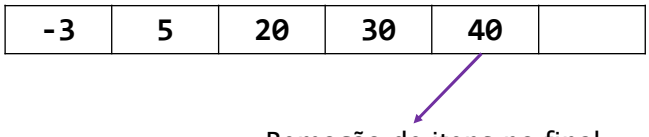
Fila de Prioridades e Lista Simplesmente Encadeada

Profa. Divani Barbosa Gavinier
Aula 9 – Estrutura de Dados

FILA DE PRIORIDADES

É uma estrutura de dados mais especializada que uma pilha ou fila. Como em uma fila comum, uma fila de prioridade tem uma frente e um final. Porém em uma fila de prioridades os itens são ordenados por valor-chave, de modo que o item com a chave mais baixa esteja na frente e a mais alta no fim.

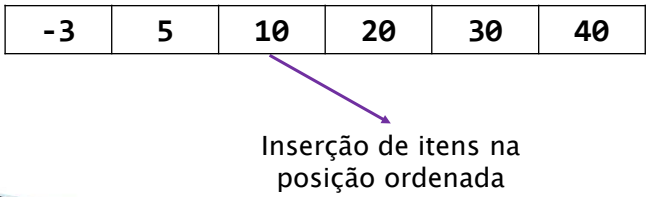


Remoção de itens no final
Itens com a chave mais alta

Vamos inserir o valor 10 ..

FILA DE PRIORIDADES

É uma estrutura de dados mais especializada que uma pilha ou fila. Como em uma fila comum, uma fila de prioridade tem uma frente e um final. Porém em uma fila de prioridades os itens são ordenados por valor-chave, de modo que o item com a chave mais baixa esteja na frente e a mais alta no fim.

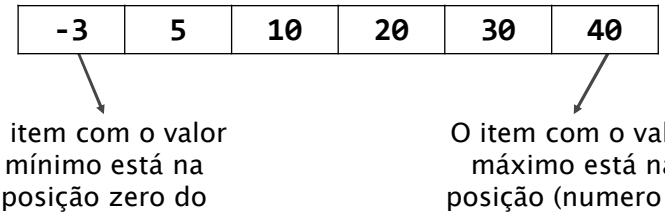


Inserção de itens na posição ordenada

3

FILA DE PRIORIDADES

É uma estrutura de dados mais especializada que uma pilha ou fila. Como em uma fila comum, uma fila de prioridade tem uma frente e um final. Porém em uma fila de prioridades os itens são ordenados por valor-chave, de modo que o item com a chave mais baixa esteja na frente e a mais alta no fim.



O item com o valor mínimo está na posição zero do vetor

O item com o valor máximo está na posição (numero de itens do vetor – 1)

4

Classe

Campos

```
int nItens;
int tam_max;
long[] itens;
```

Construtor

```
public FilaPrior(int n) {
  itens = new long[n];
  tam_max = n;
  nItens = 0;
}
```

Métodos

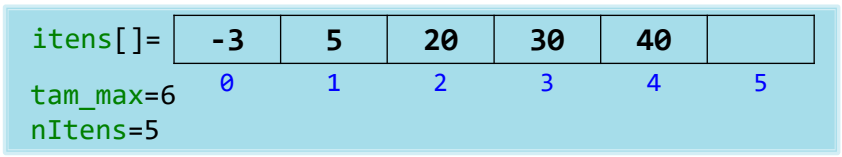
push (enqueue): Insere itens na fila
pop (dequeue): Retira itens da fila
front: Retorna o próximo item da fila a ser removido, sem retira-lo (consulta)
empty: Verifica se a fila esta vazia
full: Verifica se a fila esta cheia
max: Retorna o item com o valor máximo
min: Retorna o item com o valor mínimo

5

EMPTY e FULL

Entrada: Nenhuma
Saída: Verdadeiro ou falso

```
public boolean empty() { return (nItens == 0 ); }
public boolean full()  { return (nItens == tam_max ); }
```



FRONT

Entrada: Nenhuma
Saída: Próximo item a ser removido (maior item)

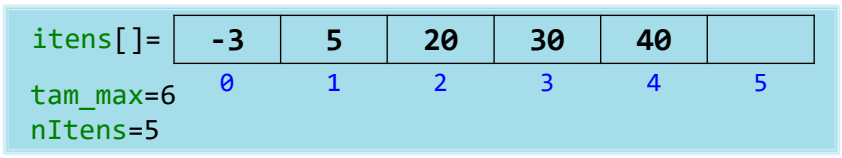
```
public long front() {
  return itens[ nItens-1 ];
}
```

6

MIN e MAX

Entrada: Nenhuma
Saída: Menor item no caso de min e Maior item no caso de max

```
public long max() { return itens[ nItens-1 ]; }
public long min() { return itens[ 0 ]; }
```



POP (dequeue)

Entrada: Nenhuma
Saída: Não tem retorno

```
public void pop() { nItens--; }
```

7

PUSH (enqueue)

Entrada: Valor do item a ser adicionado na fila (valor)
Saída: Não tem retorno

```
public void push(long valor) {
 if (full()) { // se fila cheia
    System.out.println("ATENCAO FILA CHEIA");
    return;
 }
 if (empty()) itens[ nItens ] = valor; // Se fila vazia
 else { // Se fila não vazia
      int i, j;
      for (i=0; i<nItens; i++) {
          if (itens[ i ]>valor)
             break;
      }
      for (j=nItens; j>i; j--) {
          itens[ j ] = itens[ j-1 ];
      }
      itens[ i ]=valor;
 } // fim else se fila não vazia
 nItens++; // incrementa um item no numero de itens
} // fim método push
```

PUSH (enqueue)

Entrada: Valor do item a ser adicionado na fila (valor)
Saída: Não tem retorno

```
public void push(long valor) {
 if (full()) { // se fila cheia
    System.out.println("ATENCAO FILA CHEIA");
    return;
 }
 if (empty()) itens[ nItens ] = valor; // Se fila vazia
 else { // Se fila não vazia
      int i, j;
      for (i=0; i<nItens; i++) {
          if (itens[ i ]>valor)
             break;
      }
      for (j=nItens; j>i; j--) {
          itens[ j ] = itens[ j-1 ];
      }
      itens[ i ]=valor;
 } // fim else se fila não vazia
 nItens++; // incrementa um item no numero de itens
} // fim método push
```

Trecho de Código que trata da inserção de itens de maneira ordenada

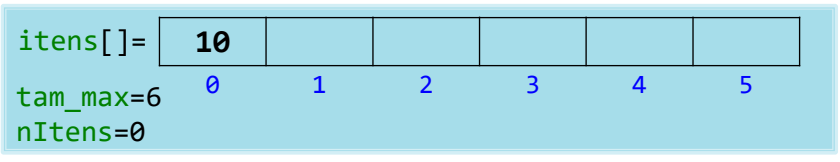
```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 10 (valor=10)



10

Trecho de Código que trata da inserção de itens de maneira ordenada

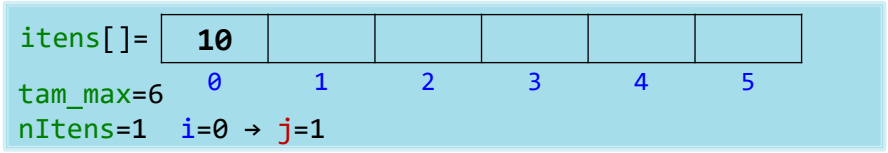
```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 5 (valor=5)



Trecho de Código que trata da inserção de itens de maneira ordenada

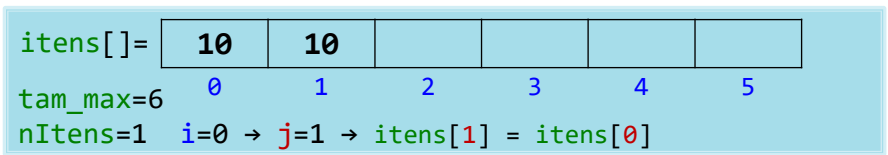
```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 5 (valor=5)



Trecho de Código que trata da inserção de itens de maneira ordenada

```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 5 (valor=5)

itens[]=	5	10				
	0	1	2	3	4	5

tam_max=6
nItens=1 i=0 → j=1 → itens[1] = itens[0] → itens[0]=5

Trecho de Código que trata da inserção de itens de maneira ordenada

```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 20 (valor=20)

itens[]=	5	10				
	0	1	2	3	4	5

tam_max=6
nItens=2 i=2

Trecho de Código que trata da inserção de itens de maneira ordenada

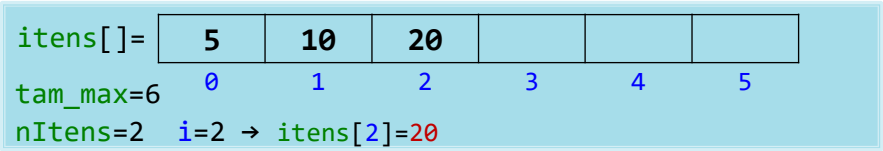
```
int i, j;
for (i=0; i<nItens; i++) {
    if (itens[ i ]>valor) break;
}
for (j=nItens; j>i; j--) {
    itens[ j ] = itens[ j-1 ];
}
itens[ i ]=valor;
```

Procurando posição para inserir, quando encontrar i = posição

Movendo os itens restantes para novas posições

Inserindo item na posição correta

Inserindo item 20 (valor=20)



Implementação Fila Prioridades

```
class FilaPrior {
  private long[] itens;
  private int nItens;
  private int tam_max;

  public FilaPrior (int n) {
    itens = new long[n];
    tam_max = n;
    nItens = 0;
  }

  public boolean empty() { return ( nItens == 0 );         }
  public boolean full()  { return ( nItens == tam_max );   }
  public long front()    { return itens[ nItens-1 ];       }
  public long min()      { return itens[ 0 ];              }
  public long max()      { return itens[ nItens-1 ];       }
  public void pop()      { nItens--;                       }

                                                  // Continuação ...
```

```
// Continuação ...

  public void push(long valor) {
   if (full()) {
      System.out.println("ATENCAO FILA CHEIA");
      return;
   }
   if (empty()) itens[ nItens ] = valor; // Se fila vazia
   else { // Se fila não vazia
        int i, j;
        for (i=0; i<nItens; i++) { // Encontrando a posição
            if (itens[ i ]>valor)
               break; // encontrou posição → sai
        }
        for (j=nItens; j>i; j--) // Movendo maiores para cima
            itens[ j ] = itens[ j-1 ];
        itens[ i ]=valor; // inserindo na posição correta
      } // fim if se fila não vazia
   nItens++; // incrementa um item no numero de itens
  } // fim método push

} // fim classe FilaPrior
                                                    // Continua ...
```

```
// Continuação ...
class FilaPriorApp {
  public static void main(String[] args) {

    FilaPrior fp = new FilaPrior(6);

    System.out.println("Enfilerando 6 itens ordenadamente");
    System.out.println("Valores Inseridos: 10, 5, 20, 40, 30 e -3");
    fp.push(10);  fp.push(5);  fp.push(20);
    fp.push(40);  fp.push(30); fp.push(-3);

    System.out.println("Valor minimo = " + fp.min());
    System.out.println("Valor maximo = " + fp.max());

    System.out.println("Removendo todos os itens da fila");
    System.out.print("Valores Removidos:");
    while (!fp.empty()) { // Ate esvaziar
      System.out.print(" " + fp.front());
      fp.pop();
    }
    System.out.println("\n");
  } // fim programa principal
} // fim classe principal FilaPriorApp
```

Resumo

- Pilha, Fila e Fila de Prioridades são estruturas de dados usadas para simplificar operações de programação.
- Nelas apenas um item de dados pode ser acessado.
- Uma fila pode ser implementada como uma fila circular, que é baseada em um vetor cujo os índices circundam do final para o inicio.
- Uma fila de prioridades permite acesso ao maior item e consulta ao menor.
- Essas estruturas também podem ser implementadas com o mecanismo de lista encadeada (ao invés de vetores).

19

Gráfico Big O:

Demonstra como tempos de execução são afetados pelo numero de

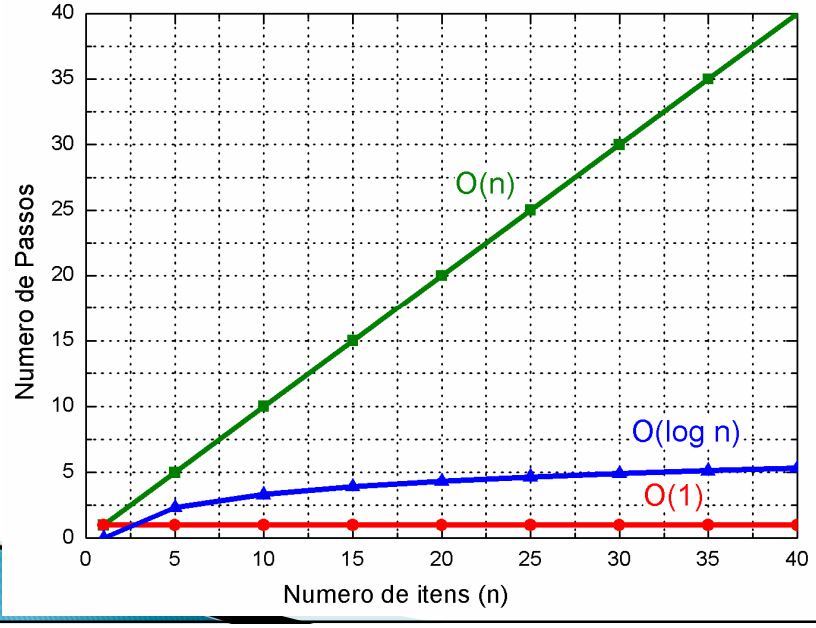
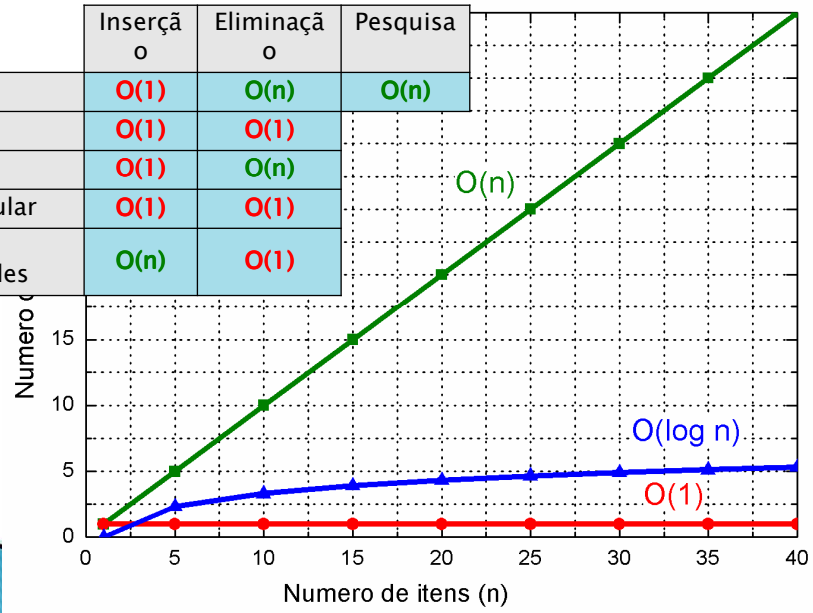


Tabela de Estruturas de Armazenamento de Propósito Geral (vetor e vetor ordenado) e Propósito Especial (pilha, fila e fila circular)

	Inserção	Eliminação	Pesquisa
Vetor	O(1)	O(n)	O(n)
Pilha	O(1)	O(1)	
Fila	O(1)	O(n)	
Fila Circular	O(1)	O(1)	
Fila Prioridades	O(n)	O(1)	



Quando usar o quê?

Filas (estrutura de armazenamento de **propósito especial**)

	Inserção	Eliminação
Fila	O(1)	O(n)
Fila Circular	O(1)	O(1)
Fila Prioridades	O(n)	O(1)

- Fila e Fila Circular são uteis quando é desejável acesso apenas ao primeiro item de dados inserido (FIFO).
- A Fila de Prioridades é útil quando é desejável acesso rápido ao maior e menor item chave.
- Por ser implementada como um vetor simples essa estrutura possui inserção lenta, porém, comparada com outra estrutura de mesmo propósito (Heap, veremos na Aula 15), ela é mais simples e adequada quando o numero de itens não é alto ou a velocidade de inserção não é critica.
- Uma fila de prioridades é uma estrutura de proposito especial, diferente do vetor ordenado que é de proposito geral. Numa fila de prioridades a remoção de itens é sempre no final (maior ou menor item).

Atividades

1. Uma fila de prioridade poderia ser usada para manter:
   a) Passageiros a serem apanhados por um taxi em diferentes partes da cidade;
   b) Teclas pressionadas em um teclado do computador;
   c) Quadrados de um tabuleiro de xadrez em um programa de jogo;
   d) Planetas um uma simulação do sistema solar.
2. Verdadeiro ou falso: pelo menos um dos métodos na classe FilaPrior usa uma pesquisa linear.
3. Suponha que você insira 15, 25, 35 e 45 em uma fila circular. Então você remove três itens. Qual deles é deixado?

6. Construa um programa que leia uma linha de texto do teclado até que a tecla enter seja pressionada e separe suas letras em duas filas circulares: filaVogais e filaConsoantes. Após a separação, imprima o conteúdo de cada fila na tela (removendo-os).
5. Escreva um método para a classe FilaPrior que exiba o conteúdo do primeiro item inserido até o ultimo. Somente exiba, não remova-os.
6. Escreva um método para a classe FilaCircular que exiba o conteúdo da fila do primeiro item inserido até o ultimo. Somente exiba, não remova-os.

Lista

Definição:
Um tipo de estrutura de dados (conjunto de dados) dispostos e acessíveis para remoção e inserção de itens de maneira dinâmica.

Adicionar ou remover itens em qualquer ponto.

Cada item é alocado na memória de maneira individual, sendo assim, pode crescer (aumentar o tamanho) conforme o limite de memória disponível pelo sistema operacional.

Lista Simplesmente Encadeada

Lista encadeada simples ou simplesmente ligada.

Cada elemento possui um elo (endereço) para o próximo elemento.

Classe Nó:

```
class No {
   public int item;
   public No prox;
}
```

espaço para armazenamento da informação

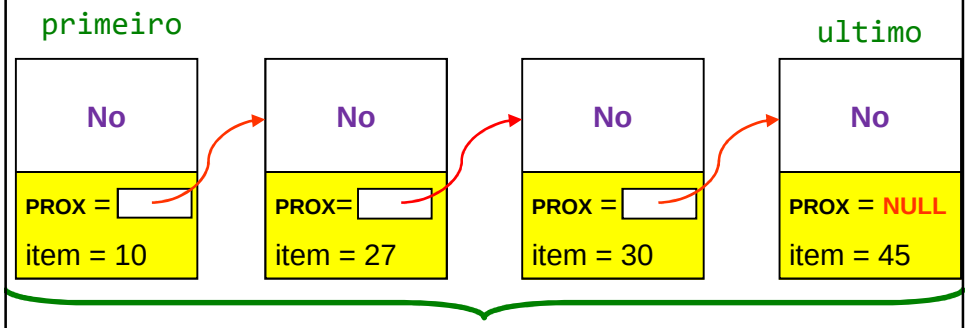
Um espaço para armazenar uma referência da localização na memória onde o próximo elemento se encontra.

Definição Nó em Java:

```
class No {
   public double item;
   public No prox;
}
```

É uma classe que contém um item de dados e um ou mais elos (vínculos).

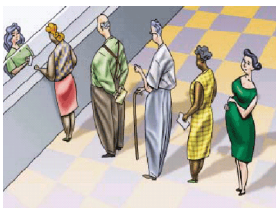
Sendo que este vínculo funciona como um ponteiro que aponta para a posição de outro nó na memoria do computador.



ESTRUTURA DE DADOS DO TIPO LISTA SIMPLESMENTE ENCADEADA
(todos os elementos ou nodos)

Estruturas de propósito especifico

Alocadas como vetor:



```
class Fila {
 public int[] item;
 public int fim, tam_max;
...
```

```
class Pilha {
 public int[] item;
 public int topo, tam_max;
...
```

Alocados como vetor:

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

Ocupados
Alocados

Fig. Considerando 30 posições da memória

Poderá crescer até o tamanho máximo de 10 elementos, que é o maior espaço livre continuo possível.

Estruturas de propósito especifico

Alocadas como variável:

Ao contrário do vetor, na estrutura de dados do tipo lista os itens são adicionados dinamicamente, ou seja, conforme a necessidade do usuário e da capacidade do sistema onde ela será operada.

```
class No {
 public int item;
 public No prox;
}
```

```
class No {
 public int item;
 public No ant;
}
```

Alocados como variável:

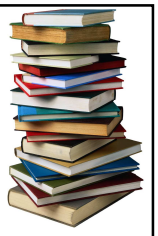
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30

Ocupados
Alocados

Fig. Considerando 30 posições da memória

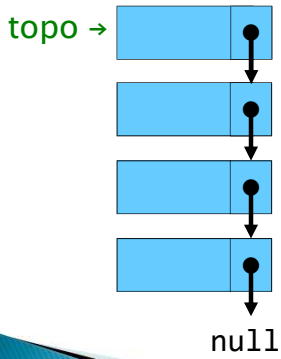
Poderá crescer até o tamanho máximo disponível na memória (nesse caso 23 espaços)

Pilha como Lista Encadeada



Política de escalonamento LIFO

Cada elemento possui um elo (endereço) para outro elemento.

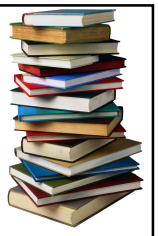


Elemento ou Nó para Pilha:

```
class No {
  public char item;
  public No ant;
}
```

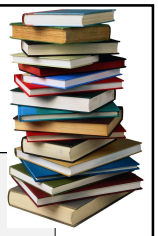
Elo para outro Nó

Classe para Pilha Encadeada



```
class PilhaSE {

  private No topo;

  public PilhaSE() { topo = null; }

  public boolean empty() { return (topo == null); }

  public char top()  { return topo.item; }

  public void pop() { if (!empty()) topo = topo.ant; }

  public void push(char valor) {
     No novo = new No(); // Aloca memoria para novo Nó
     novo.item = valor;
     novo.ant = topo;
     topo = novo;
  }

} // fim classe Pilha como Lista Encadeada
```

Exemplo de uso da Classe para Pilha Encadeada

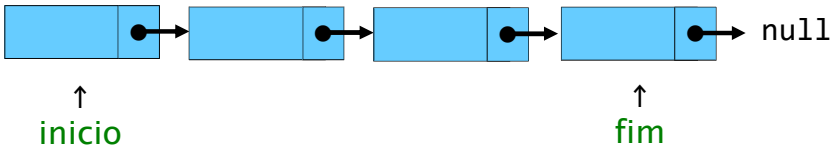


```
class PilhaSEApp {
   public static void main (String[] args) {

      PilhaSE p = new PilhaSE();

      System.out.println("Inserindo os caracteres: FATEC");
      p.push('F'); p.push('A'); p.push('T'); p.push('E'); p.push('C');

      System.out.print("Item do topo: " + p.top());

      System.out.print("\nRemovendo todos os caracteres: ");
      while(!p.empty()) {
         System.out.print(p.top());
         p.pop();
      }
      System.out.println("\n");

   }
}
```

Fila como Lista Encadeada

Política de escalonamento FIFO

Cada elemento possui um elo (endereço) para outro elemento.



Elemento ou Nó para Filha:

```
class No {
  public char item;
  public No prox;
}
```

Elo para outro Nó

Classe para Fila Encadeada

```
class FilaSE {
  private No inicio, fim;

  public FilaSE()            { inicio = fim = null;       }
  public boolean empty() { return (inicio == null); }
  public char front()        { return inicio.item;        }

  public void pop()     {
     if (empty()) return; // se vazio
     if (inicio.prox == null) fim = null; // se somente um item
     inicio = inicio.prox;
  }

  public void push(char valor) {
    No novo = new No(); // Aloca memoria para novo Nó
    novo.item = valor;
    novo.prox = null;
    if (empty()) inicio = novo; // se vazio
    else fim.prox = novo; // senão insere no fim
    fim = novo;
  }
} // fim classe FilaSE
```

Exemplo de uso da Classe para Fila Encadeada

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

      FilaSE f = new FilaSE();

      System.out.println("Inserindo os caracteres: FATEC");
      f.push('F'); f.push('A'); f.push('T'); f.push('E'); f.push('C');

      System.out.print("Item da frente: " + f.front());

      System.out.print("\nRemovendo todos os caracteres: ");
      while(!f.empty()) {
         System.out.print(f.front());
         f.pop();
      }
      System.out.println("\n");

   }
}
```

Tabela de Estruturas de Armazenamento

de Propósito Especial

	Inserção	Eliminação	Comentário
Pilha como vetor	O(1)	O(1)	Elimina item inserido mais recentemente
Pilha como Lista Encadeada	O(1)	O(1)	
Fila como vetor	O(1)	O(n)	Elimina item inserido menos recentemente
Fila Circular como vetor	O(1)	O(1)	
Fila como Lista Encadeada	O(1)	O(1)	

Atividades

1. Crie um programa que gerencie uma PILHA ENCADEADA de TAREFAS a serem cumpridas.
   As tarefas são Strings que descrevem uma ação a ser executada.
   - O usuário deverá ter duas opções:
   - Inserir tarefa na pilha; e
   - Obter a próxima tarefa da pilha.

2. Considere uma coleção de nomes de sites da web (String) e seus respectivos links na Internet (String) armazenados através de uma Lista Simplesmente Encadeada. Escreva um programa que leia do usuário o nome de um site, busque o seu link correspondente na lista, imprima na tela, e ao mesmo tempo mova o nó que contém o nome buscado para o início da lista, de forma que ele possa ser encontrado mais rapidamente na próxima vez que for buscado.

Atividades aula de ontem

3. Reescreva os métodos de inserção de itens da classe Lista Simplesmente Encadeada de Strings para que os mesmos não permitam a inserção de itens já existentes na lista.

4. Escreva um método para classe Lista Simplesmente Encadeada que retorne o ultimo item.

5. Escreva um método para classe Lista Simplesmente Encadeada que remova o ultimo item, chame-o de remove_ultimo.