



PECS | Programa de Pós-Graduação em
Engenharia de Computação e Sistemas

ALGORITMOS E ESTRUTURAS DE DADOS

Disciplina: ALGORITMOS E ESTRUTURAS DE DADOS

Carga Horária: 60h

Professor: **Dr. Reinaldo**

Listas: Tipo de Armazenamento

O tipo de armazenamento de uma lista linear pode ser classificado de acordo com a posição relativa (sempre contígua ou não) na memória de dois nós consecutivos na lista.

– **Lista linear com alocação estática de memória**

- Também chamada de Lista Seqüencial
- Nós em posições contíguas de memória
- Geralmente representado por arrays
- Útil para implementar filas e pilhas (variáveis para controlar fim e início)

Listas: Tipo de Armazenamento

- Lista linear com alocação dinâmica de memória
 - Também chamada de Lista Encadeada
 - Posições de memória são alocadas a medida que são necessárias
 - Nós encontram-se aleatoriamente dispostos na memória e são interligados por ponteiros, que indicam a próxima posição da tabela
 - Nós precisam de um campo a mais: campo que indica o endereço do próximo nó.

O que é uma lista ligada?

Dados: Cada nó contém informações relevantes que serão armazenadas e manipuladas pela estrutura de dados.

Ponteiros: Os nós possuem ponteiros que apontam para outros nós relacionados, formando a conectividade da estrutura.

Tipo: O tipo de dado armazenado no nó determina suas características e funcionalidades dentro da estrutura.



Lista linear geral com alocação estática

- Também chamada de Lista Seqüencial
- Suponhamos uma lista geral de números inteiros L que, em certo momento, possui os seguintes 5 elementos:

8 -3 2 0 -5

- Declarando um vetor de inteiros de nome $L[0..Max-1]$, disporemos a lista nas primeiras posições do vetor, de modo que o primeiro nó da lista ocupe a posição 0 do vetor, e indicaremos seu término por uma variável inteira de nome size que indica a posição do último elemento. Teremos

0	1	2	3	4	5	6
8	-3	2	0	-5		

Lista linear geral com alocação estática

- Inserir 7 no fim da lista

0	1	2	3	4	5	6
8	-3	2	0	-5	7	

- Retirar 0

0	1	2	3	4	5	6
8	-3	2	-5	7		

- Retirar primeiro elemento

0	1	2	3	4	5	6
-3	2	-5	7			

Lista linear geral com alocação estática

- Inserir 9 no início da lista

0	1	2	3	4	5	6
9	-3	2	-5	7		

- Inserir 8 no final da lista

0	1	2	3	4	5	6
9	-3	2	-5	7	8	

- Remover 3

0	1	2	3	4	5	6
9	-3	2	7	8		

Criação da Classe Lista.java

```
package projetolistaligada;

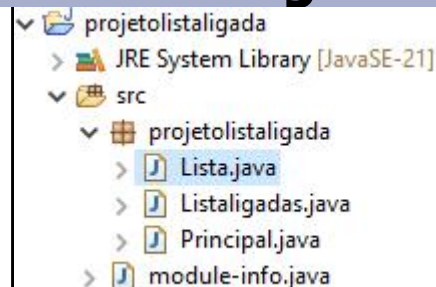
public class Lista {
    /**
     * @param elemento
     * @param proximo
     */
    private Object elemento;
    private Lista proximo;

    public Lista(Object elemento, Lista proximo) {
        super();
        this.elemento = elemento;
        this.proximo = proximo; }

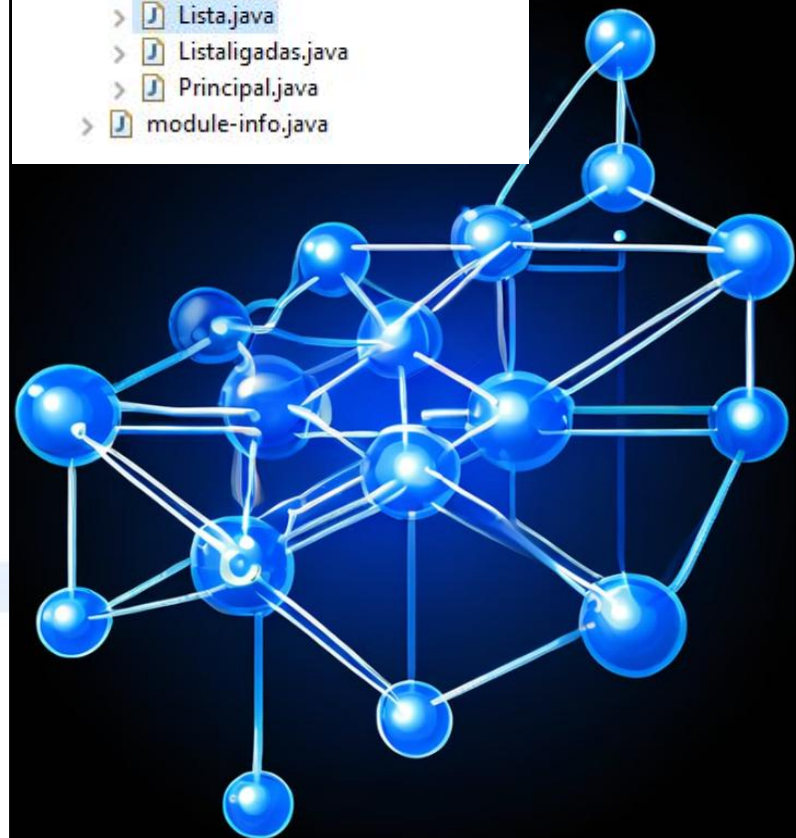
    public Lista getProximo() {
        return proximo; }

    public void setProximo(Lista proximo) {
        this.proximo = proximo; }

    public Object getElemento() {
        return elemento; }}}
```

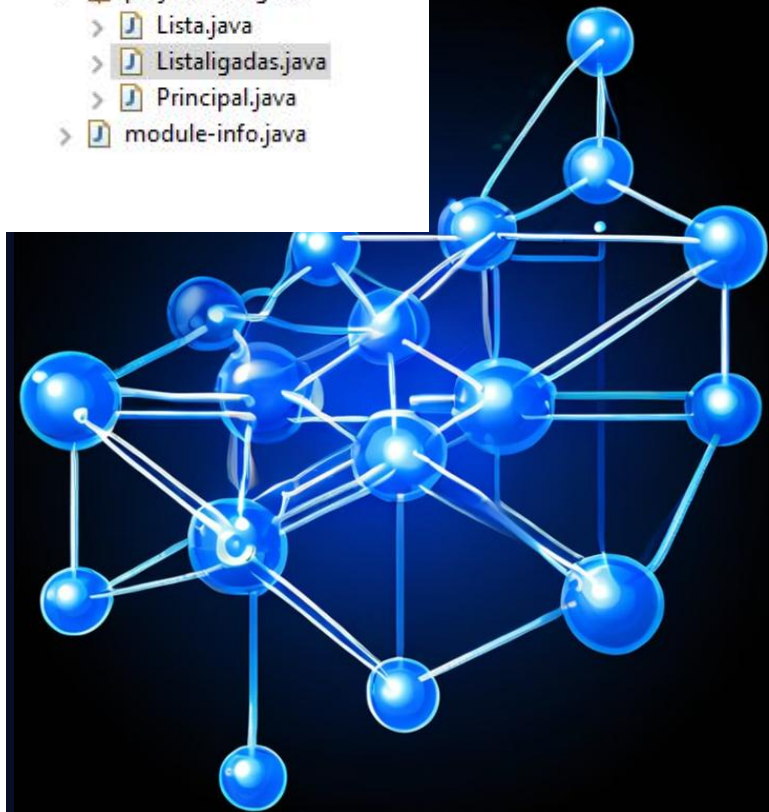
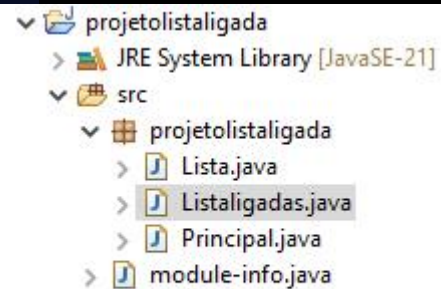
A screenshot of the Eclipse IDE's project explorer showing a project named 'projetolistaligada'. The project is expanded, showing a 'src' folder which contains four Java files: 'Lista.java', 'Listaligadas.java', 'Principal.java', and 'module-info.java'. The 'Lista.java' file is highlighted with a blue selection bar.

```
▼ projetolistaligada
  > JRE System Library [JavaSE-21]
  ▼ src
    ▼ projetolistaligada
      > Lista.java
      > Listaligadas.java
      > Principal.java
      > module-info.java
```



Criação da Classe Listaligada.java

```
package projetolistaligada;  
  
public class Listaligadas {  
  
    public void adicionaNoComeco(Object elemento) {}  
  
    public void adiciona(Object elemento) {}  
  
    public void adiciona(int posicao, Object elemento) {}  
  
    public Object busca(int posicao) { return null; }  
  
    public void remove(int posicao) {}  
  
    public int tamanho() { return 0; }  
  
    public boolean localiza(Object o) { return false; }  
}
```



Inserção de elementos em uma lista ligada

1

Identificar o local de inserção

Primeiro, é preciso determinar onde o novo elemento será inserido na lista.

2

Criar um novo nó

Alocar memória para um novo nó e preencher seu valor e ponteiro.

3

Atualizar os ponteiros

Ajustar os ponteiros dos nós adjacentes para incorporar o novo elemento.

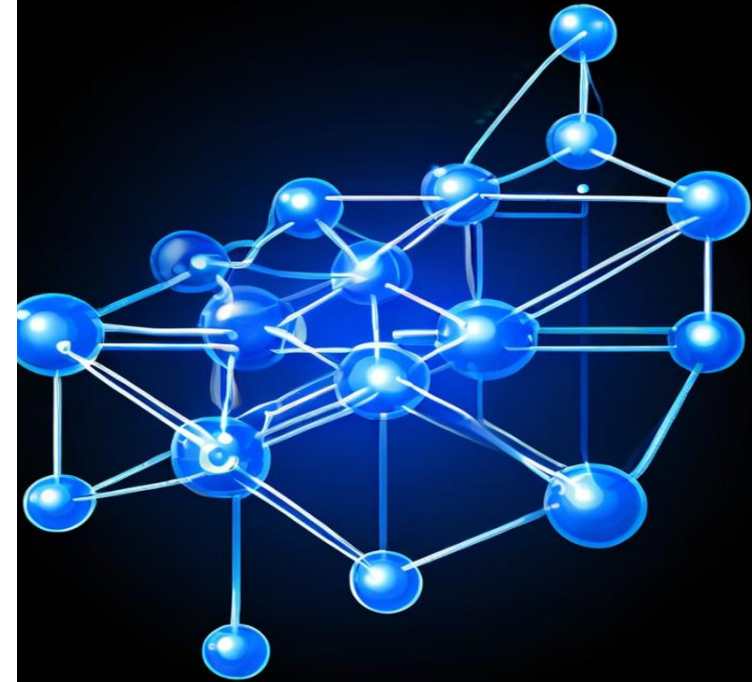


Criação do método Adiciona no começo da lista

```
public class Listaligadas {  
  
    private Lista primeiro = null;  
    private Lista ultimo = null;  
    private int totalDeElementos = 0;  
  
    public void adicionaNoComeco(Object elemento) {  
        Lista novoelemento = new Lista(elemento, primeiro);  
        this.primeiro = novoelemento;  
        if(this.totalDeElementos == 0) {  
            this.ultimo = this.primeiro;  
        }  
  
        this.totalDeElementos++;    }
```

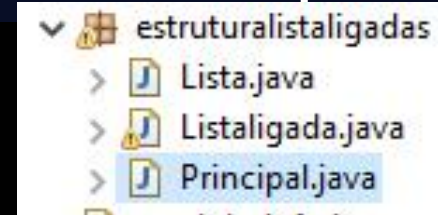
Criação do método toString

```
public String toString () {  
  
    if(this.totalDeElementos == 0) {  
        return "()"; }  
  
    Lista atual = primeira;  
    StringBuilder builder = new StringBuilder("(");  
    for(int i = 0; i < totalDeElementos; i++) {  
        builder.append(atual.getElemento());  
        builder.append(",");  
        atual = atual.getProximo();  
    }  
    builder.append(")");  
    return builder.toString(); }}
```



Criação da classe Principal

```
public class Principal {  
    public static void main(String[] args) {  
  
        Listaligada lista = new Listaligada();  
  
        System.out.println(lista);  
        lista.adicionaNoComeco("ANDERSON");  
        System.out.println(lista);  
        lista.adicionaNoComeco("FRANK");  
        System.out.println(lista);  
        lista.adicionaNoComeco("LORENA");  
        System.out.println(lista);    }  
}
```

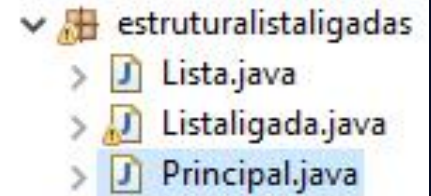


Criação do método Adicionafinal

```
public void adicionafinal(Object elemento) {  
    if(this.totalDeElementos == 0) {  
        adicionaNoComeco(elemento);  
    } else {  
        Lista novofinal = new Lista(elemento, null);  
        this.ultimo.setProximo(novofinal);  
        this.ultimo = novofinal;  
        this.totalDeElementos++;  
    }  
}
```

Criação da classe Principal

```
public static void main(String[] args) {  
    Listaligadas lista = new Listaligadas();  
    System.out.println(lista);  
    lista.adicionaNoComeco("Anderson");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Frank");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Lorena");  
    System.out.println(lista);  
    lista.adicionafinal("Rita");  
    System.out.println(lista); } }
```

A screenshot of an IDE's file explorer showing a project named "estruturalistaligadas". It contains three Java files: "Lista.java", "Listaligada.java", and "Principal.java". The "Principal.java" file is currently selected and highlighted in blue.

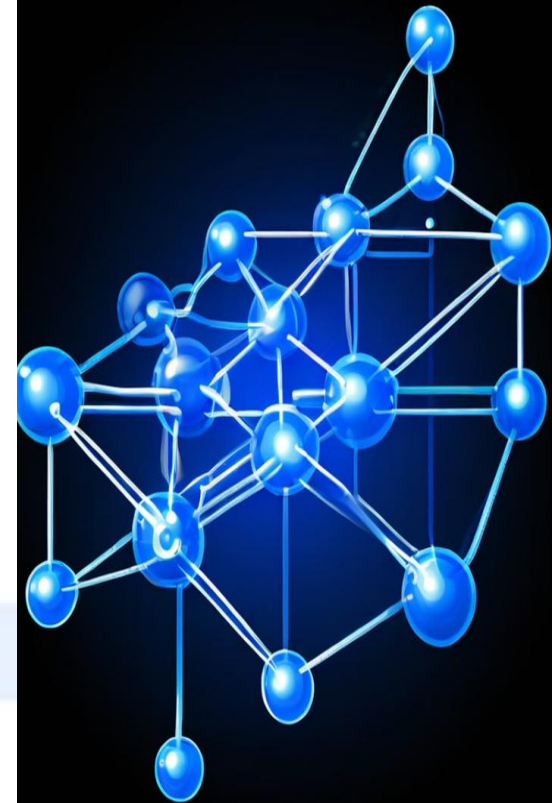
estruturalistaligadas
 > Lista.java
 > Listaligada.java
 > Principal.java

Verificar posição ocupada

```
private boolean posicao elemento(int posicao) {  
    return posicao >= 0 &&  
        posicao < this.totalDeElementos; }  
}
```

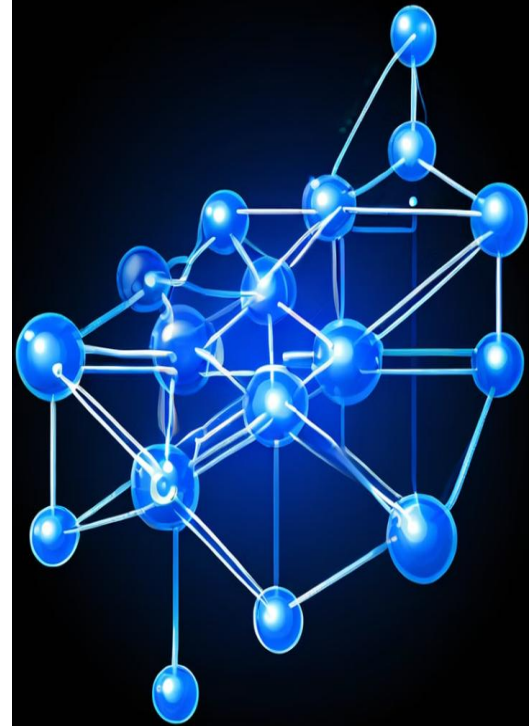

Criação do método busca

```
public Lista busca(int posicao) {  
    if(!posicaoOcupada(posicao)) {  
        throw new IllegalArgumentException  
            ("posicao inexistente"); }  
  
    Lista atual = primeiro;  
    for(int i = 0; i < posicao; i++) {  
        atual = atual.getProximo(); }  
    return atual;  
}
```




Criação do método Adiciona em qualquer posição na lista

```
public void adiciona(int posicao, Object elemento) {  
    Lista anterior = this.busca(posicao - 1);  
    Lista meio = new Lista(elemento, anterior.getProximo());  
    anterior.setProximo(meio);  
    this.totalDeElementos++;  
}
```



Criação da classe Principal

```
public static void main(String[] args) {  
    Listaligadas lista = new Listaligadas();  
    System.out.println(lista);  
    lista.adicionaNoComeco("Anderson");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Frank");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Lorena");  
    System.out.println(lista);  
    lista.adicionafinal("Rita");  
    lista.adiciona(3, "Roberto");  
    System.out.println(lista);  
    Object Dados = lista.localiza(2);  
    System.out.println(Dados);  
}
```



- estruturalistaligadas
 - Lista.java
 - Listaligada.java
 - Principal.java

Criação do método Adiciona em qualquer posição na lista

```
public void remove(int posicao) {  
    if(this.totalDeElementos == 0) {  
        throw new IllegalArgumentException("lista vazia");  
    }  
  
    this.primeiro = this.primeiro.getProximo();  
    this.totalDeElementos--;  
  
    if(this.totalDeElementos == 0) {  
        this.ultimo = null;  
    }  
}  
  
public int tamanho() {  
    return this.totalDeElementos; }  
}
```



Busca de elementos em uma lista ligada

1

Percorrer a lista

Começar do primeiro nó e avançar até encontrar o elemento desejado.

2

Comparar valores

Comparar o valor de cada nó com o elemento que se está buscando.

3





Encontrar o elemento

Parar quando o valor do nó corresponder ao elemento procurado.

A busca de elementos em uma lista ligada é realizada de forma sequencial, começando pelo primeiro nó e avançando até encontrar o elemento desejado. Durante a busca, os valores de cada nó são comparados com o elemento que se está procurando. Quando o valor do nó corresponde ao elemento procurado, a busca é concluída com sucesso.

Criação da classe Principal

```
public static void main(String[] args) {  
    Listaligadas lista = new Listaligadas();  
    System.out.println(lista);  
    lista.adicionaNoComeco("Anderson");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Frank");  
    System.out.println(lista);  
    lista.adicionaNoComeco("Lorena");  
    System.out.println(lista);  
    lista.adicionafinal("Rita");  
    lista.adiciona(3, "Roberto");  
    System.out.println(lista);  
    Object Dados = lista.localiza(2);  
    System.out.println(Dados);  
    lista.remove(1);  
    System.out.println(lista);|  
}
```

▼  estruturalistaligadas
 >  Lista.java
 >  Listaligada.java
 >  Principal.java