



PUC Minas

TAD FLEXÍVEL

AULÃO DE AEDS 2



Tópicos



CLASSE CELULA

PILHA

FILA

LISTA SIMPLES

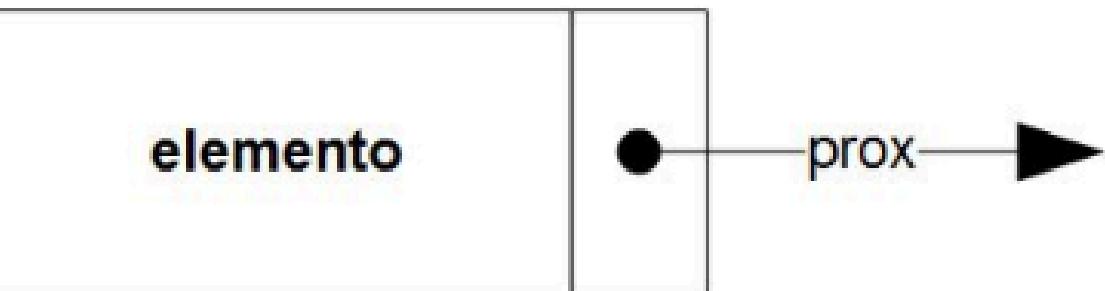
LISTA DUPLA

MATRIZ

Classe Celula

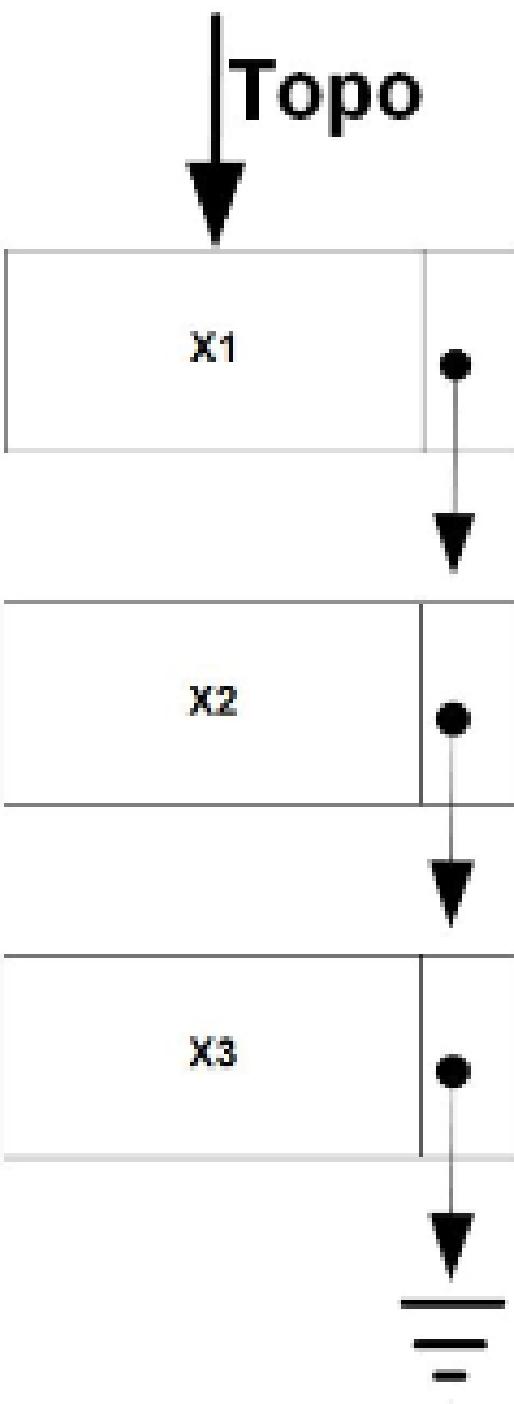
Essa será a classe célula utilizada em todas as estruturas **simplesmente encadeadas**.

```
class Celula {  
    public int elemento;  
    public Celula prox;  
    public Celula() {  
        this(0);  
    }  
    public Celula (int x) {  
        this.elemento = x;  
        this.prox = null;  
    }  
}
```



Pilha

Primeiro, vamos combinar a forma como vamos imaginar a estrutura.



Pilha

- INSERIR
- REMOVER
- MOSTRAR

Pilha - inserir

Antes de chegar no Inserir o construtor já rodou e criou uma célula **Topo** que aponta para Null

O que você faria para inserir uma célula nova? Lembre- se de seguir as restrições da pilha.

```
class Pilha {  
    private Celula topo;  
    public Pilha () {  
        topo = null;  
    }  
    public void inserir(int x) { ... }  
    public int remover() { ... }  
    public void mostrar() { ... }  
}
```

↓ Topo
= :

Pilha - inserir

A estratégia é criar uma nova célula temporária e a definir como novo Topo.

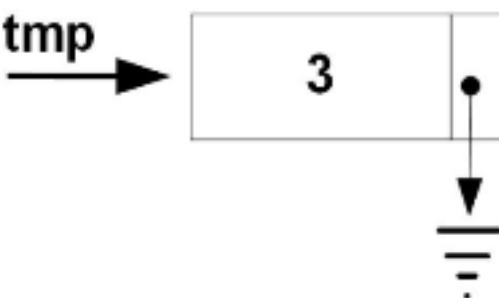
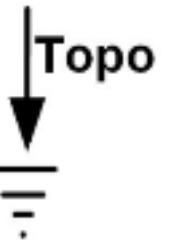
```
class Pilha {  
    private Celula topo;  
    public Pilha () {  
        topo = null;  
    }  
    public void inserir(int x) { ... }  
    public int remover() { ... }  
    public void mostrar() { ... }  
}
```

```
public void inserir(int x) {  
    Celula tmp = new Celula(x);  
    tmp.prox = topo;  
    topo = tmp;  
    tmp = null;  
}
```

↓
Topo
=

Pilha - inserir

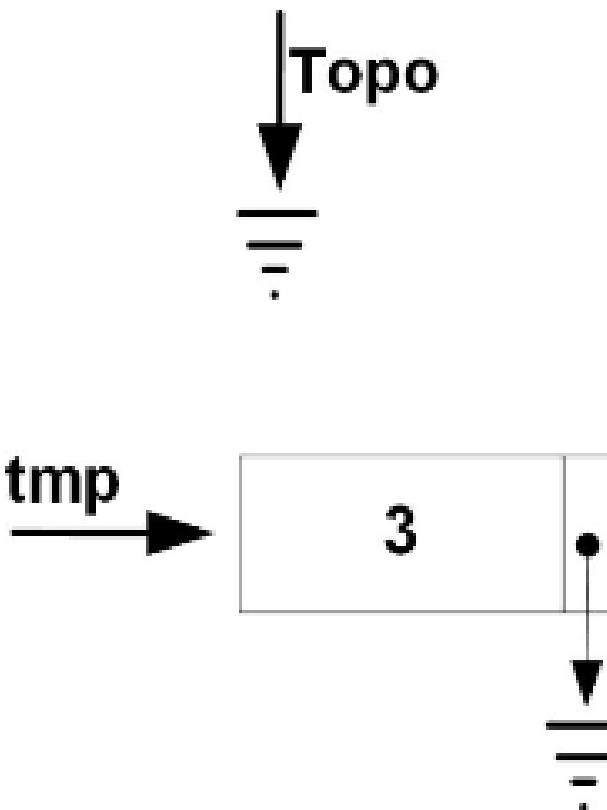
```
public void inserir(int x) { //Inserir(3)  
    Celula tmp = new Celula(x);  
    tmp.prox = topo;  
    topo = tmp;  
    tmp = null;  
}
```



Pilha - inserir

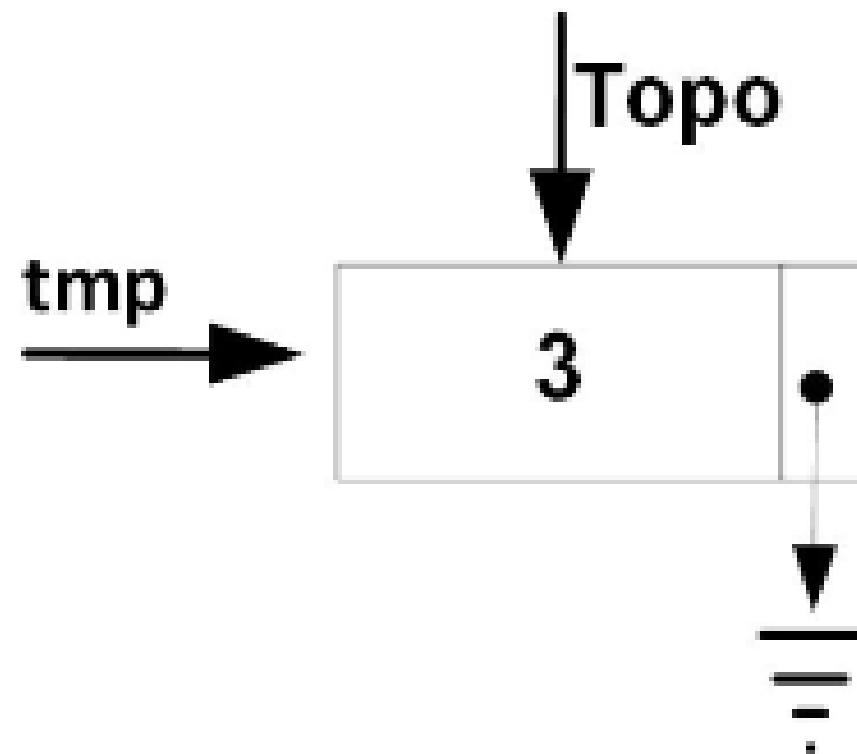
Nesse caso não temos ninguém então Topo aponta para null portando tmp.prox continua apontando pra Null

```
public void inserir(int x) { //Inserir(3)
    Celula tmp = new Celula(x);
    tmp.prox = topo;
    topo = tmp;
    tmp = null;
}
```



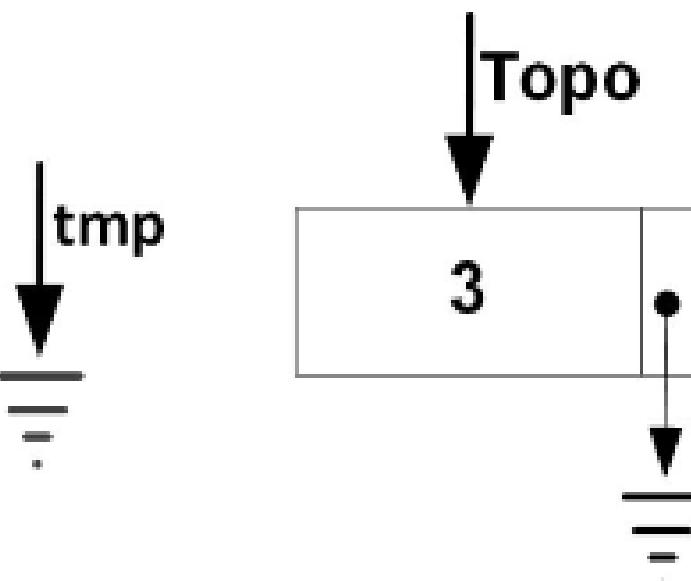
Pilha - inserir

```
public void inserir(int x) { //Inserir(3)
    Celula tmp = new Celula(x);
    tmp.prox = topo;
    topo = tmp;
    tmp = null;
}
```

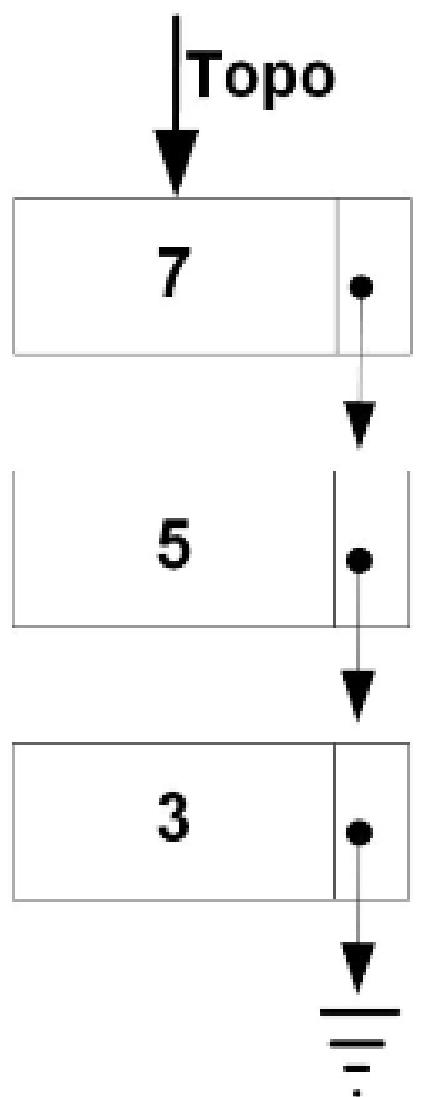


Pilha - inserir

```
public void inserir(int x) { //Inserir(3)
    Celula tmp = new Celula(x);
    tmp.prox = topo;
    topo = tmp;
    tmp = null;
}
```



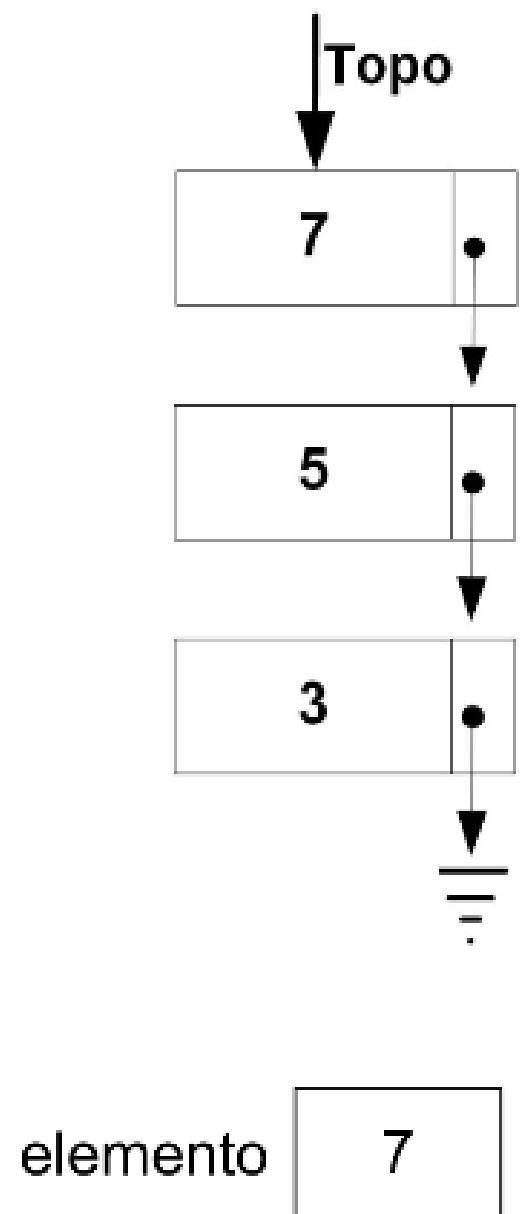
```
class Pilha {  
    private Celula topo;  
    public Pilha () {  
        topo = null;  
    }  
    public void inserir(int x) { ... }  
    public int remover() { ... }  
    public void mostrar() { ... }  
}
```



Pilha - Remover

O que você faria para remover a célula do topo? Seguindo as restrições da pilha.

```
public int remover() throws Exception {  
    if (topo == null)  
        throw new Exception("Erro!");  
    int elemento = topo.elemento;  
    Celula tmp = topo;  
    topo = topo.prox;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```

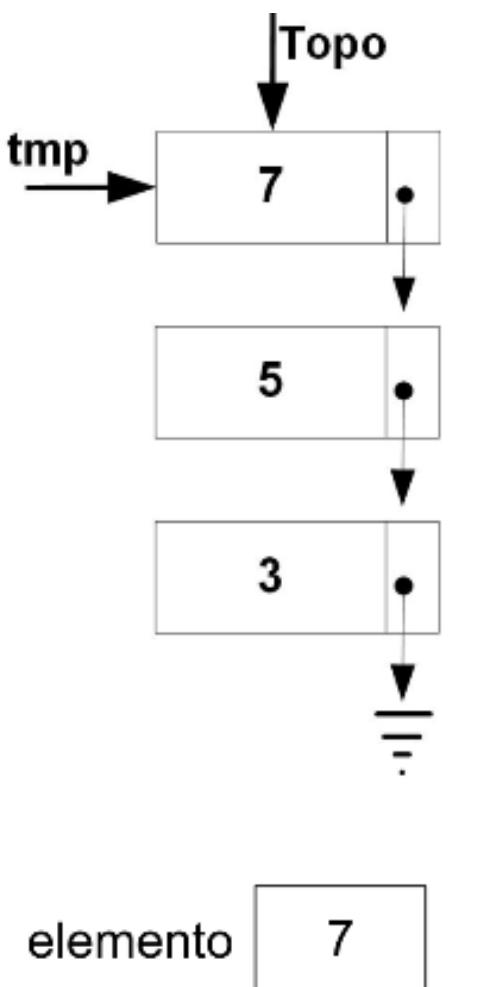


Pilha - Remover

```

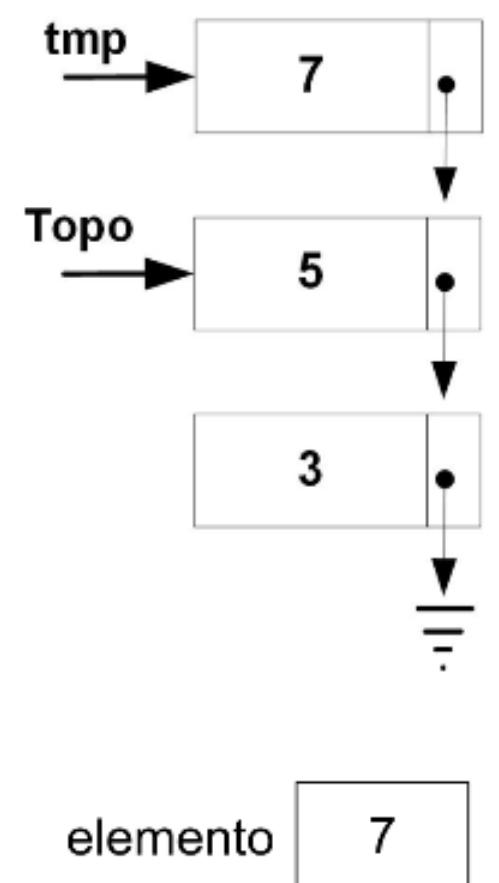
public int remover() throws Exception {
    if (topo == null)
        throw new Exception("Erro!");
    int elemento = topo.elemento;
    Celula tmp = topo;
    topo = topo.prox;
    tmp.prox = null;
    tmp = null;
    return elemento;
}

```



Pilha - Remover

```
public int remover() throws Exception {  
    if (topo == null)  
        throw new Exception("Erro!");  
    int elemento = topo.elemento;  
    Celula tmp = topo;  
    topo = topo.prox;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```

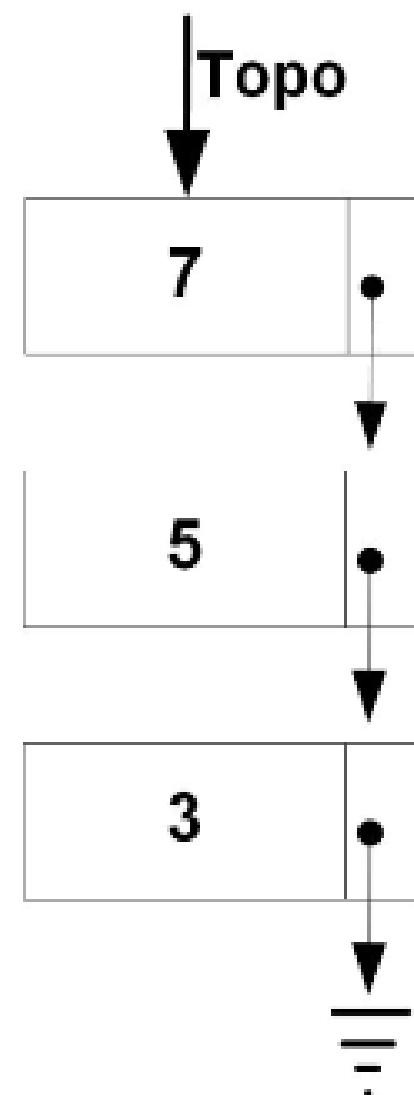


Pilha - Remover

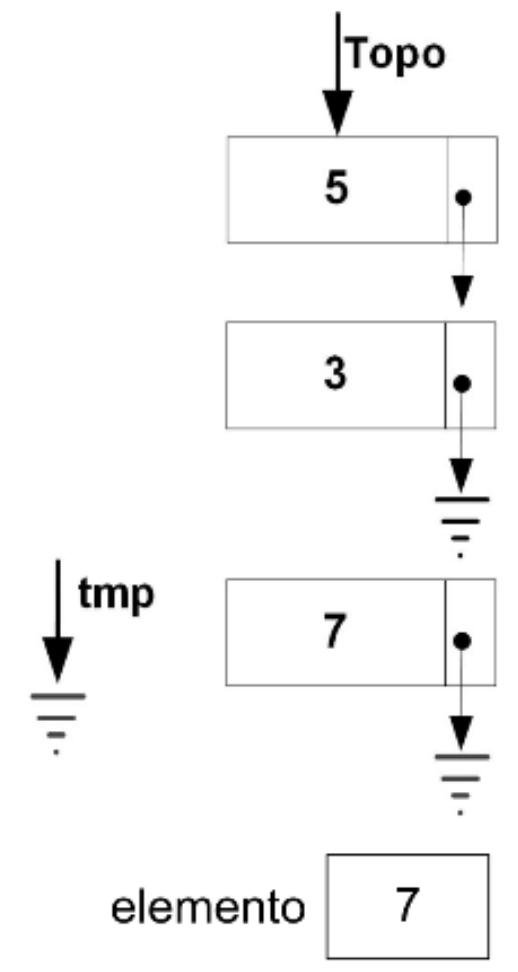
Pilha - Remover

ATENÇÃO: 3 e 7 não estão ligados

```
public int remover() throws Exception {  
    if (topo == null)  
        throw new Exception("Erro!");  
    int elemento = topo.elemento;  
    Celula tmp = topo;  
    topo = topo.prox;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```



```
public int remover() throws Exception {  
    if (topo == null)  
        throw new Exception("Erro!");  
    int elemento = topo.elemento;  
    Celula tmp = topo;  
    topo = topo.prox;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```



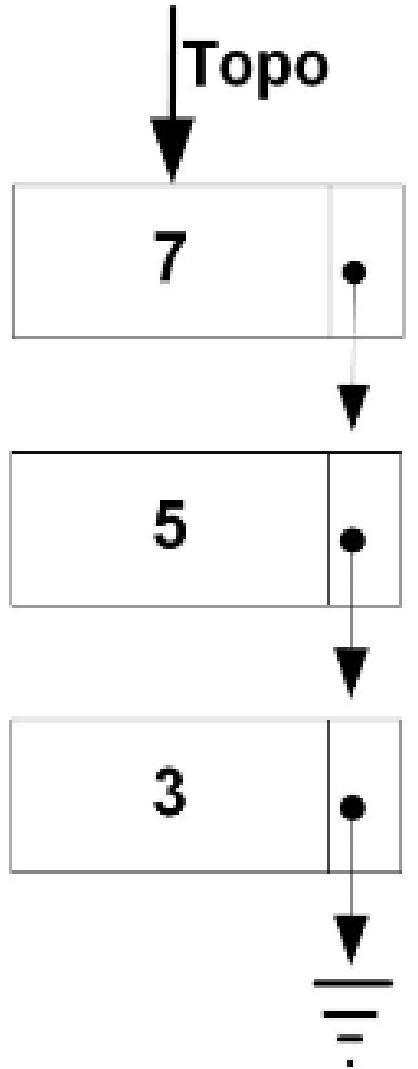
Pilha - Remover

ATENÇÃO: 3 e 7 não estão ligados

Pilha - Mostrar

O que você faria para mostrar todas as células? Seguindo as restrições da Pilha.

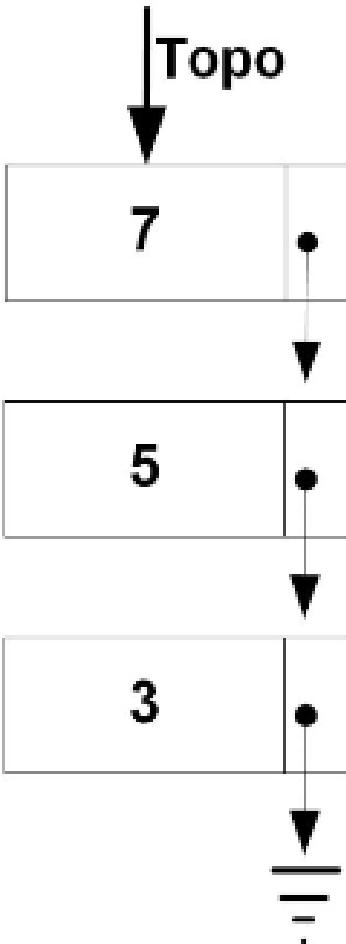
```
class Pilha {  
    private Celula topo;  
    public Pilha () {  
        topo = null;  
    }  
    public void inserir(int x) { ... }  
    public int remover() { ... }  
    public void mostrar() { ... }  
}
```



Pilha - Mostrar

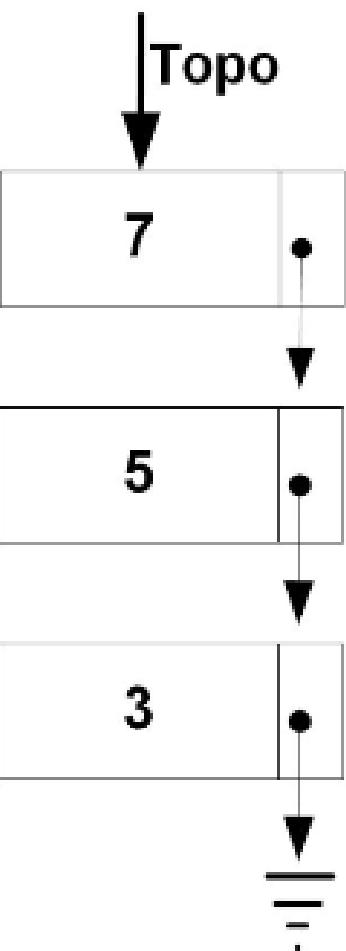
A estratégia é a partir do topo descer até a base da pilha, seguindo os ponteiros.

```
public void mostrar() {  
    System.out.print("[ ");  
    for (Celula i = topo; i != null; i = i.prox){  
        System.out.print(i.elemento + " ");  
    }  
    System.out.println("]");  
}
```



Pilha - Exercícios

Seja nossa Pilha, faça um método que retorna a soma dos elementos contidos na mesma.



Pilha - Exercícios

Seja nossa Pilha, faça um método que retorna a soma dos elementos contidos na mesma.

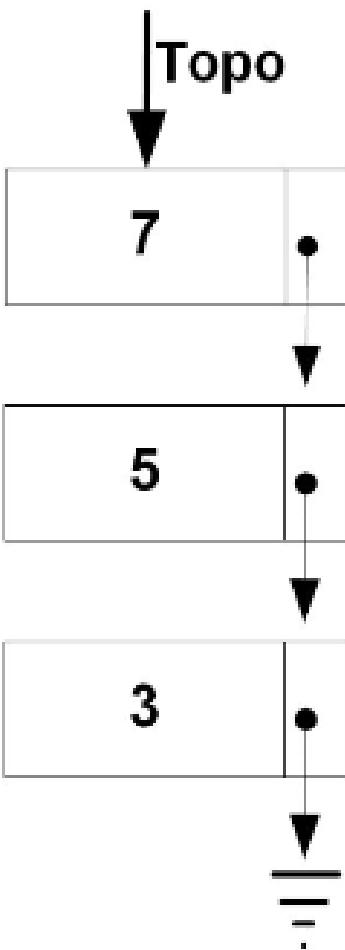
Pilha - Exercícios

Seja nossa Pilha, faça um método que retorna a soma dos elementos contidos na mesma.

```
int somar() {  
    int resp = 0;  
    for (Celula i = topo; i != null; i = i.prox) {  
        resp += i.elemento;  
    }  
    return resp;  
}
```

Pilha - Exercícios

Seja nossa Pilha, faça um método **RECUSIVO** que retorna a soma dos elementos contidos na mesma.



Pilha - Exercícios

Seja nossa Pilha, faça um método **RECUSIVO** que retorna a soma dos elementos contidos na mesma.

Tabnine | Edit | Test | Explain | Document

```
public int somarRec(Celula i) {  
    int soma = 0;  
    if (i != null) {  
        soma = i.elemento + somarRec(i.prox);  
    }  
  
    return soma;  
}
```

Tabnine | Edit | Test | Explain | Document

```
public int somarRec() {  
    return somarRec(topo);  
}
```

Pilha - Exercícios

Seja nossa Pilha, faça um método **RECUSIVO** para mostrar os elementos da pilha na ordem em que os mesmos foram inseridos.

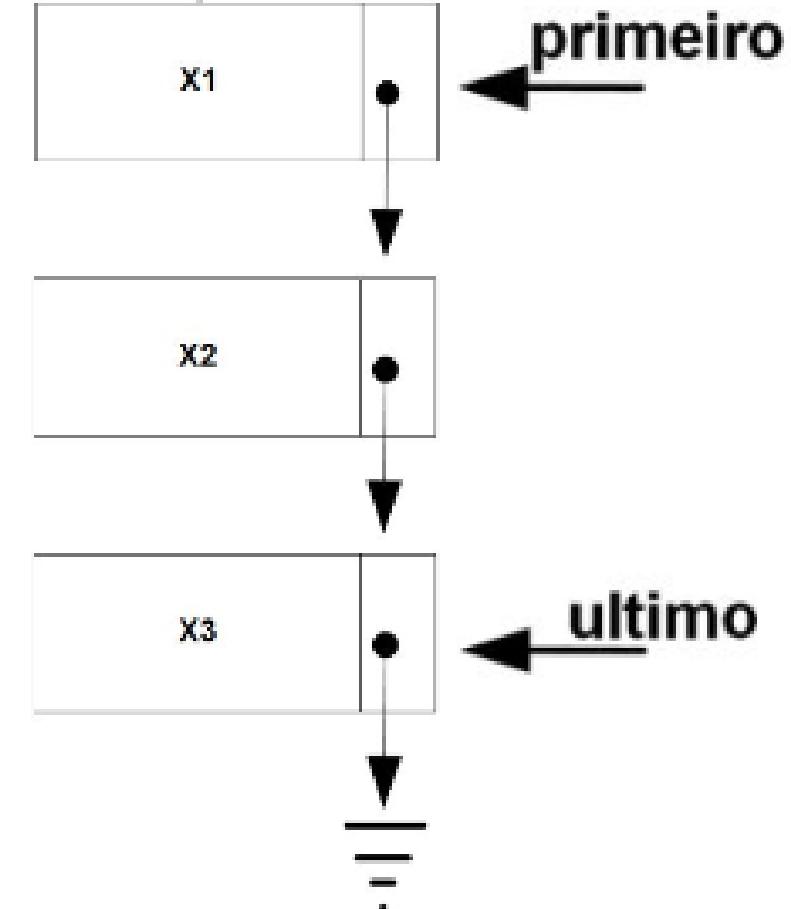
Pilha - Exercícios

Seja nossa Pilha, faça um método **RECUSIVO** para mostrar os elementos da pilha na ordem em que os mesmos foram inseridos.

```
public static void imprimirPilhaOrdemInsercao(Stack<Integer> pilha) {
    if (pilha.topo == -1) {
        return;
    }

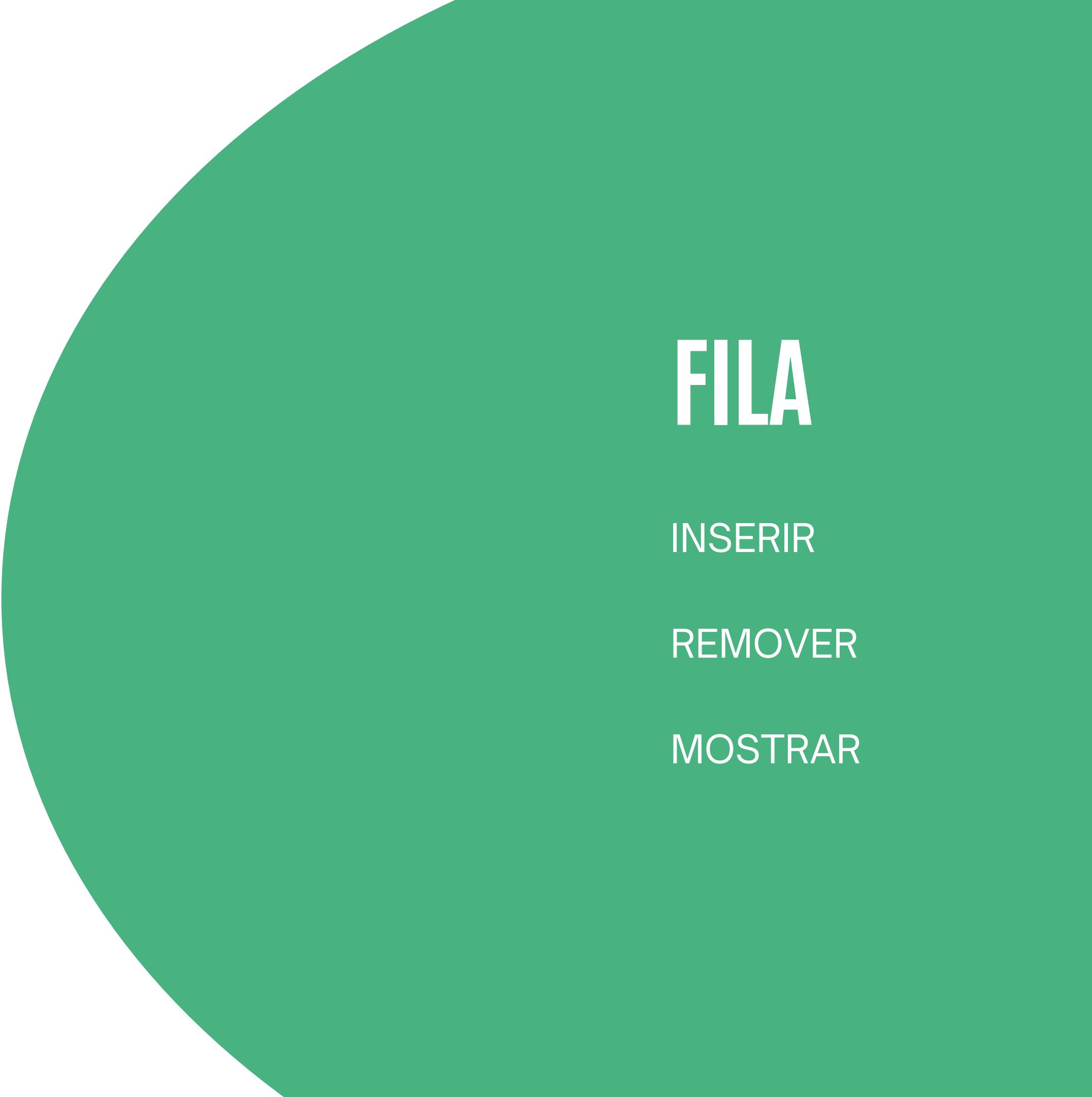
    int elemento = pilha.pop();
    imprimirPilhaOrdemInsercao(pilha);
    System.out.print(elemento + " ");
    pilha.push(elemento);
}
```

```
class Celula {  
    public int elemento;  
    public Celula prox;  
    public Celula() {  
        this(0);  
    }  
    public Celula (int x) {  
        this.elemento = x;  
        this.prox = null;  
    }  
}
```



FILA

Vamos combinar uma forma de visualizar a fila e relembrar a estrutura de célula que estamos usando.



FILA

INSERIR

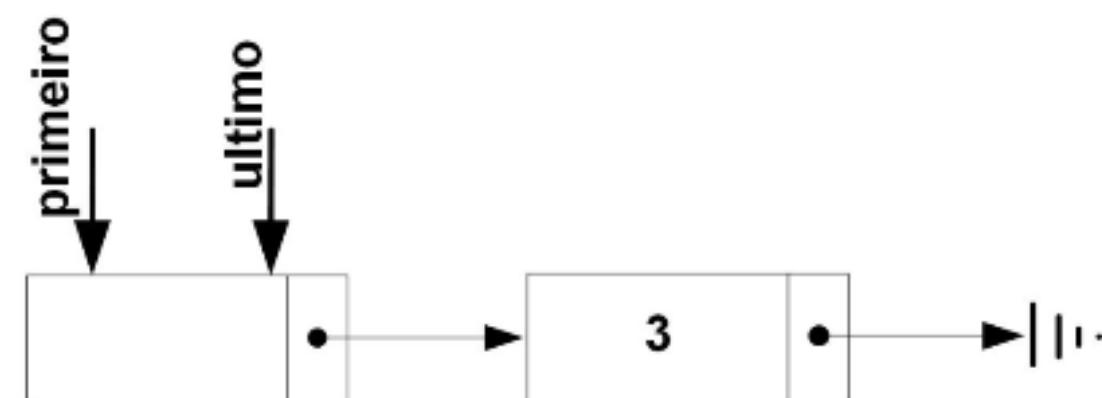
REMOVER

MOSTRAR

```
public void inserir(int x) { //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```

FILA - Inserir

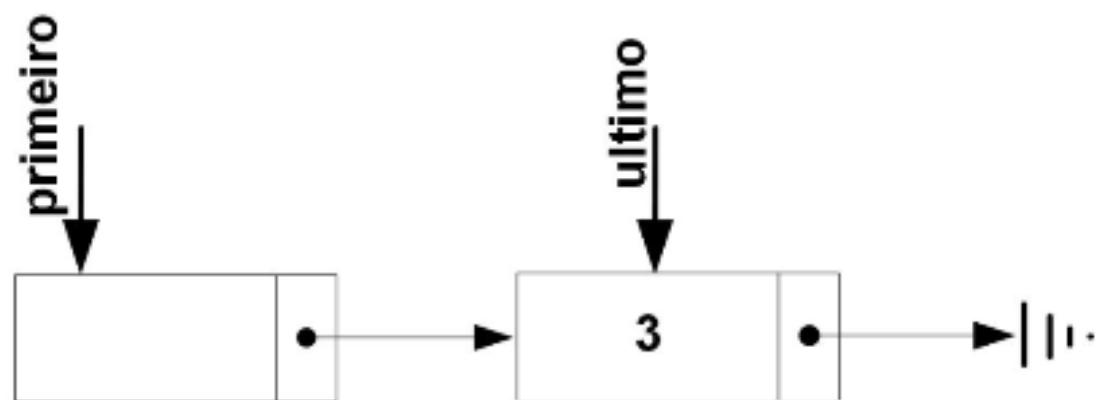
```
public void inserir(int x) { //Inserir(3)  
    ultimo.prox = new Celula(x);  
    ultimo = ultimo.prox;  
}
```



FILA - Inserir

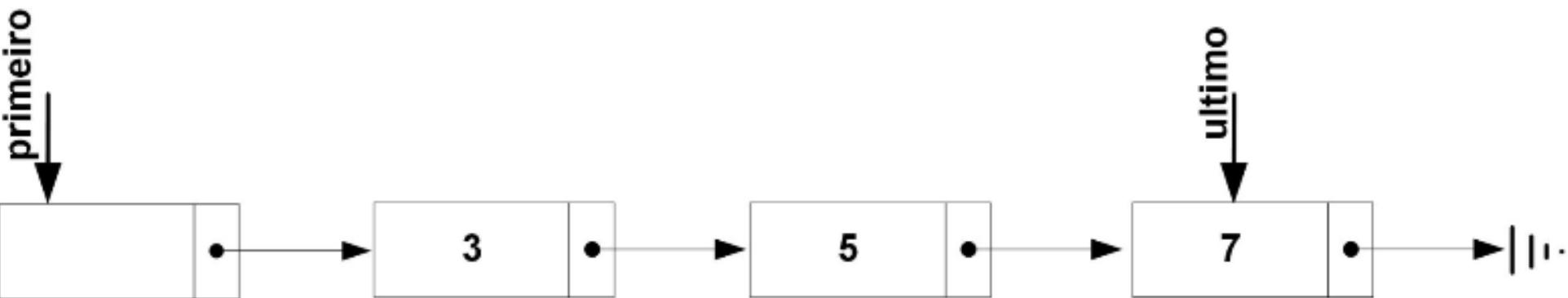
```
public void inserir(int x) { //Inserir(3)
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```

FILA - Inserir



FILA - Remover

```
public int remover() throws Exception{  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula tmp = primeiro;  
    primeiro = primeiro.prox;  
    int elemento = primeiro.elemento;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```



FILA - Remover

A estratégia é transformar o segundo elemento na célula cabeça, assim teremos removido do início.

Se o primeiro for igual ao último, a fila está vazia.

```
public int remover() throws Exception{
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = null;
    tmp = null;
    return elemento;
}
```



FILA - Remover

```

public int remover() throws Exception{
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = null;
    tmp = null;
    return elemento;
}

```

FILA - Remover

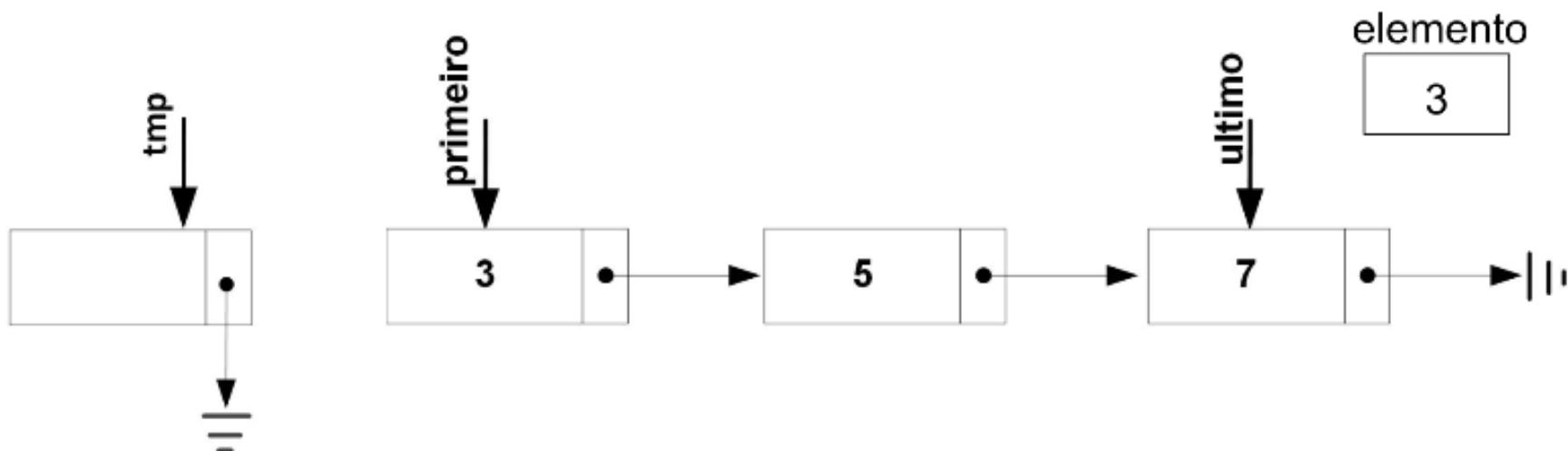
A estratégia é transformar o segundo elemento na célula cabeça, assim teremos removido do início.

Se o primeiro for igual ao último, a fila está vazia.



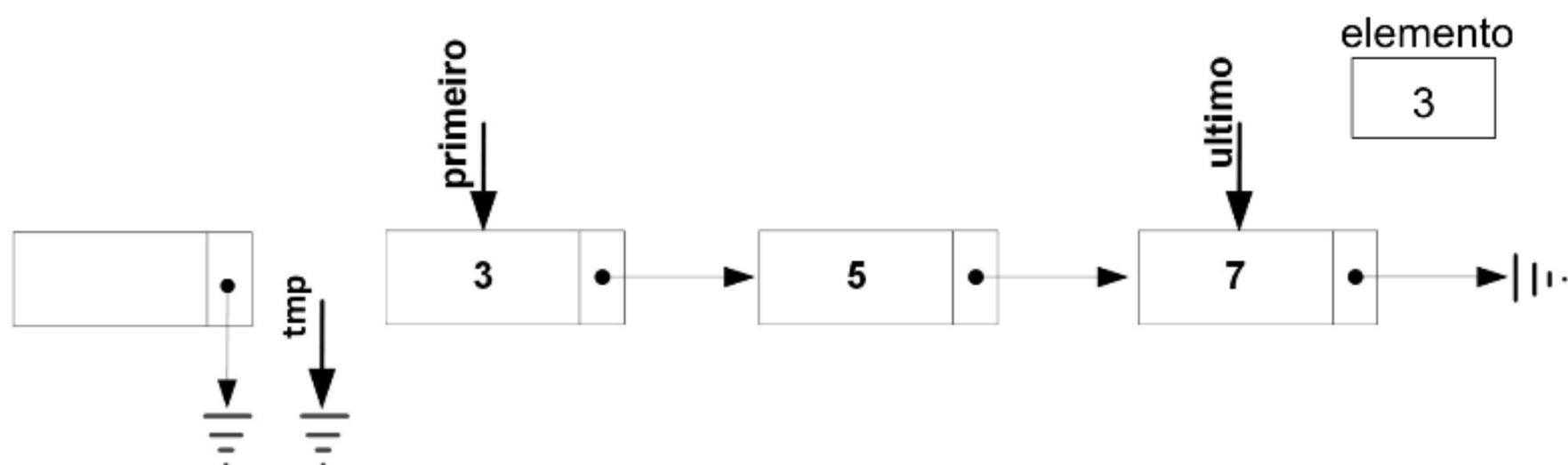
```
public int remover() throws Exception{
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = null;
    tmp = null;
    return elemento;
}
```

FILA - Remover



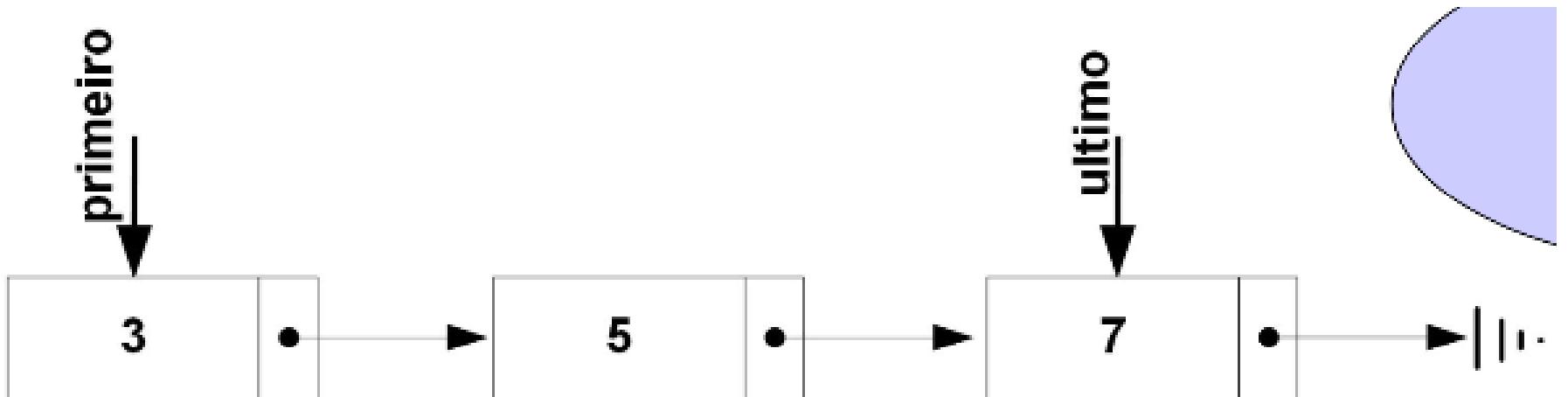
```
public int remover() throws Exception{
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = null;
    tmp = null;
    return elemento;
}
```

FILA - Remover



FILA - Mostrar

```
* Mostra os elementos separados por espacos.  
*/  
public void mostrar() {  
    System.out.print("[ ");  
  
    for(Celula i = primeiro.prox; i != null; i = i.prox) {  
        System.out.print(i.elemento + " ");  
    }  
  
    System.out.println("] ");  
}
```



FILA - Mostrar

FILA - Exercícios

Seja nossa Fila, faça um método para retornar o terceiro elemento supondo que o mesmo existe.

```
int retornarTerceiroElemento(){  
    return (primeiro.prox.prox.prox.elemento);  
}
```

FILA - Exercícios

Seja nossa Fila, faça um método para retornar o terceiro elemento supondo que o mesmo existe.

FILA - Exercícios

Seja nossa Fila, faça um método que inverta a ordem dos seus elementos.

```
void inverter () {  
    Celula fim = ultimo;  
    while (primeiro != fim){  
        Celula nova = new Celula (primeiro.prox.elemento);  
        nova.prox = fim.prox;  
        fim.prox = nova;  
        Celula tmp = primeiro.prox;  
        primeiro.prox = tmp.prox;  
        nova = tmp = tmp.prox = null;  
        if (ultimo == fim) {ultimo = ultimo.prox; }  
    }  
    fim = null;  
}
```

FILA - Exercícios

Seja nossa Fila, faça um método que inverta a ordem dos seus elementos.

Lista Simples

INserir no Início

INserir no Fim

INserir

Remover no Início

Remover no Fim

Remover

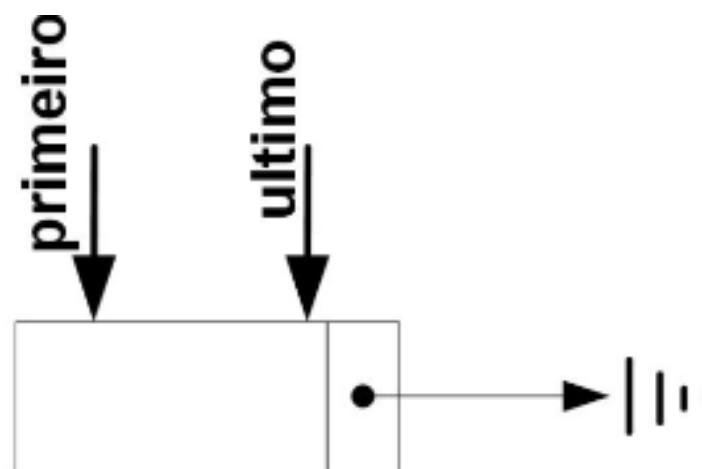
Lista Simples

```
class Lista {  
    private Celula primeiro, ultimo;  
    public Lista () {  
        primeiro = new Celula();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Simples

Criação da Celula cabeça, assim como nas demais estruturas passadas (Pilha e Fila).

```
class Lista {  
    private Celula primeiro, ultimo;  
    public Lista () {  
        primeiro = new Celula();  
        ultimo = primeiro;  
    }  
    public void inserirInício(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInício() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```



Lista Simples

Os métodos Inserir Fim e Remover Início são iguais aos métodos da Fila.

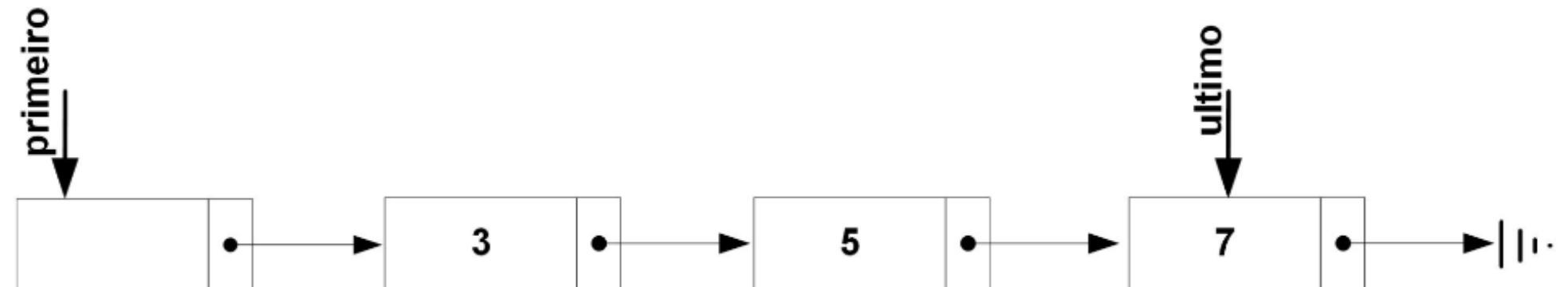
```
class Lista {  
    private Celula primeiro, ultimo;  
    public Lista () {  
        primeiro = new Celula();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```



Lista Simples

O método Mostrar é igual ao da Fila/Pilha

```
class Lista {  
    private Celula primeiro, ultimo;  
    public Lista () {  
        primeiro = new Celula();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```



Lista Simples

Sendo assim, nos restou os seguintes 4 métodos

```
class Lista {  
    ...  
    public void inserirInicio(int x) { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
}
```



Lista Simples Inserir Início

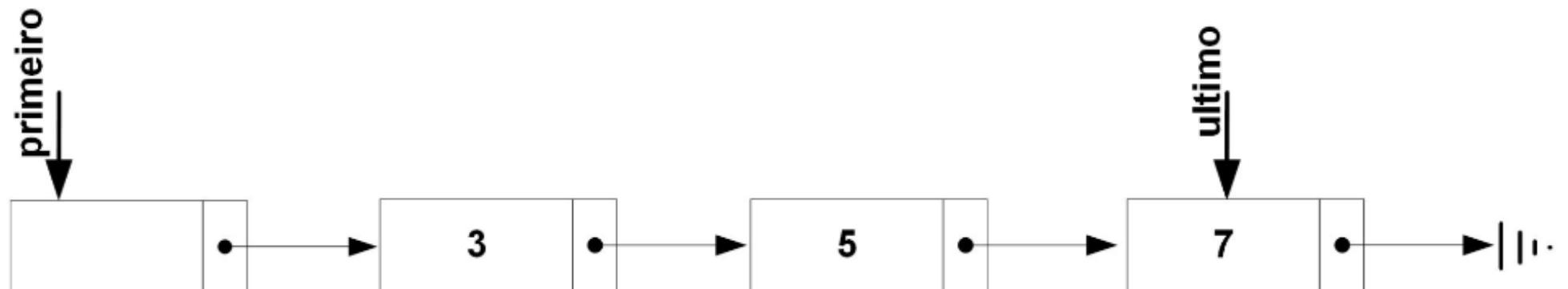
```
class Lista {  
    ...  
    public void inserirInicio(int x) { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
}
```



Lista Simples Inserir Início

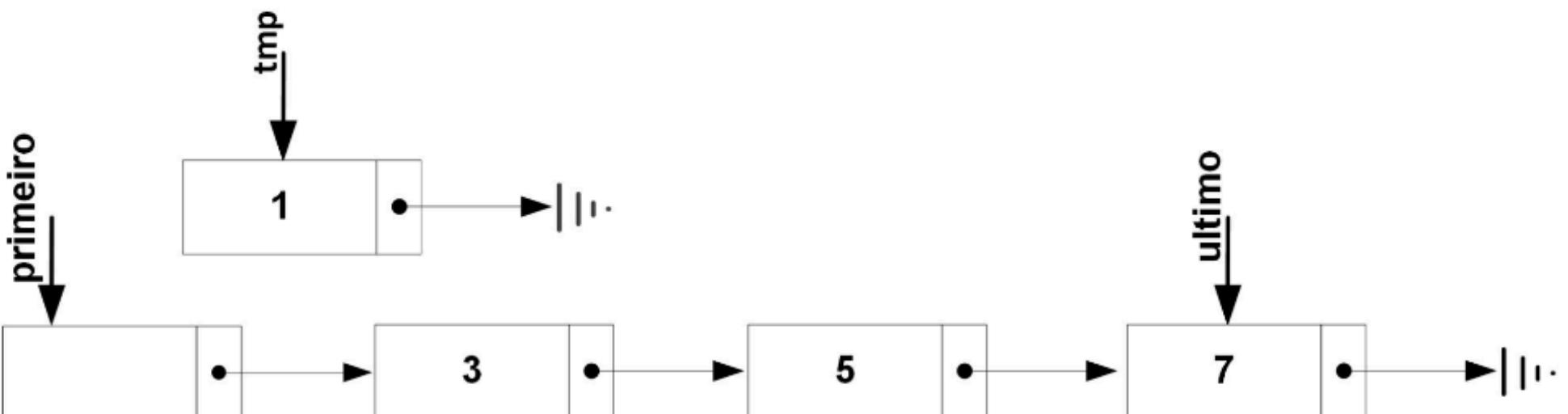
Sendo assim, nos restou os
seguintes 4 métodos

```
//inserirInicio(1)
public void inserirInicio(int x) {
    Celula tmp = new Celula(x);
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    }
    tmp = null;
}
```



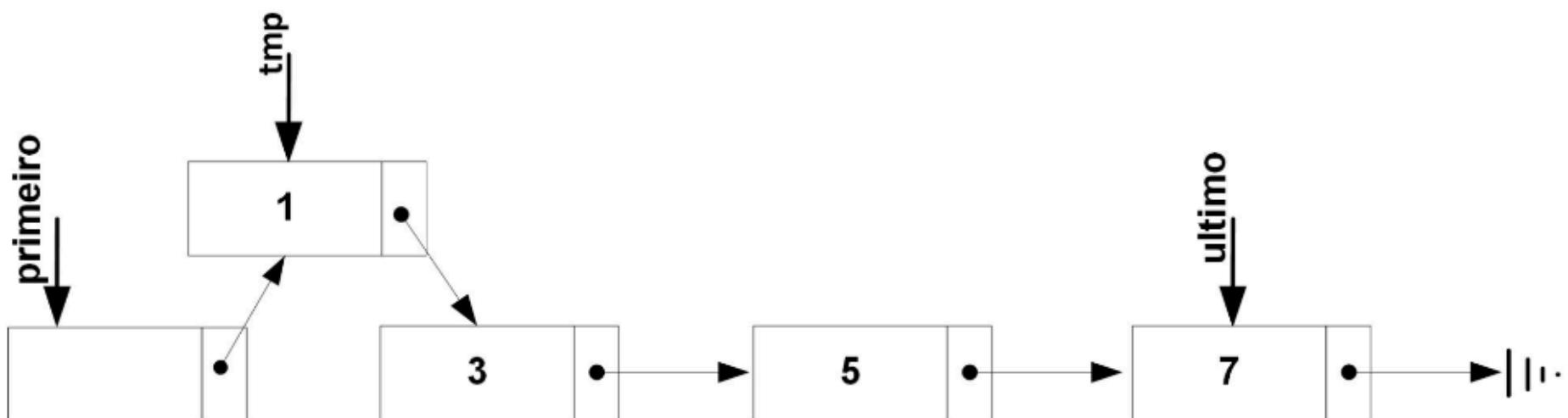
Lista Simples Inserir Início

```
//inserirInicio(1)
public void inserirInicio(int x) {
    Celula tmp = new Celula(x);
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    }
    tmp = null;
}
```



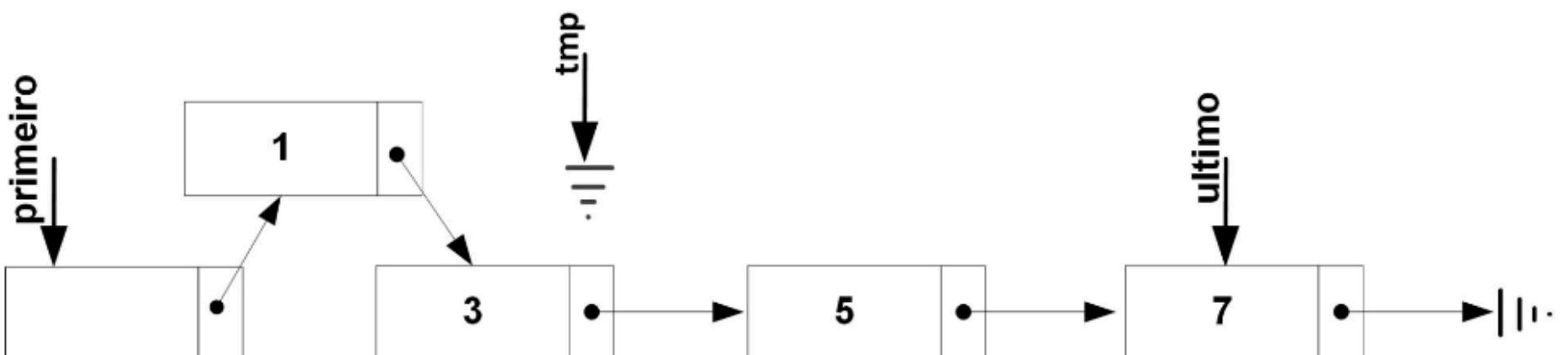
Lista Simples Inserir Início

```
//inserirInicio(1)
public void inserirInicio(int x) {
    Celula tmp = new Celula(x);
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    }
    tmp = null;
}
```



Lista Simples Inserir Início

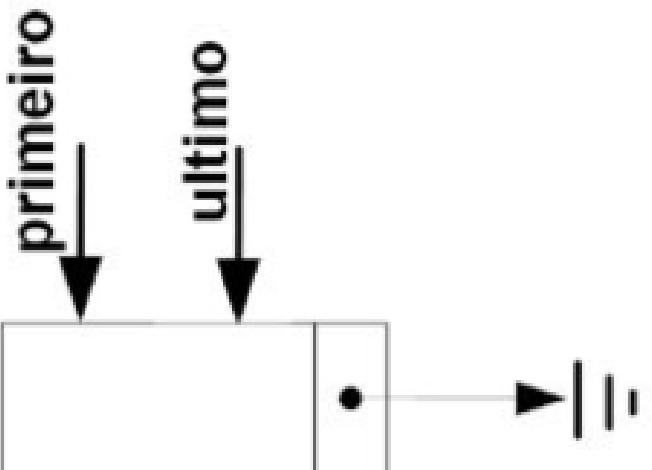
```
//inserirInicio(1)
public void inserirInicio(int x) {
    Celula tmp = new Celula(x);
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    }
    tmp = null;
}
```



Lista Simples Inserir Início

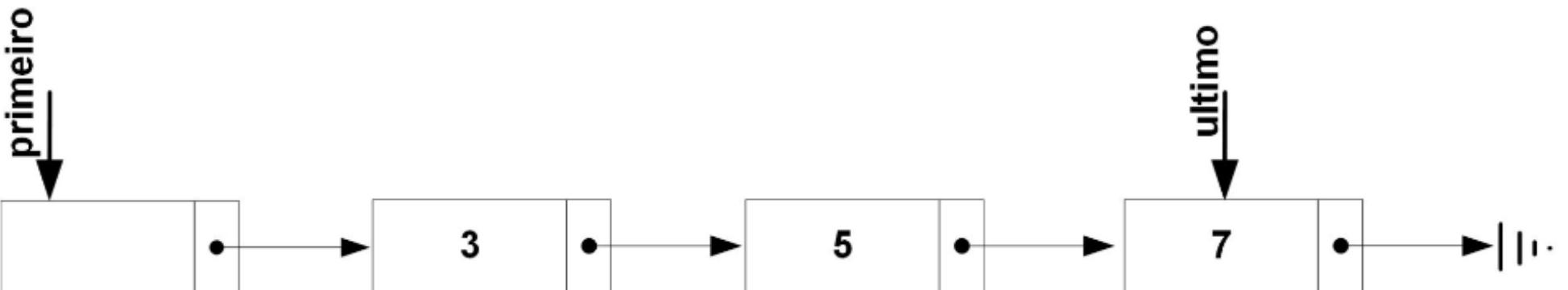
Como ficaria o método
Inserir Início desta figura?

```
//inserirInicio(1)
public void inserirInicio(int x) {
    Celula tmp = new Celula(x);
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    }
    tmp = null;
}
```



Lista Simples Remover Fim

```
class Lista {  
    ...  
    public void inserirInicio(int x) { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
}
```



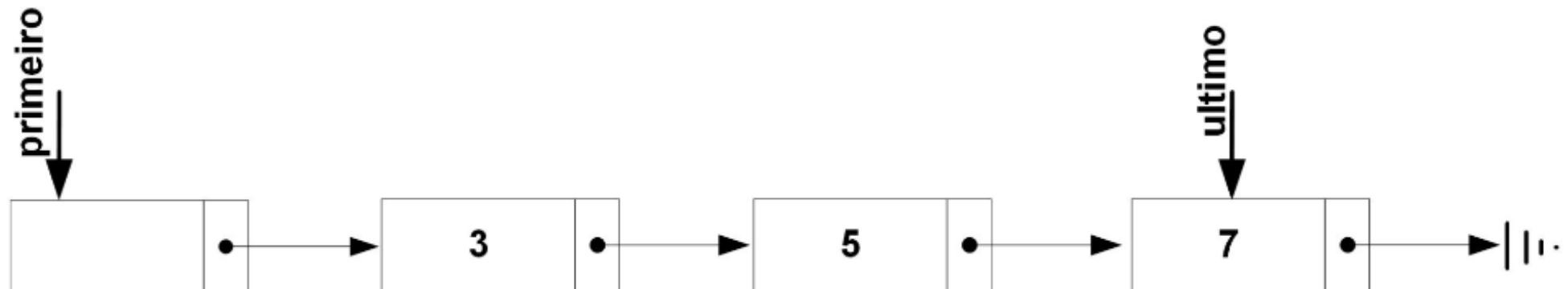
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



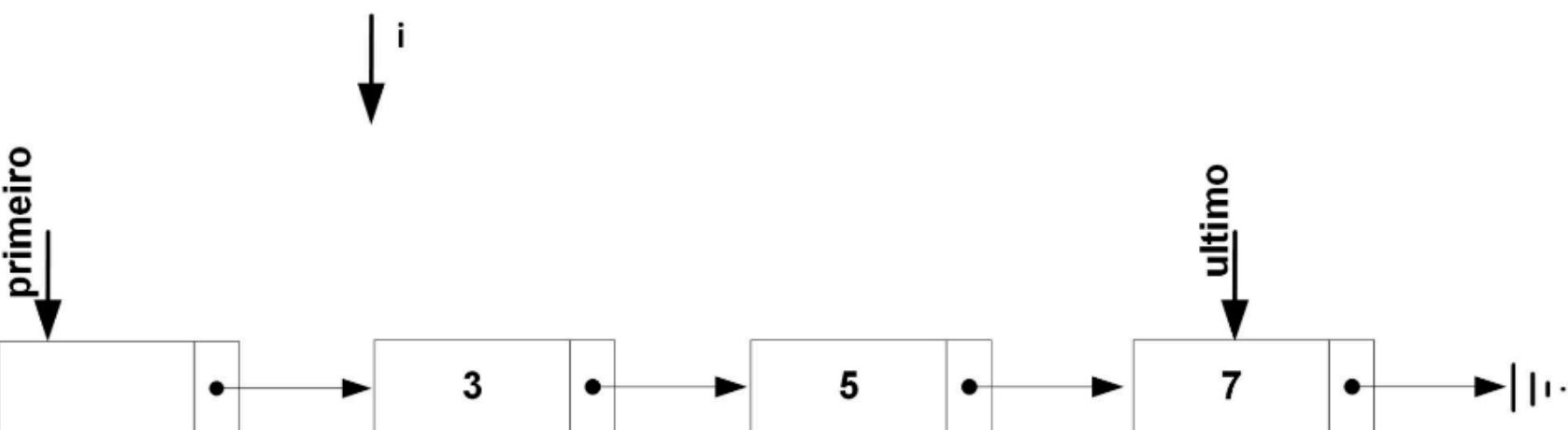
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;      false  
    return elemento;  
}
```



Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;      true  
    return elemento;  
}
```



Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



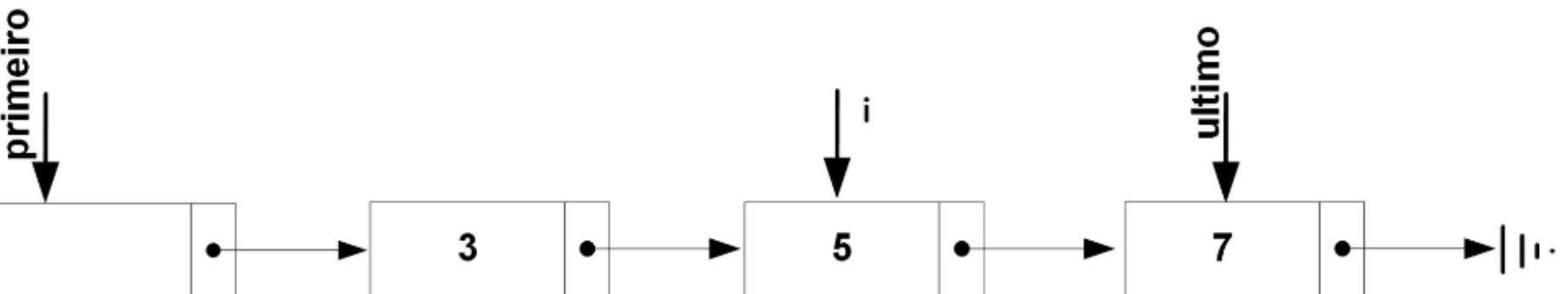
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;      true  
    return elemento;  
}
```



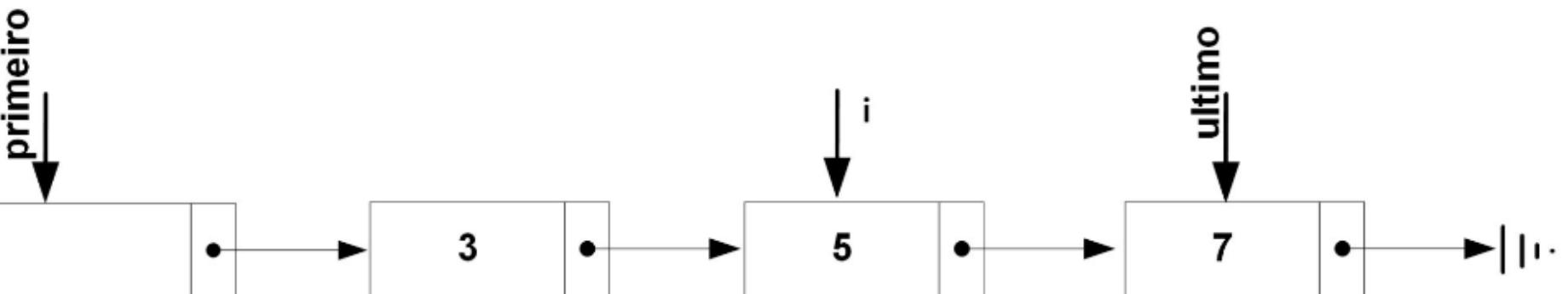
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



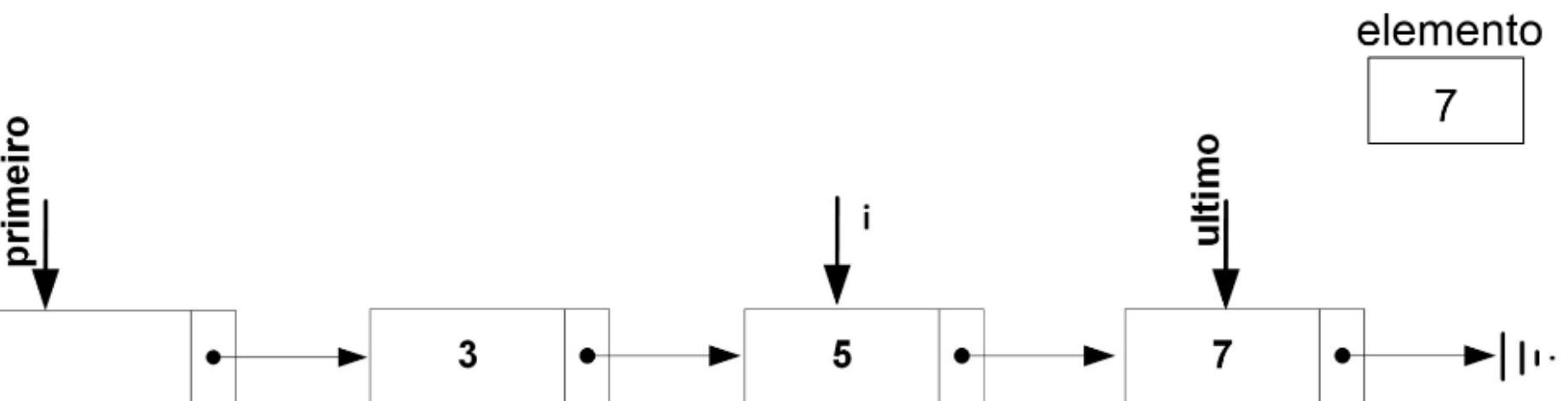
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;           false  
    return elemento;  
}
```



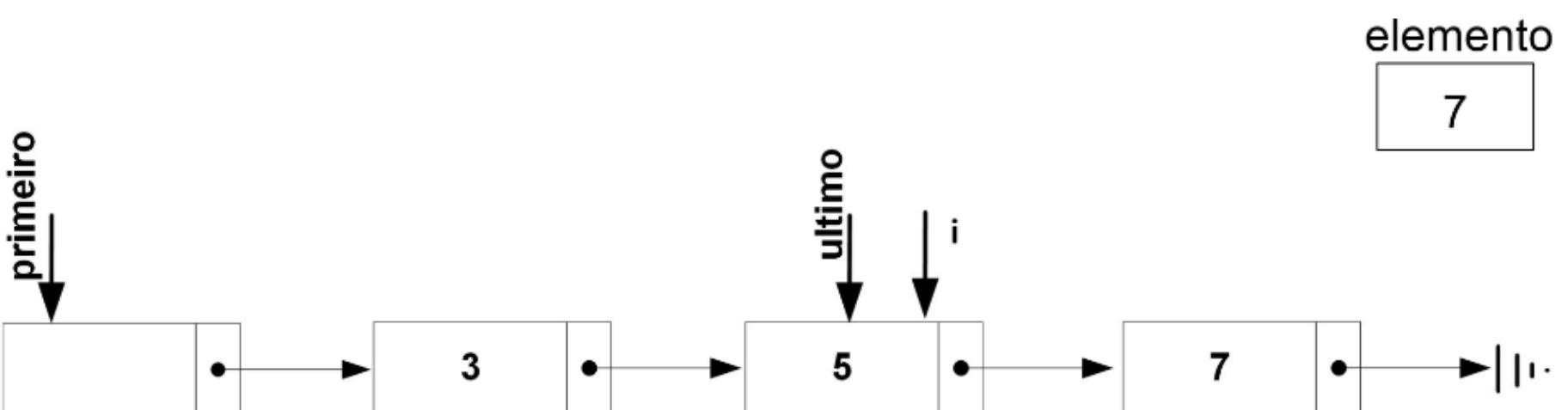
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



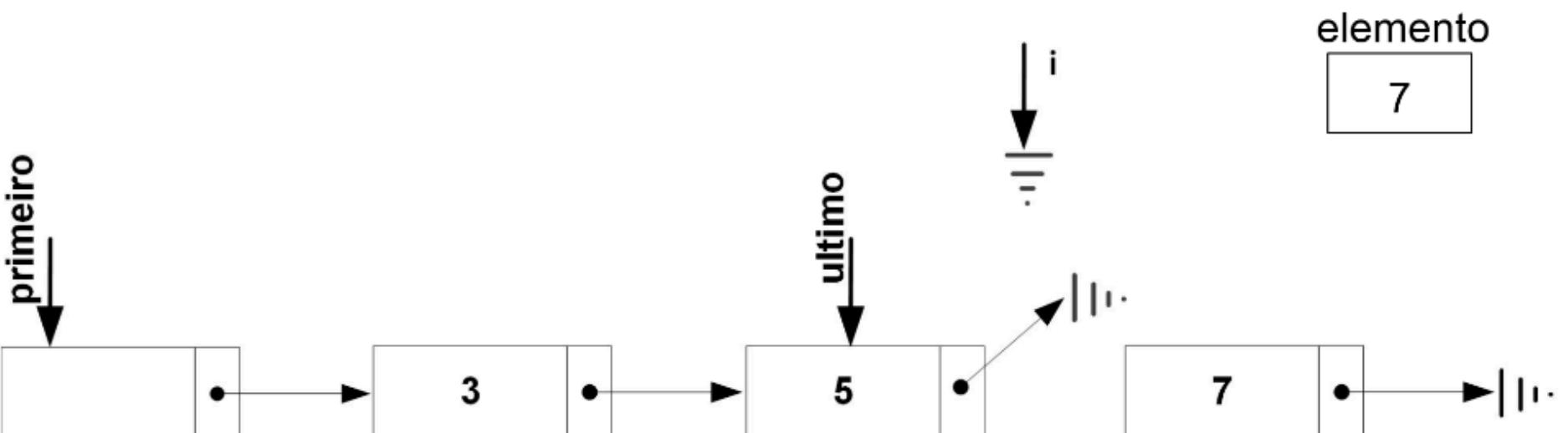
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



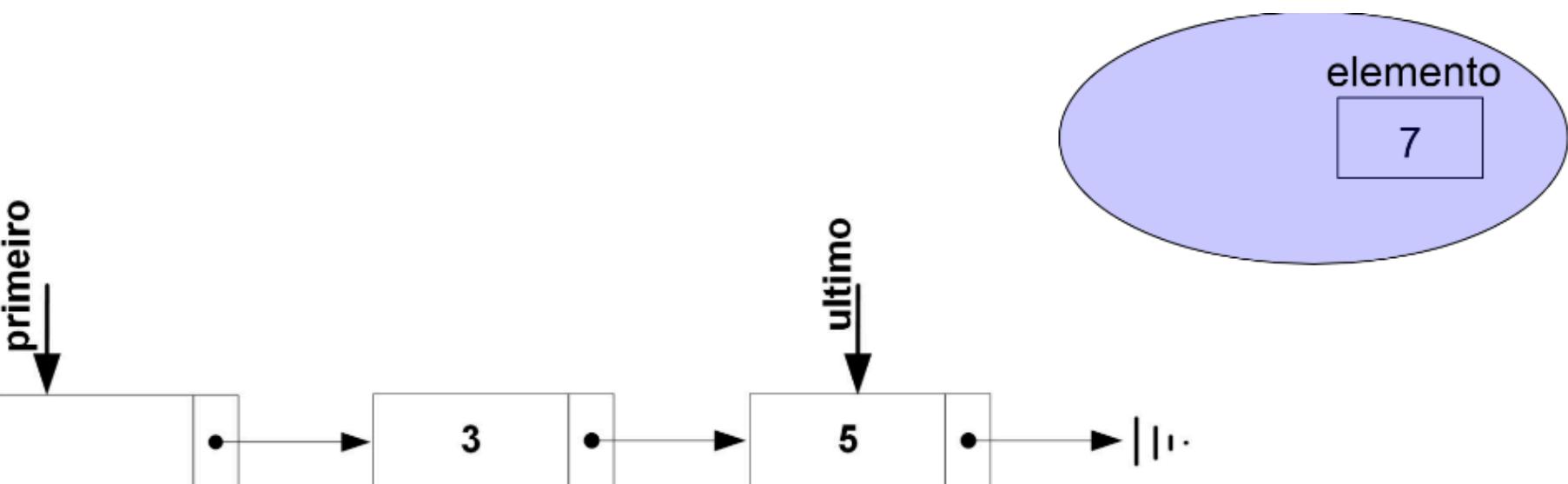
Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



Lista Simples Remover Fim

```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
    i = ultimo.prox = null;  
    return elemento;  
}
```



Lista Simples Inserir

```
class Lista {  
    ...  
    public void inserirInicio(int x) { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
}
```

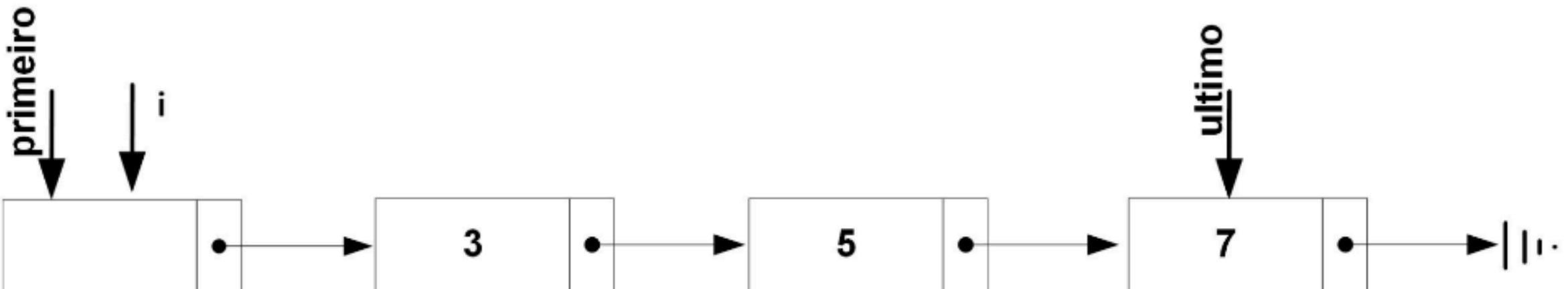
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



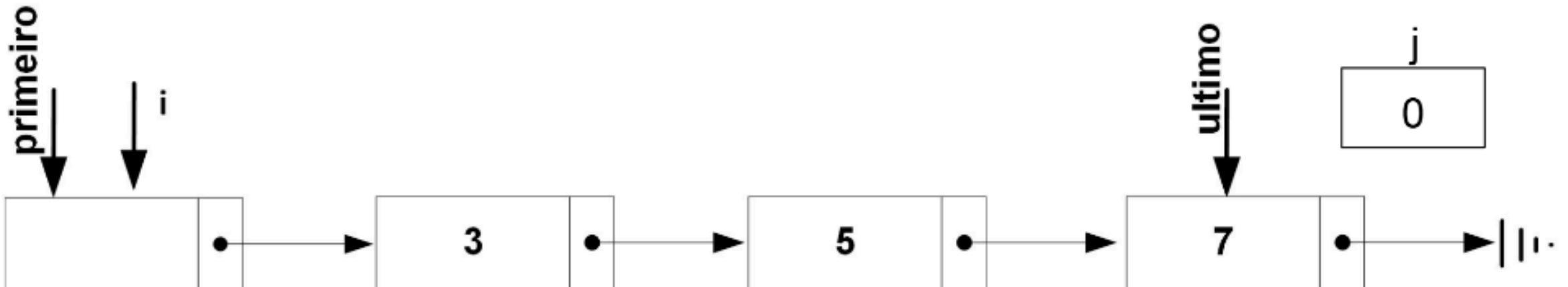
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception { //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0){ inserirInicio(x);}
    } else if (pos == tamanho){ inserirFim(x);}
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



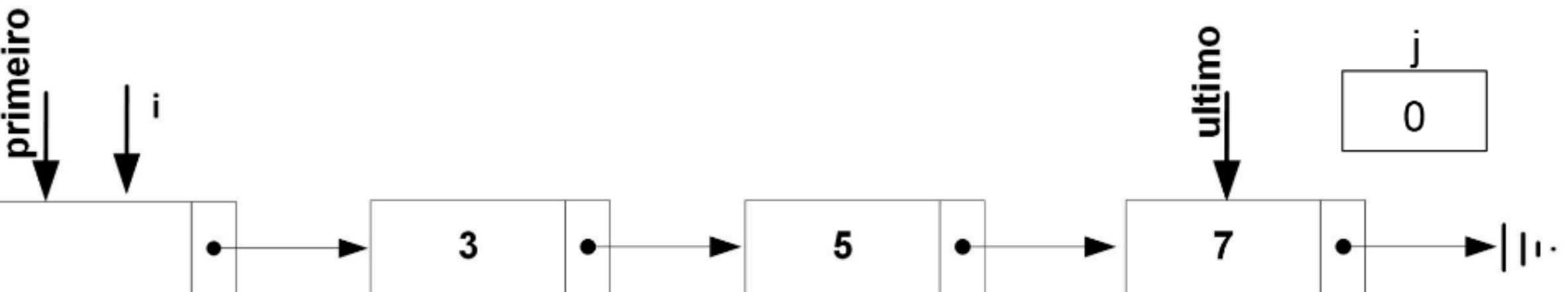
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```



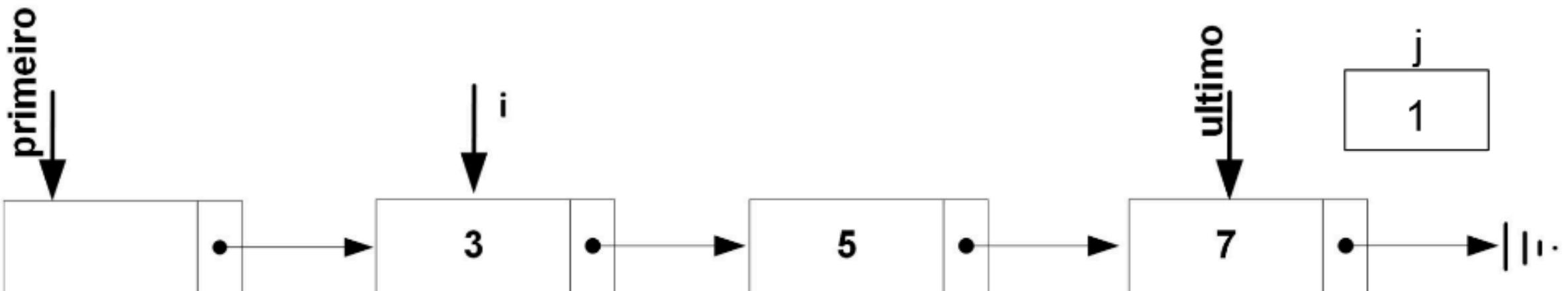
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0;j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



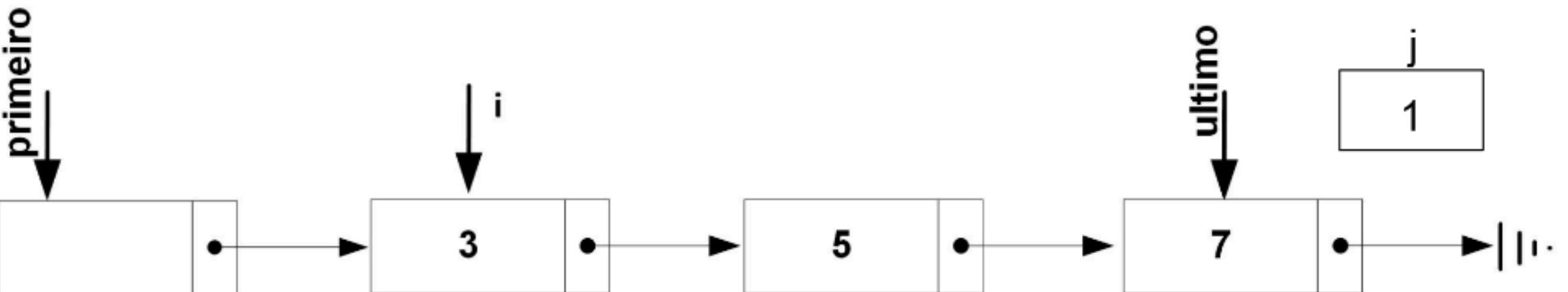
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception { //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0){ inserirInicio(x);}
    } else if (pos == tamanho){ inserirFim(x);}
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



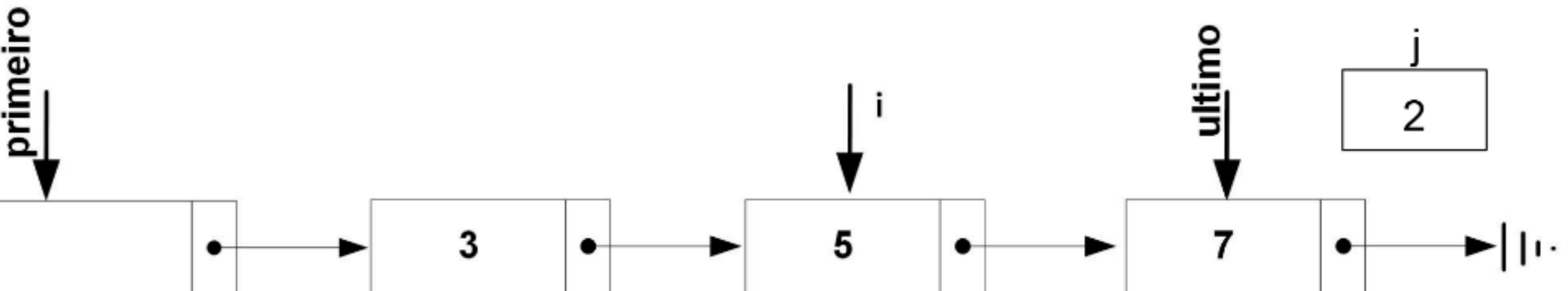
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0;j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```



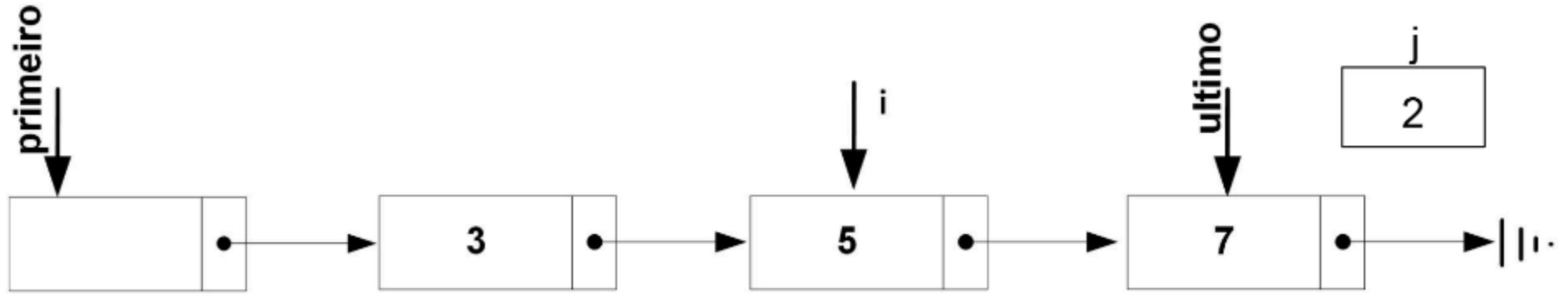
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
} else if (pos == 0){           inserirInicio(x);
} else if (pos == tamanho){    inserirFim(x);
} else {
    Celula i = primeiro;
    for(int j = 0; j < pos; j++, i = i.prox);
    Celula tmp = new Celula(x);
    tmp.prox = i.prox;
    i.prox = tmp;
    tmp = i = null;
} }
```



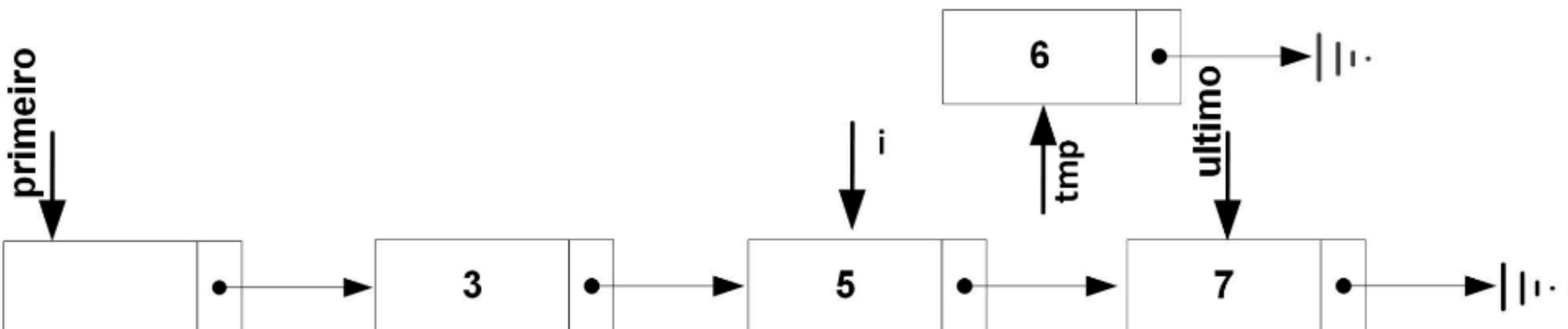
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
} else if (pos == 0){           inserirInicio(x);
} else if (pos == tamanho){    inserirFim(x);
} else {
    Celula i = primeiro;
    for(int j = 0; j < pos; j++, i = i.prox);
    Celula tmp = new Celula(x);
    tmp.prox = i.prox;
    i.prox = tmp;
    tmp = i = null;
}
}
```



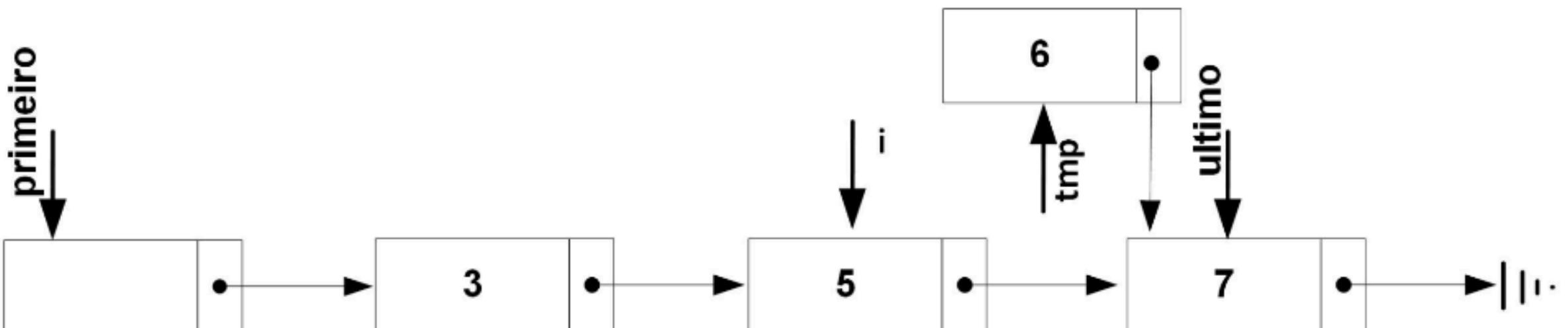
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0){ inserirInicio(x);
    } else if (pos == tamanho){ inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



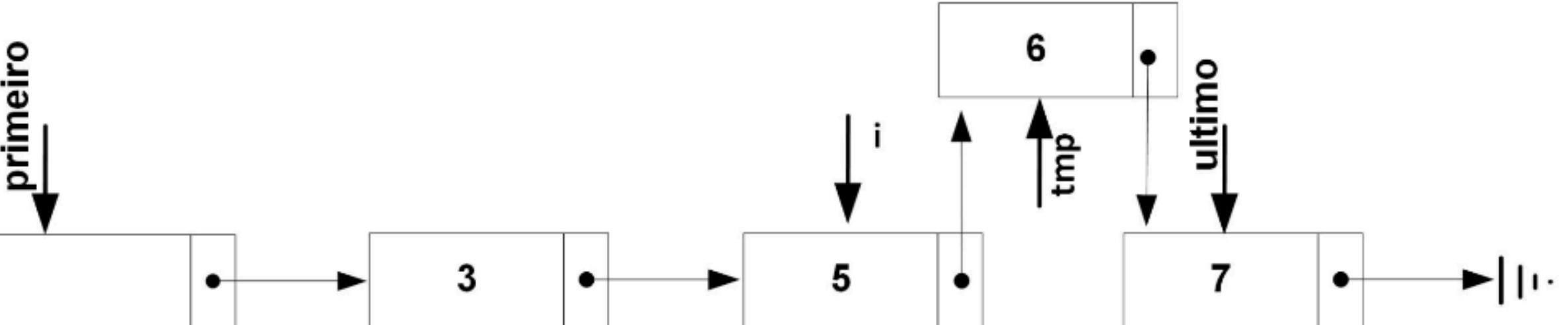
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    } }
```



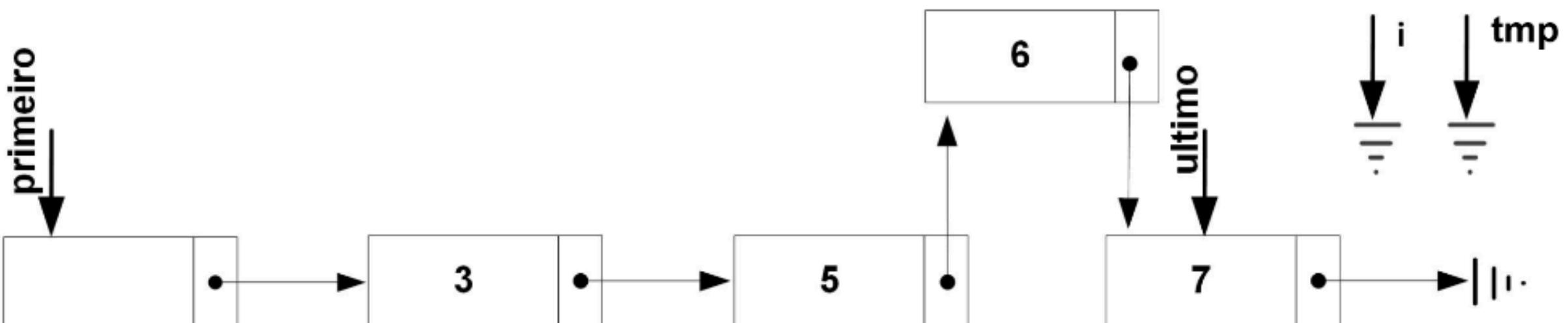
Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception { //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){ inserirInicio(x);
    } else if (pos == tamanho){ inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```



Lista Simples Inserir

```
public void inserir(int x, int pos) throws Exception {      //Inserir(6, 2)
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){ throw new Exception("Erro!");
    } else if (pos == 0){           inserirInicio(x);
    } else if (pos == tamanho){    inserirFim(x);
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```

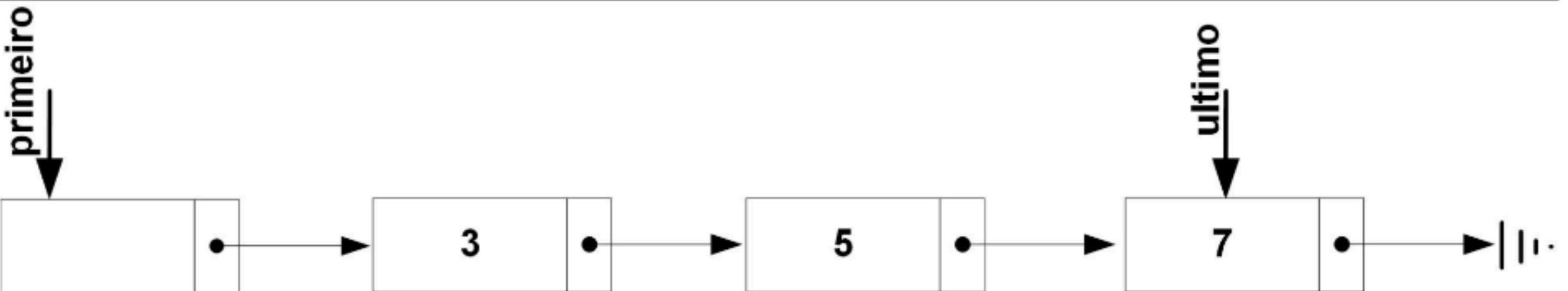


Lista Simples Remover

```
class Lista {  
    ...  
    public void inserirInicio(int x) { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
}
```

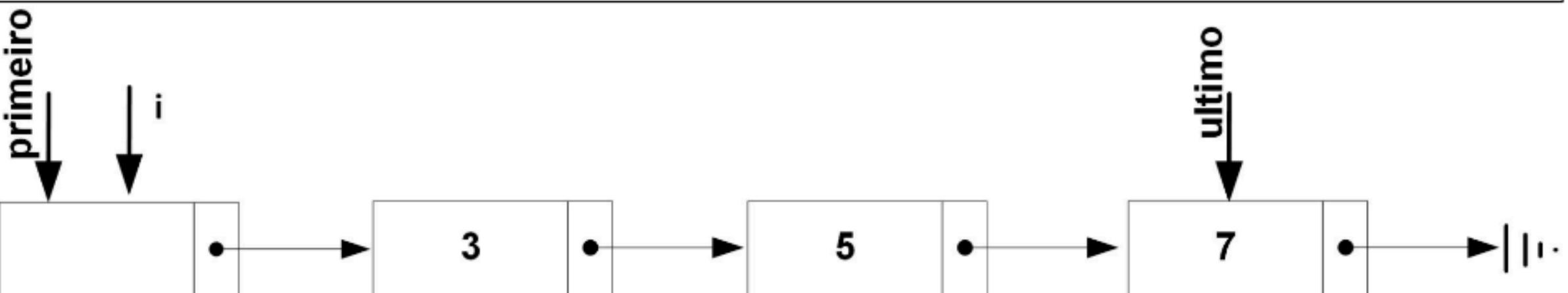
Lista Simples Remover

```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                     elemento = removerInicio(); }
    } else if (pos == tamanho - 1){           elemento = removerFim(); }
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;      i.prox = tmp.prox;
        tmp.prox = null;             i = tmp = null;
    }
    return elemento;
}
```



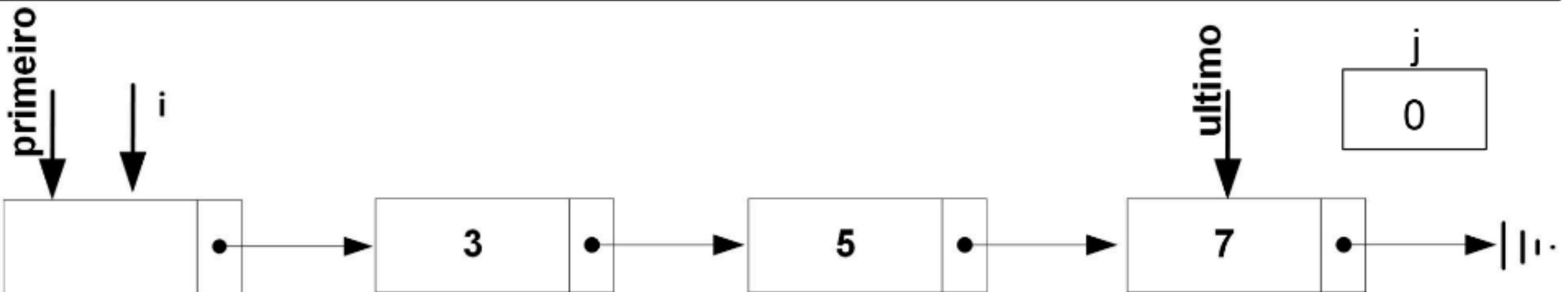
Lista Simples Remover

```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                     elemento = removerInicio();}
    } else if (pos == tamanho - 1){           elemento = removerFim();}
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;      i.prox = tmp.prox;
        tmp.prox = null;             i = tmp = null;
    }
    return elemento;
}
```



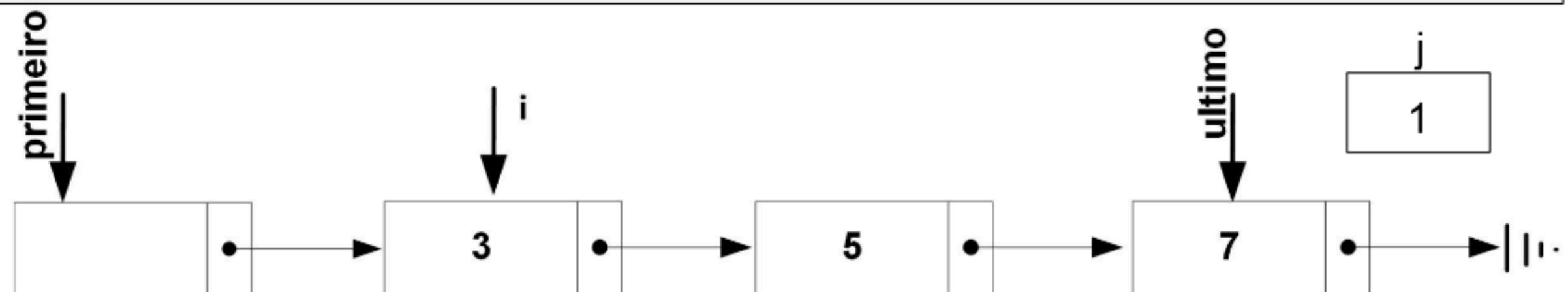
Lista Simples Remover

```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                                elemento = removerInicio();
    } else if (pos == tamanho - 1){                      elemento = removerFim();
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;      i.prox = tmp.prox;
        tmp.prox = null;              i = tmp = null;
    }
    return elemento;
}
```



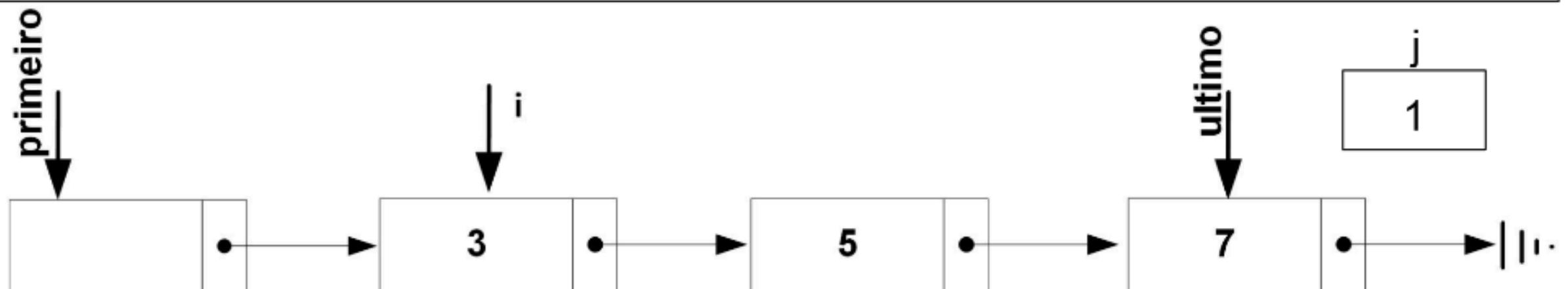
Lista Simples Remover

```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                     elemento = removerInicio(); }
    } else if (pos == tamanho - 1){           elemento = removerFim(); }
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;      i.prox = tmp.prox;
        tmp.prox = null;             i = tmp = null;
    }
    return elemento;
}
```



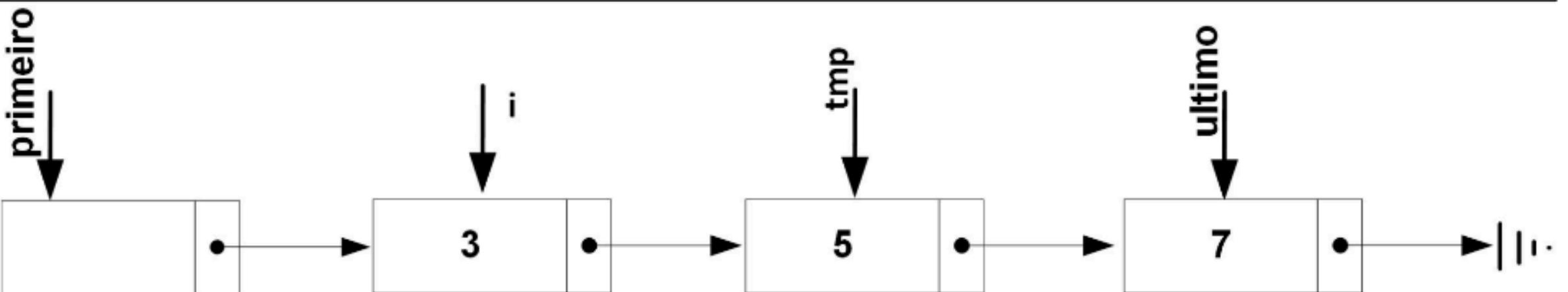
Lista Simples Remover

```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                               elemento = removerInicio();
    } else if (pos == tamanho - 1){                     elemento = removerFim();
    } else {
        Celula i = primeiro;
        for(int j = 0;j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;           i.prox = tmp.prox;
        tmp.prox = null;                  i = tmp = null;
    }
    return elemento;
}
```



Lista Simples Remover

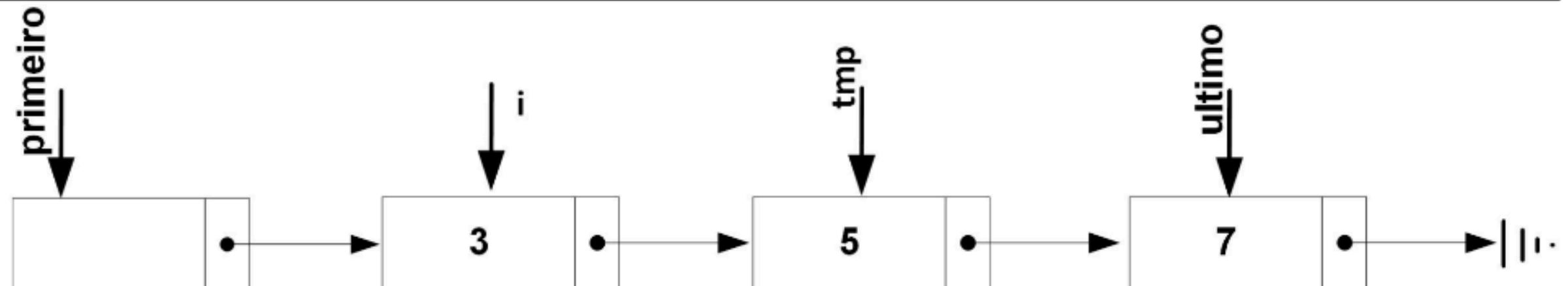
```
public int remover(int pos) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                               elemento = removerInicio();
    } else if (pos == tamanho - 1){                     elemento = removerFim();
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;           i.prox = tmp.prox;
        tmp.prox = null;                  i = tmp = null;
    }
    return elemento;
}
```



Lista Simples Remover

```
public int remover(int pos) throws Exception { //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) { elemento = removerInicio(); }
    } else if (pos == tamanho - 1){ elemento = removerFim(); }
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;
        tmp.prox = null;
        i.prox = tmp.prox;
        i = tmp = null;
    }
    return elemento;
}
```

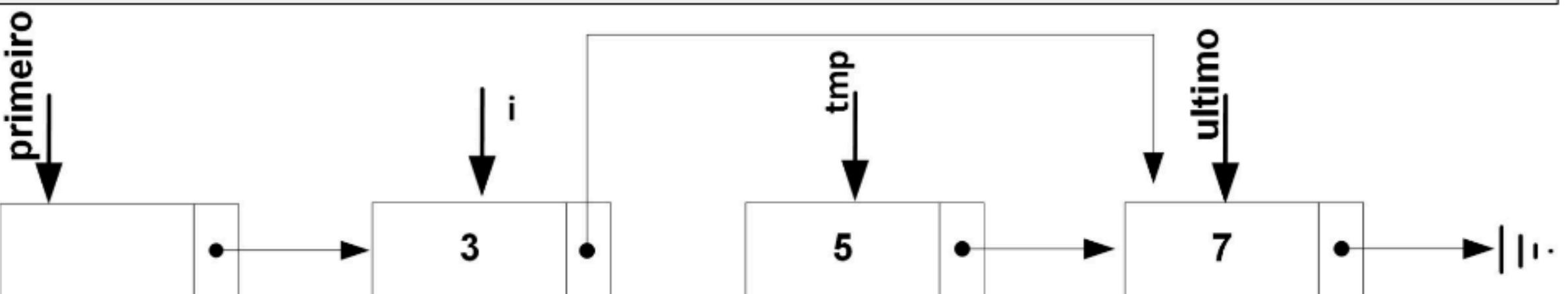
elemento
5



Lista Simples Remover

```
public int remover(int pos) throws Exception { //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) { elemento = removerInicio(); }
    } else if (pos == tamanho - 1){ elemento = removerFim(); }
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento; i.prox = tmp.prox;
        tmp.prox = null; i = tmp = null;
    }
    return elemento;
}
```

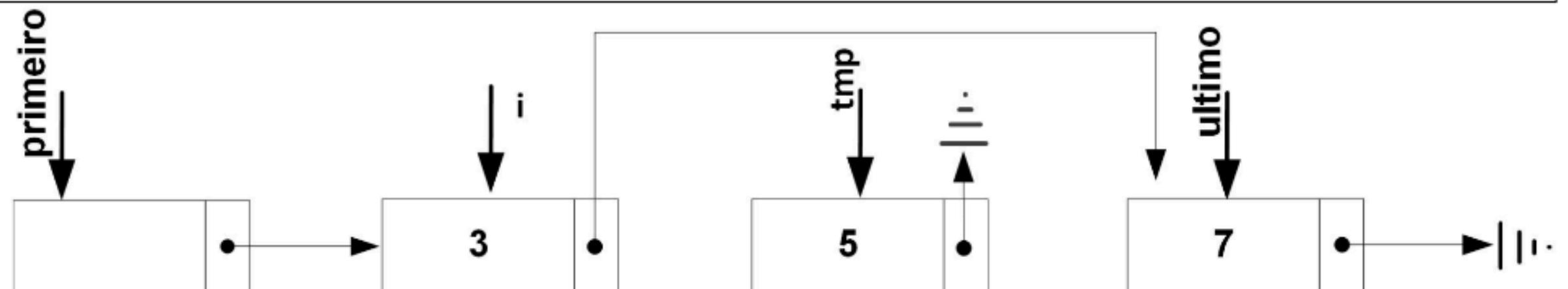
elemento
5



Lista Simples Remover

```
public int remover(int pos) throws Exception {           //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) {                               elemento = removerInicio();
    } else if (pos == tamanho - 1){                     elemento = removerFim();
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;          i.prox = tmp.prox;
        tmp.prox = null;                  i = tmp = null;
    }
    return elemento;
}
```

elemento
5



Lista Simples Remover

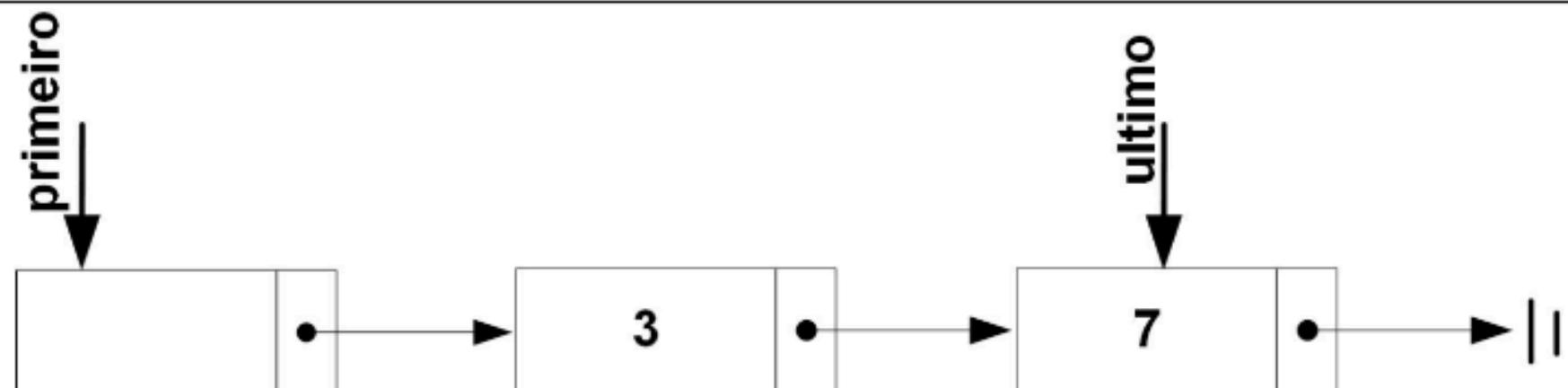
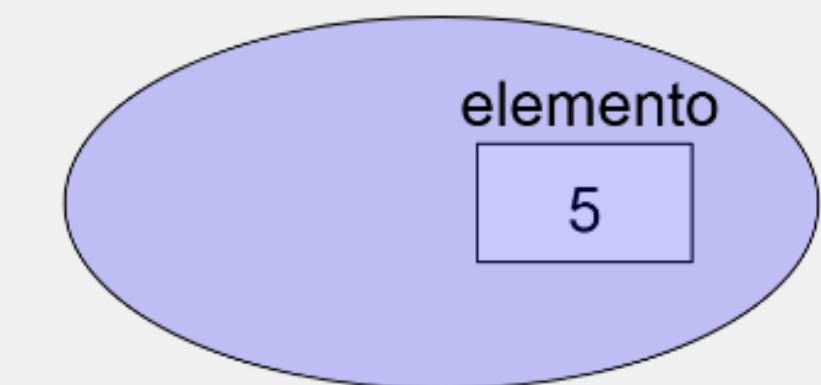
```
public int remover(int pos) throws Exception { //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");}
    } else if (pos == 0) { elemento = removerInicio(); }
    } else if (pos == tamanho - 1){ elemento = removerFim(); }
    } else {
        Celula i = primeiro;
        for(int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento; i.prox = tmp.prox;
        tmp.prox = null; i = tmp = null;
    }
    return elemento;
}
```

elemento
5



Lista Simples Remover

```
public int remover(int pos ) throws Exception {          //remover(1)
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo || pos < 0 || pos >= tamanho){ throw new Exception("Erro!");
} else if (pos == 0) {                                elemento = removerInicio();
} else if (pos == tamanho - 1){                      elemento = removerFim();
} else {
    Celula i = primeiro;
    for(int j = 0; j < pos; j++, i = i.prox);
    Celula tmp = i.prox;
    elemento = tmp.elemento;      i.prox = tmp.prox;
    tmp.prox = null;              i = tmp = null;
}
return elemento;
}
```



Lista Dupla

INserir no INÍCIO

INserir no FIM

INserir

REmover no INÍCIO

REmover no FIM

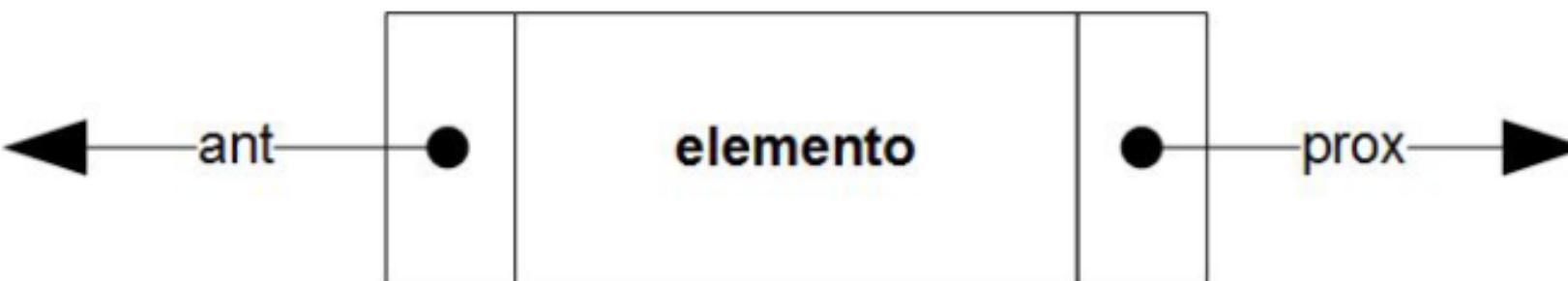
REmover

Lista Dupla

Classe CelulaDupla

Similar a classe **Celula** antes vista, o principal diferencial é a criação de mais um ponteiro.

```
class CelulaDupla {  
    public int elemento;  
    public CelulaDupla prox, ant;  
    public CelulaDupla () {  
        this(0);  
    }  
    public CelulaDupla (int x) {  
        this.elemento = x;  
        this.prox = this.ant = null;  
    }  
}
```



Lista Dupla

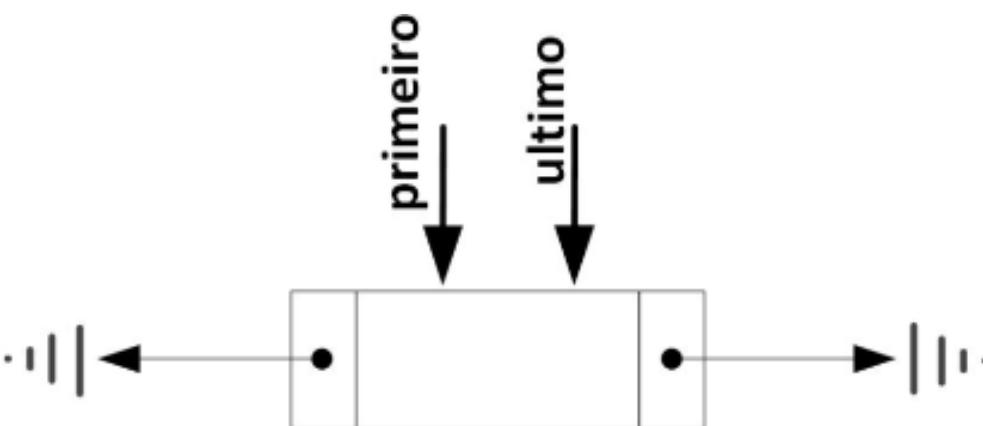
Similar a Lista Simples, contudo considerando o ponteiro **ant**.

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla

Similar a Lista Simples, contudo considerando o ponteiro **ant**.

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```



Lista Dupla

Inserir Início

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla

Inserir Início

//LISTA DUPLA

```
public void inserirInicio(int x) {  
    CelulaDupla tmp = new CelulaDupla(x);  
    tmp.ant = primeiro;  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    } else {  
        tmp.prox.ant = tmp;  
    }  
    tmp = null;  
}
```

//LISTA SIMPLES

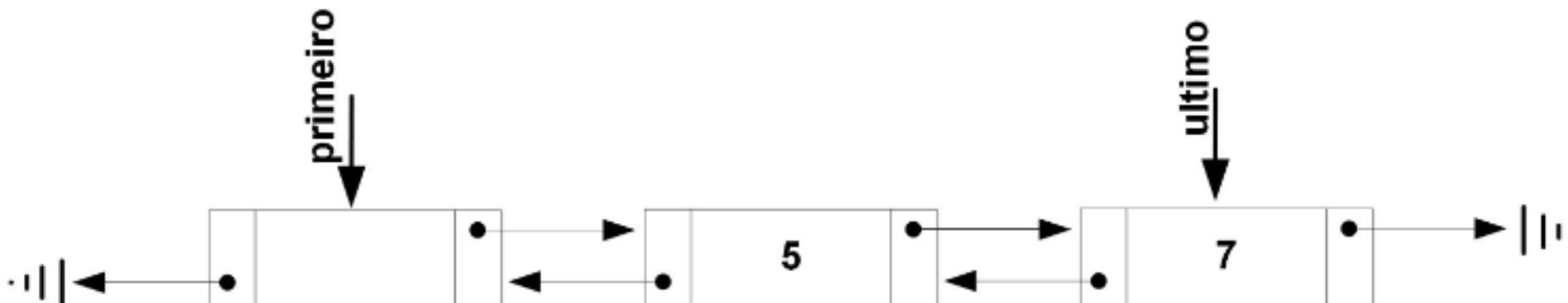
```
public void inserirInicio(int x) {  
    Celula tmp = new Celula(x);  
  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    }  
    tmp = null;  
}
```

Lista Dupla

Inserir Início

Supondo uma lista com os elementos 5 e 7, vamos inserir o 3 no início.

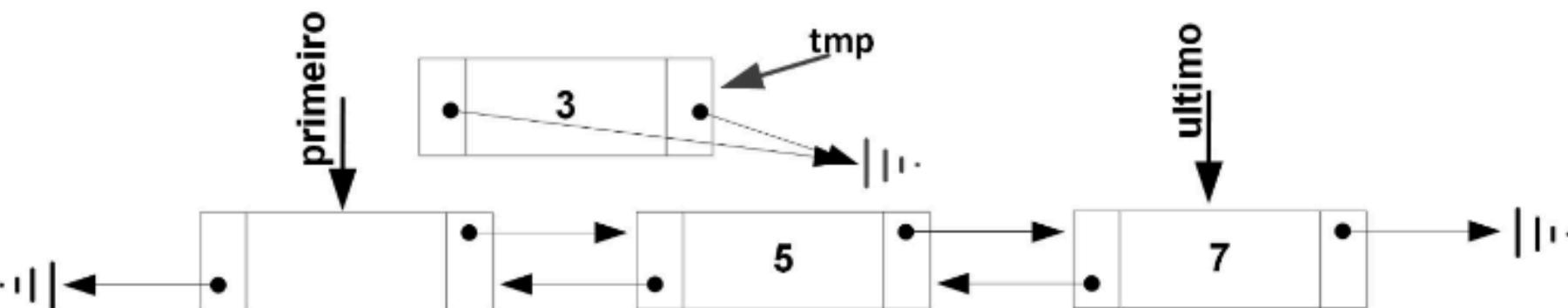
```
//LISTA DUPLA
public void inserirInicio(int x) {
    CelulaDupla tmp = new CelulaDupla(x);
    tmp.ant = primeiro;
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    } else {
        tmp.prox.ant = tmp;
    }
    tmp = null;
}
```



Lista Dupla

Inserir Início

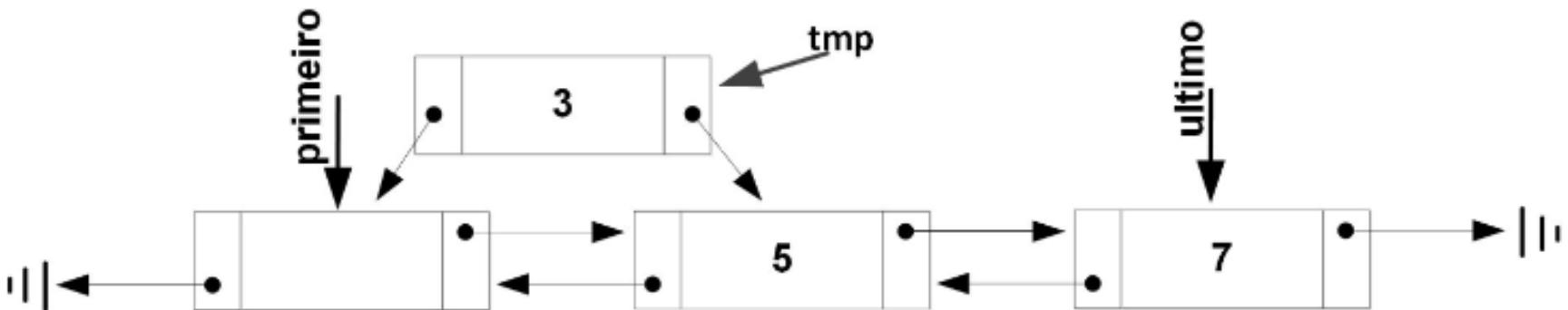
```
//Inserindo o 3 no início
public void inserirInicio(int x) {
    CelulaDupla tmp = new CelulaDupla(x);
    tmp.ant = primeiro;
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    } else {
        tmp.prox.ant = tmp;
    }
    tmp = null;
}
```



Lista Dupla

Inserir Início

```
//Inserindo o 3 no início
public void inserirInicio(int x) {
    CelulaDupla tmp = new CelulaDupla(x);
    tmp.ant = primeiro;
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    } else {
        tmp.prox.ant = tmp;
    }
    tmp = null;
}
```

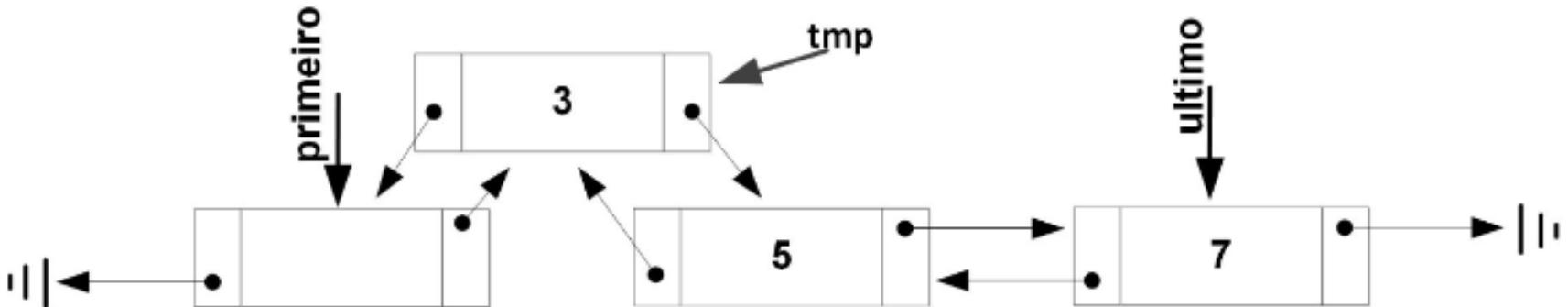


Lista Dupla

Inserir Início

//Inserindo o 3 no início

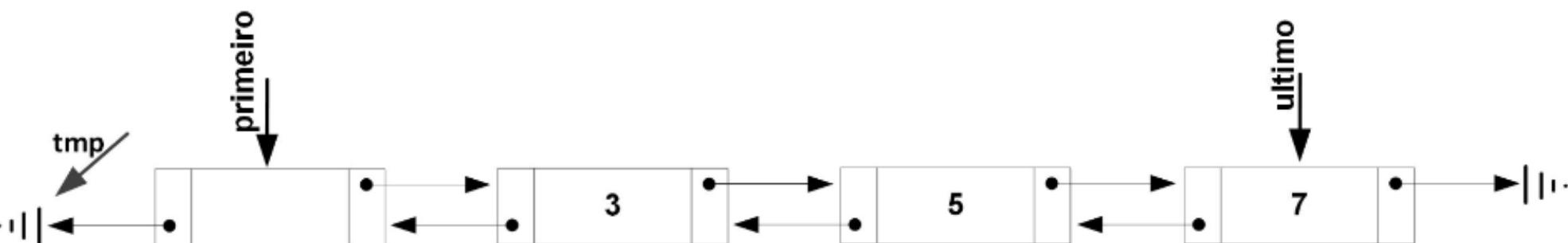
```
public void inserirInicio(int x) {  
    CelulaDupla tmp = new CelulaDupla(x);  
    tmp.ant = primeiro;  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    } else {  
        tmp.prox.ant = tmp;  
    }  
    tmp = null;  
}
```



Lista Dupla

Inserir Início

```
//Inserindo o 3 no início
public void inserirInicio(int x) {
    CelulaDupla tmp = new CelulaDupla(x);
    tmp.ant = primeiro;
    tmp.prox = primeiro.prox;
    primeiro.prox = tmp;
    if (primeiro == ultimo) {
        ultimo = tmp;
    } else {
        tmp.prox.ant = tmp;
    }
    tmp = null;
}
```



Lista Dupla

Inserir Fim

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla

Inserir Fim

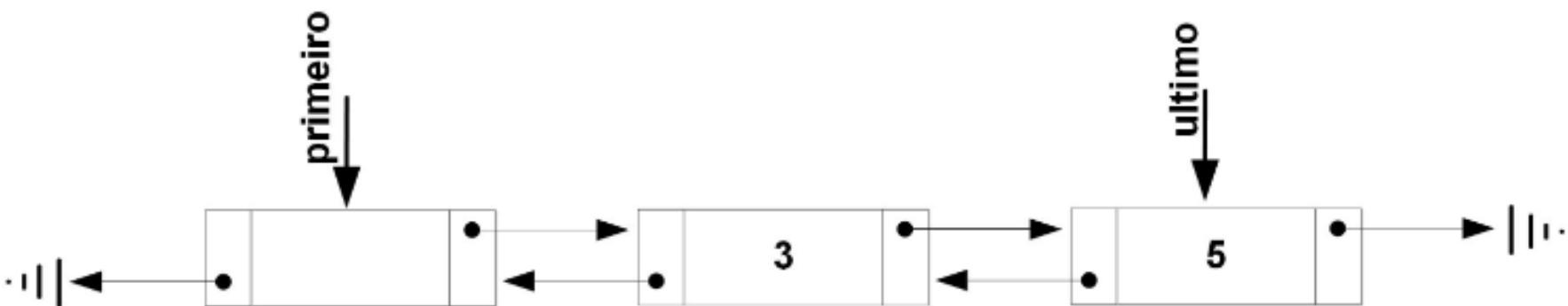
```
//LISTA DUPLA
public void inserirFim(int x) {
    ultimo.prox = new CelulaDupla(x);
    ultimo.prox.ant = ultimo;
    ultimo = ultimo.prox;
}
```

```
//LISTA SIMPLES
public void inserirFim(int x) {
    ultimo.prox = new Celula(x);
    ultimo = ultimo.prox;
}
```

Lista Dupla Inserir Fim

Supondo uma lista com os elementos 3 e 5, vamos inserir o 7 no fim.

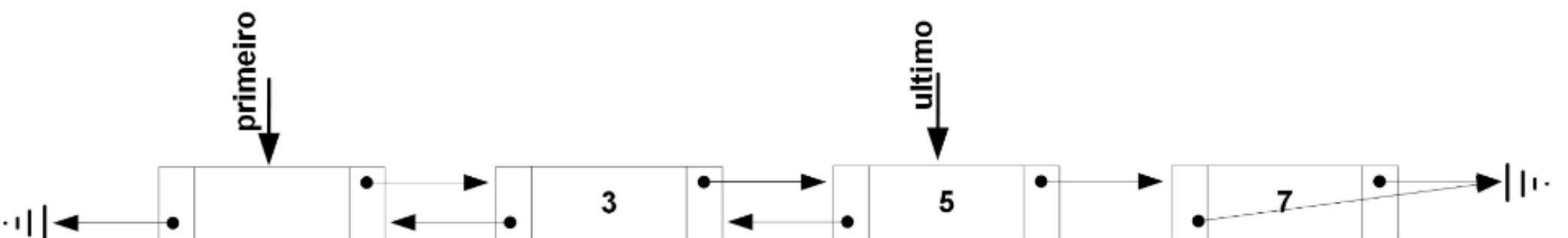
```
//LISTA DUPLA
public void inserirFim(int x) {
    ultimo.prox = new CelulaDupla(x);
    ultimo.prox.ant = ultimo;
    ultimo = ultimo.prox;
}
```



Lista Dupla Inserir Fim

Supondo uma lista com os elementos 3 e 5, vamos inserir o 7 no fim.

```
//LISTA DUPLA
public void inserirFim(int x) {
    ultimo.prox = new CelulaDupla(x);
    ultimo.prox.ant = ultimo;
    ultimo = ultimo.prox;
}
```

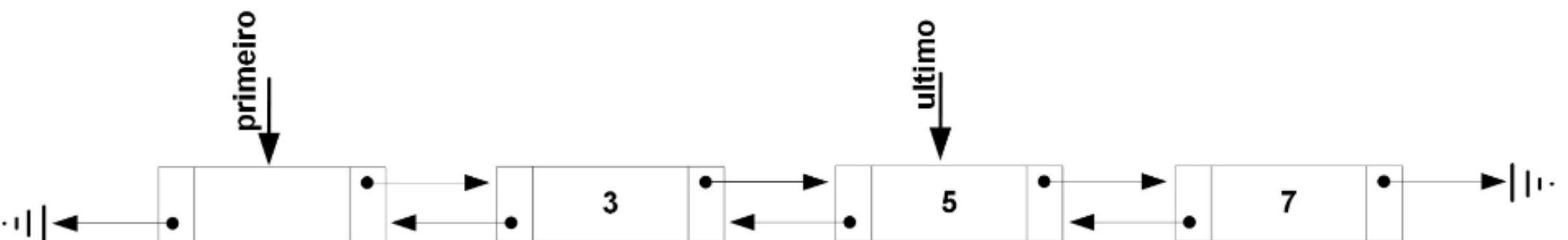


Lista Dupla Inserir Fim

Supondo uma lista com os elementos 3 e 5, vamos inserir o 7 no fim.

//LISTA DUPLA

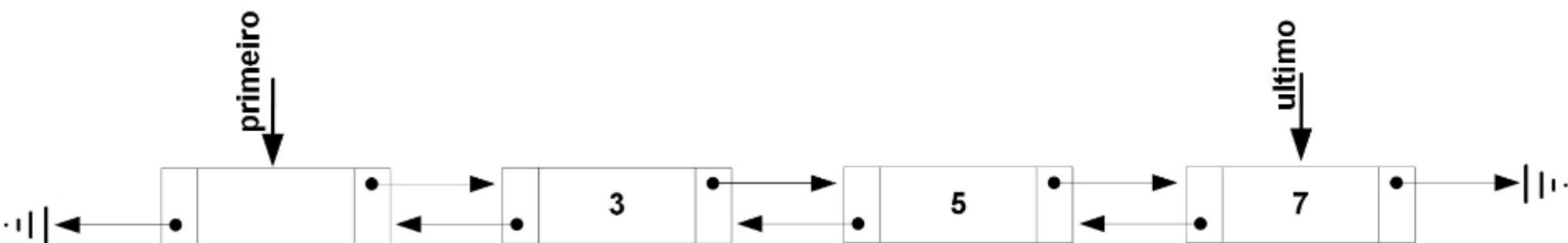
```
public void inserirFim(int x) {  
    ultimo.prox = new CelulaDupla(x);  
    ultimo.prox.ant = ultimo;  
    ultimo = ultimo.prox;  
}
```



Lista Dupla Inserir Fim

Supondo uma lista com os elementos 3 e 5, vamos inserir o 7 no fim.

```
//LISTA DUPLA
public void inserirFim(int x) {
    ultimo.prox = new CelulaDupla(x);
    ultimo.prox.ant = ultimo;
    ultimo = ultimo.prox;
}
```



Lista Dupla Remover Início

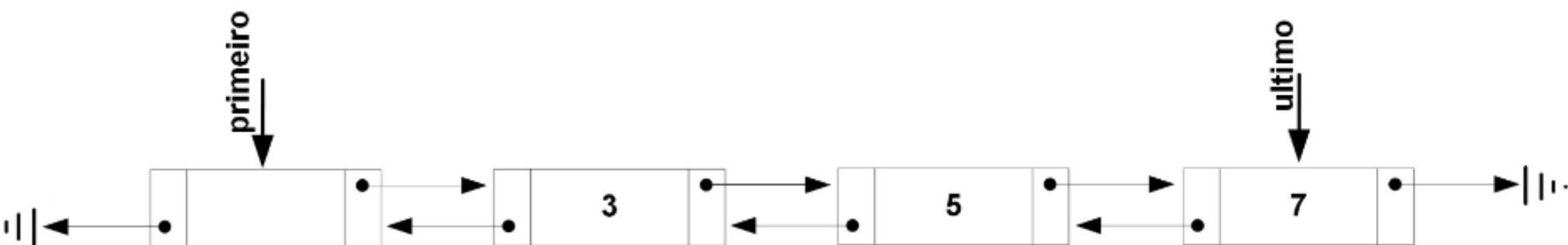
```
//LISTA DUPLA
public int removerInicio() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    CelulaDupla tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = primeiro.ant = null;
    tmp = null;
    return elemento;
}
```

```
//LISTA SIMPLES
public int removerInicio() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = null;
    tmp = null;
    return elemento;
}
```

Lista Dupla Remover Início

Supondo uma lista com os elementos 3, 5 e 7, execute o remove no início.

```
//LISTA DUPLA
public int removerInicio() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    CelulaDupla tmp = primeiro;
    primeiro = primeiro.prox;
    int elemento = primeiro.elemento;
    tmp.prox = primeiro.ant = null;
    tmp = null;
    return elemento;
}
```



Lista Dupla Remover Fim

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... } // This line is highlighted in purple  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla Remover Fim

```
//LISTA DUPLA
public int removerFim() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");

    int elemento = ultimo.elemento;
    ultimo = ultimo.ant;
    ultimo.prox.ant = null;
    ultimo.prox = null;
    return elemento;
}
```

```
//LISTA SIMPLES
public int removerFim() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");
    Celula i;
    for(i = primeiro; i.prox != ultimo; i = i.prox);
    int elemento = ultimo.elemento;
    ultimo = i;

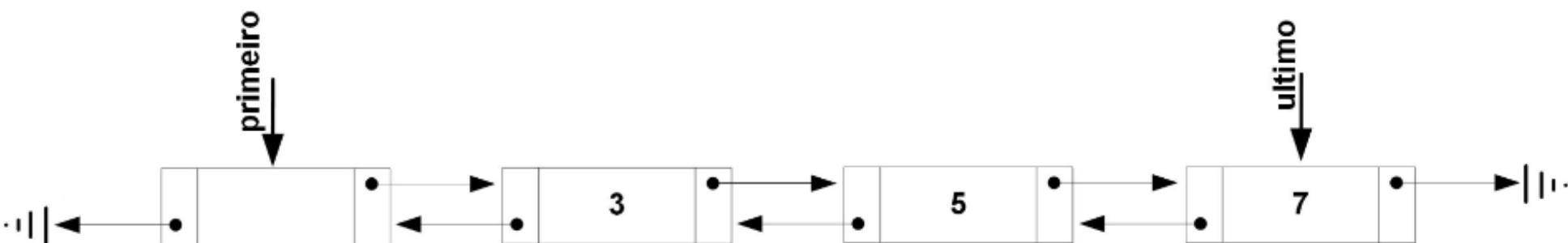
    i = ultimo.prox = null;
    return elemento;
}
```

Lista Dupla Remover Fim

Supondo uma lista com os elementos 3, 5 e 7, execute o remover no início.

```
//LISTA DUPLA
public int removerFim() throws Exception {
    if (primeiro == ultimo)
        throw new Exception("Erro!");

    int elemento = ultimo.elemento;
    ultimo = ultimo.ant;
    ultimo.prox.ant = null;
    ultimo.prox = null;
    return elemento;
}
```



Lista Dupla

Inserir

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla Inserir

```
//LISTA DUPLA
public void inserir(int x, int pos) throws
Exception {
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){
        throw new Exception("Erro!");
    } else if (pos == 0){ inserirInicio(x);
    } else if (pos == tamanho){ inserirFim(x);
    } else {
        CelulaDupla i = primeiro;
        for (int j = 0; j < pos; j++, i = i.prox);

        CelulaDupla tmp = new CelulaDupla(x);
        tmp.ant = i;
        tmp.prox = i.prox;
        tmp.ant.prox = tmp.prox.ant = tmp;
        tmp = i = null;
    }
}
```

```
//LISTA SIMPLES
public void inserir(int x, int pos) throws
Exception {
    int tamanho = tamanho();
    if (pos < 0 || pos > tamanho){
        throw new Exception("Erro!");
    } else if (pos == 0){ inserirInicio(x);
    } else if (pos == tamanho){ inserirFim(x);
    } else {
        Celula i = primeiro;
        for (int j = 0; j < pos; j++, i = i.prox);

        Celula tmp = new Celula(x);
        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```

Lista Dupla Remover

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```

Lista Dupla Remover

//LISTA DUPLA

```
public int remover(int pos) throws Exception {
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo){
        throw new Exception("Erro!");
    } else if (pos < 0 || pos >= tamanho){
        throw new Exception("Erro!");
    } else if (pos == 0){
        elemento = removerInicio();
    } else if (pos == tamanho - 1){
        elemento = removerFim();
    } else {
        CelulaDupla i = primeiro.prox;
        for (int j = 0; j < pos; j++, i = i.prox);
        i.ant.prox = i.prox;
        i.prox.ant = i.ant;
        elemento = i.elemento;
        i.prox = i.ant = null;
        i = null;
    }
    return elemento;
}
```

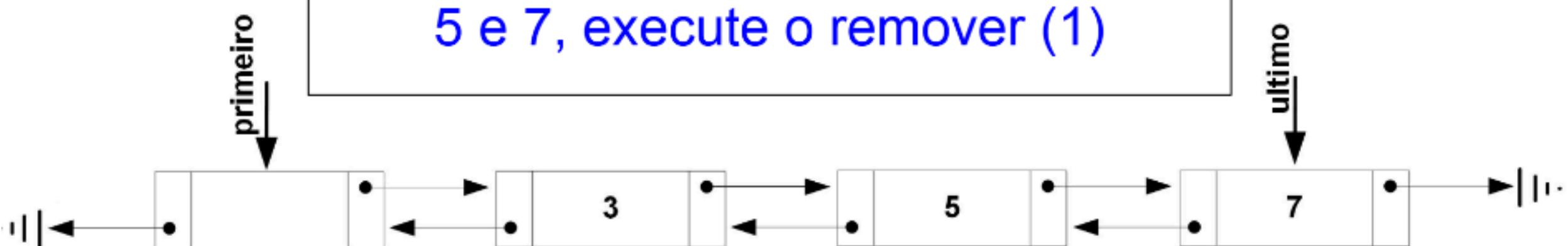
//LISTA SIMPLES

```
public int remover(int pos) throws Exception {
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo){
        throw new Exception("Erro!");
    } else if (pos < 0 || pos >= tamanho){
        throw new Exception("Erro!");
    } else if (pos == 0){
        elemento = removerInicio();
    } else if (pos == tamanho - 1){
        elemento = removerFim();
    } else {
        Celula i = primeiro;
        for (int j = 0; j < pos; j++, i = i.prox);
        Celula tmp = i.prox;
        elemento = tmp.elemento;
        i.prox = tmp.prox;
        tmp.prox = null;
        i = tmp = null;
    }
    return elemento;
}
```

Lista Dupla Remover

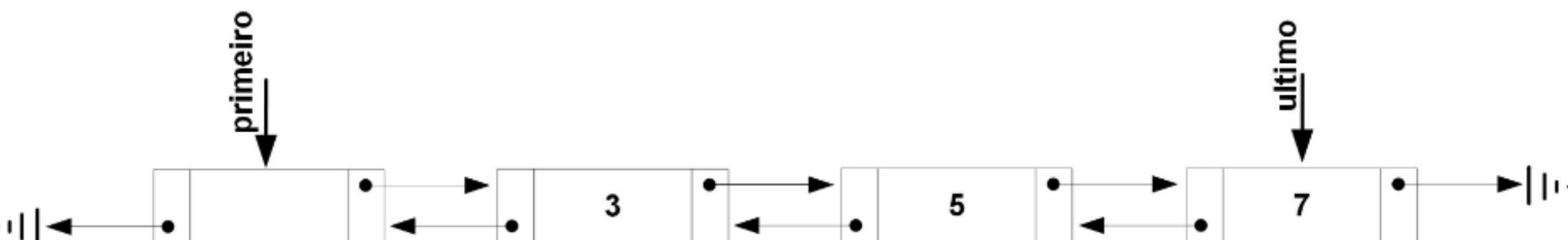
```
//LISTA DUPLA
public int remover(int pos) throws Exception {
    int elemento, tamanho = tamanho();
    if (primeiro == ultimo){
        throw new Exception("Erro!");
    } else if (pos < 0 || pos >= tamanho){
        throw new Exception("Erro!");
    } else if (pos == 0){
        elemento = removerInicio();
    } else if (pos == tamanho - 1){
        elemento = removerFim();
    } else {
        CelulaDupla i = primeiro;
        i.ant.prox = i.prox;
        elemento = i.elemento;
    }
    return elemento;
}
```

Exercício: Supondo a lista com o 3, 5 e 7, execute o remover (1)



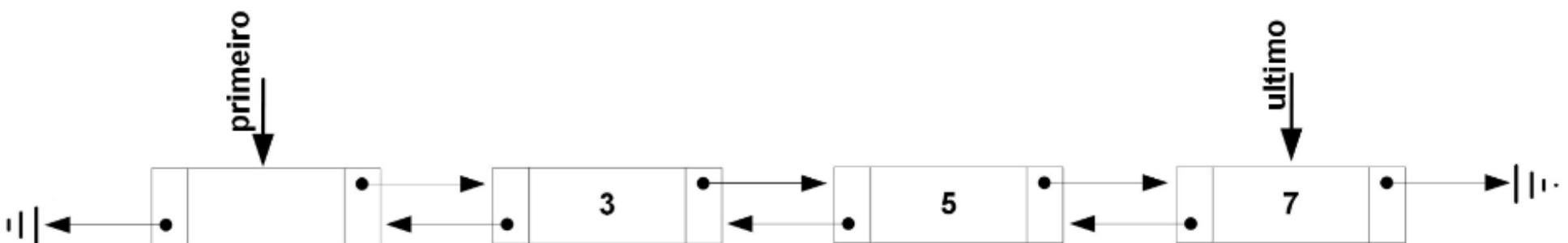
Lista Dupla Mostrar

```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ... }  
    public int removerFim() { ... }  
    public void inserir(int x, int pos) { ... }  
    public int remover(int pos) { ... }  
    public void mostrar() { ... }  
}
```



Lista Dupla Mostrar

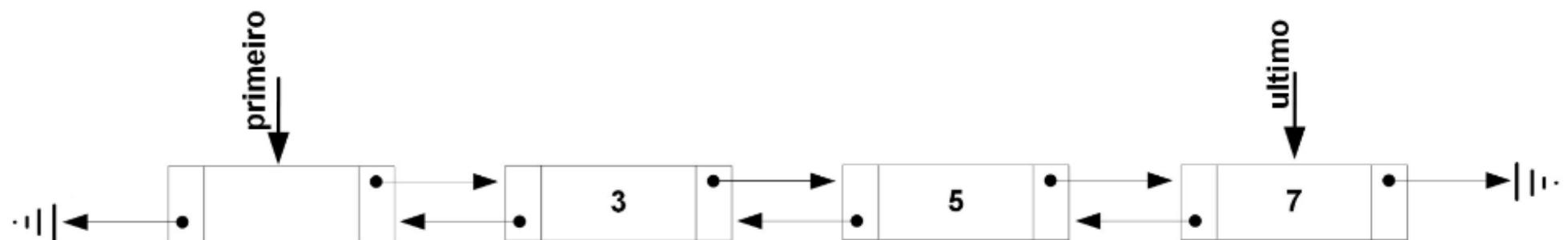
Mostrar começando pelo **íncio** e indo até o **fim**.



Lista Dupla Mostrar

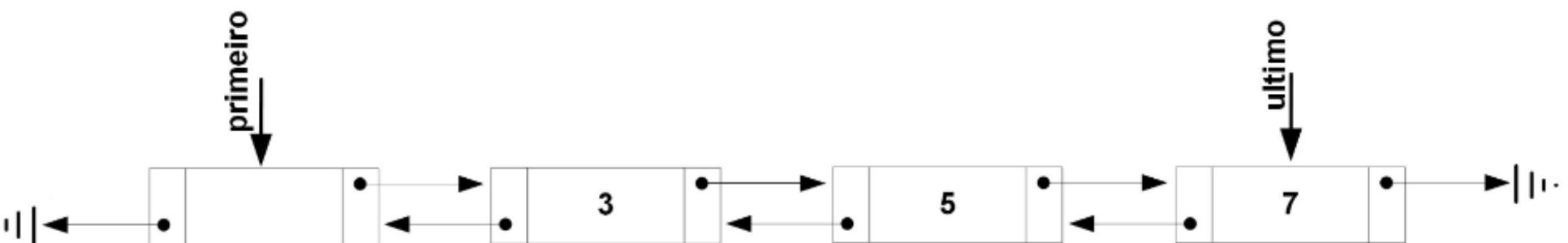
Mostrar começando pelo **íncio** e indo até o **fim**.

```
public void mostrar() {  
    for (CelulaDupla i = primeiro.prox; i != null; i = i.prox) {  
        System.out.print(i.elemento + " ");  
    }  
}
```



Lista Dupla Mostrar

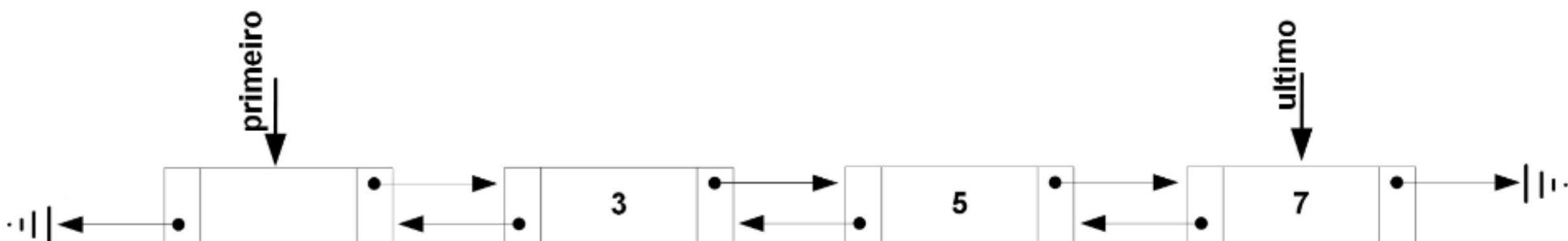
Mostrar do começando
pelo **fim** e indo até o **início**.



Lista Dupla Mostrar

Mostrar do começando
pelo **fim** e indo até o **início**.

```
public void mostrar() {  
    for (CelulaDupla i = ultimo; i != null; i = i.ant) {  
        System.out.print(i.elemento + " ");  
    }  
}
```



Lista Dupla Mostrar

[Tabnine](#) | [Edit](#) | [Test](#) | [Explain](#) | [Document](#)

```
public void mostrarProx() {
    for (CelulaDupla i = primeiro.prox; i != null; i = i.prox) {
        System.out.print(i.elemento + " ");
    }
}
```

[Tabnine](#) | [Edit](#) | [Test](#) | [Explain](#) | [Document](#)

```
public void mostrarAnt() {
    for (CelulaDupla i = ultimo; i != null; i = i.ant) {
        System.out.print(i.elemento + " ");
    }
}
```

Matriz Flexível

```
class Celula {
    public int elemento;
    public Celula superior;
    public Celula inferior;
    public Celula esquerda;
    public Celula direita;

    public Celula() {
        this(elemento:0);
    }

    public Celula(int elemento) {
        this.elemento = elemento;
        this.superior = null;
        this.inferior = null;
        this.esquerda = null;
        this.direita = null;
    }
}

class Matriz {
    private Celula inicio;
    private int linhas;
    private int colunas;

    // Construtor no próximo slide
}
```

Matriz Flexível

Construtor da Matriz

```
public Matriz(int linhas, int colunas) {
    if (linhas <= 0 || colunas <= 0) {
        throw new IllegalArgumentException("Dimensões da matriz devem ser positivas.");
    }

    this.linhas = linhas;
    this.colunas = colunas;

    // Cria a primeira célula (canto superior esquerdo)
    this.inicio = new Celula();
    Celula celulaAtual = this.inicio;

    // Cria o restante da primeira linha
    for (int j = 1; j < colunas; j++) {
        celulaAtual.direita = new Celula();
        celulaAtual.direita.esquerda = celulaAtual;
        celulaAtual = celulaAtual.direita;
    }

    Celula linhaAnterior = this.inicio;

    // Cria as linhas subsequentes
    for (int i = 1; i < linhas; i++) {
        // Cria a primeira célula da nova linha e a conecta com a linha de cima
        linhaAnterior.inferior = new Celula();
        linhaAnterior.inferior.superior = linhaAnterior;
        celulaAtual = linhaAnterior.inferior;
        Celula celulaDeCima = linhaAnterior;

        // Cria o restante da nova linha, conectando horizontalmente e verticalmente
        for (int j = 1; j < colunas; j++) {
            celulaDeCima = celulaDeCima.direita;

            celulaAtual.direita = new Celula();
            celulaAtual.direita.esquerda = celulaAtual;
            celulaAtual.direita.superior = celulaDeCima;
            celulaDeCima.inferior = celulaAtual.direita;

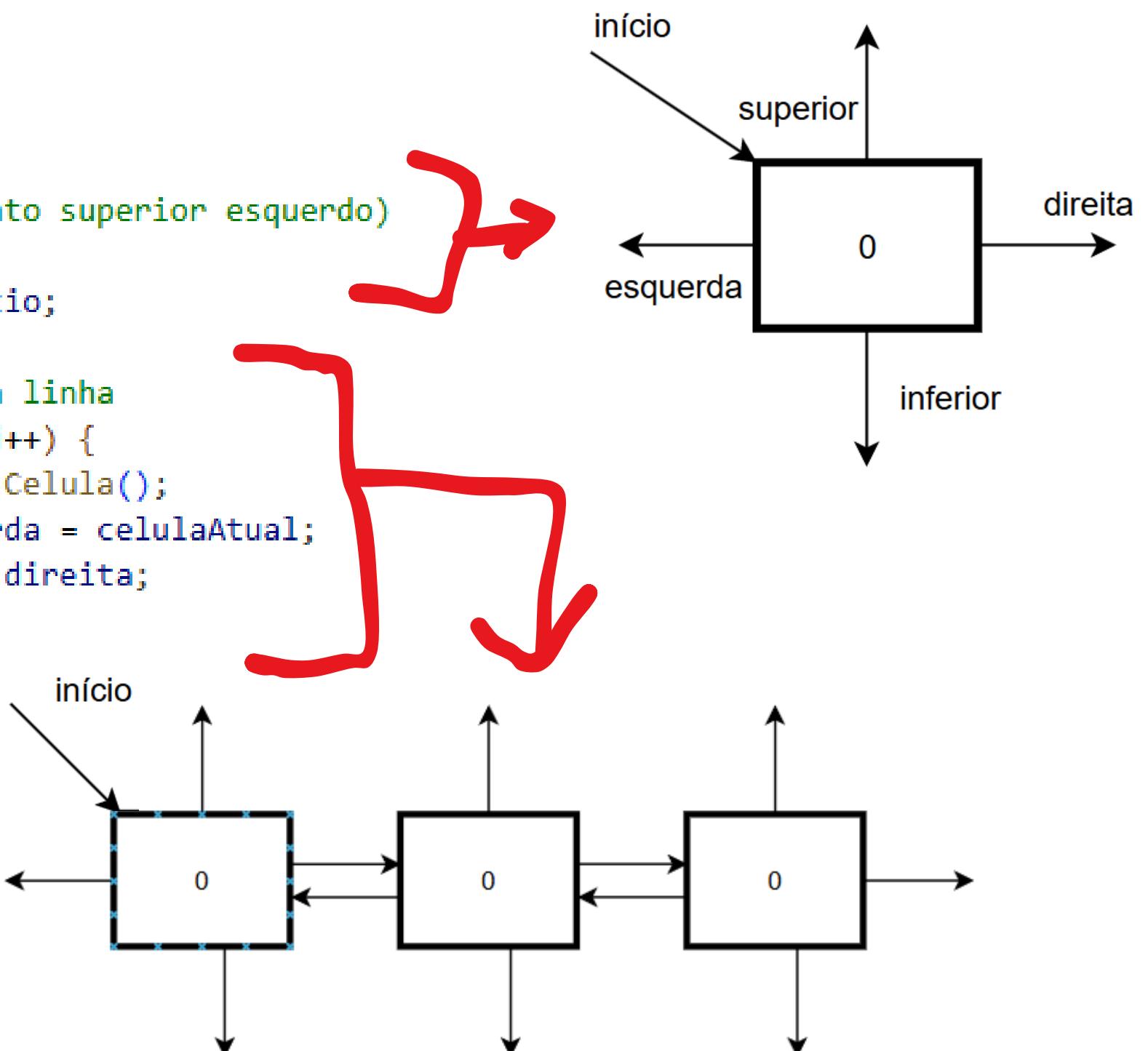
            celulaAtual = celulaAtual.direita;
        }
        linhaAnterior = linhaAnterior.inferior;
    }
}
```

Matriz Flexível

Vamos por partes!!

(Exemplo de Matriz 3x3)

```
public Matriz(int linhas, int colunas) {  
    if (linhas <= 0 || colunas <= 0) {  
        throw new IllegalArgumentException("Dimensões da matriz devem ser positivas.");  
    }  
  
    this.linhas = linhas;  
    this.colunas = colunas;  
  
    // Cria a primeira célula (canto superior esquerdo)  
    this.inicio = new Celula();  
    Celula celulaAtual = this.inicio;  
  
    // Cria o restante da primeira linha  
    for (int j = 1; j < colunas; j++) {  
        celulaAtual.direita = new Celula();  
        celulaAtual.direita.esquerda = celulaAtual;  
        celulaAtual = celulaAtual.direita;  
    }  
}
```



Matriz Flexível

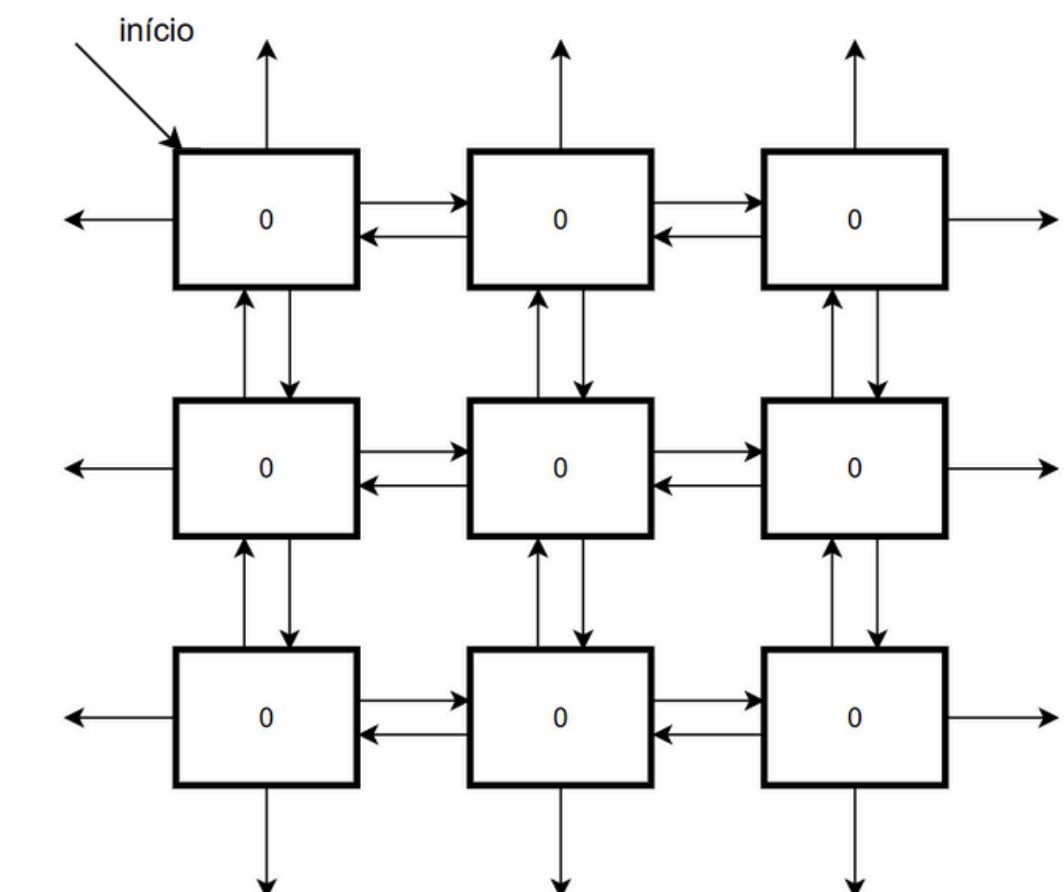
```
Celula linhaAnterior = this.inicio;

// Cria as linhas subsequentes
for (int i = 1; i < linhas; i++) {
    // Cria a primeira célula da nova linha e a conecta com a linha de cima
    linhaAnterior.inferior = new Celula();
    linhaAnterior.inferior.superior = linhaAnterior;
    celulaAtual = linhaAnterior.inferior;
    Celula celulaDeCima = linhaAnterior;

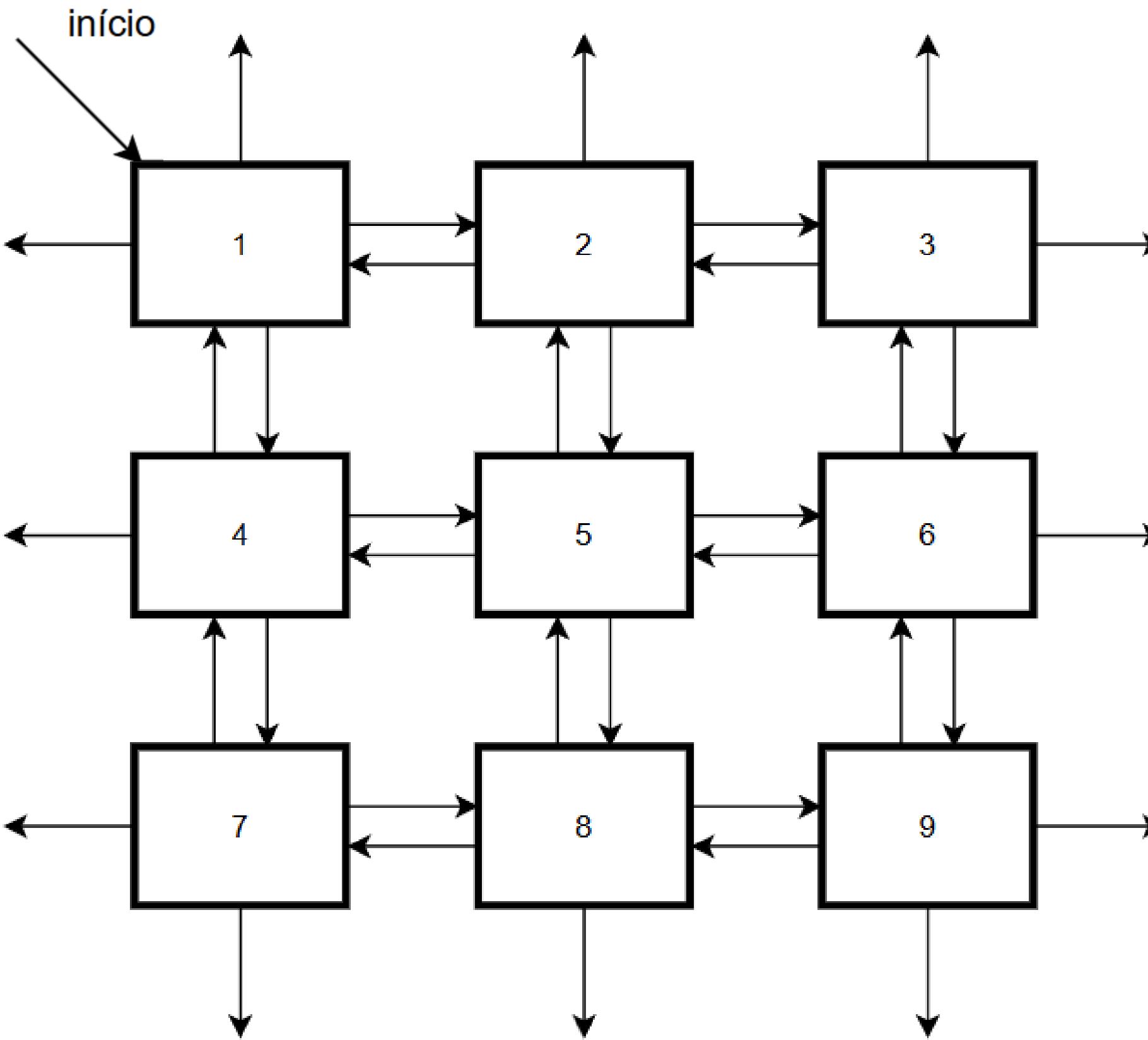
    // Cria o restante da nova linha, conectando horizontalmente e verticalmente
    for (int j = 1; j < colunas; j++) {
        celulaDeCima = celulaDeCima.direita;

        celulaAtual.direita = new Celula();
        celulaAtual.direita.esquerda = celulaAtual;
        celulaAtual.direita.superior = celulaDeCima;
        celulaDeCima.inferior = celulaAtual.direita;

        celulaAtual = celulaAtual.direita;
    }
    linhaAnterior = linhaAnterior.inferior;
}
```



Matriz Flexível



Matriz Flexível

```
public void inserir(int linha, int coluna, int elemento) {
    if (linha >= this.linhas || coluna >= this.colunas || linha < 0 || coluna < 0) {
        System.err.println("Erro: Posição (" + linha + ", " + coluna + ") é inválida!");
        return;
    }

    // Caminha até a linha desejada
    Celula celulaAlvo = this.inicio;
    for (int i = 0; i < linha; i++) {
        celulaAlvo = celulaAlvo.inferior;
    }

    // Caminha até a coluna desejada
    for (int j = 0; j < coluna; j++) {
        celulaAlvo = celulaAlvo.direita;
    }

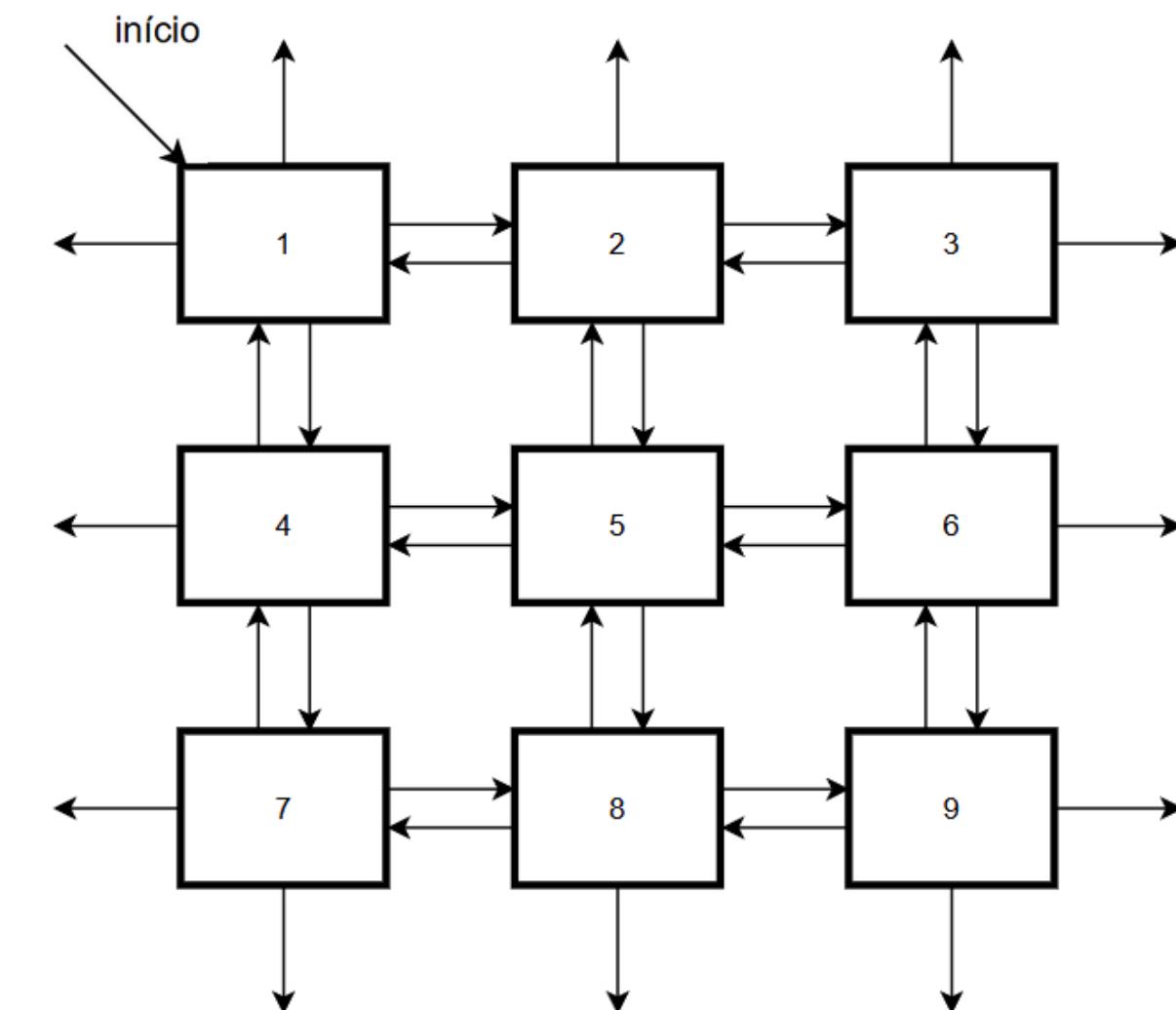
    celulaAlvo.elemento = elemento;
}
```

Matriz Flexível

Saída no terminal:

1	2	3
4	5	6
7	8	9

```
public void exibir() {  
    for (Celula i = this.inicio; i != null; i = i.inferior) {  
        for (Celula j = i; j != null; j = j.direita) {  
            System.out.print(j.elemento + "\t");  
        }  
        System.out.println();  
    }  
}
```



Grupo Monitoria

