

Tema 2.4.1 TAD lineales

TAD Lista: Implementación con una Lista Simplemente Enlazada

Estructura de Datos y Algoritmos (EDA)

Contenidos

- ▶ 2.1. ¿Qué es un TAD Lineal?
- ▶ 2.2. TAD Pila
- ▶ 2.3. TAD Cola
- ▶ 2.4. TAD Lista
 - ▶ **2.4.1 Implementación con una Lista Simplemente Enlazada**
 - ▶ 2.4.2 Implementación con una Lista Doblemente Enlazada

Objetivos

- ▶ Al final de la clase, los estudiantes deben ser capaces de:
 - ▶ Especificar formalmente un TAD Lista
 - ▶ Implementar un TAD Lista usando una Lista Simplemente Enlazada
 - ▶ Explicar qué ventajas del uso de last (referencia al último elemento (last) en la Lista)

Clase SNode

```
public class SNode {  
  
    public String elem;  
    public SNode next;  
  
    public SNode(String e) {  
        elem = e;  
    }  
  
}
```

TAD Lista

- ▶ Especificación formal:
 - ▶ Secuencia de elementos $\{a_1, a_2, \dots, a_n\}$ donde cada elemento tiene un único predecesor (excepto el primero que no tiene predecesor) y un único sucesor (excepto el último que no tiene sucesor).
 - ▶ Las operaciones básicas de un TAD Lista son:
 - ▶ **Agregar** un elemento a la Lista
 - ▶ **Eliminar** un elemento de la Lista
 - ▶ **Consultar** un elemento de la Lista

Especificación Formal de un TAD Lista (operaciones de agregar)

```
public interface IList {  
  
    public void addFirst(String newElem);  
  
    public void addLast(String newElem);  
  
    public void insertAt(int index, String newElem);  
}
```

Especificación Formal de un TAD Lista (operaciones de consulta)

```
public boolean isEmpty();  
  
public boolean contains(String elem);  
  
public int getSize();  
  
public int getIndexOf(String elem);  
  
public String getFirst();  
  
public String getLast();  
  
public String getAt(int index);  
  
public String toString();
```

Especificación Formal de un TAD Lista (Operación de borrado)

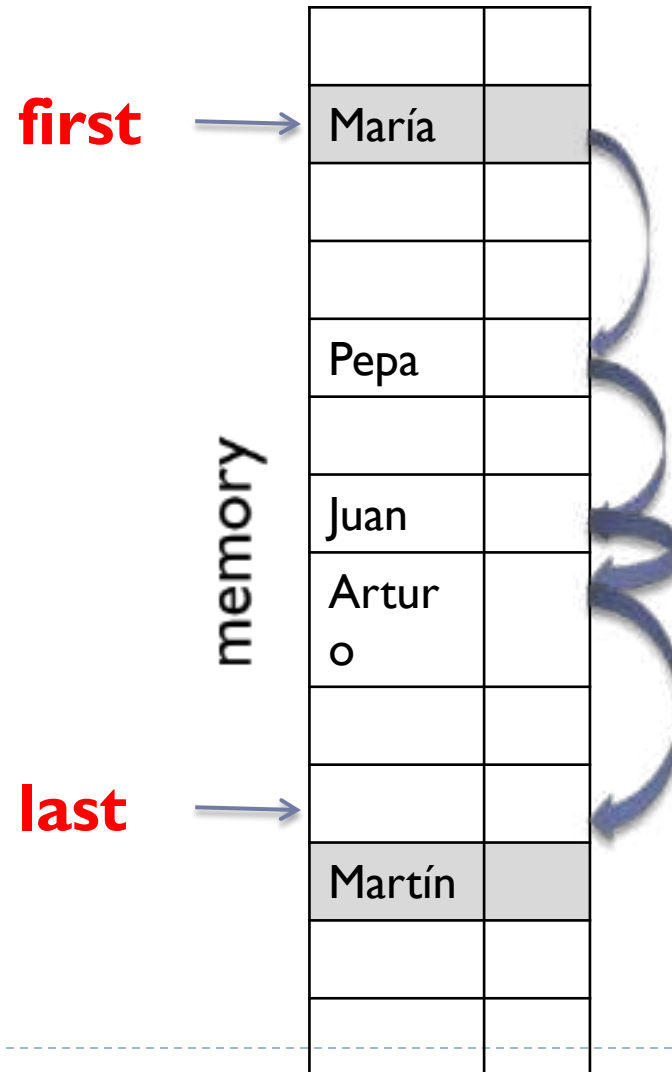
```
public void removeFirst();
```

```
public void removeLast();
```

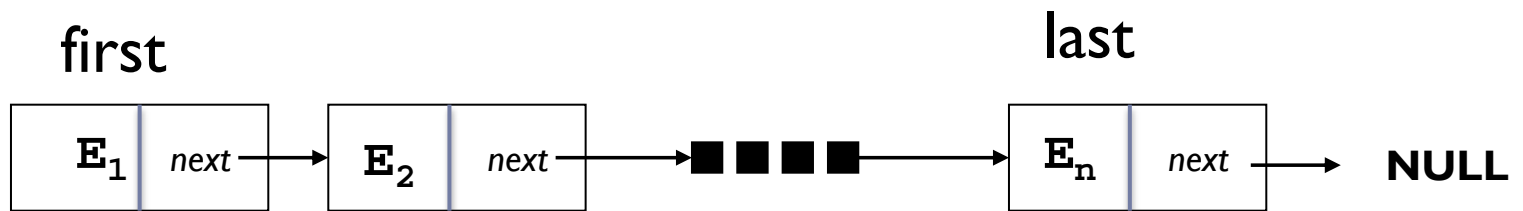
```
public void removeAll(String elem);
```

```
public void removeAt(int index);
```

Implementación de un TAD Lista usando una Lista Simplemente Enlazada



Implementación de un TAD Lista usando una Lista Simplemente Enlazada



```
public class SList implements IList {  
  
    public SNode first;  
    public SNode last;  
    int size; //by default is 0  
}
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método empty)

first → **NULL**

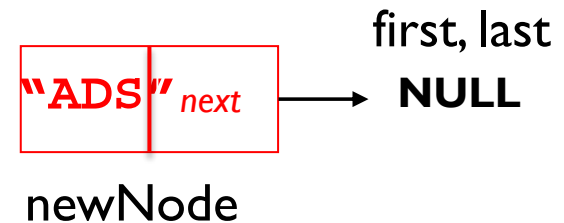
```
public boolean isEmpty() {  
    return (first == null);  
}
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)

Agrega “ADS” en una Lista vacía

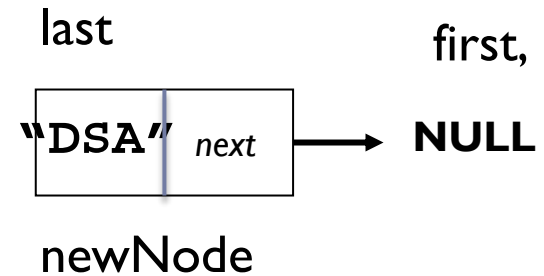
`list.addFirst("ADS")`



```
Snode newNode=new SNode("ADS");  
newNode.next=first;  
if (first==null) last=newNode;  
first=newNode;
```

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)

`list.addFirst("DSA")`

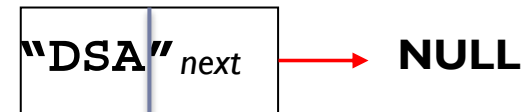


```
Snode newNode=new SNode( "DSA" );  
newNode.next=first;  
if (first==null) last=newNode;  
first=newNode;
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)

first, last



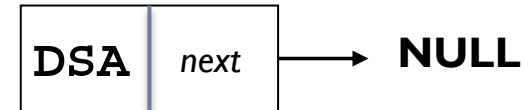
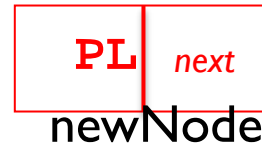
```
list.addFirst("DSA")
```

```
Snode newNode=new SNode("DSA");  
newNode.next=first;  
if (first==null) last=newNode;  
first=newNode;
```

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)

Ahora, agrega "PL" en el inicio de la Lista

first,last

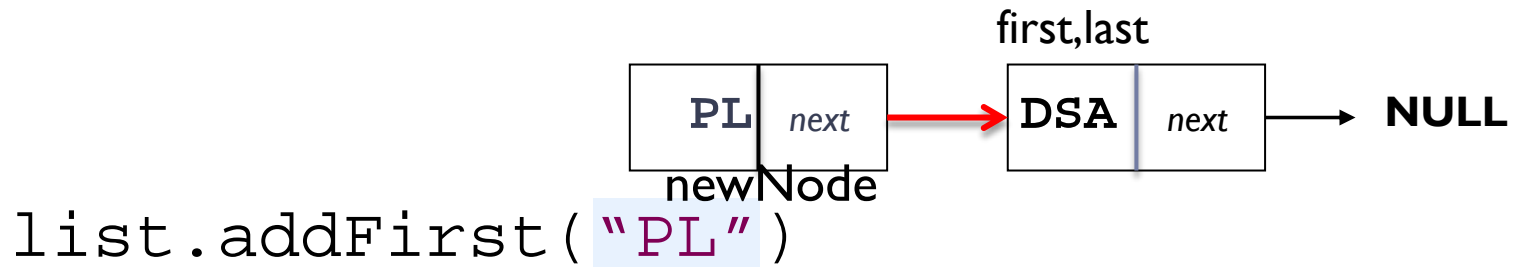


```
list.addFirst( "PL" )
```

```
    Snode newNode=new SNode( "PL" );  
    newNode.next=first;  
    if (first==null) last=newNode;  
    first=newNode;
```

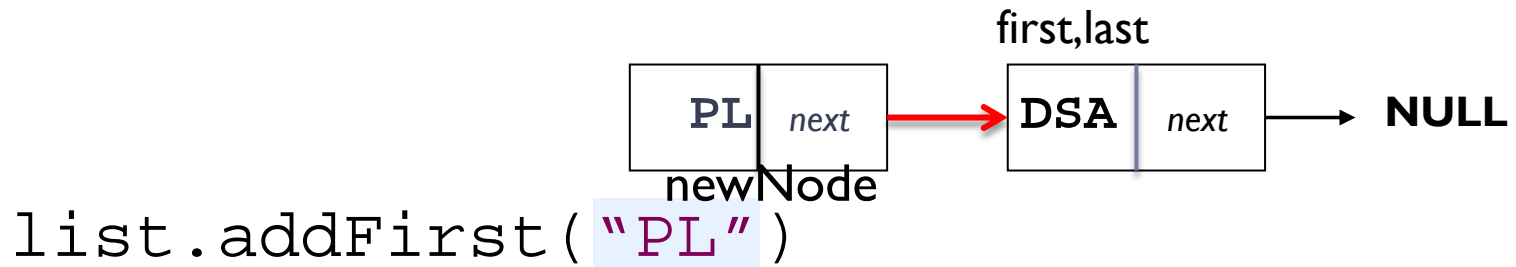
A diagram illustrating the execution of the `addFirst` method. A blue line starts from the `list.addFirst` call, goes up and then right to point at the `Snode newNode` line. Another blue line starts from the `newNode` line, goes down and then right to point at the `newNode.next=first` line. A third blue line starts from the `newNode.next=first` line, goes down and then right to point at the `if (first==null)` line. A fourth blue line starts from the `if (first==null)` line, goes down and then right to point at the `first=newNode` line.

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)



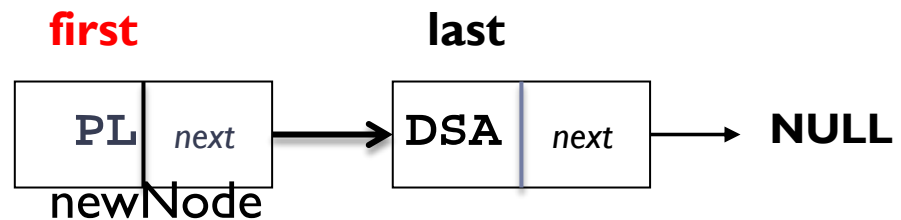
```
Snode newNode=new SNode("PL");
newNode.next=first;
if (first==null) last=newNode;
first=newNode;
```


Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)



```
Snode newNode=new SNode("PL");  
newNode.next=first;  
if (first==null) last=newNode;  
first=newNode;
```

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)



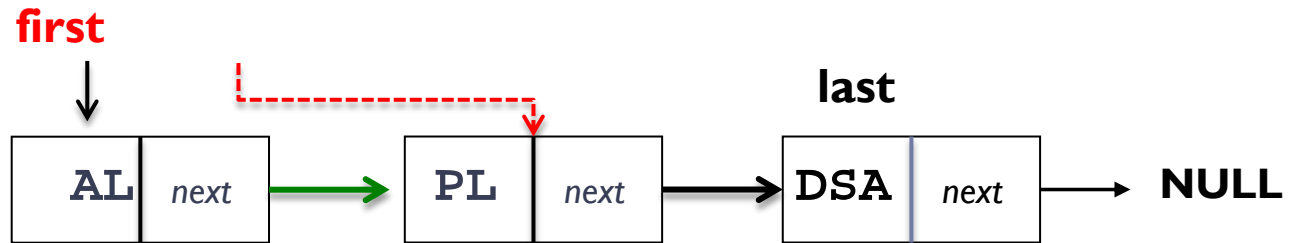
```
list.addFirst("PL")
```

```
Snode newNode=new SNode("PL");  
newNode.next=first;  
if (first==null) last=newNode;  
first=newNode;
```

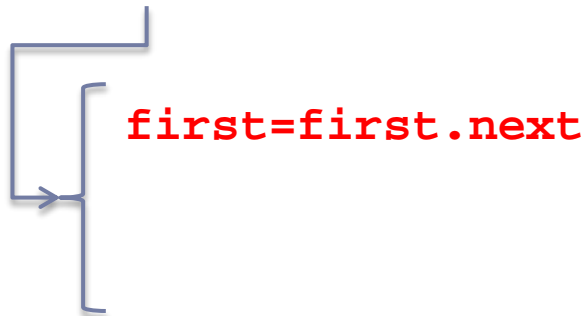
Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addFirst)

```
public void addFirst(String newElem) {  
  
    SNode newNode = new SNode(newElem);  
    newNode.next = first;  
    //If list is empty, last has also to reference to newNode  
    if (first==null) last=newNode;  
    //we set the new first node  
    first = newNode;  
    size++;  
}
```

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeFirst)

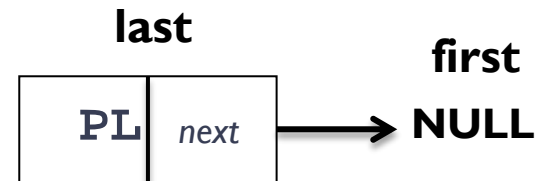
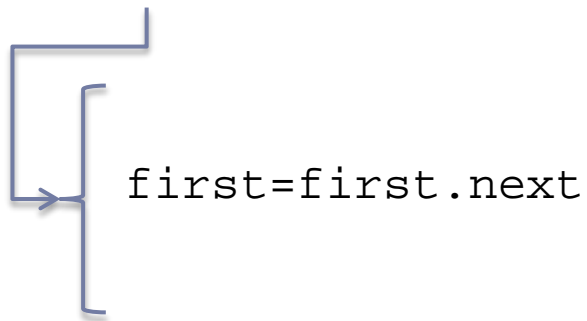


```
list.removeFirst()
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeFirst)

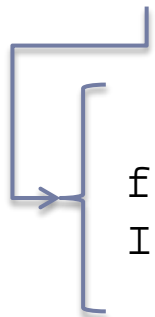
```
list.removeFirst()
```



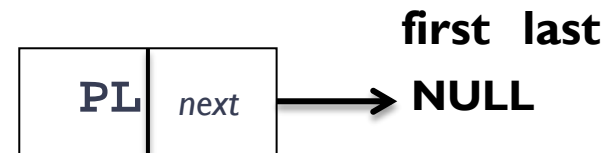
- ¿Funciona cuando la lista solo tiene un nodo?
- Si la lista está vacía, ¿qué sucede?

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeFirst)

```
list.removeFirst( )
```



```
first=first.next  
If (first==null) last=null;
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada

(Método removeFirst)

```
public void removeFirst() {  
    if (!isEmpty()) {  
        first = first.next;  
        if (first==null) last=null;  
        size--;  
    }  
}
```

¡¡Advertencia!! En realidad, el nodo no se elimina, solo es ignorado
Java garbage collection lo eliminará

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addLast)

```
list.addLast("HomeLand")
```



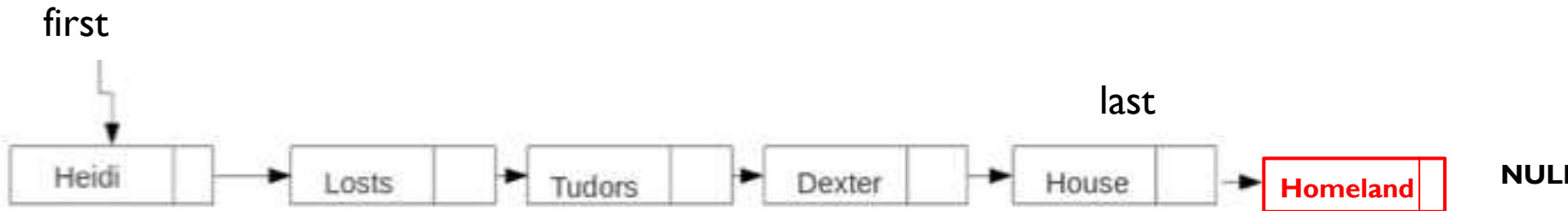
Primer paso: crear un nuevo nodo para almacenar “HomeLand”

```
SNode node = new SNode (newElem);
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addLast)

```
lista.addLast("HomeLand")
```

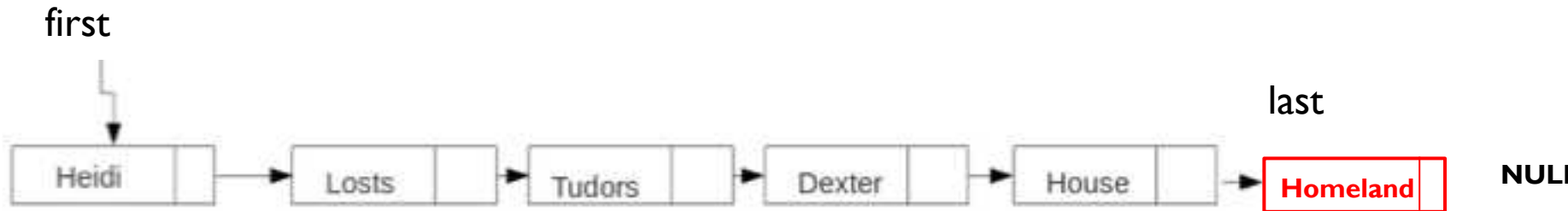


Segundo paso: last.next debe hacer referencia al nuevo nodo

last.next=newNode;

Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addLast)

```
lista.addLast("HomeLand")
```



Tercer paso: actualizar el último

last=newNode;

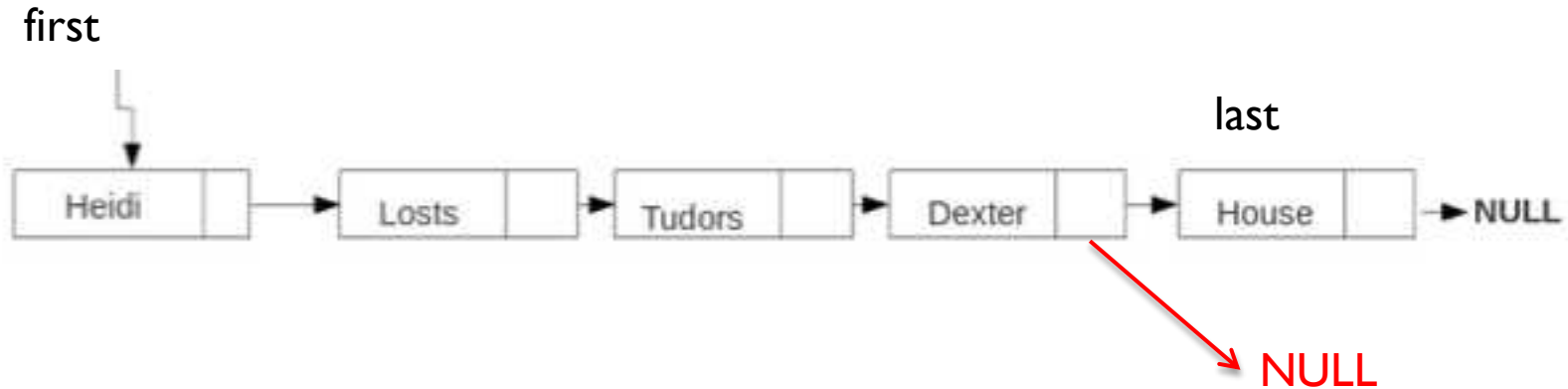
Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método addLast)

```
public void addLast(String newElem) {  
  
    SNode node = new SNode(newElem);  
    if (isEmpty()) addFirst(newElem);  
    else {  
        last.next=node;  
        last=node;  
        size++;  
    }  
  
}
```



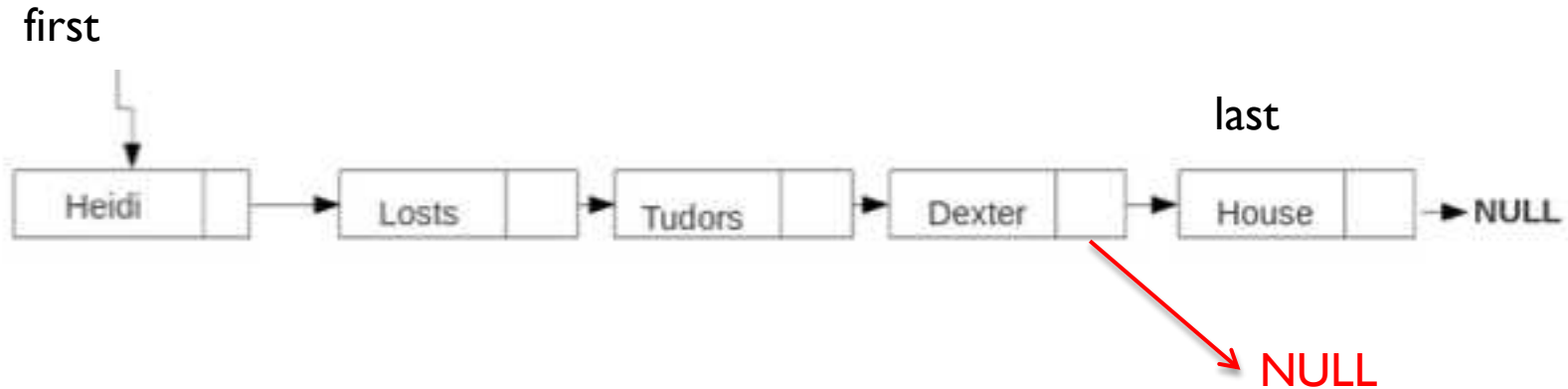
Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

```
lista.removeLast();
```



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

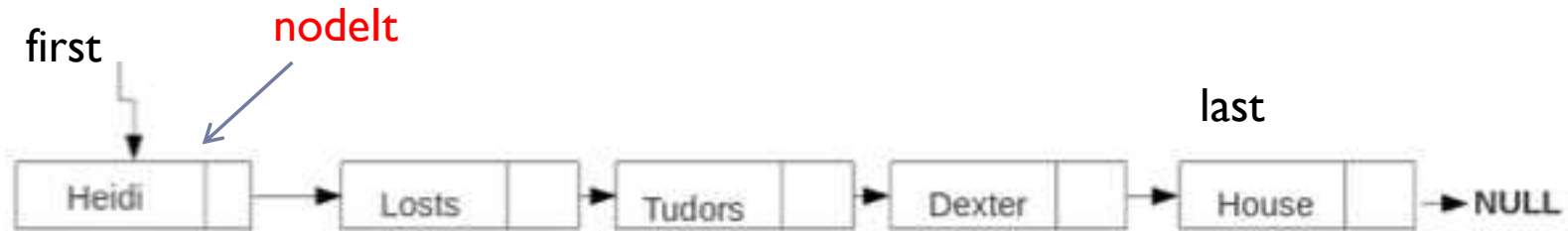
```
lista.removeLast();
```



Idea: tenemos que llegar al penúltimo nodo de la lista. Cuando se alcanza este nodo, entonces tenemos que modificar su próxima referencia a NULL.

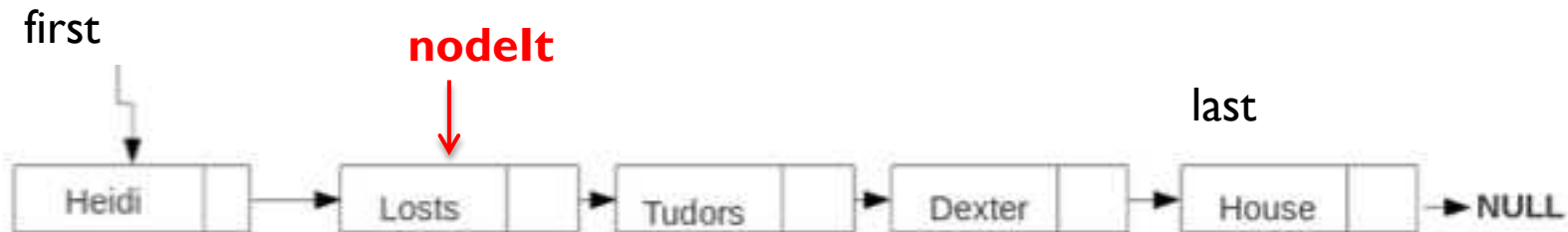
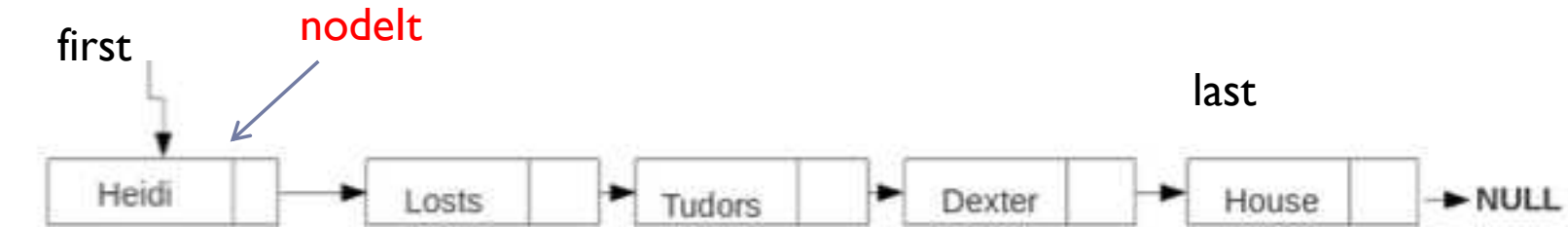
Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

`lista.removeLast()`



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

`lista.removeLast()`

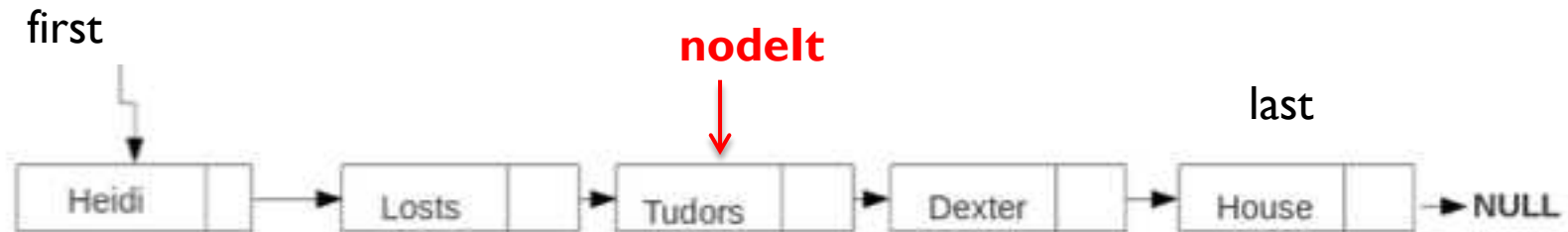


`nodeIt = nodeIt.next;`



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

`lista.removeLast()`



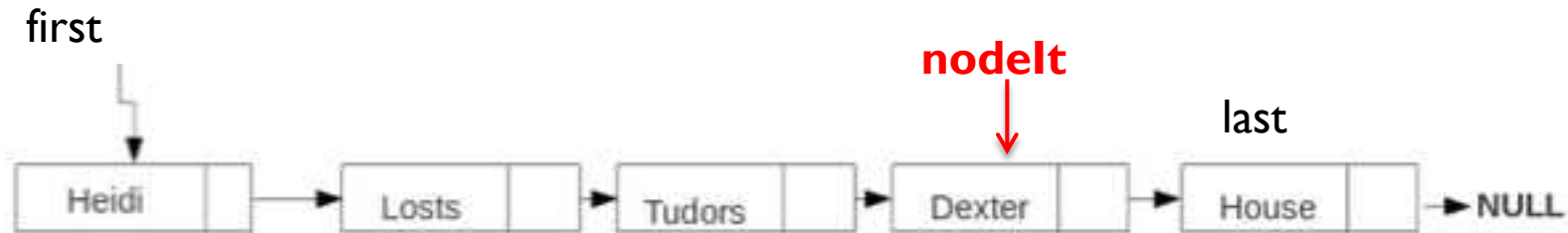
`nodeIt = nodeIt.next;`

¿Cuándo tenemos que parar?



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

`lista.removeLast()`



```
while (nodelt!=last) {  
    nodelt= nodelt.next;  
}
```

hasta que `nodelt.next!=last`



Implementación de un TAD Lista usando una Lista Simplemente Enlazada (Método removeLast)

```
public void removeLast() {  
    if (!isEmpty()) {  
        if (size==1)  
            removeFirst();  
        else {  
            SNode nodeIt = first;  
            while (nodeIt.next!=last) {  
                nodeIt = nodeIt.next;  
            }  
            nodeIt.next=null;  
            last=nodeIt;  
            size--;  
        }  
    }  
}
```

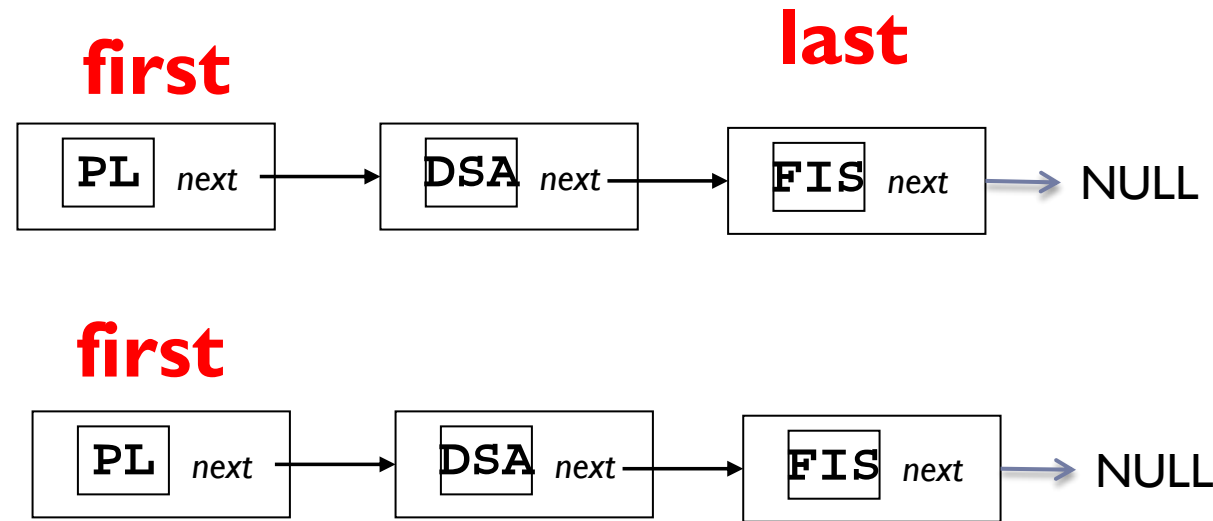
Ejercicio: implementa el resto de los métodos (Clase de laboratorio)

```
public String getFirst();  
public String getLast();  
public int getSize();  
public boolean contains(String elem);  
public int indexOf(String elem);  
  
public void insertAt(int index, String newElem);  
public String getAt(int index);  
public void removeAll(String elem);
```



Implementación sin last

- ▶ **Ejercicio:** Implementa una Lista Simplemente Enlazada sin usar last
- ▶ Compara ambas implementaciones. ¿Qué operaciones son mas eficientes usando last?



Lista Simplemente Enlazada Circular

- Ejercicio: implementa una Lista Simplemente Enlazada circular.

first

