Estrutura de Dados

AULA 4
Listas Ligadas

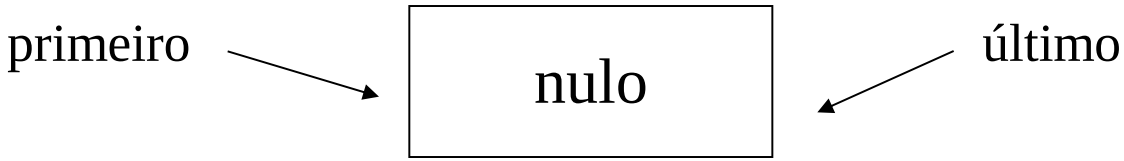
Listas Ligadas

- A lista Ligada é uma estrutura de dados utilizada na programação de computadores que se dispõe em forma de lista linear, ou seja, um elemento está seguido do outro elemento e assim por diante para todos os elementos da lista ligada.

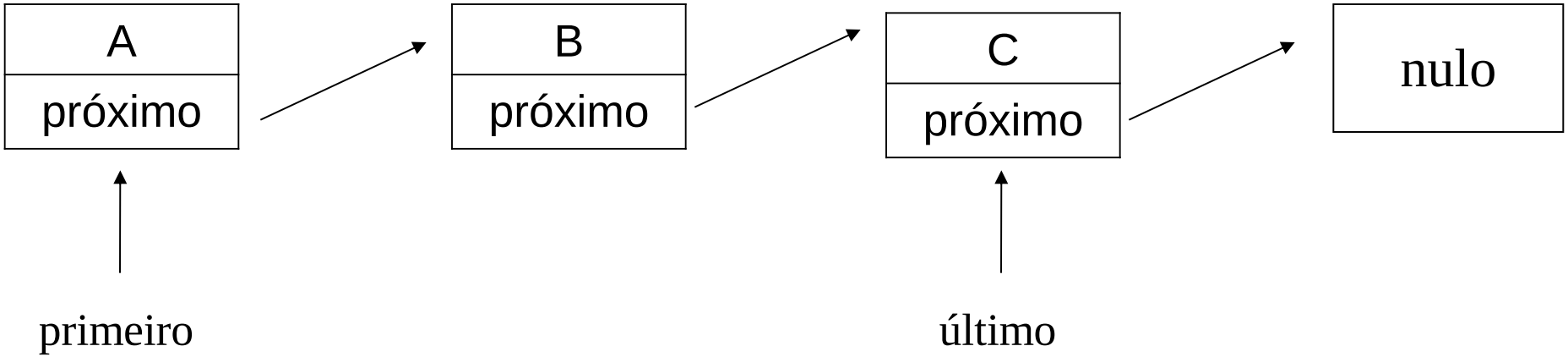
- Os elementos são dispostos de uma forma que cada elemento tenha uma ligação com o próximo elemento da lista ligada.

- Assim, os elementos não estão armazenados seqüencialmente de forma física, na ordem dos endereços que os elementos ocupam na memória. Com a lista ligada, os elementos estão armazenados de forma lógica e é a seqüência das ligações que determina a ordem dos elementos.

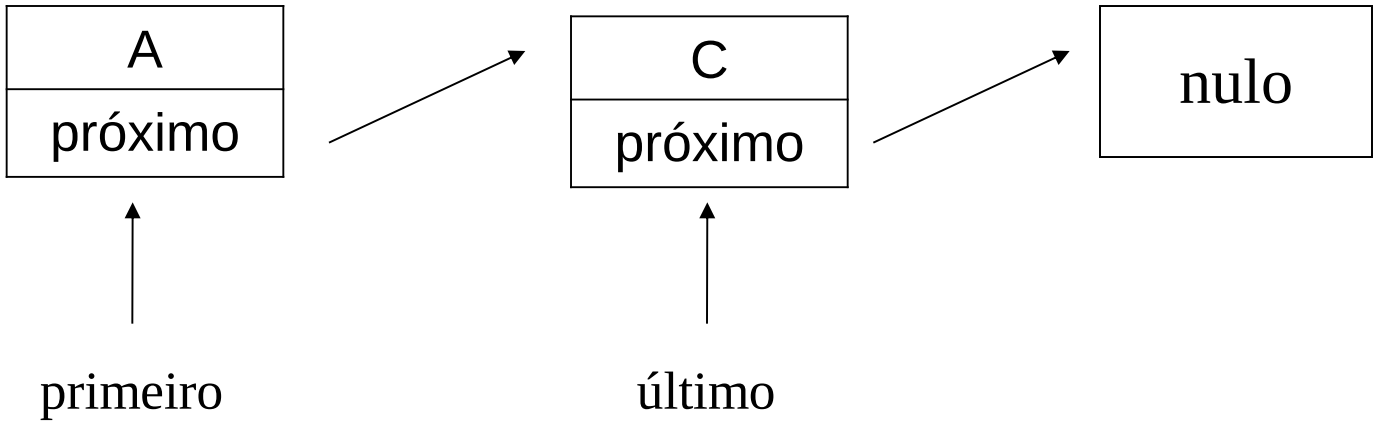
- Inicialmente a lista ligada está vazia, ou seja, não existem elementos na lista ligada. Portanto, o primeiro e o último elemento desta lista ligada são nulos.



- Vamos agora, inserir a letra A na lista ligada, em seguida a letra B e, depois, a letra C.



- Se quisermos remover um elemento da lista ligada, podemos remover qualquer elemento que esteja na lista ligada. Vamos, por exemplo, remover a letra B, observando a figura abaixo:



Operações de Lista Ligada

- ✓ inserir no início – insere um elemento no início da lista
- ✓ inserir no final – insere um elemento no final da lista
- ✓ inserir no meio – insere um elemento no meio da lista
- ✓ lista vazia – verifica se a lista está vazia
- ✓ elemento do início – mostra o elemento que está no início da lista
- ✓ elemento do final – mostra o elemento que está no final da lista
- ✓ remover – elimina um elemento que está na lista ligada
- ✓ contar nós – verifica quantos elementos existem na lista
- ✓ mostrar lista – exibe todos os elementos que a lista possui.
- ✓ buscar – verifica se determinado elemento pertence à lista.
- ✓ destruir – elimina todos os elementos da lista ligada.

Descrição da classe Lista

- Basicamente, esses métodos são:
  - construtores No() e listaLigada() - criam o nó e a lista ligada e inicia o primeiro e o último nós
  - inserirInicio() – insere um elemento no início da lista
  - inserirFinal() – insere um elemento no final da lista
  - inserirMeio() – insere um elemento no meio da lista
  - listaVazia() – verifica se a lista está vazia
  - elementoInicio() – mostra o elemento que está no início da lista
  - elementoFinal() – mostra o elemento que está no final da lista
  - remover() – elimina um elemento que está na lista ligada
  - contarNos() – verifica quantos elementos existem na lista
  - mostrarlista() – exibe todos os elementos que a lista possui.
  - buscaNo() – verifica se determinado elemento pertence à lista.
  - destruir() – elimina todos os elementos da lista ligada.

Classe No

```
class No {
        private Object elemento;
        private No prox;
        public No (Object elem){
                elemento = elem;
                prox = null;
        }
}
```

Classe ListaLigada

```
public class ListaLigada {
        private No primeiro, ultimo;
        public ListaLigada ()  {
                primeiro = null;
                ultimo = null;
        }
```

Método listaVazia

```
public boolean listaVazia (){
        if (primeiro == null){
                return true;
        } else {
                return false;
        }
 }
```

Método inserirInicio

```
public void inserirInicio (Object elemento){
      No novoNo = new No(elemento);

      if (listaVazia()){
            ultimo = novoNo;
      } else {
            novoNo.prox = primeiro;
      }
      primeiro = novoNo;
}
```

Método inserirFinal

```
public void inserirFinal (Object elemento){
      No novoNo = new No(elemento);

       if (listaVazia()){
             primeiro = novoNo;
       } else {
             ultimo.prox = novoNo;
       }
       ultimo = novoNo;
}
```

Método contarNos

```
public int contarNos ( ){
        int tamanho = 0;
        No noTemp = primeiro;
        while (noTemp != null){
                tamanho = tamanho + 1;
                noTemp = noTemp.prox;
        }
        return tamanho;
}
```

Método inserirMeio

```
public void inserirMeio(Object elemento, int posicao){
        No novoNo = new No(elemento);
        No noTemp = primeiro;
        int nroNos, posAux=1;
        nroNos = contarNos();
        if (posicao <= 1){
                inserirInicio(novoNo);
        } else {
                if (posicao > nroNos){
                     inserirFinal(novoNo);
                } else {
                     while (posAux < (posicao – 1)){
                                noTemp = noTemp.prox;
                                posAux = posAux + 1;
                     }
                     novoNo.prox = noTemp.prox;
                     noTemp.prox = novoNo;
                }
        }
}
```

Método remover

```
public void remover( Object elemento){
        No noTemp = primeiro;
        No noAnt = null;
        if (primeiro.elemento.equals(elemento)){
                primeiro = primeiro.prox;
        } else {
                while ((noTemp != null) && (!noTemp.elemento.equals(elemento))){
                    noAnt = noTemp;
                    noTemp = noTemp.prox;
                }
                if (noTemp != null){
                    noAnt.prox = noTemp.prox;
                }
                if (noTemp == ultimo){
                    ultimo = noAnt;
                }
        }
}
```

Métodos elementoInicio e elementoFinal

```
public Object elementoInicio( ) throws IOException {
	if (! listaVazia()){
		return primeiro.elemento;
	} else {
		throw new IOException();
	}
}

public Object elementoFinal( ) throws IOException {
	if (! listaVazia()){
		return ultimo.elemento;
	} else {
		throw new IOException();
	}
}
```

Método buscaNo

```
public boolean buscaNo (Object elemento){
        No noTemp = primeiro;
        boolean found = false;

        while ((noTemp != null) && (!found)){
                if (noTemp.elemento.equals(elemento)){
                    found = true;
                } else {
                    noTemp = noTemp.prox;
                }
        }
        return found;
}
```

Método mostrarlista

```
public void mostrarLista ( ){
        int i = 1;
        No noTemp = primeiro;
        while (noTemp != null){
                System.out.println("Elemento "+ noTemp.elemento+ " posição "+i);

                noTemp = noTemp.prox;

                i++;
        }
}
```

Exercícios

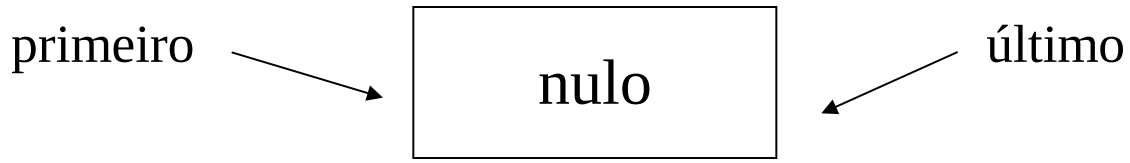
- Como fica a lista após a execução do programa abaixo

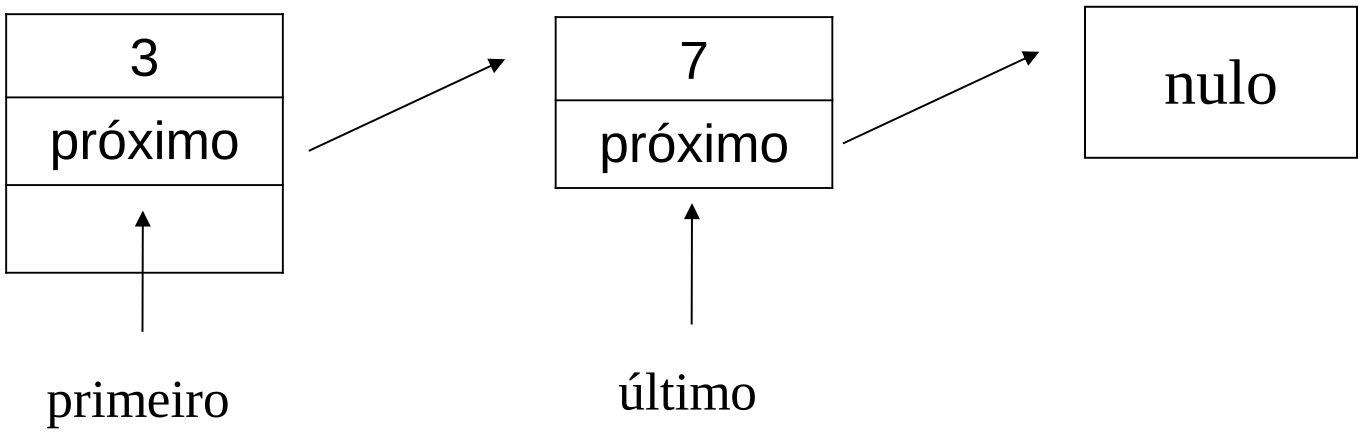
```
public class Exercicio1{
    public static void main (String arg []) {
        ListaLigada lista = new ListaLigada();
        lista.mostrarLista();
        lista.inserirInicio(new Integer(3));
        lista.inserirFinal(new Integer(7));
        lista.inserirMeio(new Integer(5),2);
        lista.inserirMeio(new Integer(4),2);
        lista.inserirMeio(new Integer(6),4);
        lista.mostrarLista();
        lista.remover(new Integer(3));
        lista.remover(new Integer(7));
        lista.remover(new Integer(25));
        lista.mostrarLista();
        lista.remover(new Integer(6));
        lista.mostrarLista();

    }
}
```

Exercícios – Outpout

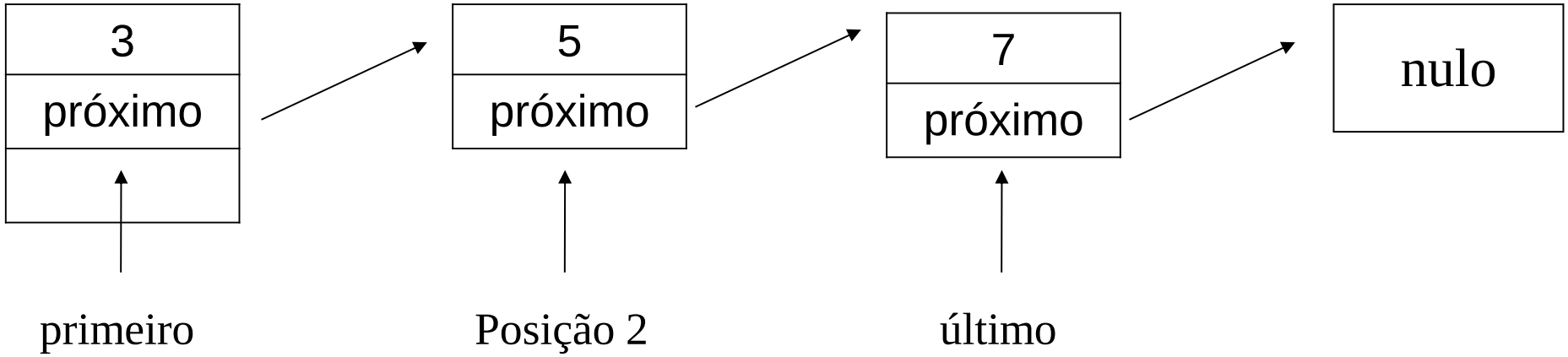
```
C:\unipJava>java TestaListas1
Lista Vazia
Elemento 3 - posicao 1.
Elemento 4 - posicao 2.
Elemento 5 - posicao 3.
Elemento 6 - posicao 4.
Elemento 7 - posicao 5.
Elemento 4 - posicao 1.
Elemento 5 - posicao 2.
Elemento 6 - posicao 3.
Elemento 4 - posicao 1.
Elemento 5 - posicao 2.
C:\unipJava>
```

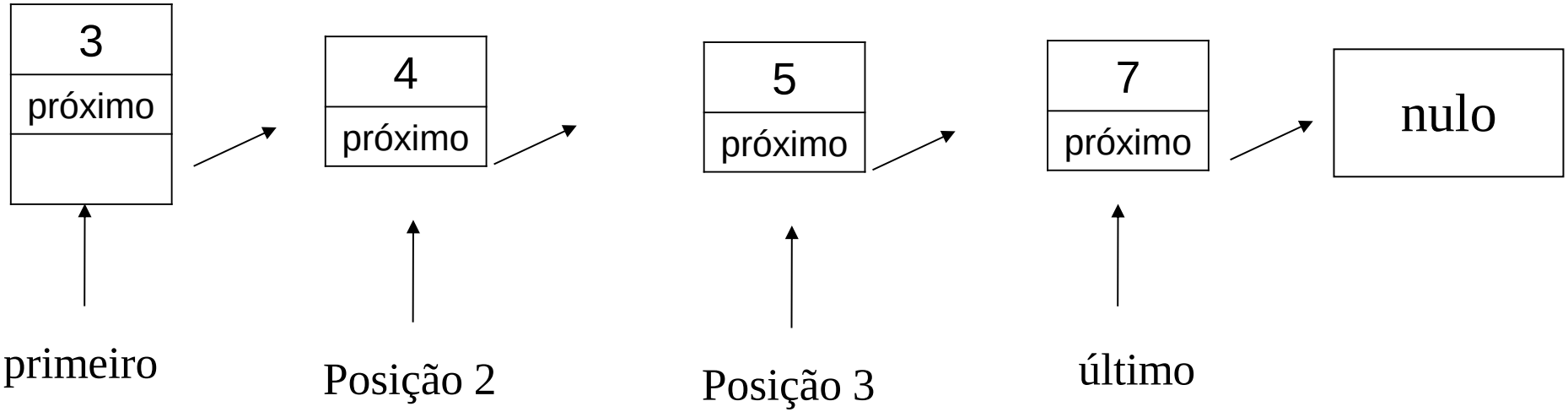
Exercícios – Outpout



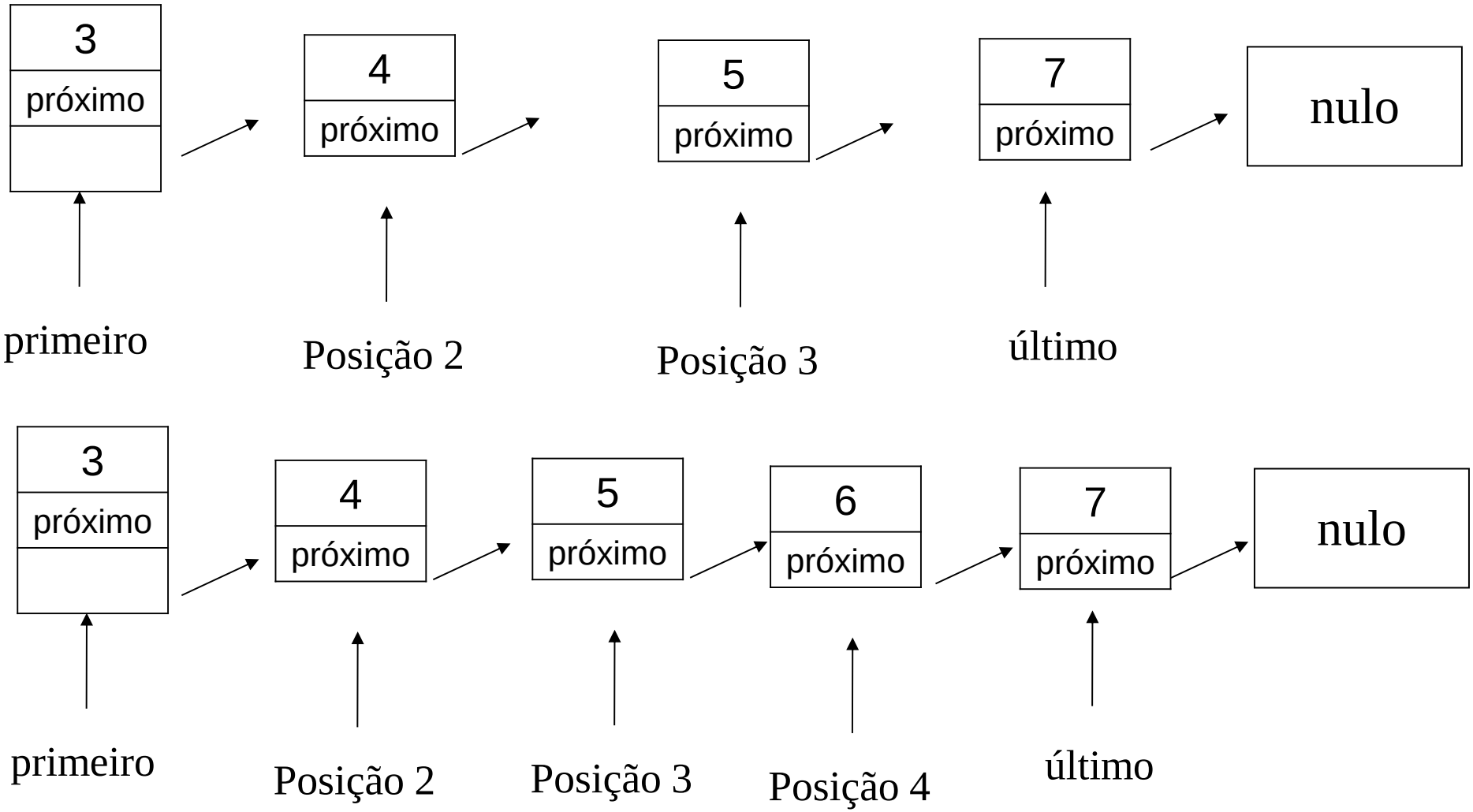


Exercícios – Outpout

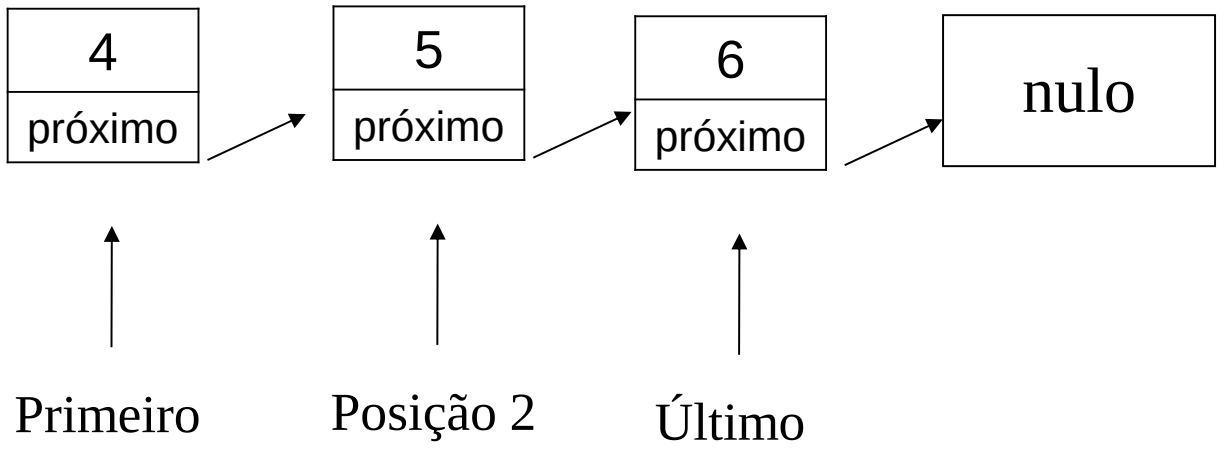


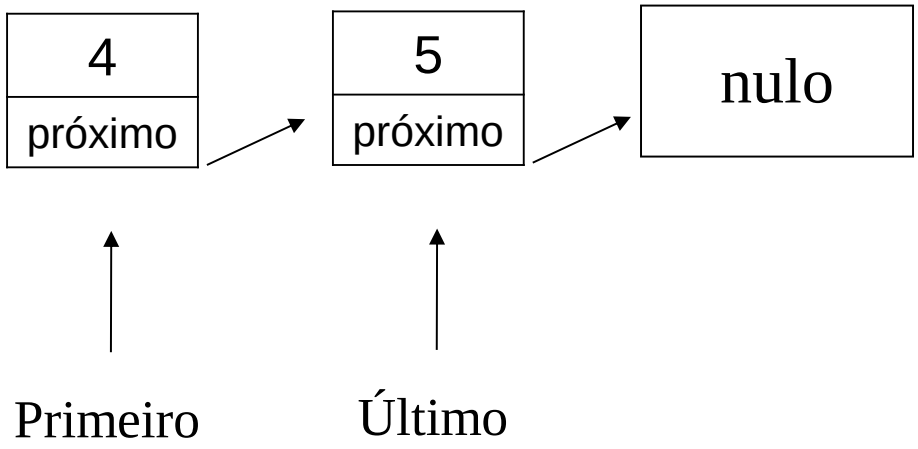


Exercícios – Outpout



Exercícios – Outpout





Exercício 1

- Implementar um novo método na classe ListaLigada chamado copia, que recebe como parâmetro um objeto da classe ListaLigada (ou seja, outra lista ligada). O método deve criar uma nova lista, cópia da lista recebida como parâmetro

Exercício 2

- Implementar um novo método na classe ListaLigada chamado uneListas, que recebe como parâmetro dois objetos da classe ListaLigada. O método deve criar uma nova lista, que é a união das duas recebidas como parâmetro, uma após a outra