

Unidade V:

Estruturas de Dados Básicas

com Alocação Flexível - Lista Dupla

Prof. Max do Val Machado

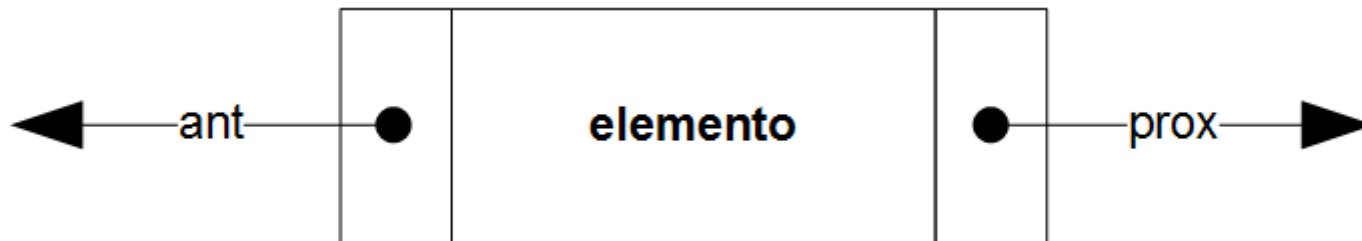


PUC Minas

Instituto de Ciências Exatas e Informática
Curso de Ciência da Computação

Classe Célula Dupla

```
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;  
    }  
}
```



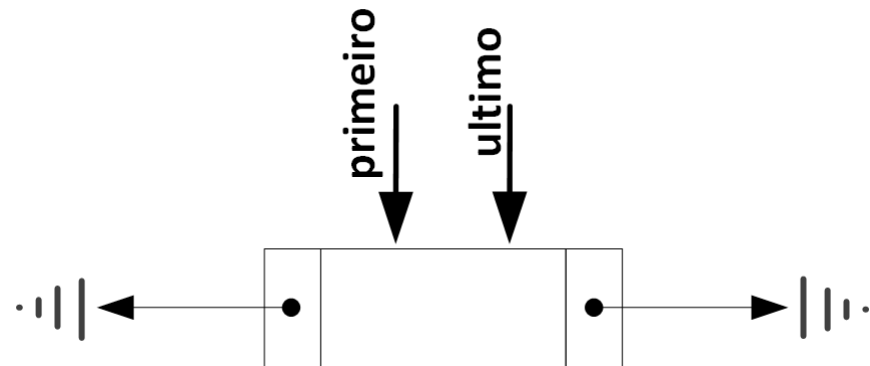
Lista Dupla Flexível

```
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() { ... }  
}
```

Similar a Lista Simples,
contudo, considerando o
ponteiro ant

Lista Dupla Flexível

```
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() { ... }  
}
```



Inserir no 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() { ... }  
}
```

Inserir 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 SIMPLES

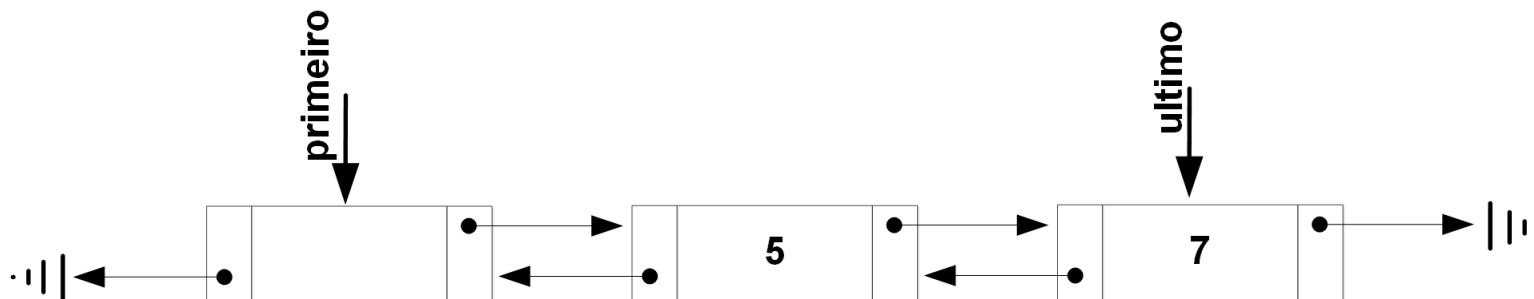
```
public void inserirInicio(int x) {  
    Celula tmp = new Celula(x);  
  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    }  
    tmp = null;  
}
```

Inserir 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;  
}
```

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



Inserir no 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;

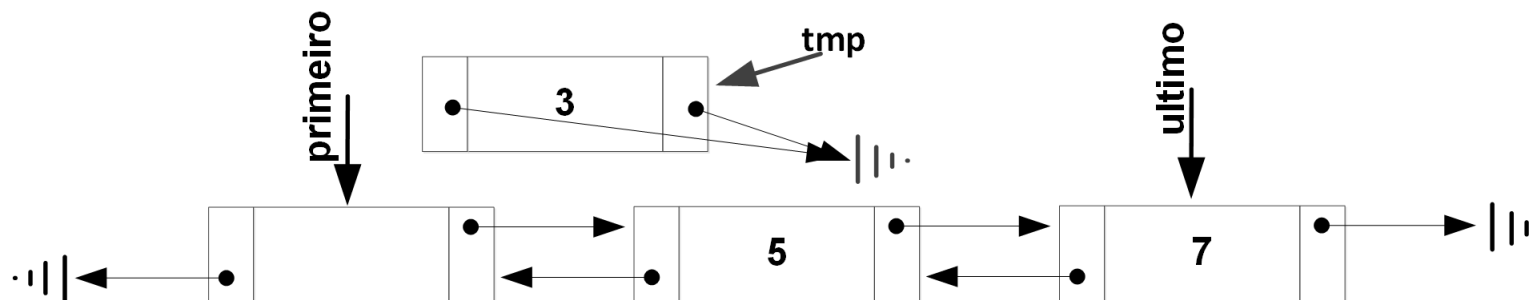
} **else** {

 tmp.prox.ant = tmp;

}

tmp = **null**;

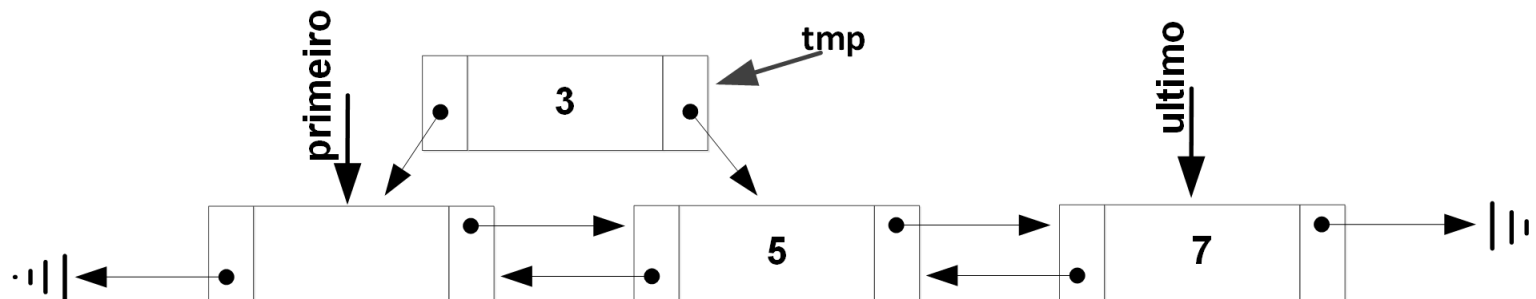
}



Inserir no Início

//Inserindo o 3 no início

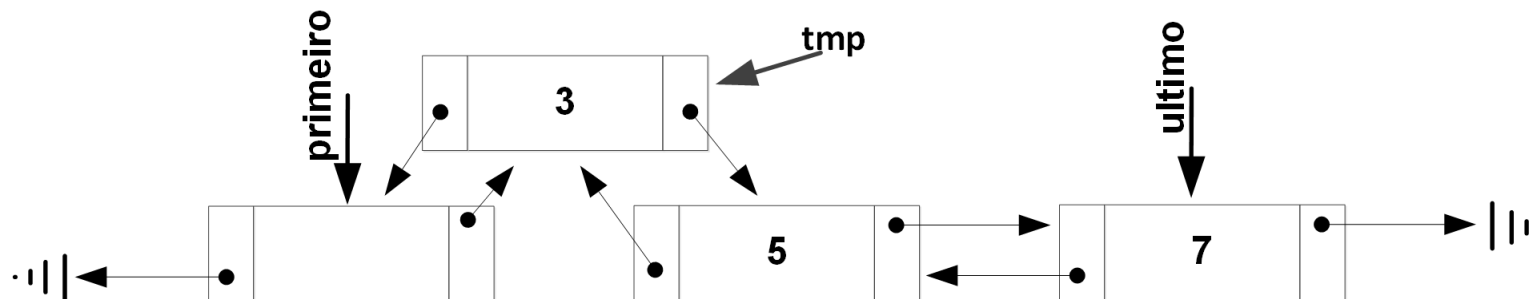
```
public void inserirInicio(int x) {  
    CelulaDupla tmp = new CelulaDupla (x);  
    tmp.ant = primeiro;  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    } else {  
        tmp.prox.ant = tmp;  
    }  
    tmp = null;  
}
```



Inserir no Início

//Inserindo o 3 no início

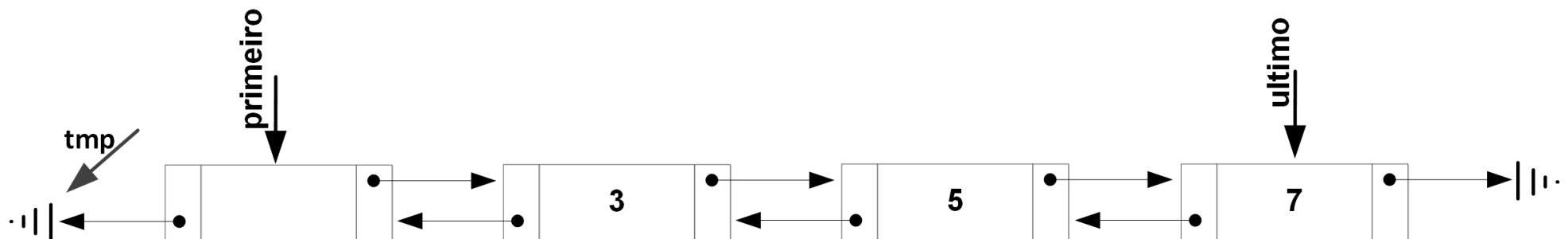
```
public void inserirInicio(int x) {  
    CelulaDupla tmp = new CelulaDupla (x);  
    tmp.ant = primeiro;  
    tmp.prox = primeiro.prox;  
    primeiro.prox = tmp;  
    if (primeiro == ultimo) {  
        ultimo = tmp;  
    } else {  
        tmp.prox.ant = tmp;  
    }  
    tmp = null;  
}
```



Inserir no 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;  
}
```



Inserir no 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() { ... }  
}
```

Inserir no Fim

//LISTA DUPLA

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

//LISTA SIMPLES

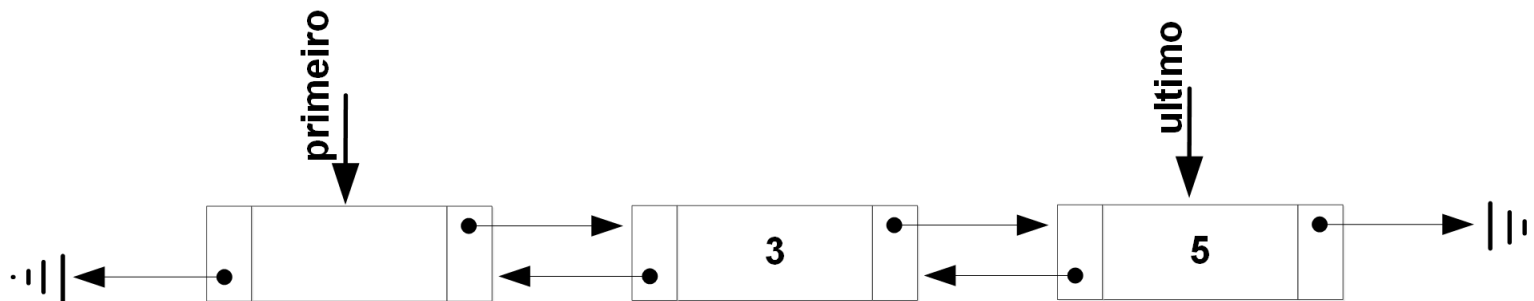
```
public void inserirFim(int x) {  
    ultimo.prox = new Celula(x);  
  
    ultimo = ultimo.prox;  
}
```

Inserir no Fim

//LISTA DUPLA

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

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



Inserir no Fim

//LISTA DUPLA

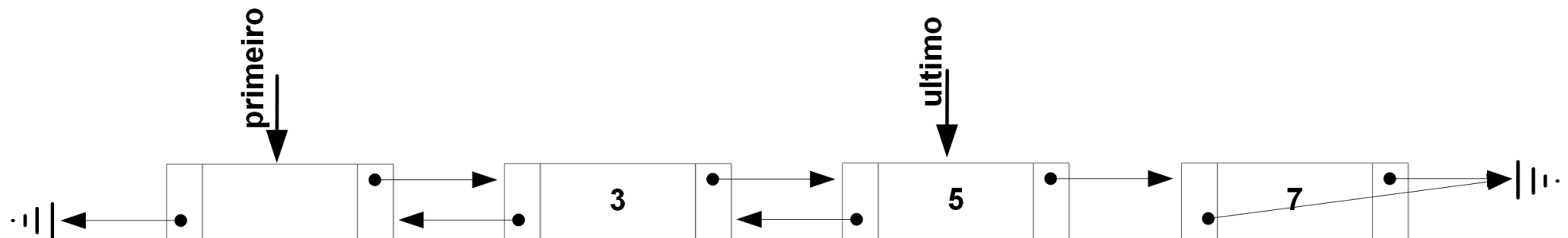
public void inserirFim(**int** x) {

ultimo.prox = **new** CelulaDupla(x);

ultimo.prox.ant = ultimo;

ultimo = ultimo.prox;

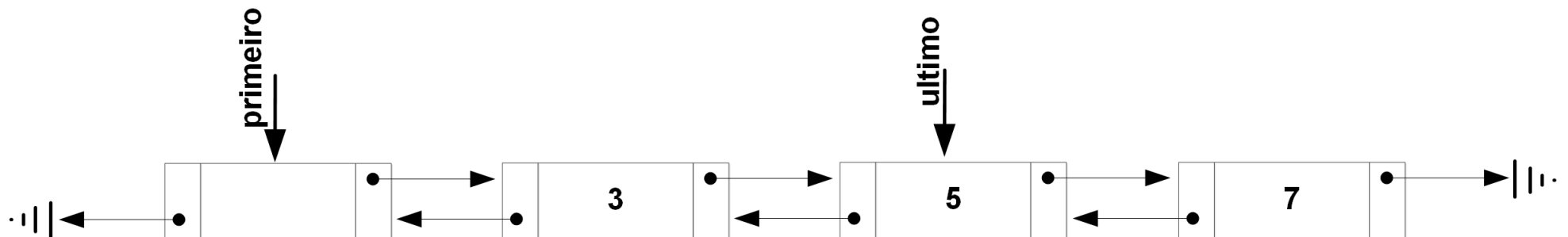
}



Inserir no Fim

//LISTA DUPLA

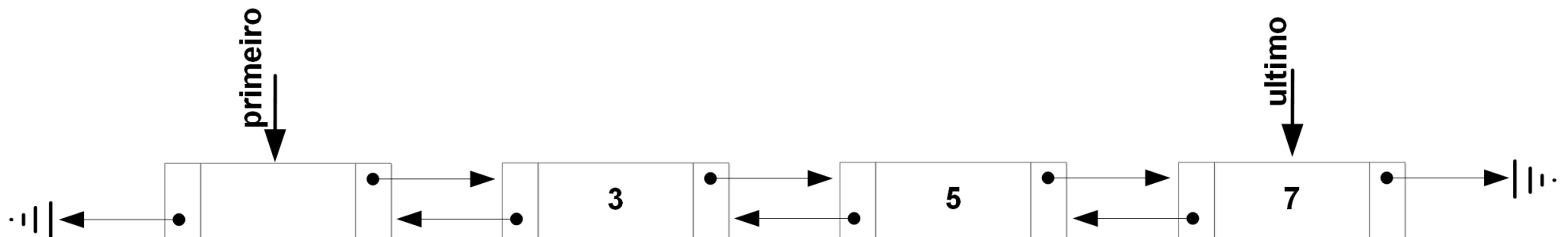
```
public void inserirFim(int x) {  
    ultimo.prox = new CelulaDupla(x);  
    ultimo.prox.ant = ultimo;  
    ultimo = ultimo.prox;  
}
```



Inserir no Fim

//LISTA DUPLA

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



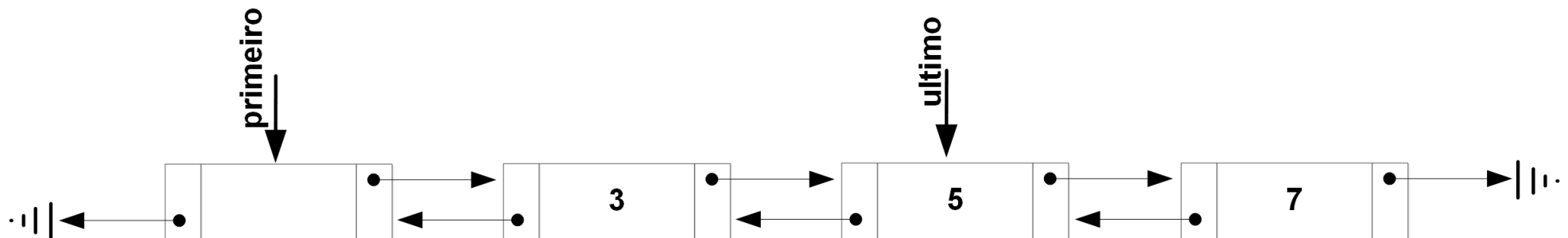
Exercício

//LISTA DUPLA

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

- A linha marcada pode ser substituída pelo código abaixo?

ultimo = ultimo.prox.ant.prox.ant.prox



Remover no 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() { ... }  
}
```

Remover 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 SIMPLES

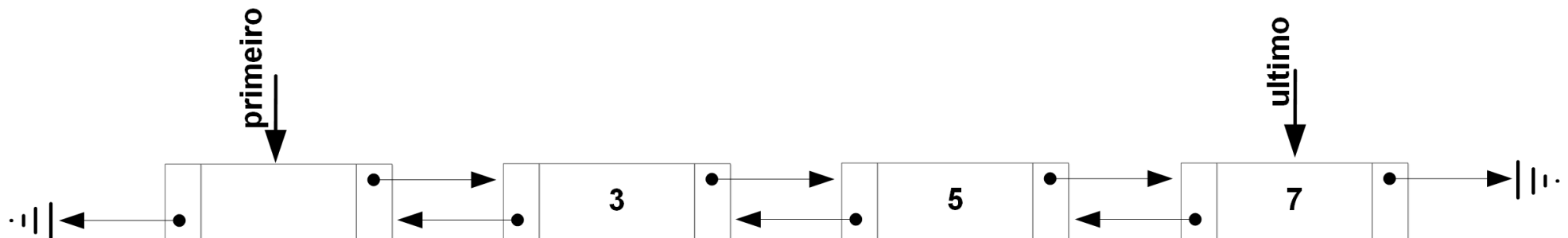
```
public int removerInicio() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
    Celula tmp = primeiro;  
    primeiro = primeiro.prox;  
    int elemento = primeiro.elemento;  
    tmp.prox = null;  
    tmp = null;  
    return elemento;  
}
```

Remover 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;  
}
```

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



Remover no 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() { ... }  
}
```

Remover no 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

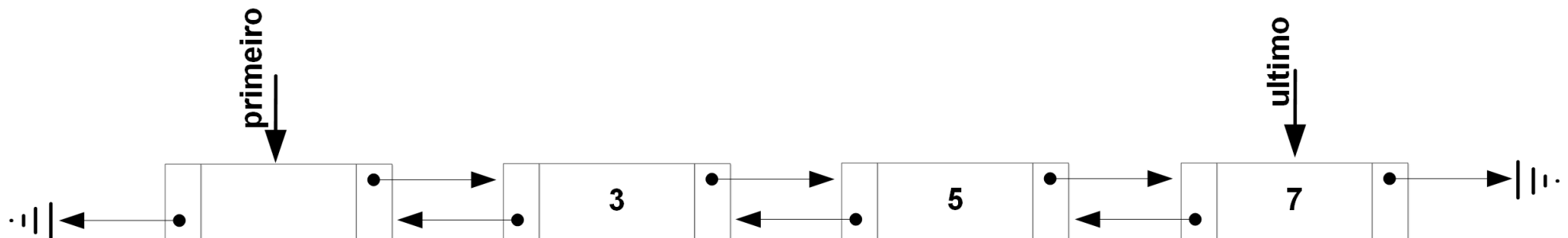
```
public int removerFim() throws Exception {  
    if (primeiro == ultimo)  
        throw new Exception("Erro!");  
  
    Celula i;  
    for(i = primeiro; i.prox != ultimo; i = i.prox);  
    int elemento = ultimo.elemento;  
    ultimo = i;  
  
    i = ultimo.prox = null;  
    return elemento;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

Remover no 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;  
}
```

Exercício: Supondo uma lista com os elementos 3, 5 e 7, execute o remover no 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() { ... }  
}
```

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);

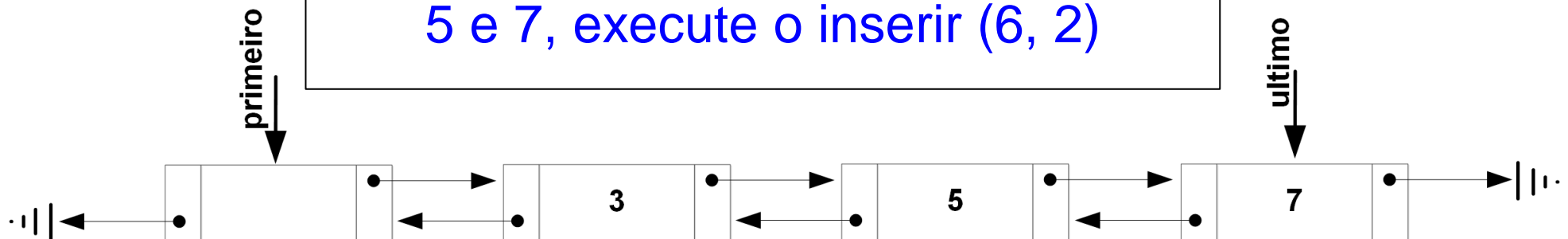
        Celula tmp = new Celula(x);

        tmp.prox = i.prox;
        i.prox = tmp;
        tmp = i = null;
    }
}
```

//LISTA DUPLA

```
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;  
    }  
}
```

Exercício: Supondo a lista com o 3, 5 e 7, execute o inserir (6, 2)



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() { ... }  
}
```

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;
}
```

Remover

//LISTA DUPLA

```
public int remover(int pos) throws Exception {  
    int elemento, tamanho = tamanho();  
    if (primeiro == ultimo){  
    } else if (pos < 0 || pos >= tamanho){  
    } else if (pos == 0){  
    } else if (pos == tamanho - 1){  
    } else {  
        CelulaDupla i = primeiro;  
        i.ant.prox = i.prox;  
        elemento = i.elemento;  
    }  
    return elemento;  
}
```

```
throw new Exception("Erro!");
```

```
throw new Exception("Erro!");
```

```
elemento = removerInicio();
```

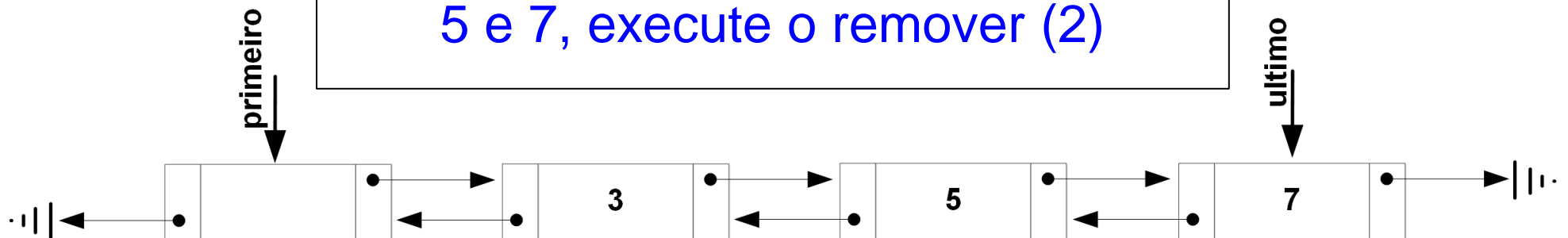
```
elemento = removerFim();
```

```
for (int j = 0; j < pos; j++, i = i.prox);
```

```
i.prox.ant = i.ant;
```

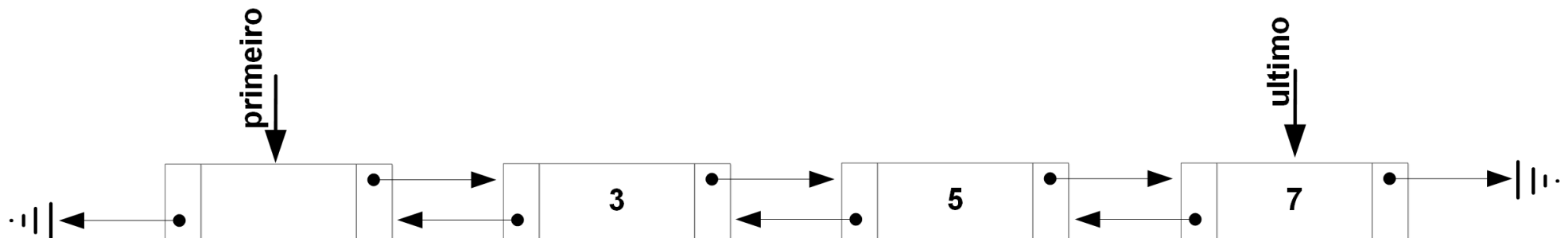
```
i = i.prox = i.ant = null;
```

Exercício: Supondo a lista com o 3, 5 e 7, execute o remover (2)



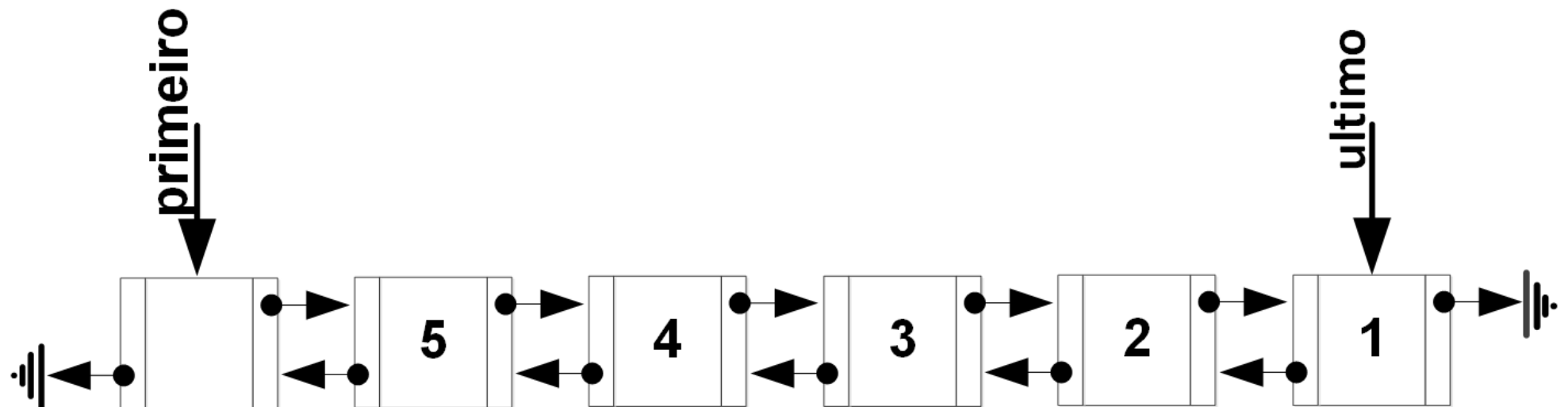
```
class ListaDupla {  
    private CelulaDupla primeiro, ultimo;  
    public ListaDupla () {  
        primeiro = new CelulaDupla();  
        ultimo = primeiro;  
    }  
    public void inserirInicio(int x) { ... }  
    public void inserirFim(int x) { ... }  
    public int removerInicio() { ...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

Exercício: Implemente
o mostrar e o execute
para uma lista com
os elementos 3, 5 e 7



Exercício

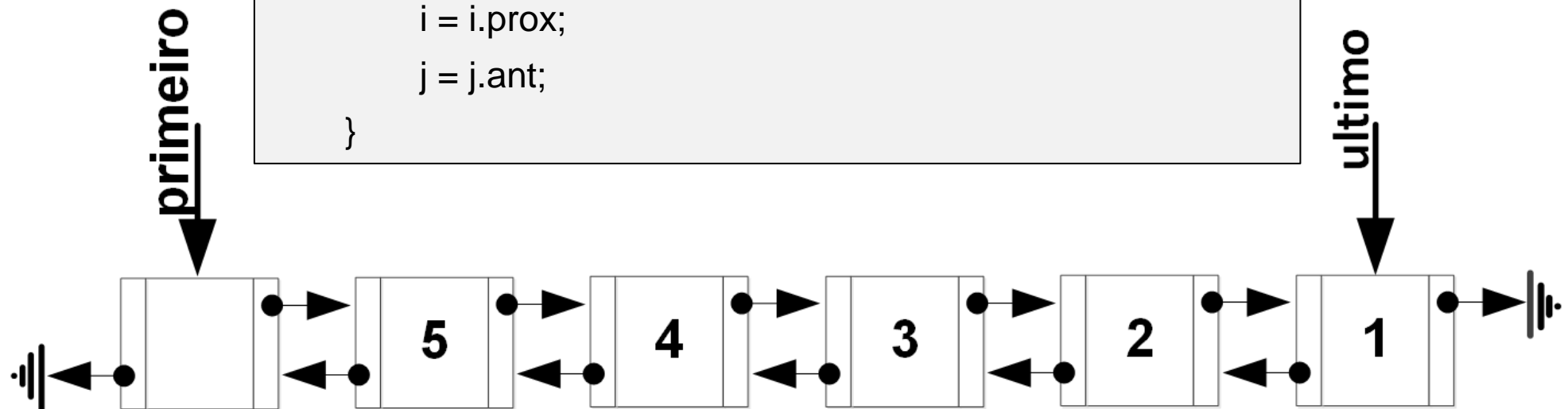
- Exercício: Faça um método que inverta a ordem dos elementos da lista dupla. No exemplo abaixo, após a inversão, os elementos ficarão na ordem crescente



Exercício

● Exercício: Faça um método que inverta a ordem dos elementos da lista dupla. No exemplo abaixo, após a inversão, os elementos ficarão na ordem crescente

```
void inverte(){  
    Celula i = primeiro.prox;    Celula j = ultimo;  
    while (i != j && j.prox != i){  
        int tmp = i.elemento;  
        i.elemento = j.elemento;  
        j.elemento = tmp;  
        i = i.prox;  
        j = j.ant;  
    }  
}
```

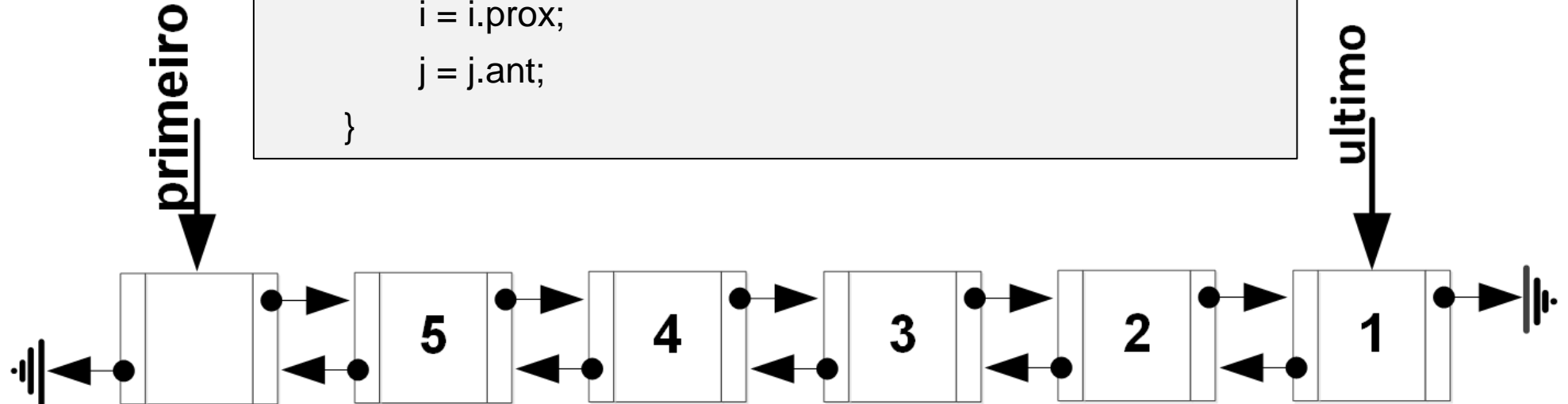


Exercício

- Exercício: Faça um método que inverta a ordem dos elementos da lista dupla. No exemplo abaixo, após a inversão, os elementos ficarão na ordem crescente

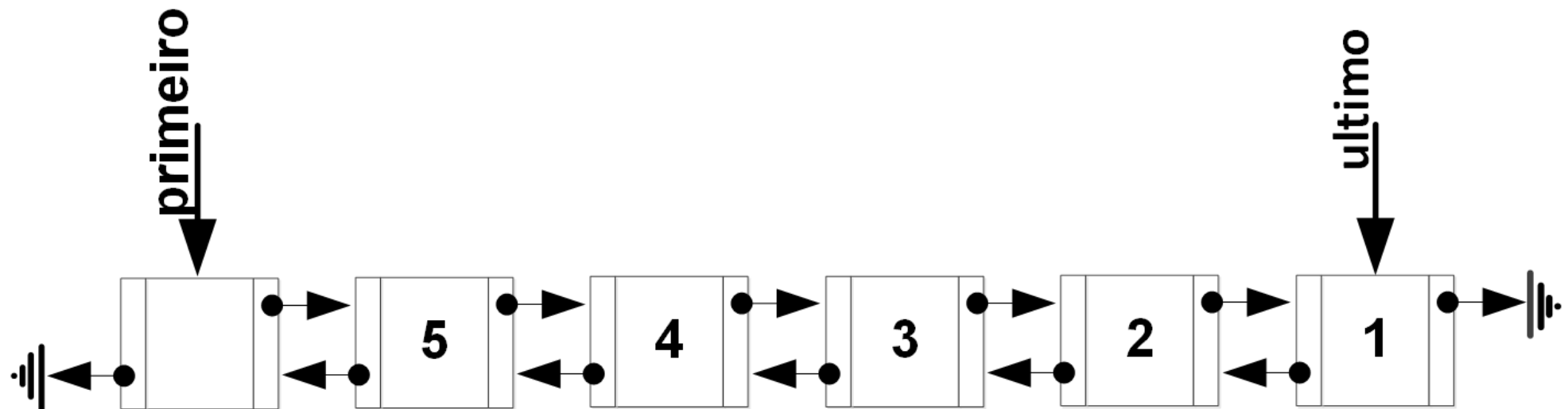
```
void inverte(){  
    Celula i = primeiro.prox;    Celula j = ultimo;  
    while (i != j && j.prox != i){  
        int tmp = i.elemento;  
        i.elemento = j.elemento;  
        j.elemento = tmp;  
        i = i.prox;  
        j = j.ant;  
    }  
}
```

Veja que a condição $i \neq j$ é para uma lista com número par de elementos e $j.prox \neq i$, para as com número ímpar de elementos



Exercício

- Exercício: Faça um método que inverta a ordem dos elementos da lista **simples**. No exemplo abaixo, após a inversão, os elementos ficarão na ordem crescente



Exercício

- Exercício: Faça um método que inverta a ordem dos elementos da lista **simples**. No exemplo abaixo, após a inversão, os elementos ficarão na ordem crescente

```
void inverte(){  
    Celula i = primeiro.prox;    Celula j = ultimo;  
    Celula k;  
    while (i != j && j.prox != i){  
        int tmp = i.elemento;  
        i.elemento = j.elemento;  
        j.elemento = tmp;  
        i = i.prox;  
        for (k = primeiro; k.prox != j; k = k.prox); j = k;  
    }
```

