UFSC-CTC-INE

INE5384 - Estruturas de Dados

# Listas como Vetores

Prof. Ronaldo S. Mello

2002/2

## Modelagem Física de Listas

- Duas alternativas de implementação:
  - Vetor (*array*)
  - Encadeamento (uso de ponteiros)

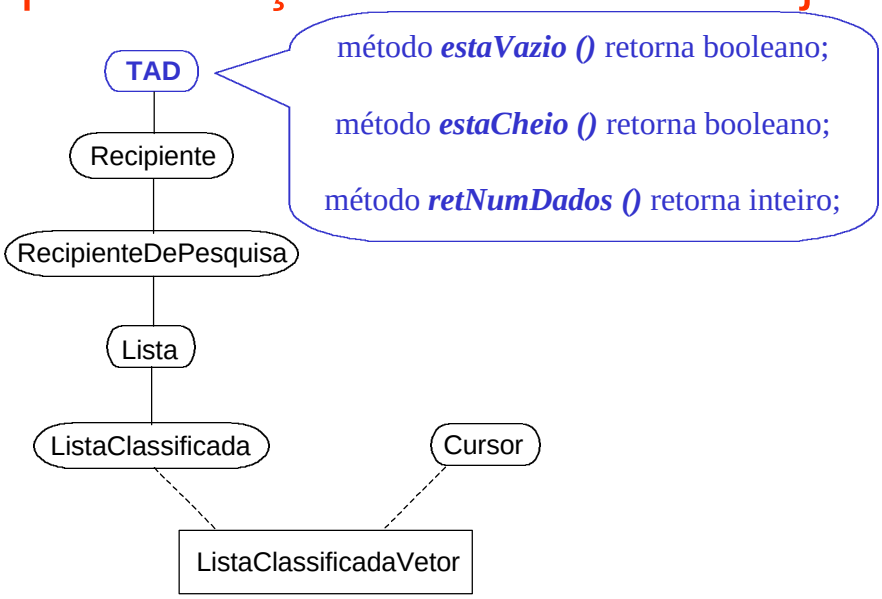
# Listas na Forma de Vetores

- Exemplo:

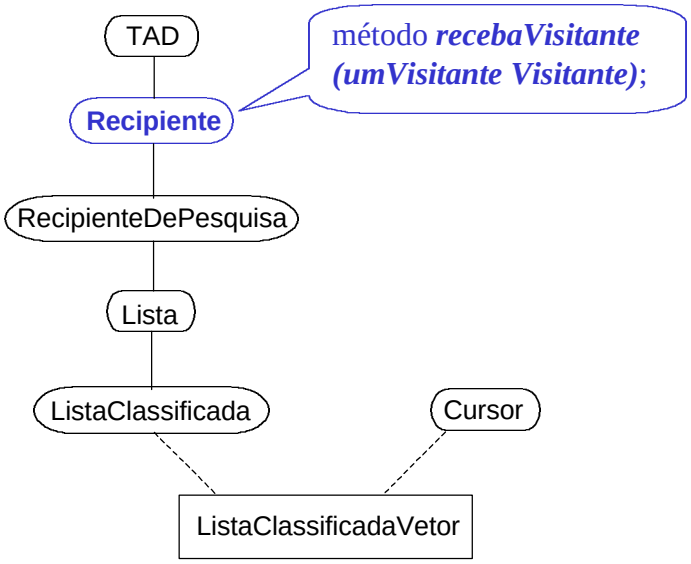
| | 0 | 1 | 2 | . . . | 9 |
|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | . . . | ^ |

- Características:
  - Lista possui tamanho máximo predefinido
  - Posições contíguas de memória
  - Acesso direto a um elemento (posições são indexadas – Exemplo: Vetor[2] = 7)

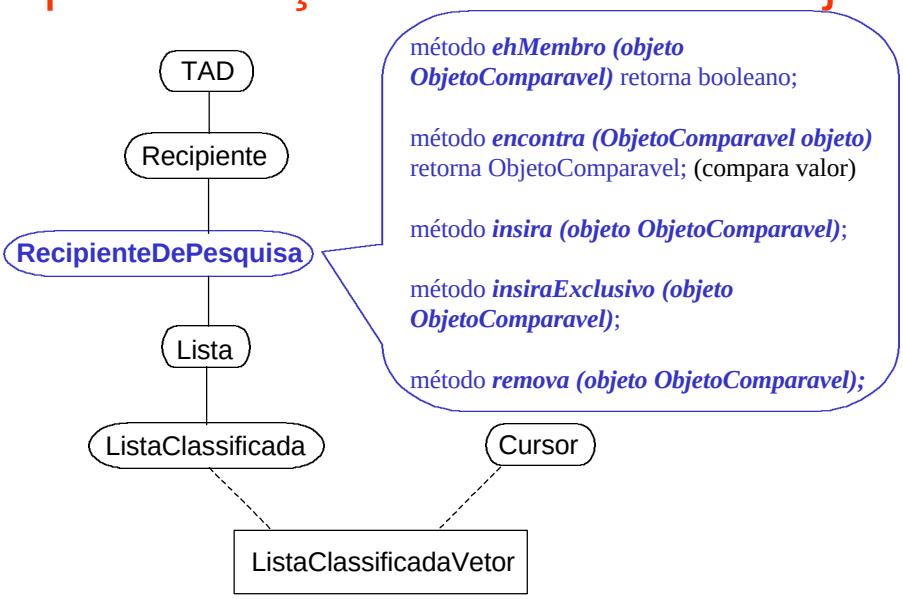
# Implementação – Padrão de Projeto



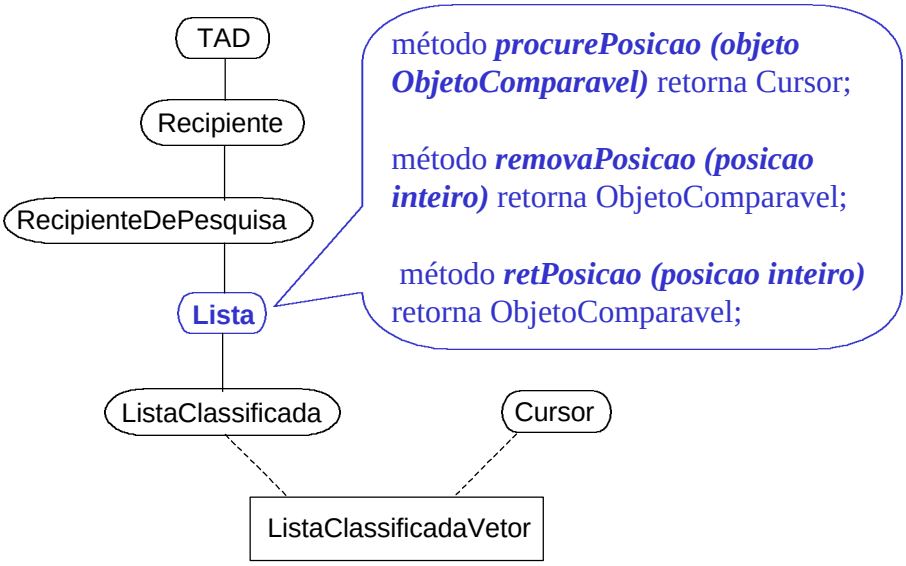
# Implementação – Padrão de Projeto



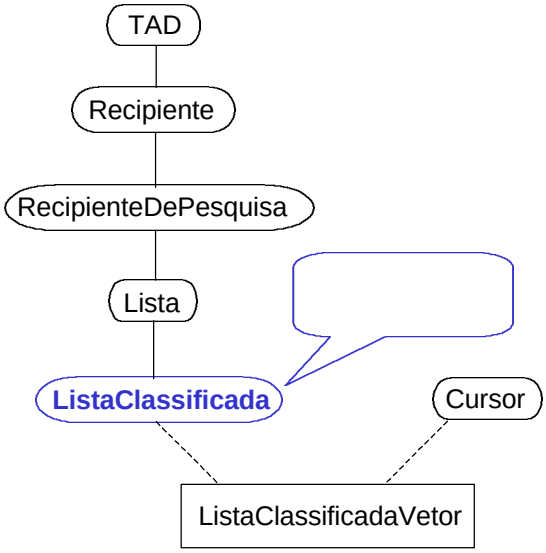

# Implementação – Padrão de Projeto



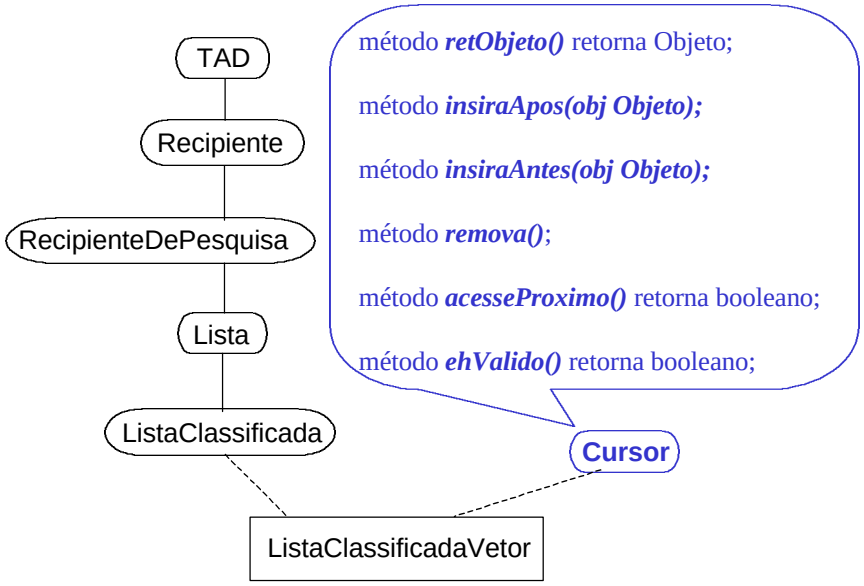
# Implementação – Padrão de Projeto




# Implementação – Padrão de Projeto



# Implementação – Padrão de Projeto




# Operações sobre o Vetor

- Supõe-se os seguintes dados:
  - Vetor (atributo ***array***)
  - Comprimento (valor retornado pelo método ***lenght***)
  - NroElementos (atributo ***count***)
- Exemplo:

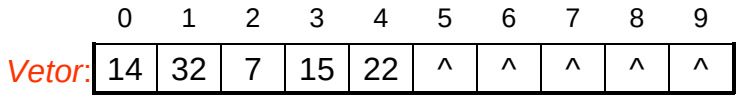
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | 15 | 22 | ^ | ^ | ^ | ^ | ^ |

*Comprimento*: 10

*NroElementos*: 5

# Inserção de Elemento

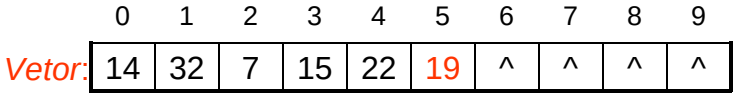
- Exemplo: inserir o número 19 (no final)




*Comprimento*: 10

*NroElementos*: 5

Inserir em *Vetor* [ *NroElementos* ]
Somente se *NroElementos* < *Comprimento*




*Comprimento*: 10

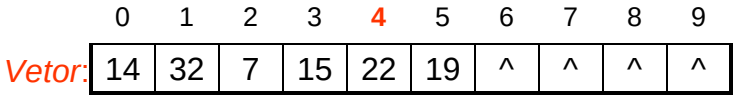
*NroElementos*: 5  6

# Implementação

```
Classe ListaClassificadaVetor
   Subclasse de RecipienteDePesquisaAbstrato
   Implementa ListaClassificada
início
   ...
   método adicioneNoFinal (objeto ObjetoComparável);
   início
         se NroElementos = Vetor.lenght então
                Exceção EstruturaCheia();
         Vetor [ NroElementos ] ← objeto;
         NroElementos ← NroElementos + 1;
   fim;
fim
```

# Inserção em uma Posição “x”

- Exemplo: inserir o número 67 na posição 4



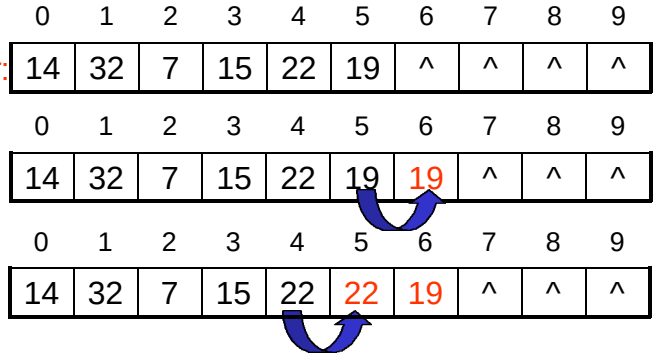

*Comprimento*: 10
*NroElementos*: 6

Inserir em: *Vetor [ posição ]*
Somente se: *posição > 0 E*
*posição <= NroElementos*
*E*
*NroElementos < Comprimento*

***Como proceder a inserção???***

# Inserção em uma Posição “x”

- Exemplo: inserir o número 67 na posição 4




*Comprimento*: 10
*NroElementos*: 6

*Nro Deslocamentos?* ***NroElementos – Posição***

# Inserção em uma Posição “x”

- Exemplo: inserir o número 67 na posição 4

| | 0 | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | 15 | 22 | 22 | 19 | ^ | ^ | ^ |

*Comprimento*: 10
*NroElementos*: 6

Feitos os deslocamentos, o elemento é inserido em *Vetor [ posição ]*

| | 0 | 1 | 2 | 3 | **4** | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | 15 | **67** | 22 | 19 | ^ | ^ | ^ |

*Comprimento*: 10
*NroElementos*: 6 → 7

# Implementação

```
Classe ListaClassificadaVetor    . . .
início
    classe MeuCursor implementa Cursor
    início
        Posição, i inteiro;

        método insiraAntes (objeto ObjetoComparável);
        início
          se Posição < 0 OU Posição >= NroElementos então
                Exceção OperaçaoIlegal();
          se NroElementos = Vetor.lenght então
                 Exceção EstruturaCheia();
          para i de NroElementos até Posição faça
                Vetor [ i ] ← Vetor [ i-1 ];
          Vetor [ Posição ] ← objeto;
          NroElementos ← NroElementos + 1;
        fim;
    fim;
fim
```

# Exclusão de Elemento

- Exemplo: exclusão no final (último elemento)

| | 0 | 1 | 2 | 3 | 4 | 5 | **6** | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | 15 | 67 | 22 | **19** | ^ | ^ | ^ |

*Comprimento*: 10

*NroElementos*: 7

Excluir em? *Vetor [ NroElementos - 1]*
Somente se? *NroElementos > 0*
O que fazer? *Vetor [ NroElementos - 1]* ¬ ^
*NroElementos* ¬ *NroElementos - 1*

| | 0 | 1 | 2 | 3 | 4 | 5 | **6** | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| *Vetor*: | 14 | 32 | 7 | 15 | 67 | 22 | ^ | ^ | ^ | ^ |

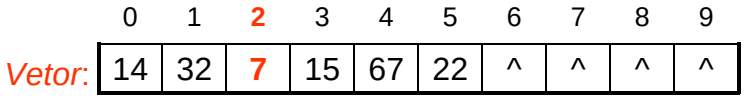
*Comprimento*: 10 *NroElementos*: 7 → 6

# Implementação

```
Classe ListaClassificadaVetor
   Subclasse de RecipienteDePesquisaAbstrato
   Implementa ListaClassificada
início
   ...
   método excluiNoFinal();
   início
        se NroElementos = 0 então
                Exceção EstruturaVazia();
        Vetor [ NroElementos -1 ] ← null;
        NroElementos ← NroElementos - 1;
   fim;
fim
```

# Exclusão em uma Posição “x”

- Exemplo: exclusão do número 7 (posição 2)




*Comprimento*: 10

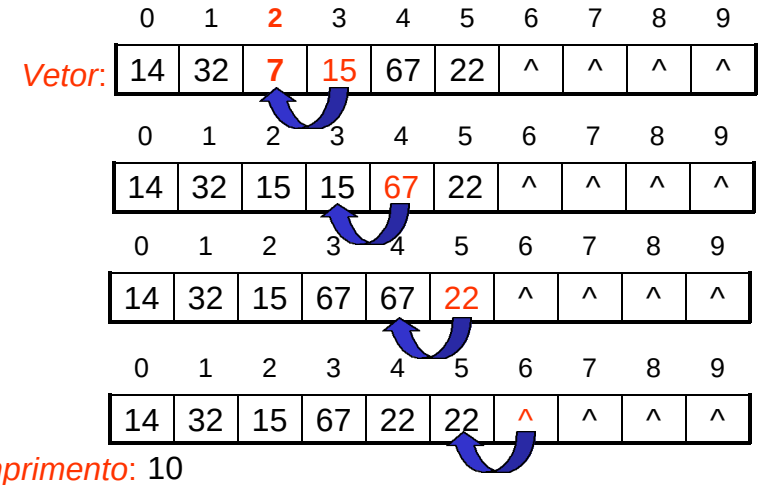
*NroElementos*: 6

Excluir somente se? *NroElementos > 0 E posição > 0 E posição < NroElementos*

***Como proceder a exclusão???***

# Exclusão em uma Posição “x”

- Exemplo: exclusão do número 7 (posição 2)



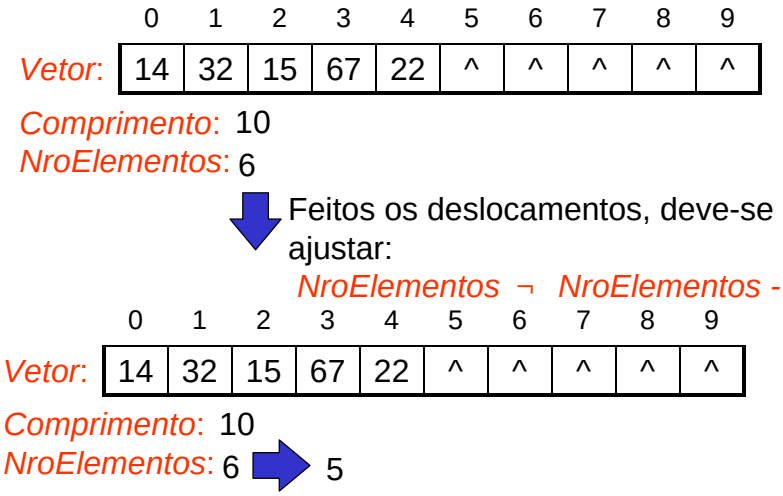

*Comprimento*: 10

*NroElementos*: 6

*Nro Deslocamentos?* ***NroElementos - Posição***

# Exclusão em uma Posição “x”

- Exemplo: exclusão do número 7 (posição 2)




# Implementação

```
Classe ListaClassificadaVetor    . . .
início
    classe MeuCursor implementa Cursor
    início
          Posição, i inteiro;

          método remove ();
          início
            se NroElementos = 0 então
                    Exceção EstruturaVazia();
            se Posição < 0 OU Posição >= NroElementos então
                    Exceção OperaçaoIlegal();
            para i de Posição até NroElementos faça
                    Vetor [ i ] ← Vetor [ i+1 ];
            NroElementos ← NroElementos - 1;
          fim;
    fim;
fim
```

# Operações de Consulta

- Consulta1: consulta a um elemento (objeto) em uma posição “x”

  Obtém o elemento (objeto) na posição “x” somente se? *posição > 0 E*
  *posição < NroElementos E*
  *NroElementos > 0*

(Método ***get*** do livro e ***retPosicao*** da interface Lista)

# Implementação

```
Classe ListaClassificadaVetor
    Subclasse de RecipienteDePesquisaAbstrato
    Implementa ListaClassificada
início
    ...
    método retPosicao(posicao inteiro) retorna ObjetoComparavel;
    início
        se NroElementos = 0 então
                Exceção EstruturaVazia();
        se posicao < 0 OU posicao >= NroElementos então
                Exceção OperaçãoIlegal();
        retorna Vetor [posicao];
    fim;
fim
```

# Operações de Consulta

- Consulta2: consulta a posição de um dado elemento (objeto)

  Obtém a posição de um dado elemento (objeto) somente se? *Elemento (objeto) está na lista e NroElementos > 0*

(Método ***findPosition*** do livro e ***procurePosição*** da interface Lista)

# Implementação

```
Classe ListaClassificadaVetor
    Subclasse de RecipienteDePesquisaAbstrato
    Implementa ListaClassificada
início
    método procurePosicao(objeto ObjetoComparavel) retorna Cursor;
    início
            i inteiro;
            se NroElementos = 0 então Exceção EstruturaVazia();
            i ← 0;
            enquanto i < NroElementos faça
            início
              se Vetor [ i ].ehIg(objeto) = VERDADEIRO então
                        retorna NOVO MeuCursor (i);
              i ← i + 1;
            fim;
            retorna NULL;
    fim;
fim
```

# Operações de Consulta

- Consulta3: busca de um elemento com um certo valor na lista - *consulta por valor*

  Retorna um elemento (objeto) cujo valor é igual ao valor de um elemento (objeto) informado. Caso não existe tal objeto, retorna-se NULL.

(Método ***find*** do livro)

# Implementação

```
Classe ListaClassificadaVetor
    Subclasse de RecipienteDePesquisaAbstrato
    Implementa ListaClassificada
início
    ...
    método encontra(objeto ObjetoComparavel) retorna
    ObjetoComparavel;
    início
          i inteiro;

          para i de 0 até NroElementos - 1 faça
            se Vetor [ i ].ehIg(objeto) = VERDADEIRO
                  retorna Vetor [ i ];
          retorna NULL;
    fim;
fim
```

# Operações de Consulta

- Consulta4: verifica se um elemento (objeto) está na lista - *consulta por igualdade de objeto*

  Retorna VERDADEIRO se um elemento (objeto) está na lista. Caso contrário, retorna FALSO.

(Método ***isMember*** do livro e ***ehMembro*** da Interface RecipienteDePesquisa)

# Implementação

```
Classe ListaClassificadaVetor
    Subclasse de RecipienteDePesquisaAbstrato
    Implementa ListaClassificada
início
    ...
    método ehMembro(objeto ObjetoComparavel) retorna
    booleano;
    início
          i inteiro;

          para i de 0 até NroElementos - 1 faça
            se Vetor [ i ] = = objeto então retorna VERDADEIRO;
          retorna FALSO;
    fim;
fim
```