ED – Lista Encadeada ou Lista Ligada

Implementação da Lista Encadeada ou Lista Ligada

1) Implementar a classe Node, contendo:
   Atributos:
     info – tipo int
     next – tipo Node

   Construtor que recebe o valor de info do nó, atribui esse valor ao atributo info e null para next
   Propriedades

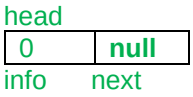
2) Implementar a classe ListaLigada, contendo:
   Atributo:
     head – tipo Node

   Construtor:
     Cria um nó, com valor de info igual a 0 ou null (dependendo do tipo do info do Node) e atribui esse nó para head (nó cabeça da lista)

```
ListaLigada()
  head← new Node(0);
```

Exemplo: Supondo que criamos um objeto lista da classe ListaLigada

```
ListaLigada lista = new ListaLigada( );
```

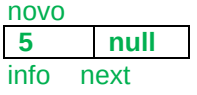


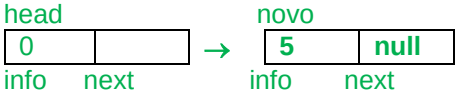
   Métodos:
   a) void InsereNode (int valor)
      - cria um objeto **novo** da classe Node com info igual ao valor
      - insere o novo nó na lista encadeada
        atribui para next de **novo** o conteúdo de head.Next
        atribui para next de head o **novo**

```
void InsereNode(int valor)
  Node novo ← new Node(valor);
  novo.Next←head.Next;
  head.Next← novo;
```

Exemplo: Supondo que queremos inserir o valor 5 na lista.

```
lista.InsereNode(5);
```

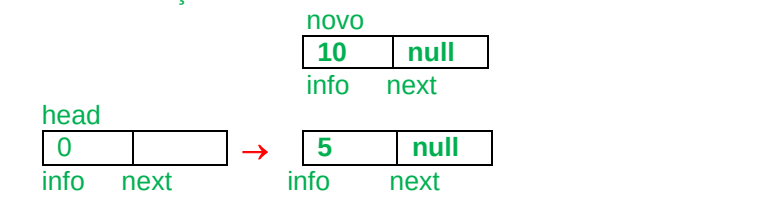




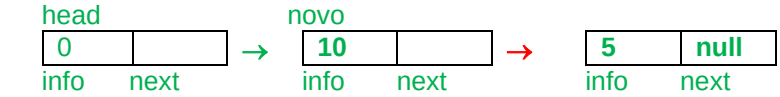
Exemplo: Supondo que agora queremos inserir o valor 10 na lista.

```
lista.InsereNode(10);
```

Antes da inserção:



Após inserção:



b) void Imprime ( )

- percorre a lista encadeada, imprimindo seus valores
  - atribui para a variável atual (do tipo Node) o conteúdo de head.Next
  - enquanto atual for diferente de null,
    - imprime o valor do elemento desse nó,
    - atribui para atual o conteúdo de atual.Next

```
void Imprime()
  Node atual←head.Next;
  enquanto atual ≠ null faça
   início
     exibe(atual.Info);
     atual←atual.Next;
   fim
```

c) Node BuscaNode (int valor)

- percorre a lista encadeada, verificando se existe um nó com o valor passado como parâmetro
- Se existir, devolve o endereço desse nó, senão devolve null

```
Node BuscaNode(int valor)
  Node atual←head.Next;
  enquanto atual ≠ null faça
   início
     se atual.Info = valor
     então retorna atual;
     senão atual←atual.Next;
   fim
  retorna null;
```

d) bool RemoveNode (int valor)

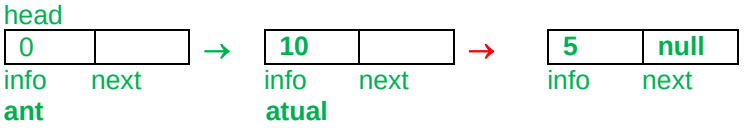
- percorre a lista encadeada, verificando se existe um nó com o valor passado como parâmetro
- Se existir, remove esse nó da lista e devolve true
- Se não existir, devolve false

```
bool RemoveNode(int valor)
  Node ant ← head;
  Node atual ← head.Next;
  enquanto atual ≠ null faça
    início
      se atual.Info = valor
      então início
              ant.Next←atual.Next;
              retorna true;
            fim
      senão início
              ant← atual;
              atual←atual.Next;
            fim
    fim
  retorna false;
```

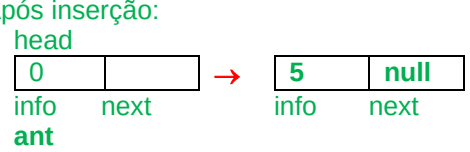
Exemplo: Supondo que queremos remover o valor 10 da lista.

```
lista.RemoveNode(10);
```

Antes da remoção:



Após inserção:



e) int Tamanho( )

- percorre a lista encadeada, calcula e devolve o tamanho da lista

```
int Tamanho()
  Node atual←head.Next;
  int tam ← 0;
  enquanto atual ≠ null faça
    início
      tam←tam + 1;
      atual←atual.Next;
    fim
  retorna tam;
```

3) Testar a classe ListaLigada, criando um objeto ListaLigada, e inserindo vários valores.

- Imprima os valores para verificar se está inserindo corretamente.
- Depois teste os métodos BuscaNode e RemoveNode, sempre imprimindo os valores da lista, para verificar se está executando corretamente.