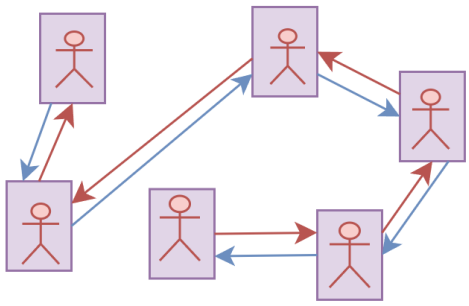
Lista Duplamente ligada

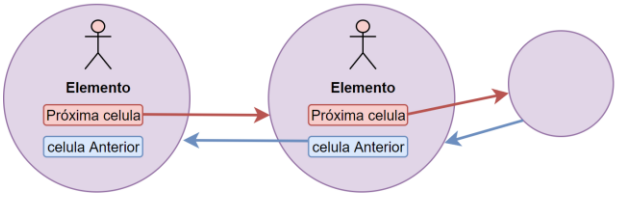
A **lista duplamente ligada** não muda a maneira como as **celulas** são guardados na memória. Porém agora cada **célula** *além de saber quem é o* ***próximo****, também sabe quem é o* ***anterior*** *a* ***ela***:



As células de uma lista duplamente ligada

Uma **célula** de uma **lista duplamente ligada** precisa além de **conhecer seu próximo**, também **conhecer seu anterior**. É possível fazer isso adicionando essas referências na célula *(além é claro de manter o elemento)*:

```
public class CelulaDupla {

    private Object elemento;

    private CelulaDupla proxima;
    private CelulaDupla anterior;
```



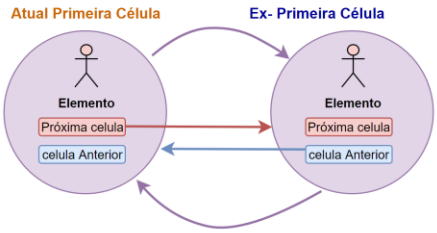
Adicionar Elementos na lista duplamente ligada

Para **inserir um elemento no começo** da **Lista Duplamente ligada** é necessário tomar cuidado para que a ***ex-primeira célula*** **aponte para sua anterior (atual primeira).**

Basicamente falando é simplesmente **fazer com que** a **nova/atual primeira célula** faça a **ex primeira ser a sua próxima**, enquanto a **ex-primeira** faz a **atual primeira** ser **sua anterior. E agora a primeira Celula da lista ligada aponto para a atual primeira:**

```java
public void adicionaNoComeco(Object elemento){

    if(totalElementos == 0 ){
        CelulaDupla novaCelula = new CelulaDupla(elemento);
        this.primeira = novaCelula;
        this.ultima = novaCelula;

    }else{

        CelulaDupla novaCelula = new CelulaDupla(elemento, this.primeira);
        this.primeira.setAnterior(novaCelula);
        this.primeira = novaCelula;
    }

    this.totalElementos ++;
}
```

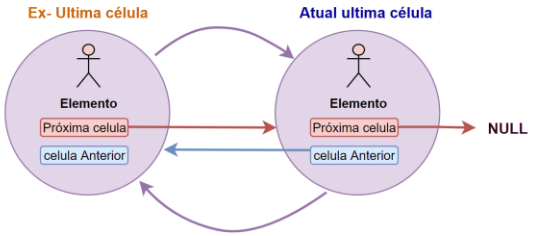
Esse parametro do construtor é a próxima celula



Adicionar no final da lista ligad também é uma operação simples, bem parecida com adicionar no começo na real. A gente só precisa cuidar dar referências entre a anterior e a próxima.

A **ex-ultima célula** precisa fazer com que a **atual última seja sua próxima**, enquanto a **atual última precisa fazer com que** a **ex-ultima seja sua anterior,** depois de estabelecer as referências basta fazer com que a **última célula da lista ligada** aponte para a **atual última criada:**

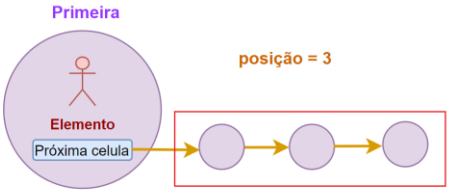
```java
public void adiciona(Object elemento){

    if(this.totalElementos == 0){
        this.adicionaNoComeco(elemento);
    }else{

        CelulaDupla novaCelula = new CelulaDupla(elemento);

        this.ultima.setProximo(novaCelula);
        novaCelula.setAnterior(this.ultima);

        this.ultima = novaCelula;

        this.totalElementos ++;

    }
}
```



Percorrer a lista duplamente ligada

Pra navegar nas células de uma lista ligada precisamos **percorrer pelas referências**. Nesse caso eu estou percorrendo **baseado em uma posição** (eu uso a posição para iterar sobre as células) **começando da primeira**:

```java
private Celula pegaCelula(int posicao){

    if(!posicaoOcupada(posicao)){
        throw new IllegalArgumentException("Posição inexistente");
    }

    Celula atual = this.primeira;

    for(int i = 0; i < posicao; i++){
        atual = atual.getProximo();
    }

    return atual;
}
```



Adicionar Elementos no “meio” da lista duplamente ligada

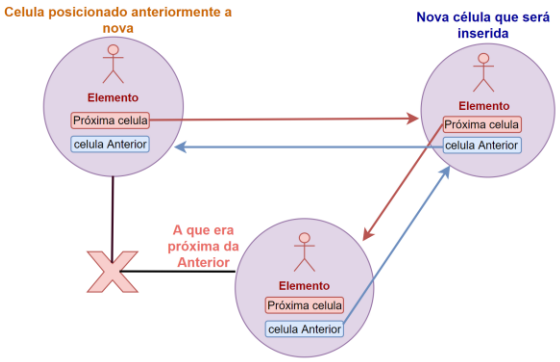
Para adicionar um elemento em uma posição específica, a ideia é que essa **nova célula** seja inserida entre as **outras células já existentes**.

Pensando nisso a lógica é **pegar a célula que está na posição anterior** a **essa nova** , a gente também precisa da **próxima célula** dessa **anterior** *(já que vamos inserir a **nova** entre elas).*

Agora basta que a **nova célula tenha sua próxima** sendo a **próxima célula** da **anterior, enquanto a anterior da nova célula** vai ser a **anterior** previamente pegada. A **célula anterior** terá s**ua próxima sendo** a **nova** e a **próxima célula da anterior** vai ter **sua anterior** sendo a **nova célula**.

No p**ortuguês claro**, eu peguei a **célula nova** e **enfiei no meio dessas duas** "**A <- (N) ->** B":

```java
public void adiciona(int posicao, Object elemento){

    if(posicao == 0){
        adicionaNoComeco(elemento);

    }else if(posicao == this.totalElementos){
        adiciona(elemento);

    }else{

        CelulaDupla celulaAnterior = this.pegaCelula( posicao: posicao - 1);
        CelulaDupla celulaProximaDaAnterior = celulaAnterior.getProximo();
        CelulaDupla novaCelula = new CelulaDupla(elemento, celulaAnterior.getProximo());

        novaCelula.setAnterior(celulaAnterior);

        celulaAnterior.setProximo(novaCelula);
        celulaProximaDaAnterior.setAnterior(novaCelula);

        this.totalElementos ++;
    }
}
```



Remover Elementos no começo e no fim da lista duplamente ligada

Remover um elemento do começo da lista ligada é simples, basta **pegar a próxima célula** da **atual primeira** e **atribuir como primeira célula da lista ligada**, com isso a **até então primeira** vai ser substituída:

```java
public void removeDoComeco(){
    if(this.totalElementos == 0){
        throw new IllegalArgumentException("Não existe uma celula no começo");
    }

    this.primeira = primeira.getProximo();
    this.totalElementos --;

    if(this.totalElementos == 0){
        this.primeira = null;
    }
}
```

Remover do fim é meio "same energy". Basta pegar a **célula anterior** da **atual última** *(famosa penúltima),* essa **penúltima** vai ser **atribuída a última da lista ligada**, portanto sua **próxima não existe** *(com isso a até então atual última deixa de existir):*

```java
public void removeDoFim(){
    if(this.totalElementos == 0){
        this.removeDoComeco();
    }else{

        CelulaDupla penultimaCelula = this.ultima.getAnterior();

        penultimaCelula.setProximo(null);
        this.ultima = penultimaCelula;

        this.totalElementos --;
    }
}
```

Remover Elementos no “meio” da lista duplamente ligada

A remoção de uma célula no meio de uma lista duplamente ligada consiste em encontrar quem vem antes do elemento de remoção e quem vai depois, e ligar ambos excluindo o atual da relação:

```java
public void remove(int posicao){
    if(posicao == 0){
        this.removeDoComeco();

    }else if(posicao == this.totalElementos){
        this.removeDoFim();

    }else{
        CelulaDupla anterior = this.pegaCelula( posicao: posicao - 1);
        CelulaDupla atual = anterior.getProximo();
        CelulaDupla proxima = atual.getProximo();

        anterior.setProximo(proxima);
        proxima.setAnterior(anterior);

        this.totalElementos --;
    }
}
```