY (0 - rear)	ARRAY (first - rear)	LISTA ENCADEADA (first e rear)
size()	O(1)	O(1)	O(1)
isEmpty()	O(1)	O(1)	O(1)
front()	O(1)	O(1)	O(1)
enqueue(o)	O(1)	O(1)	O(1)
dequeue()	O(n)	O(1)	O(1)