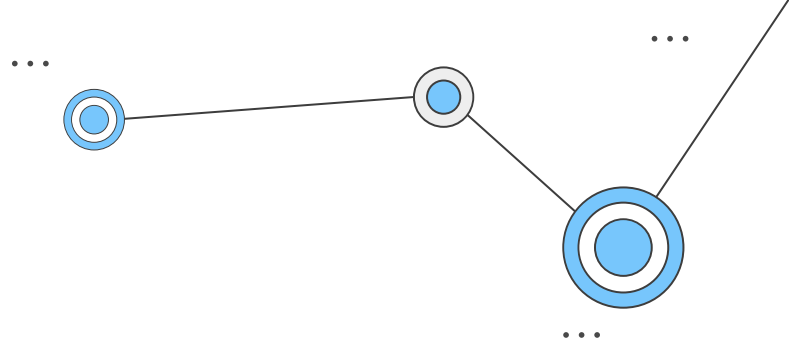
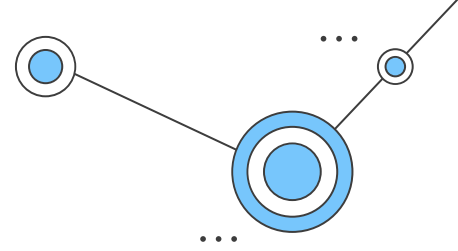


Listas Simplesmente Encadeadas

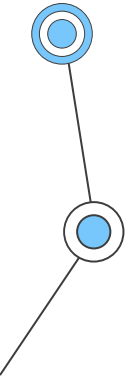
Júlio Silveira Ortiz Rocha
Rafael Azevedo Lezama
Tadeu Brasil de Souza
William Christopher Ramos Oliveira

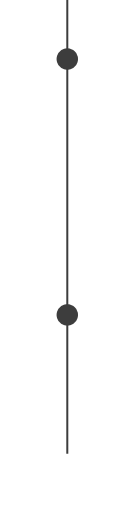
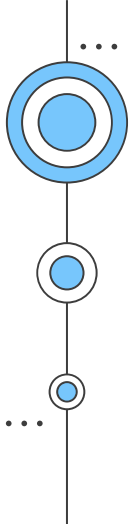


Roteiro



1. Motivação para o uso
2. Funcionamento
3. Exemplos de uso
4. Implementação prática
5. Estória ilustrativa baseada na estrutura
6. Referências
7. Ata





01

Motivação

Motivação

- Performance
- Crescimento dinâmico das listas
- Espaço de memória melhor aproveitado
- Salvamento e divisão de trabalho
- Mapeamento otimizado
- Adequado para salvar dados/itens de maior permanência e menor mudança

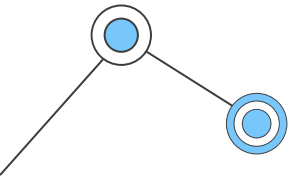
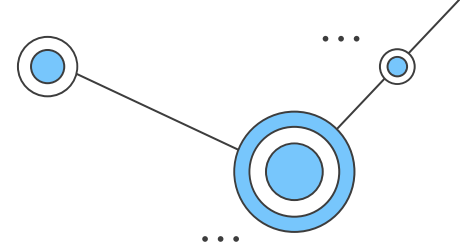
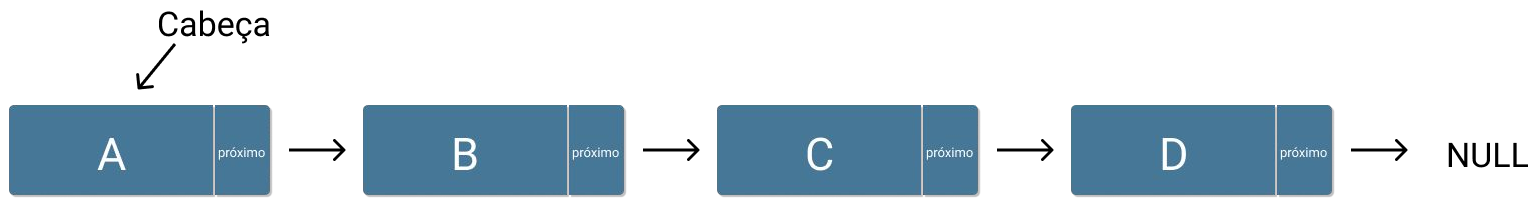




02

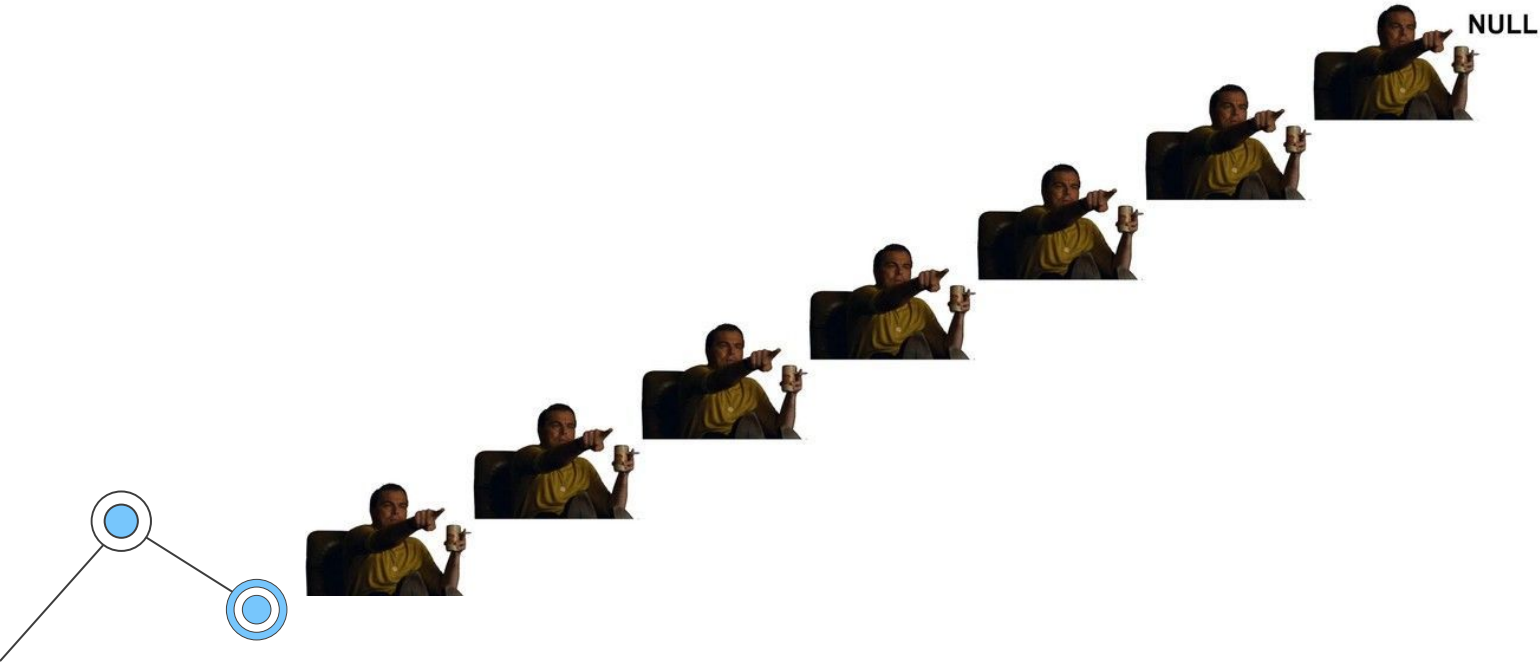
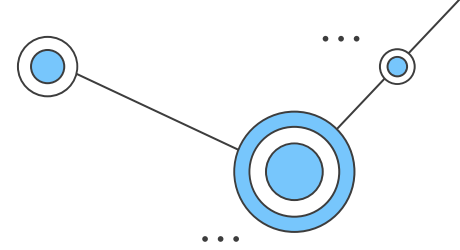
Funcionamento

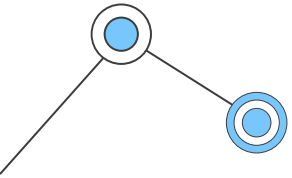
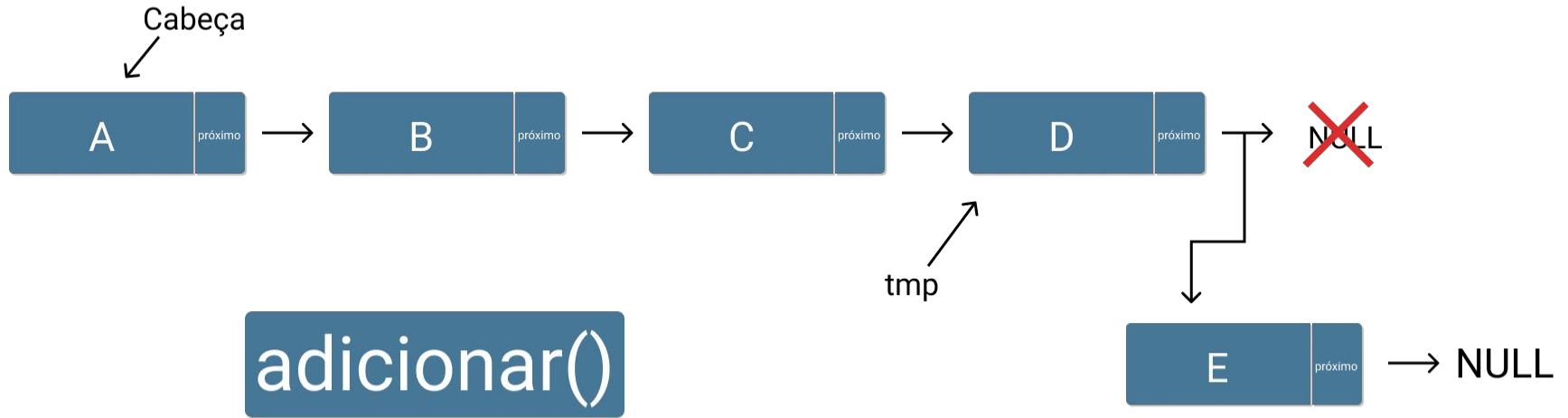
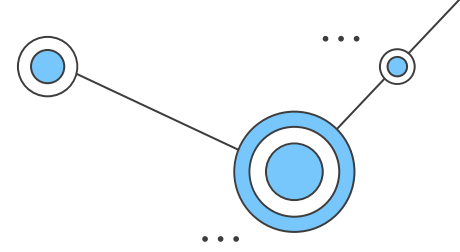


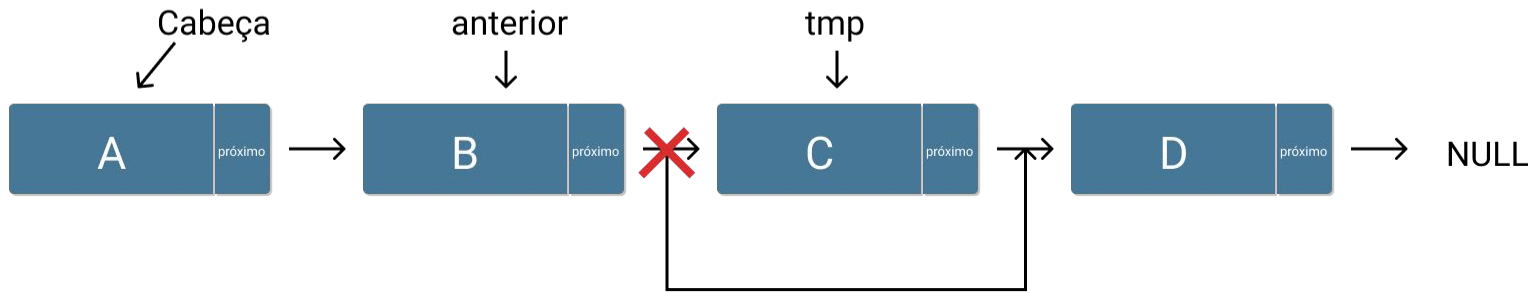


Nobody:

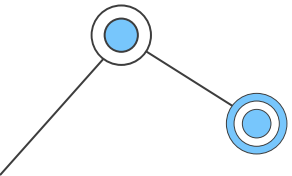
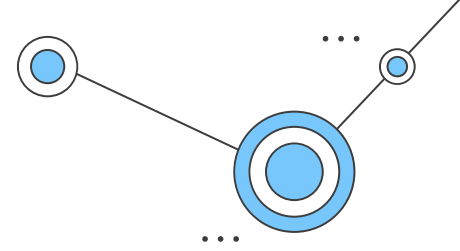
Linked Lists:

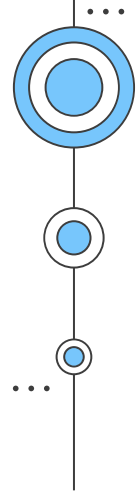






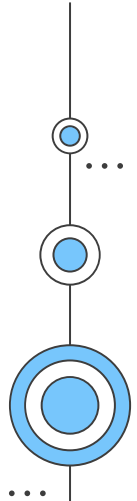
remover()



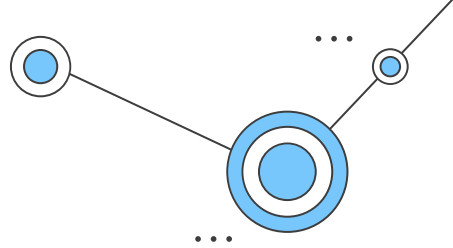


03

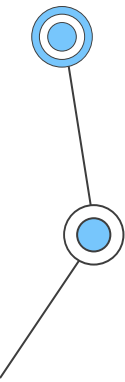
Exemplos



Como e quando usar as listas simplesmente encadeadas?



- Podem ser usadas através de estruturas prontas ("LinkedList") ou com estrutura própria.
- São ótimas para situações que demandam praticidade.
- Quando são necessárias diversas operações de inserção/remoção.
- Execução de operações com filas e pilhas.

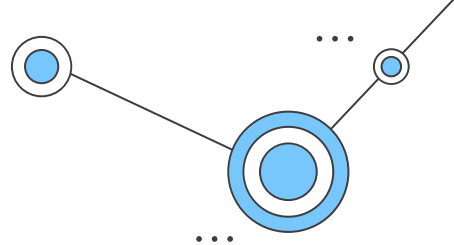


Lista com estrutura pronta

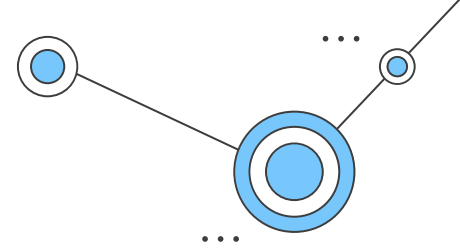
```
1  import java.util.*;
2  public class Cidade {
    Run | Debug
3      public static void main(String args[]){
4          LinkedList<String> cidade = new LinkedList<String>();
5
6          cidade.add("Uruguaiana");
7          cidade.add("Alegrete");
8          cidade.add("Itaqui");
9          cidade.add("Bage");
10         cidade.add("Livramento");
11         cidade.add("Sao Borja");
12
13         System.out.println(cidade);
14     }
15 }
16 }
```

```
[Uruguaiana, Alegrete, Itaqui, Bage, Livramento, Sao Borja]
```

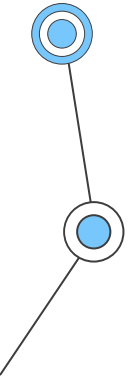
Existem inúmeros métodos p/ manipulação das listas encadeadas

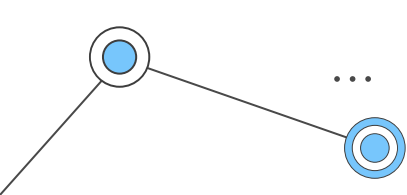


Estrutura própria

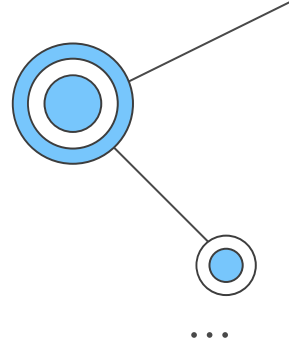


```
1 public class Celula {  
2     private Celula proximo;  
3     private Pessoa valor;  
4  
5     public Celula getProximo(){  
6         return proximo;  
7     }  
8  
9     public void setProximo(Celula proximo){  
10        this.proximo = proximo;  
11    }  
12  
13    public Pessoa getValor(){  
14        return valor;  
15    }  
16  
17    public void setValor(Pessoa valor){  
18        this.valor = valor;  
19    }  
20 }
```

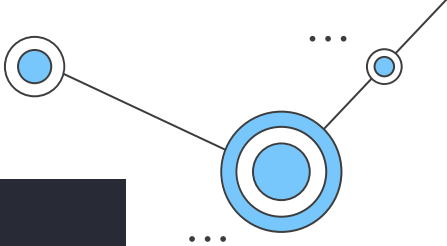




```
1 public class ListaSimplesEncadeada {
2     private Celula primeiro;
3     private Celula ultimo;
4     private Celula posAtual;
5
6     // Adicionar uma pessoa ao final da lista
7     public void adicionar(Pessoa valor){
8         Celula celula = new Celula();
9         celula.setValor(valor);
10
11         if(primeiro == null && ultimo == null){
12             primeiro = celula;
13             ultimo = celula;
14         } else{
15             ultimo.setProximo(celula);
16             ultimo = celula;
17         }
18     }
19
20     // Remove uma pessoa do final da lista
21     public void remover(){
22         if(primeiro.getProximo() != null){
23             Celula celula = this.recuperarPenultimo(this.primeiro);
24             ultimo = celula;
25             celula.setProximo(null);
26         } else {
27             primeiro = ultimo = null;
28         }
29     }
30 }
```



```
39     public boolean temProximo(){
40         if(primeiro == null){
41             return false;
42         } else if(posAtual == null) {
43             posAtual = primeiro;
44             return true;
45         } else {
46             boolean temProximo = posAtual.getProximo() != null ? true : false;
47             posAtual = posAtual.getProximo();
48             return temProximo;
49         }
50     }
51
52     public Celula getPosAtual(){
53         return this.posAtual;
54     }
55 }
```




```
1 public class ListaPrincipal {
2     Run | Debug
3     public static void main(String args[]){
4         ListaSimplesEncadeada listaEncadeada = new ListaSimplesEncadeada();
5         ListaPrincipal listaPrincipal = new ListaPrincipal();
6
7         listaPrincipal.adicionarPessoa(listaEncadeada);
8         listaPrincipal.remover(listaEncadeada);
9         System.out.println("Lista de alunos - Grupo 2 - Estrutura de Dados");
10        while(listaEncadeada.temProximo()){
11            System.out.println(listaEncadeada.getPosAtual().getValor());
12        }
13    }
14
15    private void adicionarPessoa(ListaSimplesEncadeada listaEncadeada){
16        Pessoa p1 = new Pessoa(1, "Julio", "julioortiz.aluno@unipampa.edu.br");
17        Pessoa p2 = new Pessoa(2, "Rafael", "rafaellezama.aluno@unipampa.edu.br");
18        Pessoa p3 = new Pessoa(3, "Tadeu", "tadeubrasil.aluno@unipampa.edu.br");
19        Pessoa p4 = new Pessoa(4, "William", "williamchristopher.aluno@unipampa.edu.br");
20        Pessoa p5 = new Pessoa(5, "Alan (F)", "alanfarias.aluno@unipampa.edu.br");
21        listaEncadeada.adicionar(p1);
22        listaEncadeada.adicionar(p2);
23        listaEncadeada.adicionar(p3);
24        listaEncadeada.adicionar(p4);
25        listaEncadeada.adicionar(p5);
26    }
27 }
```



04

Implementação

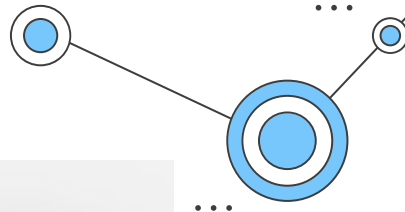




```
package lib;

public class No<T> {
    T dado;
    No<T> proximo;

    No(T d) {
        dado = d;
        proximo = null;
    }
}
```





```
package lib;

public interface IListaEncadeada<T> {

    public void adicionarInicio(T dado);

    public void adicionar(T dado);

    public void remover(T dado);

    public String exibirTodos();

    public T retornaPrimeiro();

    public T retornaUltimo();

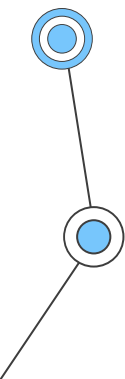
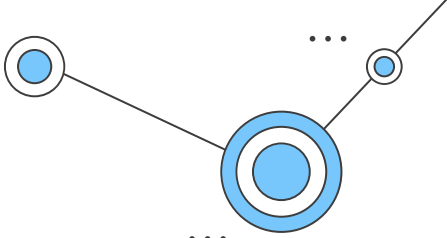
    public T proximo(T dado);

    public int tamanho();

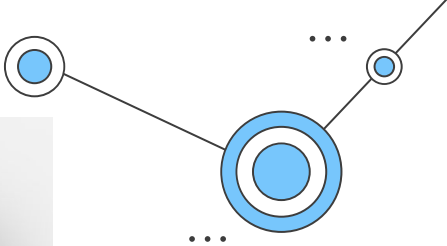
    public T buscar(int indice);

}
```





```
@Override
public void adicionarInicio(T dado) {
    No<T> novoNo = new No<>(dado);
    novoNo.proximo = cabeca;
    cabeca = novoNo;
    size++;
}
```

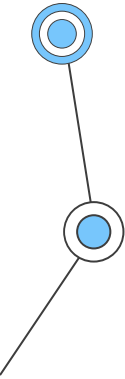


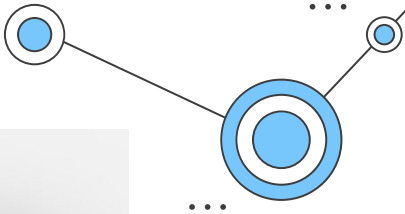
```
@Override
public void adicionar(T dado) {
    No<T> novoNo = new No<>(dado);
    novoNo.proximo = null;

    size++;

    if(cabeca == null) {
        cabeca = novoNo;
    }
    else{
        No<T> ultimo = cabeca;
        while(ultimo.proximo != null){
            ultimo = ultimo.proximo;
        }

        ultimo.proximo = novoNo;
    }
}
```

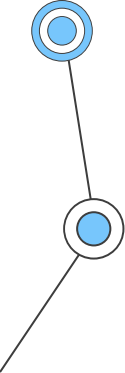
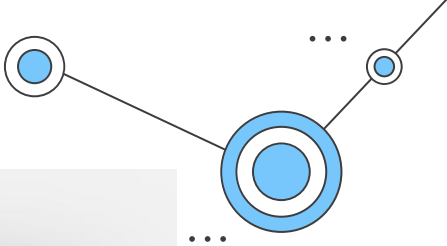




```
@Override
public void remover(T dado) {
    No<T> noAtual = cabeca;

    while(noAtual != null){
        if (noAtual.proximo.dado == dado){
            noAtual.proximo = noAtual.proximo.proximo;
            size--;
            break;
        }
        noAtual = noAtual.proximo;
    }
}
```

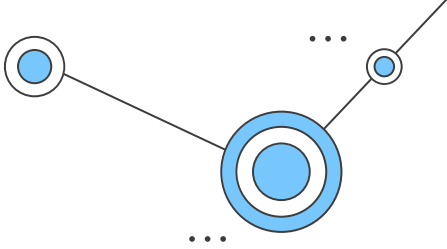




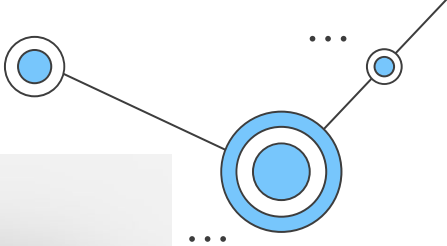
```
@Override
public String exibirTodos() {
    No<T> noAtual = cabeca;
    String str = "ListaEncadeada: ";

    while(noAtual != null){
        str += noAtual.dado + " ";
        noAtual = noAtual.proximo;
    }

    return str;
}
```



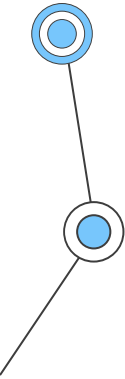
```
@Override  
public T retornaPrimeiro() {  
    return cabeca.dado;  
}
```

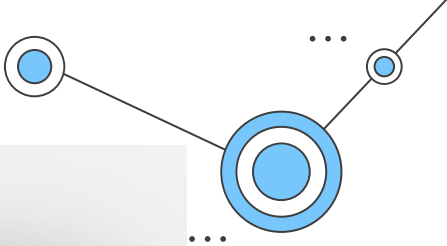


```
@Override
public T retornaUltimo() {
    No<T> noAtual = cabeca;

    while(noAtual != null){
        if (noAtual.proximo == null){
            return noAtual.dado;
        }
        noAtual = noAtual.proximo;
    }

    return null;
}
```

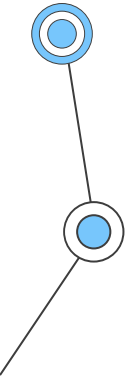


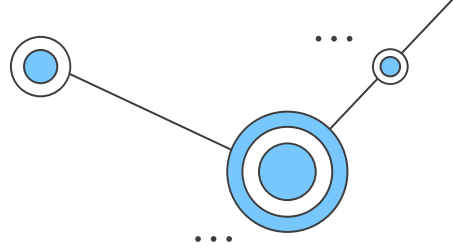


```
@Override
public T proximo(T dado){
    No<T> noAtual = cabeca;

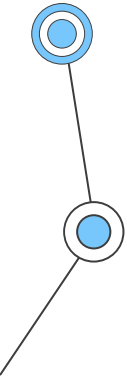
    while(noAtual != null){
        if (noAtual.dado == dado){
            return noAtual.proximo.dado;
        }
        noAtual = noAtual.proximo;
    }

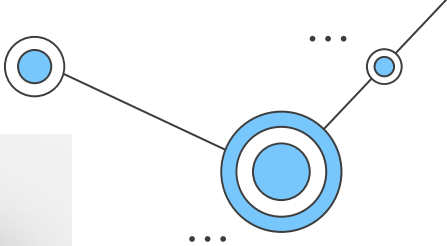
    return null;
}
```





```
@Override  
public int tamanho(){  
    return size;  
}
```





```
@Override
public T buscar(int indice){
    if(indice >= 0 && indice < size){
        No<T> noAtual = cabeza;
        int i = 0;

        while(noAtual != null){
            if(i == indice){
                return noAtual.dado;
            }
            noAtual = noAtual.proximo;
            i++;
        }

        throw new IndexOutOfBoundsException("Index out of bounds!");
    }
}
```



```
package model.entities;

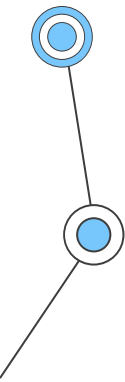
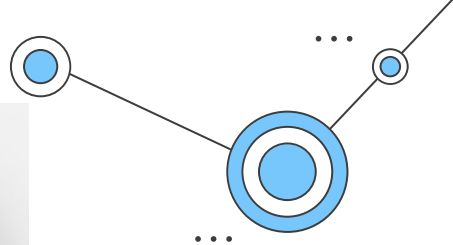
import lib.ListaEncadeada;

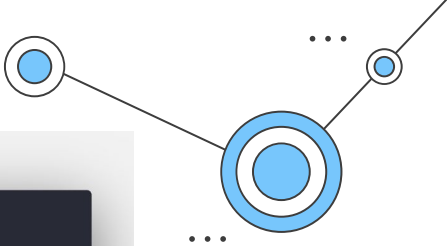
public class Snake {
    private ListaEncadeada<Position> tail = new ListaEncadeada<>();
    private Position pos;

    public Snake() {
        this.pos = new Position(2, 15 / 2);
        tail.adicionarInicio(pos);
        grow(1, 15/2);
    }

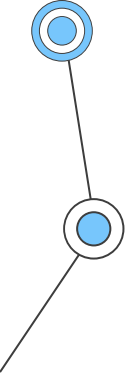
    public void grow(int x, int y) {
        tail.adicionar(new Position(x, y));
    }

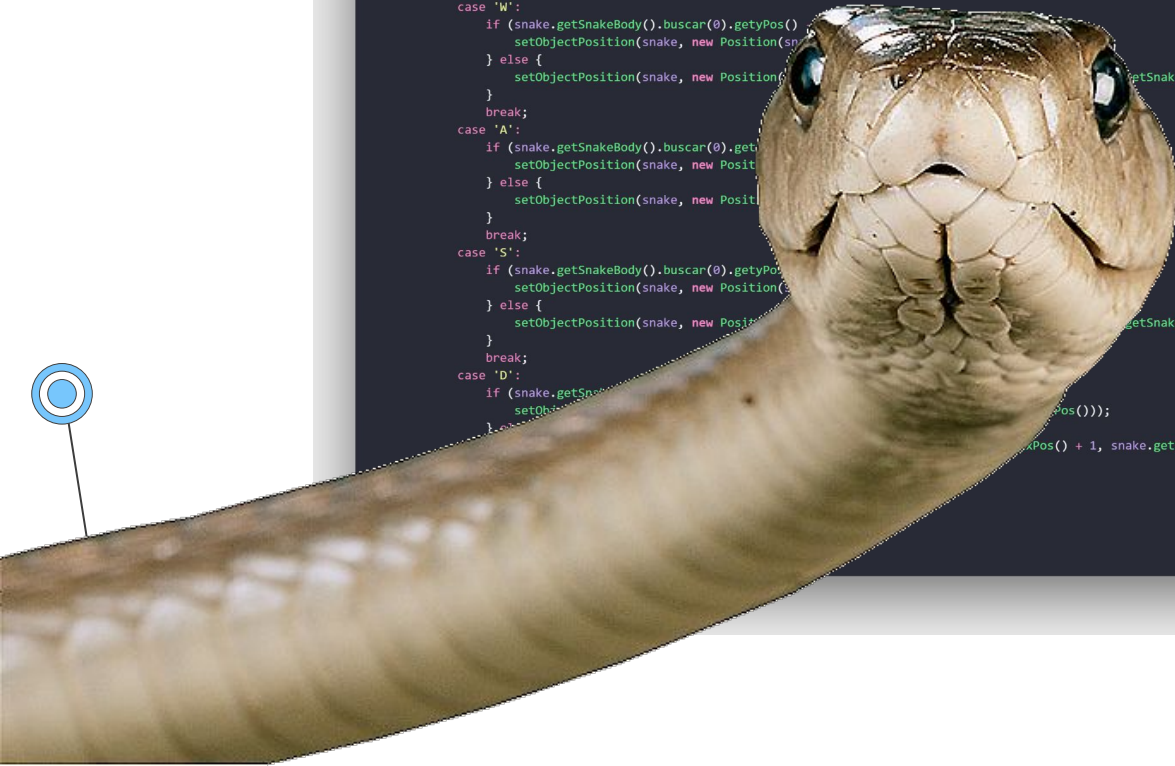
    public ListaEncadeada<Position> getSnakeBody() {
        return tail;
    }
}
```



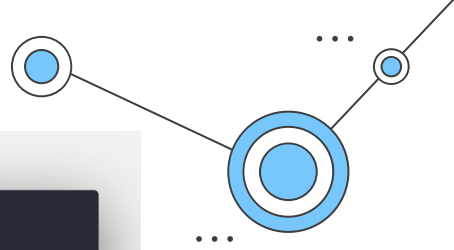


```
public void move() {
    for (int i = snake.getSnakeBody().tamanho() - 1; i > 0; i--) {
        snake.getSnakeBody().buscar(i).setxPos(snake.getSnakeBody().buscar(i - 1).getxPos());
        snake.getSnakeBody().buscar(i).setyPos(snake.getSnakeBody().buscar(i - 1).getyPos());
    }
    switch (direction) {
        case 'W':
            if (snake.getSnakeBody().buscar(0).getyPos() - 1 < 0) {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), 15));
            } else {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos() - 1));
            }
            break;
        case 'A':
            if (snake.getSnakeBody().buscar(0).getxPos() - 1 < 0) {
                setObjectPosition(snake, new Position(15, snake.getSnakeBody().buscar(0).getyPos()));
            } else {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos() - 1, snake.getSnakeBody().buscar(0).getyPos()));
            }
            break;
        case 'S':
            if (snake.getSnakeBody().buscar(0).getyPos() + 1 > 15) {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), 0));
            } else {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos() + 1));
            }
            break;
        case 'D':
            if (snake.getSnakeBody().buscar(0).getxPos() + 1 > 15) {
                setObjectPosition(snake, new Position(0, snake.getSnakeBody().buscar(0).getyPos()));
            } else {
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos() + 1, snake.getSnakeBody().buscar(0).getyPos()));
            }
            break;
    }
    flag = direction;
}
```





```
public void move() {  
    for (int i = snake.getSnakeBody().tamanho() - 1; i > 0; i--) {  
        snake.getSnakeBody().buscar(i).setxPos(snake.getSnakeBody().buscar(i - 1).getxPos());  
        snake.getSnakeBody().buscar(i).setyPos(snake.getSnakeBody().buscar(i - 1).getyPos());  
    }  
    switch (direction) {  
        case 'W':  
            if (snake.getSnakeBody().buscar(0).getyPos() != 0) {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos() - 1));  
            } else {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos()));  
            }  
            break;  
        case 'A':  
            if (snake.getSnakeBody().buscar(0).getxPos() != 0) {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos() - 1, snake.getSnakeBody().buscar(0).getyPos()));  
            } else {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos()));  
            }  
            break;  
        case 'S':  
            if (snake.getSnakeBody().buscar(0).getyPos() != 10) {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos() + 1));  
            } else {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos()));  
            }  
            break;  
        case 'D':  
            if (snake.getSnakeBody().buscar(0).getxPos() != 10) {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos() + 1, snake.getSnakeBody().buscar(0).getyPos()));  
            } else {  
                setObjectPosition(snake, new Position(snake.getSnakeBody().buscar(0).getxPos(), snake.getSnakeBody().buscar(0).getyPos()));  
            }  
            break;  
    }  
}
```





05

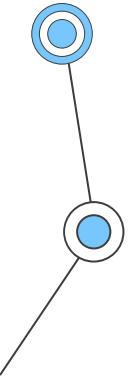
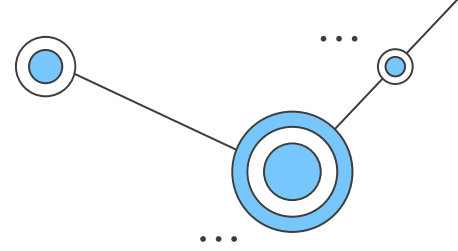
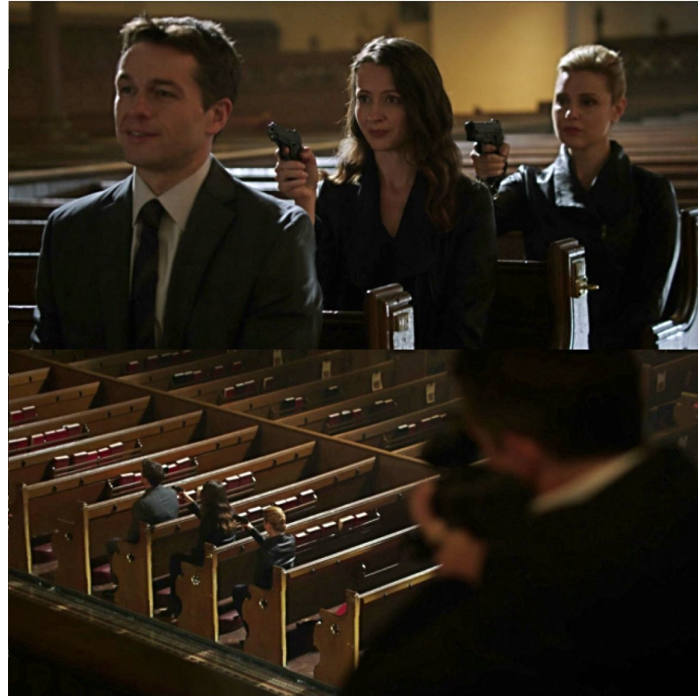
Memes



when you ask stack overflow how
to get the first element in a linked
list



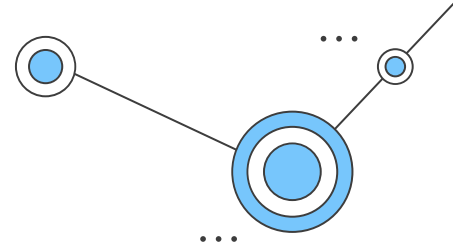
Linked Lists be like





**Obrigado pela
atenção!**

REFERÊNCIAS



"Listas encadeadas em Java." Preciso estudar sempre. Disponível em:

["https://precisoestudarsempre.blogspot.com/2014/12/listas-encadeadas-em-java.html"](https://precisoestudarsempre.blogspot.com/2014/12/listas-encadeadas-em-java.html).

"LinkedList (Java Platform SE 7)". Disponível em:

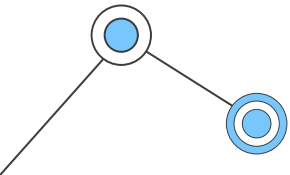
["https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html"](https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html).

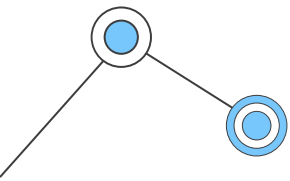
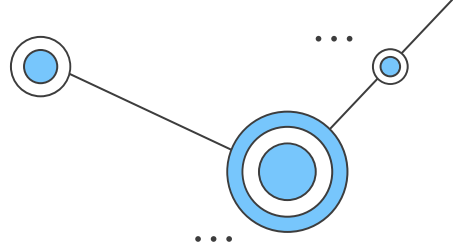
"Listas Encadeadas - DEV Community." CUNHA, Carolina. Disponível em:

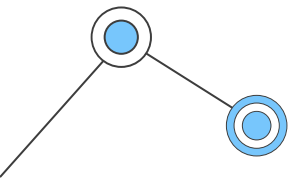
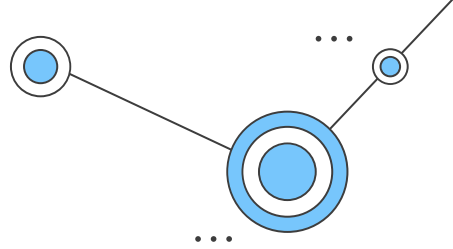
["https://dev.to/ccunha/listas-encadeadas-157"](https://dev.to/ccunha/listas-encadeadas-157).

"Linked List | Set 1 (Introduction)". GeeksforGeeks, 8 de março de 2013,

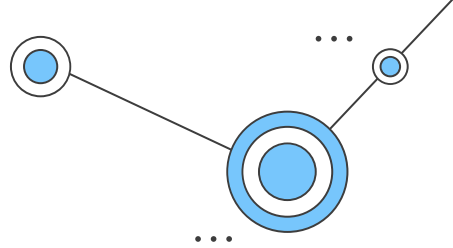
<https://www.geeksforgeeks.org/linked-list-set-1-introduction/>.







ATA



Júlio Silveira → Template dos slides / Exemplos de uso (códigos básicos e implementação usando a classe "Cédula").

Rafael Lezama → Implementação prática das listas.

Tadeu → Motivação.

William Oliveira → Funcionamento das listas/imagens do funcionamento/história (memes).

