

Lista Simplesmente Encadeada

Estrutura de Dados I
Profa. Janaina Fontana Biffi Duarte

Introdução

- Definição

- Lista é uma estrutura que armazena um conjunto de elementos em sequência.

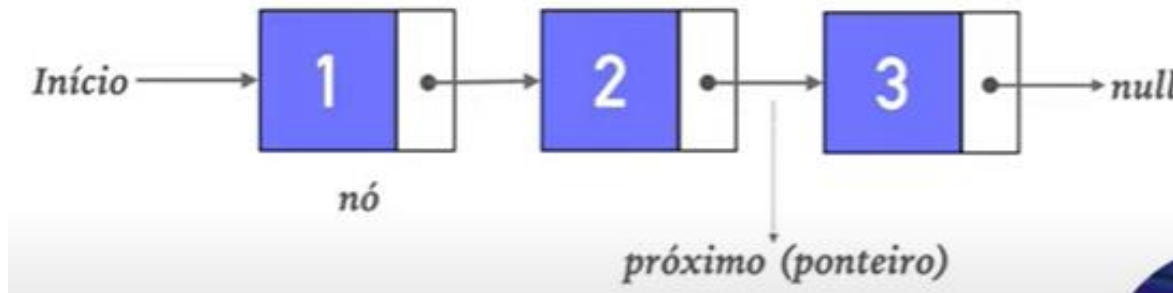
- O VETOR é uma LISTA?

Sim, pois além de ser um conjunto de elementos do mesmo tipo, eles estão em sequência, fisicamente um ao lado do outro na memória.

Introdução

E se os elementos não estiverem um do lado do outro, posso ter uma LISTA?

Sim, desde que eu pegue cada um dos elementos e rotule quem é o próximo, criando uma **LISTA ENCADEADA**.



Nesse caso não temos uma ordem física, mas sim uma ordem virtual.

Introdução

- As listas encadeadas são excelentes estruturas para **alocarmos dinamicamente** a memória.
- Quando temos um vetor seu tamanho é fixo.
 - Caso seja necessário incluir mais elementos do que o tamanho do vetor, não será possível.
 - Caso o espaço definido seja subutilizado, teremos um desperdício de memória.

Introdução

- Utilizando as listas encadeadas podemos adicionar e remover elementos sob demanda, de forma que:
 - Conforme forem sendo adicionados mais elementos é utilizada mais memória e;
 - Conforme forem sendo removidos elementos, a memória vai sendo liberada.

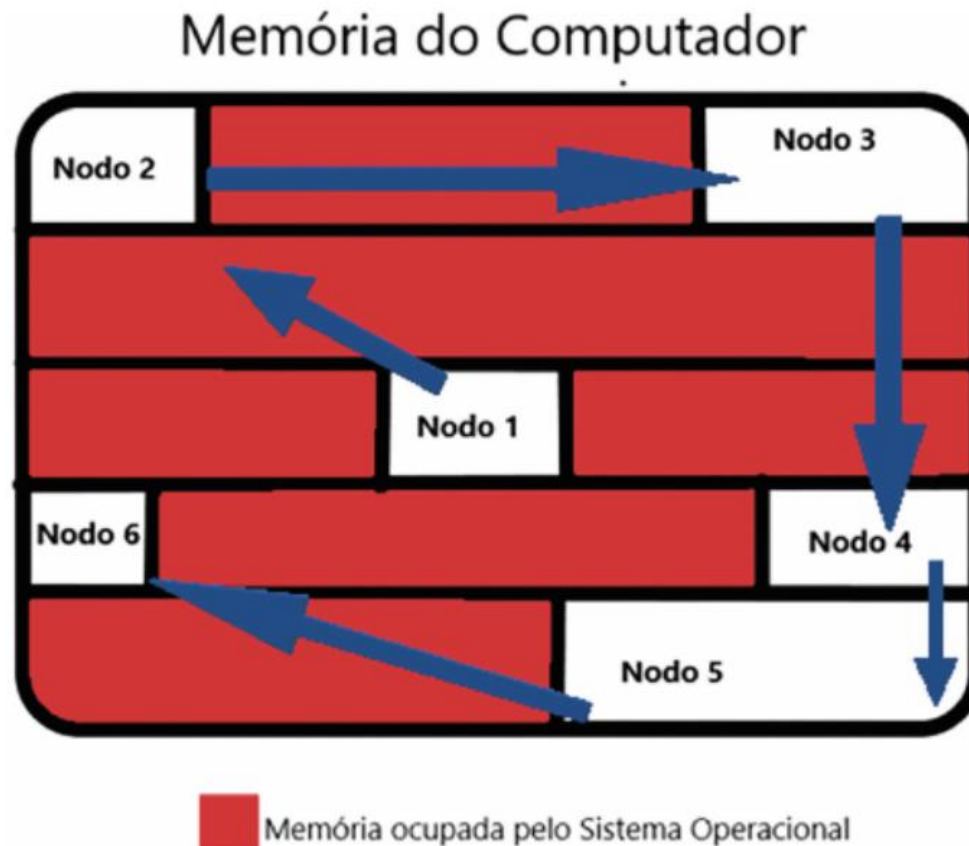
Lista Linear Encadeada

“Listas lineares ligadas ou encadeadas são listas dinâmicas, isto é, seu tamanho pode aumentar ou diminuir no momento da execução de um programa, e cada nó é formado por dois campos: o dado propriamente dito e um ponteiro. O ponteiro contém o endereço do próximo nó da lista.”

Características

- Estrutura dinâmica;
- Cada item em uma lista é conhecido como **nó** ou **célula**;
- Uma lista que não tem nós é chamada de **vazia** ou **nula**;
- Ponteiros;
- Cada nó faz referência ao próximo nó (sucessor).
- Sabemos que a lista terminou quando o ponteiro para o próximo elemento é **null**.
- Cabeça da lista: primeiro nó
- Cauda da lista: restante dos nós

Posição dos Nós na Memória



Operações sobre listas

- Inserir no início da lista
- Inserir no final da lista
- Inserir em posição específica
- Quantidade de nós (tamanho)
- Percorrer a lista (Imprimir)
- Pesquisar um nó ou uma posição específica
- Remover do início da lista
- Remover do final da lista
- Remover de qualquer posição

Mão na massa

PASSO 1 - Fazer um programa principal que atenda ao menu:

- 1 - Inserir no início da lista
- 2 - Inserir no final da lista
- 3 - Inserir em posição específica
- 4 - Quantidade de nós (tamanho)
- 5 - Percorrer a lista (Imprimir)
- 6 - Pesquisar um nó ou uma posição específica
- 7 - Remover do início da lista
- 8 - Remover do final da lista
- 9 - Remover de qualquer posição
- 10 - Sair

Programa principal - 1

```
1  package app;
2
3  import java.util.Scanner;
4
5  public class Program {
6      Run | Debug
7      public static void main(String[] args) {
8          Scanner sc = new Scanner(System.in);
9
10         int op;
11         do {
12             showMenu();
13             op = sc.nextInt();
14
15             switch (op) {
16                 case 1: {
17                     System.out.print("Entrei na opção 1");
18                     break;
19                 }
20                 case 2: {
21                     System.out.println("Entrei na opção 2");
22                     break;
23                 }
24             }
25         } while (op != 0);
26     }
27 }
```

Programa principal - 2

```
23     case 3: {
24         System.out.println("Entrei na opção 3");
25         break;
26     }
27     case 4: {
28         System.out.println("Entrei na opção 4");
29         break;
30     }
31     case 5: {
32         System.out.println("Entrei na opção 5");
33         break;
34     }
35     case 6: {
36         System.out.println("Entrei na opção 6");
37         break;
38     }
39     case 7: {
40         System.out.println("Entrei na opção 7");
41         break;
42     }
```

Programa principal - 3

```
43         case 8: {
44             System.out.println("Entrei na opção 8");
45             break;
46         }
47         case 9: {
48             System.out.println("Entrei na opção 9");
49             break;
50         }
51         case 10: {
52             System.out.println("Fim do programa!");
53             break;
54         }
55         default:
56             System.out.println("Opcao inválida!");
57     }
58     while (op != 10);
59
60     sc.close();
61 }
62
```

Programa principal - 4

```
63  public static void showMenu() {  
64      System.out.println("1 - Inserir no início da lista");  
65      System.out.println("2 - Inserir no final da lista");  
66      System.out.println("3 - Inserir em posição específica");  
67      System.out.println("4 - Quantidade de nós (tamanho)");  
68      System.out.println("5 - Percorrer a lista");  
69      System.out.println("6 - Pesquisar um nó ou uma posição específica");  
70      System.out.println("7 - Remover do início da lista");  
71      System.out.println("8 - Remover do final da lista");  
72      System.out.println("9 - Remover de qualquer posição");  
73      System.out.println("10 - Sair");  
74  }  
75 }  
76
```

Mão na massa


PASSO 2 - Implementar a classe do Nó

```
1  package util;
2
3  public class Node {
4
5      private Double value; //Atributo valor
6      private Node next;    //Aponta para o próximo nó, que também é do tipo Node
7
8      public Double getValue() {
9          return value;
10     }
11
12     public void setValue(Double value) {
13         this.value = value;
14     }
15
16     public Node getNext() {
17         return next;
18     }
19
20     public void setNext(Node next) {
21         this.next = next;
22     }
23 }
24
```

Mão na massa

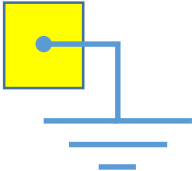
PASSO 3 - Implementar a classe List

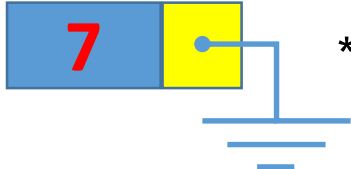
Esboço inicial

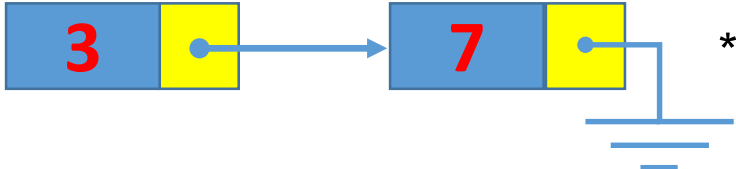
```
1  package util;
2
3  public class List {
4
5      private Node head;
6
7      public void addIni(Double value) {
8          //1 - Inserir no início da lista
9      }
10
11 } 
```


Mão na massa

1 – Inserir no início da lista

List =  * Situação inicial

List =  * Inserção do primeiro elemento

List =  * Inserção do segundo elemento no início

Mão na massa

1 – Inserir no início da lista

- Passos:
 - Instanciar um novo nó (Node);
 - Atribui o dado ao nó;
 - Faz ele apontar para a antiga cabeça da lista;
 - O nó inserido passa a ser a cabeça.

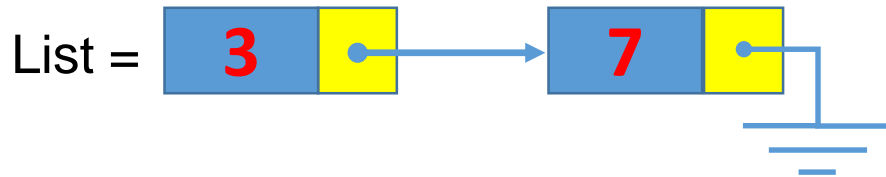
Mão na massa

1 – Inserir no início da lista

```
1  package util;
2
3  public class List {
4
5      private Node head;
6
7      public void addIni(Double value) {
8          //1 - Inserir no início da lista
9          Node node = new Node();
10         node.setValue(value);
11         node.setNext(head);
12         head = node;
13     }
14 }
```

Mão na massa

5 – Percorrer a lista (Imprimir)



Saída da impressão: [3.0 7.0]

Para isso, vamos reescrever o método ***toString()*** da classe, para que ele concatene todos os elementos de dentro da Lista em uma única String.

Mão na massa

5 – Percorrer a lista (Imprimir)

- Passos:
 - Iniciar pela cabeça da lista;
 - Inicia a string da saída com "[";
 - Enquanto o ponteiro para o próximo Node não for nulo:
 - Concatena na string;
 - Vai para o próximo Node.
 - Concatena na string da saída "]"

Mão na massa

5 – Percorrer a lista (Imprimir)

```
15  @Override
16  public String toString() {
17      //5 - Percorrer a lista (Imprimir)
18
19      //Aqui estamos utilizando a classe StringBuffer
20      //que é muito útil para otimizar a construção de Strings potencialmente grandes
21      StringBuffer sb = new StringBuffer();
22      sb.append("[");
23
24      Node p = head;
25      while (p != null) {
26          sb.append(p.getValue() + " ");
27          p = p.getNext();
28      }
29
30      sb.append("]");
31      return sb.toString();
32  }
33 }
```

Mão na massa – Testes

Testando as opções 1 e 5

- Alterando o programa principal

```
1  package app;
2
3  import java.util.Scanner;
4  import util.List;
5
6  public class Program {
7      Run | Debug
8      public static void main(String[] args) {
9          Scanner sc = new Scanner(System.in);
10
11         List list = new List();
12
13         int op;
14         do {
15             showMenu();
16             op = sc.nextInt();
17
18             switch (op) {
19                 case 1: {
20                     System.out.print("Digite um número: ");
21                     double value = sc.nextDouble(); //lê o número informado
22                     list.addIni(value);
23                     break;
24                 }
25             }
26         }
27     }
28 }
```

Mão na massa – Testes

Testando as opções 1 e 5

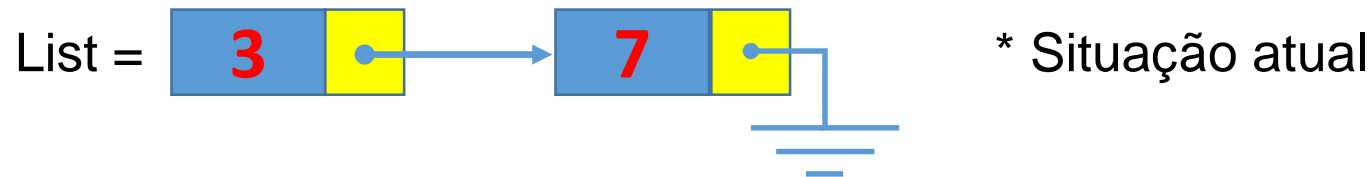
- Alterando o programa principal

```
35     }  
36     case 5: {  
37         System.out.println(list.toString());  
38         break;  
39     }
```


.... Continuamos na próxima aula

Mão na massa

2 – Inserir no final da lista



* Inserção do terceiro elemento no final

Mão na massa

2 – Inserir no final da lista

- Passos:
 - Instanciar um novo nó (Node);
 - Atribui o dado ao nó;
 - Alterar o ponteiro do último nó para que aponte para o novo nó;
 - O último nó passa a ser o nó inserido.

Mão na massa

```
1  package util;
2
3  public class List {
4
5      private Node head;
6      private Node lastNode;
7      private int totalDeElementos = 0;
8
9      public void addFim(Double value) {
10         //2 - Inserir no final da lista
11         if (totalDeElementos == 0) {
12             addIni(value);
13         } else {
14             Node node = new Node();
15             node.setValue(value);
16             lastNode.setNext(node);
17             lastNode = node;
18             totalDeElementos++;
19         }
20     }
```

Mão na massa

- Alterações na função addIni por conta da criação do lastNode e totalDeElementos.

```
22     public void addIni(Double value) {
23         //1 - Inserir no início da lista
24         Node node = new Node();
25         node.setValue(value);
26         node.setNext(head);
27         head = node;
28
29         if (totalDeElementos == 0) {
30             lastNode = head;
31         }
32         totalDeElementos++;
33     }
```

Mão na massa – Testes

Testando a opção 2

- Alterando o programa principal

```
24     case 2: {  
25         System.out.print("Digite um número: ");  
26         double value = sc.nextDouble(); //lê o número informado  
27         list.addFim(value);  
28         break;  
29     }
```

Mão na massa

6 – Pesquisar um nó ou uma posição específica

```
3 public class List {
4
5     private Node head;
6     private Node lastNode;
7     private int totalDeElementos = 0;
8     → private final String NAO_EXISTE = "Posição não existe.";
9 }
```

==

```
75 public Node buscaNoPosicao(int posicao) {
76     //6 - Pesquisar um nó ou uma posição específica
77     if (!(posicao >= 0 && posicao <= totalDeElementos)) {
78         throw new IllegalArgumentException(NAO_EXISTE);
79     }
80     Node noAtual = head;
81     for (int i = 0; i < posicao; i++) {
82         noAtual = noAtual.getNext();
83     }
84     return noAtual;
85 }
```

Mão na massa

6 – Pesquisar um nó ou uma posição específica

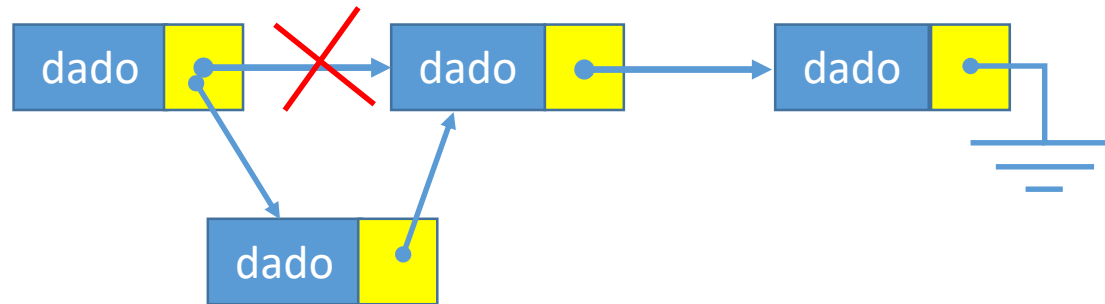
Testando a opção 6

- Alterando o programa principal

```
46     case 6: {  
47         System.out.print("Digite uma posição: ");  
48         int posicao = sc.nextInt(); //lê o número informado  
49         System.out.print("O valor da posição "+posicao+" é: "+list.buscaNoPosicao(posicao).getValue());  
50         break;  
51     }
```


Mão na massa

3 – Inserir em uma posição específica (ex. Posição 1)



Mão na massa

3 – Inserir em uma posição específica

- **Passos:**
 - Percorrer a lista até chegar no nó anterior;
 - Instanciar um novo nó;
 - Atribui o dado ao nó;
 - Alterar o ponteiro do novo nó para que aponte para o mesmo ponteiro do nó anterior;
 - Alterar o ponteiro do nó anterior para que aponte para o novo nó.

Mão na massa

```
36 public void addPosicao(int posicao, Double value) {
37     //3 - Inserir em posição específica
38     if (posicao == 0) { //está pedindo para adicionar na primeira posição
39         addIni(value);
40     } else if (posicao == totalDeElementos) { //está pedindo para adicionar no fim
41         addFim(value);
42     } else { //adicionar no meio
43         Node nodeAnterior = buscaNoPosicao(posicao - 1);
44         Node nodeNovo      = new Node();
45         nodeNovo.setValue(value);
46         nodeNovo.setNext(nodeAnterior.getNext());
47         nodeAnterior.setNext(nodeNovo);
48         totalDeElementos++;
49     }
50 }
```

Mão na massa – Testes

Testando a opção 3

- Alterando o programa principal

```
30     case 3: {  
31         System.out.print("Digite uma posição: ");  
32         int posicao = sc.nextInt(); //lê o número informado  
33         System.out.print("Digite um valor: ");  
34         Double value = sc.nextDouble(); //lê o número informado  
35         list.addPosicao(posicao, value);  
36         break;  
37     }
```

Mão na massa

4 – Quantidade de nós (tamanho)

- Esta função não tem segredo, pois já temos um atributo que possui a informação.

```
49 public int tamanho() {  
50     //4 - Quantidade de nós (tamanho)  
51     return totalDeElementos;  
52 }  
53
```

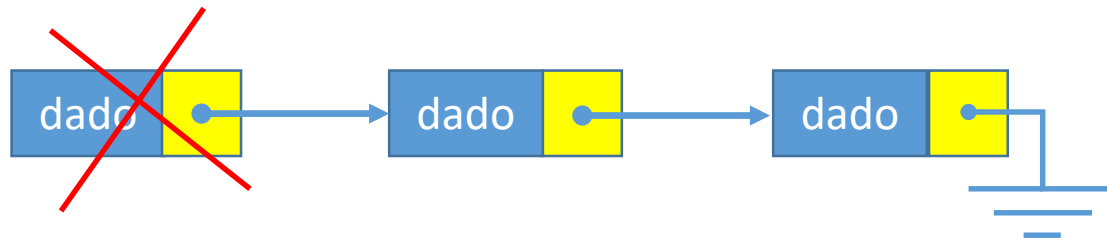
Testando a opção 4

- Alterando o programa principal

```
34  ▾  
35  ⚡ case 4: {  
36      System.out.print("O número de elementos da lista é: "+list.tamanho());  
37      break;  
38  }
```

Mão na massa

7 – Remover do início da lista



Mão na massa

7 – Remover do início da lista

- Passos:
 - Alterar a cabeça da lista, para que ela seja o nó para onde a cabeça está apontando no momento;

Mão na massa

```
3 public class List {
4
5     private Node head;
6     private Node lastNode;
7     private int totalDeElementos = 0;
8     private final String NAO_EXISTE = "Posição não existe.";
9     → private final String LISTA_VAZIA = "Lista está vazia.";
10 }
```

...

```
88 public void removeInicio() {
89     //7 - Remover do início da lista"
90     if (totalDeElementos == 0) {
91         throw new RuntimeException(LISTA_VAZIA);
92     }
93     head = head.getNext();
94     totalDeElementos--;
95
96     if (totalDeElementos == 0) {
97         lastNode = null;
98     }
99 }
```


Mão na massa – Testes

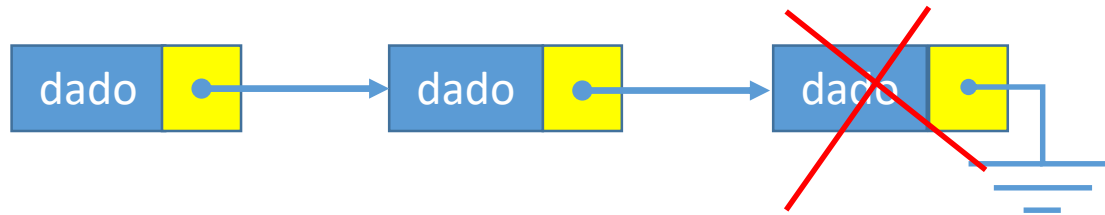
Testando a opção 7

- Alterando o programa principal

```
52     case 7: {  
53         list.removeInicio();  
54         break;
```

Mão na massa

8 – Remover do final da lista



Mão na massa

8 – Remover do final da lista

- Passos:
 - Posiciona no penúltimo nó;
 - Seta null para o ponteiro;
 - Atualiza o último da lista para ser o penúltimo nó.

Mão na massa

```
101 public void removeFinal() {  
102     //8 - Remover do final da lista  
103     if (totalDeElementos == 0) {  
104         throw new RuntimeException(LISTA_VAZIA);  
105     }  
106     if (totalDeElementos == 1) {  
107         removeInicio();  
108     } else {  
109         Node nodepenultimo = buscaNoPosicao(totalDeElementos - 2);  
110         nodepenultimo.setNext(next:null);  
111         lastNode = nodepenultimo;  
112         totalDeElementos--;  
113     }  
114 }
```

Mão na massa – Testes

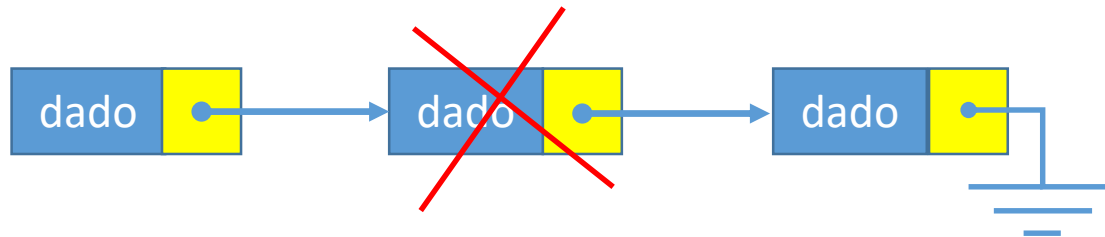
Testando a opção 8

- Alterando o programa principal

```
56      case 8: {  
57          list.removeFinal();  
58          break;
```

Mão na massa

9 – Remover de uma posição específica



Mão na massa

9 – Remover de uma posição específica

- Passos:
 - Localiza o nó anterior (noAnterior);
 - Localiza o nó atual (noAtual) (por meio do ponteiro do noAnterior);
 - Localiza o próximo nó (noProximo) (por meio do ponteiro do noAtual);
 - Atualiza a informação do ponteiro do noAnterior para o noProximo;
 - Atualiza o ponteiro do noAtual para null.

Mão na massa

```
116 public void removePosicao(int posicao) {
117     //9 - Remover de uma posição específica
118     if (!(posicao >= 0 && posicao <= totalDeElementos)) {
119         throw new IllegalArgumentException(NAO_EXISTE);
120     }
121     if (posicao == 0) {
122         removeInicio();
123     }
124     if (posicao == totalDeElementos - 1) {
125         removeFinal();
126     } else {
127         Node noAnterior = buscaNoPosicao(posicao - 1);
128         Node atual = noAnterior.getNext();
129         Node proximo = atual.getNext();
130         noAnterior.setNext(proximo);
131         atual.setNext(next:null);
132         totalDeElementos--;
133     }
134 }
```


Mão na massa – Testes

Testando a opção 9

- Alterando o programa principal

```
60      case 9: {  
61          System.out.print("Digite uma posição: ");  
62          int posicao = sc.nextInt(); //lê o número informado  
63          list.removePosicao(posicao);  
64          break;  
65      }
```