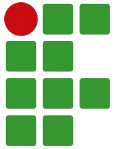


INSTITUTO FEDERAL
Sertão Pernambucano
Campus Salgueiro

Estrutura de Dados e Algoritmos com Java

Prof. Heraldo Gonçalves Lima Junior
heraldo.junior@ifsertao-pe.edu.br

1. Listas Duplamente Encadeadas (continuação...)

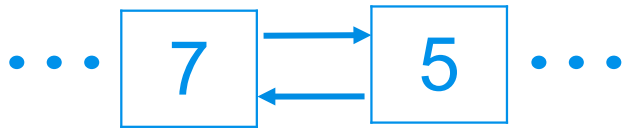
1.1. Adicionando de qualquer posição

- Separamos os casos em que a inserção é no começo ou no fim porque podemos reaproveitar os métodos já implementados.
- Sobra o caso em que a inserção é no meio da Lista, ou seja, entre dois nós existentes. Neste caso, devemos ajustar as referências para o novo nó ser apontado corretamente pelos dois nós relacionados a ele (o anterior e o próximo). E também fazer o novo nó apontar para o anterior e o próximo.

1.1. Adicionando de qualquer posição

- Exemplo de inserção em qualquer posição em **lista não vazia**:

ANTES DA REMOÇÃO



DEPOIS DA REMOÇÃO

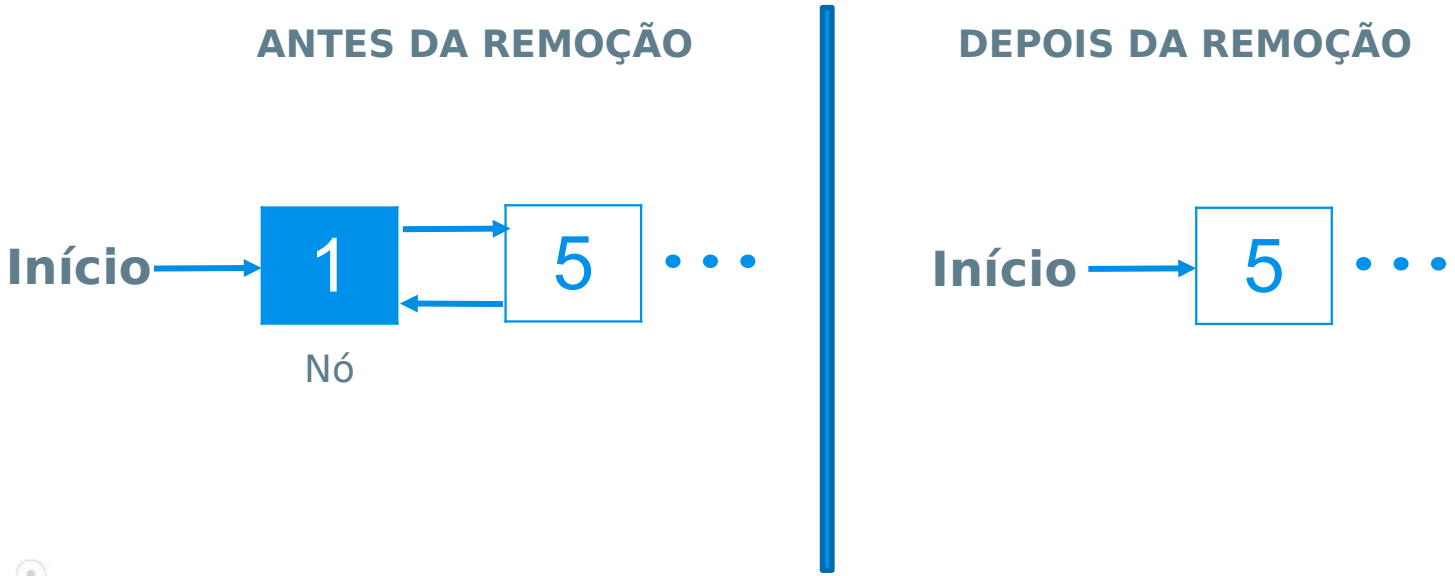
1.1. Adicionando de qualquer posição

```
public void adiciona(int posicao, Object elemento) {
    if(this.posicaoValida(posicao)) {
        if(posicao == 0) {
            this.adicionaNoInicio(elemento);
        }else {
            No anterior = this.pegaNo(posicao-1);
            No proximo = anterior.getProximo();
            No novo = new No(elemento, proximo);
            anterior.setProximo(novo);
            proximo.setAnterior(novo);
            novo.setAnterior(anterior);
            this.totalDeElementos++;
        }
    }else if(posicao == this.totalDeElementos){
        this.adicionaNoFinal(elemento);
    }else{
        System.out.println("Posição inválida!");
    }
}
```

1.2. Removendo do começo da Lista

- Esta operação é idêntica em ambos os tipos de Lista encadeada (simples ou dupla). Ela apenas deve avançar a referência **INÍCIO** para o segundo nó e tomar cuidado com a caso da Lista ficar vazia pois, neste caso, a referência **FIM** deve ser atualizada também.

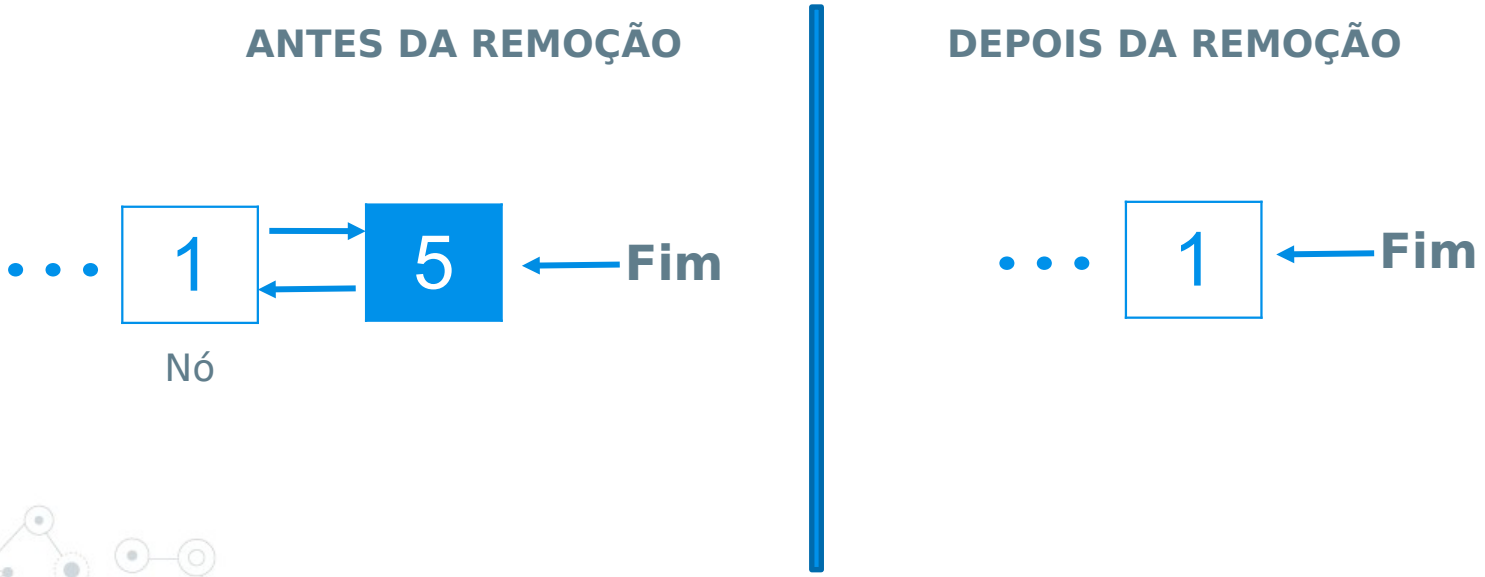
1.2. Removendo do começo da Lista



1.2. Removendo do começo da Lista

```
public void removeDoInicio() {
    if(this.totalDeElementos > 1) {
        No novoInicio = this.inicio.getProximo();
        novoInicio.setAnterior(null);
        this.inicio = novoInicio;
    }else {
        this.fim = null;
        this.inicio = null;
    }
    this.totalDeElementos--;
}
```

1.3. Removendo do fim da Lista



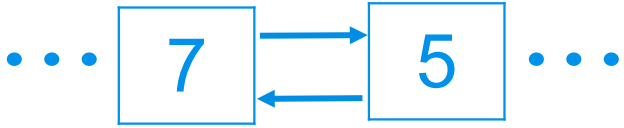
1.3. Removendo do fim da Lista

```
public void removeDoFinal() {
    if(this.totalDeElementos > 1) {
        No NovoFim = this.fim.getAnterior();
        NovoFim.setProximo(null);
        this.fim = this.fim.getAnterior();
        this.totalDeElementos--;
    }
    else {
        this.removeDoInicio();
    }
}
```

1.3. Removendo de qualquer posição

ANTES DA REMOÇÃO

DEPOIS DA REMOÇÃO



1.3. Removendo de qualquer posição

```
public void remove(int posicao) {
    if(this.posicaoValida(posicao)) {
        if(posicao == 0) {
            this.removeDoInicio();
        }else if(posicao == this.totalDeElementos-1) {
            this.removeDoFinal();
        }else {
            No no = this.pegaNo(posicao);
            No anterior = no.getAnterior();
            No proximo = no.getProximo();
            anterior.setProximo(proximo);
            proximo.setAnterior(anterior);
            this.totalDeElementos--;
        }
    }else{
        System.out.println("Posição inválida!");
    }
}
```

2. LinkedList Java

2.1. LinkedList Java

- A classe LinkedList faz o papel da nossa Lista Ligada dentro da bibilioteca do Java.
- Ela possui os mesmos métodos que a ArrayList, e adiciona alguns outros, como o **addFirst(Object)**, **removeFirst()**, **addLast(Object)** e **removeLast()**, que operam no começo e no fim da Lista em tempo constante.

2.1. LinkedList – Métodos a mais

addFirst()	Adiciona no início
addLast()	Adiciona no final
removeFirst()	Remove no início
removeLast()	Remove no final
getFirst()	Pega o início
getLast()	Pega o final

CALMA, RESPIRA!

Obrigado!

Perguntas?


heraldo.junior@ifsertao-pe.edu.br


heraldolimajr.com.br