UNIVERSIDAD NACIONAL DE CAJAMARCA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS

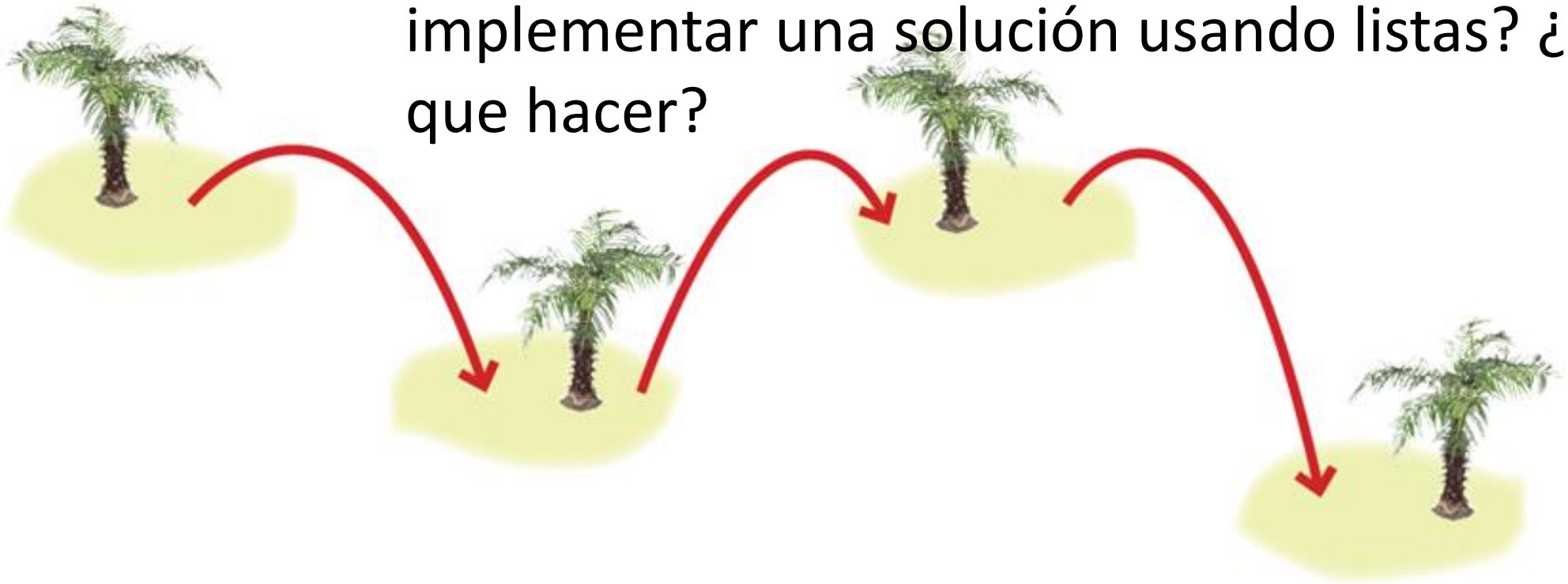
ALGORITMOS Y ESTRUCTURA DE DATOS II
05: LISTAS ENLAZADAS DOBLES

INTRODUCCIÓN

Semana 05 - Listas enlazadas dobles

PLANIFICANDO MI VIAJE

- Imagine que quiere registrar todas las ciudades qu
durante su viaje, partiendo de un punto A hasta lle
destino, y retornar siguiendo el mismo camino ¿Có
implementar una solución usando listas? ¿Qué can
que hacer?



RECORDEMOS

- ¿Qué ventajas tiene una lista con dos referencias respecto de la que solo tiene una?
- ¿Que problemas identifica aún en una lista con dos referencias?
- Si tuviese que utilizar los métodos de una lista enlazada, ¿cuáles seria de utilidad para atender una cola?
- Identifique un método de una lista enlazada y describa los pasos a seguir para su implementación de manera gráfica

¿CÓMO LO SOLUCIONARÍA?

- Que pasaría si me piden que muestre la lista en orden inverso, o dado cualquier nodo en la lista conocer el anterior?

LOGRO ESPERADO

- Al término de la sesión, el estudiante construye un menú de opciones mostrando las funciones principales de una lista doblemente enlazada, verificando el planteamiento de su solución.

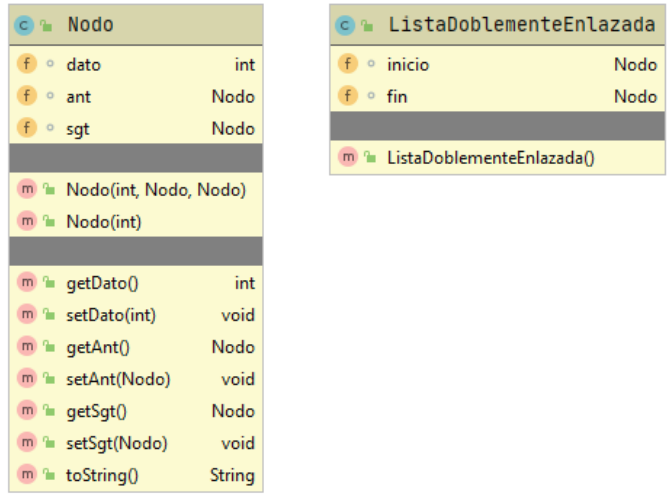
DESARROLLO DEL TEMA

Estructura de Datos

LISTAS ENLAZADAS DOBLES

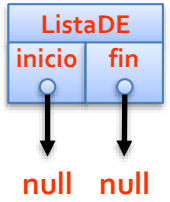
- Hasta ahora, el recorrido de una lista se ha realizado hacia adelante. Existen numerosas aplicaciones en las que es conveniente poder acceder a los elementos o nodos de una lista en cualquier orden, tanto hacia adelante como hacia atrás

- Un nodo de una lista doblemente enlazada tiene dos referencias para enlazar con los nodos izquierdo y derecho, además de la parte correspondiente al campo dato.
- Se tiene la clase ListadoblementeEnlazada que tiene las referencias inicio y fin. Cuando una lista está vacía las referencias valen null.

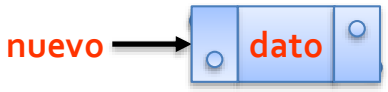


INSERTA AL INICIO EN UNA LISTA VACÍA

- Se tiene la clase ListaDoblementeEnlazada (ListaDE) con dos referencias al inicio y fin de la lista
- Insertamos en una lista vacía
  - Se parte de una lista vacía y el nodo a insertar (nuevo)

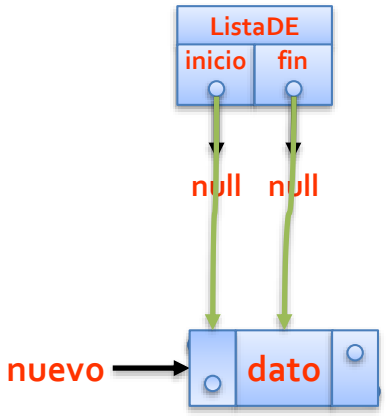



```
Nodo nuevo = new Nodo(dato);
```



INSERTA AL INICIO EN UNA LISTA VACÍA

- Se tiene la clase ListaDoblementeEnlazada (ListaDE) con dos referencias al inicio y fin de la lista
- Insertamos en una lista vacía
  - Se parte de una lista vacía y el nodo a insertar (nuevo)
  - Luego inicio y fin referencian nuevo

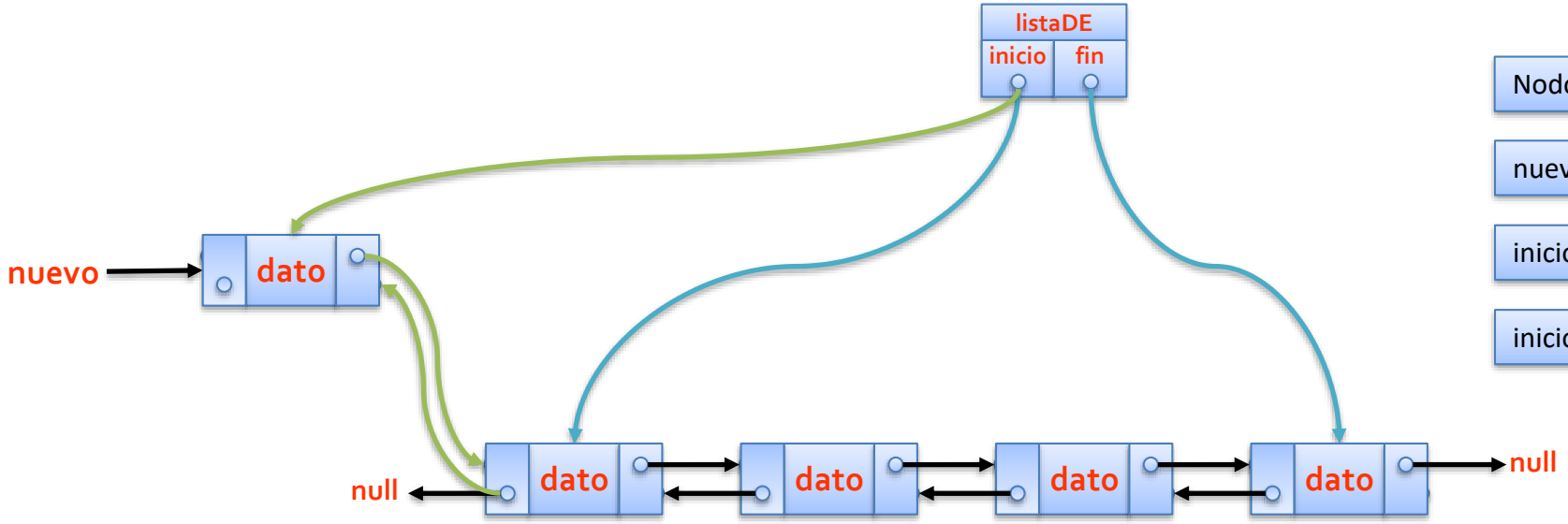



```
Nodo nuevo = new Nodo(dato);
```

```
inicio = fin = nuevo;
```

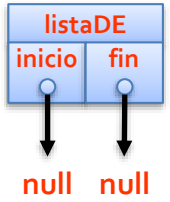
INSERTA AL INICIO EN UNA LISTA NO VACÍA

- Si la lista está vacía
  - Se parte de una lista con elementos y el nodo a insertar (nuevo)
  - Luego nuevo.sgt referencie a inicio
  - A continuación se hace que inicio.ant referencie a nuevo
  - Finalmente inicio referencia a nuevo



INSERTAR AL FINAL EN UNA LISTA VACÍA

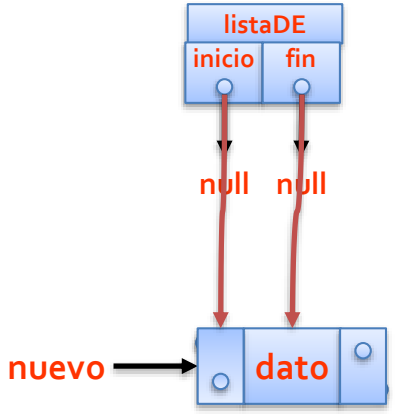
- Al igual que en el ejemplo anterior:
- Insertamos en una lista vacía
  - Partimos con una lista vacía y el nodo a insertar




```
Nodo nuevo = new Nodo (dato);
```

INSERTAR AL FINAL EN UNA LISTA VACÍA

- Al igual que en el ejemplo anterior:
- Insertamos en una lista vacía
  - Partimos con una lista vacía y el nodo a insertar
  - Hacemos de que inicio y fin apunten a nuevo

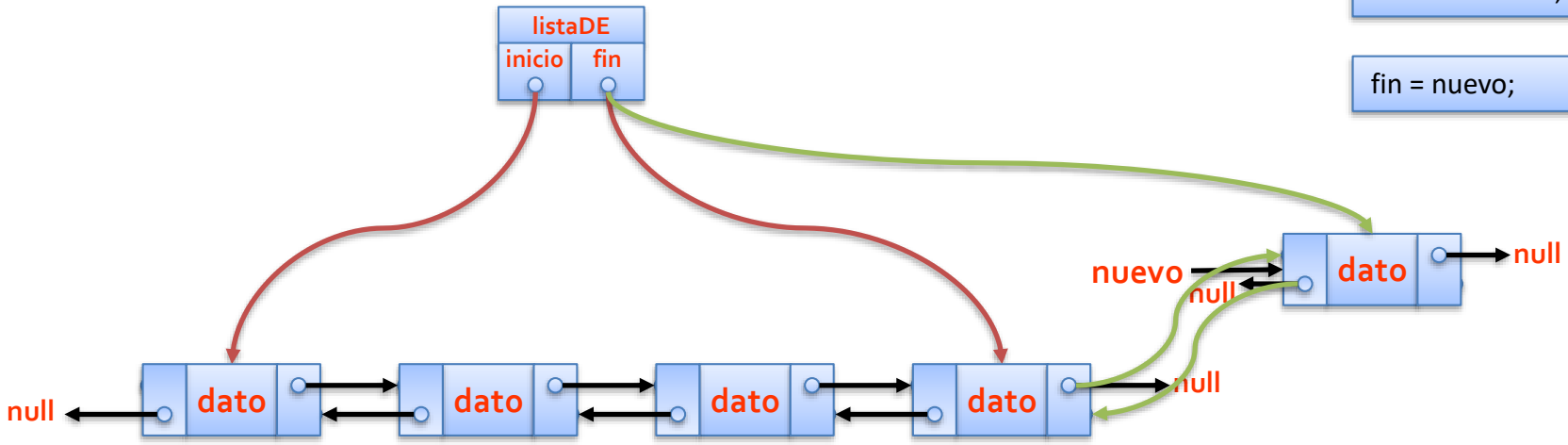



```
Nodo nuevo = new Nodo (dato)

inicio = nuevo;

fin = nuevo;
```

INSERTAR AL FINAL EN UNA LISTA

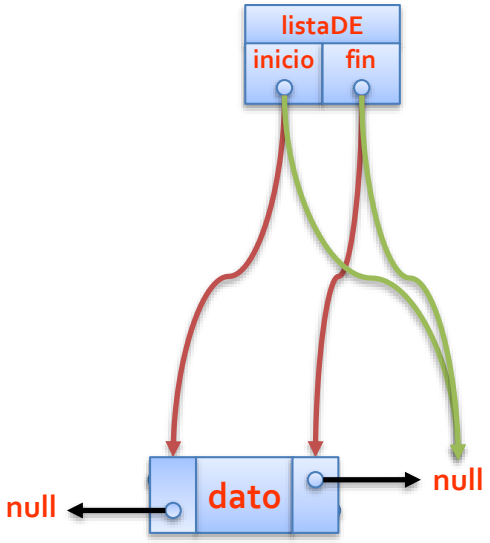
- Insertamos en una lista no vacía
  - Se parte de una lista con elementos y el nodo a insertar
  - Se hace que sgt de fin referencie a nuevo
  - Luego el ant de nuevo referencia a fin
  - Finalmente fin sea referencia a nuevo

```
Nodo nuevo = new Nodo (dato);
fin.sgt = nuevo;
nuevo.ant = fin;
fin = nuevo;
```



ELIMINA EL PRIMER ELEMENTO EN UNA LISTA

- Si la lista tiene un solo elemento
  - Inicio referencia al siguiente elemento (null)
  - Hacemos que lista.fin también referencie a null
  - El recolector de basura libera la memoria

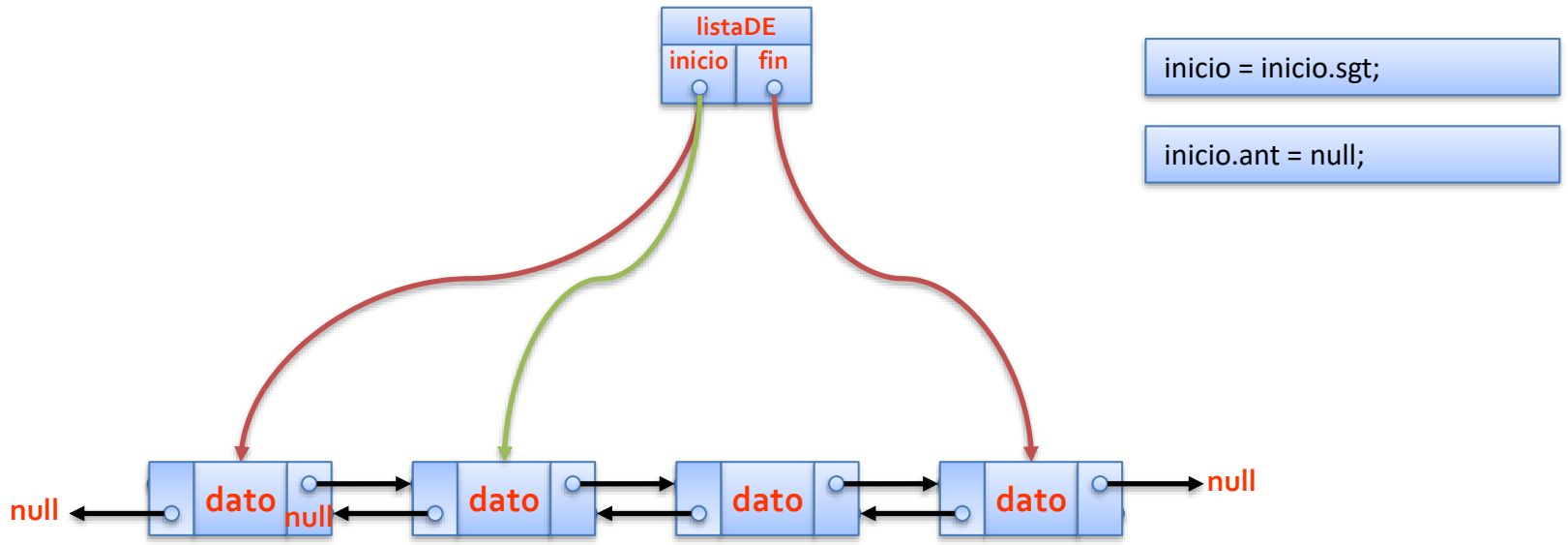



```
inicio = inicio.sgt;
```

```
fin = null;
```

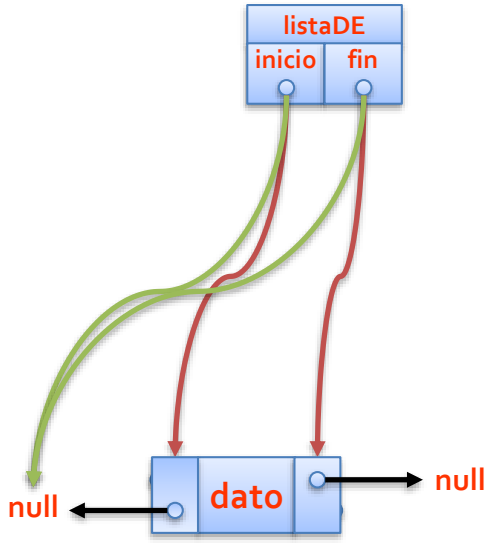
ELIMINA EL PRIMER ELEMENTO EN UNA LISTA

- Si la lista tiene más de un elemento
  - Inicio referencia al siguiente elemento
  - ant de inicio se establece en null
  - El recolector de basura libera la memoria



ELIMINA EL ÚLTIMO ELEMENTO EN UNA LISTA

- Si la lista tiene un solo elemento
  - fin referencia al elemento anterior (null)
  - Ahora inicio también referencia a null
  - El recolector de basura libera la memoria

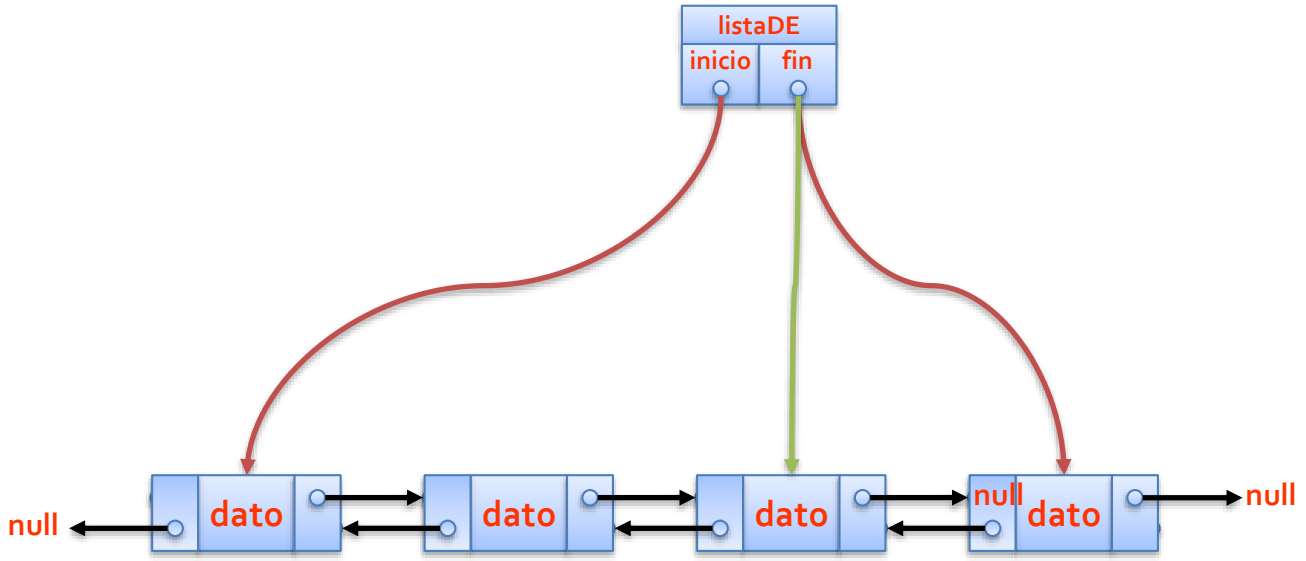



```
fin = fin.ant;
```

```
inicio = null;
```

ELIMINA EL ÚLTIMO ELEMENTO EN UNA LISTA

- Si la lista tiene más de un elemento
  - fin referencia al elemento anterior
  - Hacemos que siguiente de fin sea igual a null
  - El recolector de basura libera la memoria




```
fin = fin.ant;
```

```
fin.sgt = null;
```

OTRAS FUNCIONES


```
@Override
public String toString() {
    String s = "\n[";
    Nodo iterador = inicio;
    while (iterador != null){
        s += iterador.getDato()+ ", ";
        iterador = iterador.getSgt();
    }
    if (!estaVacia()){
        s = s.substring(0,s.length() -2);
    }
    s += "]\n";
    return s;
}
```

```
public ListaDoblementeEnlazada() {
    this.inicio = null;
    this.fin = null;
}
```

```
public boolean estaVacia(){
    return inicio == null;
}
```

EVALUACIÓN DEL TEMA DESARROLLADO

Reflexionemos!

RECORDEMOS

- ¿Qué diferencia existe en el nodo de una lista DE con una lista simple?
- ¿Qué ventajas tiene lista doblemente enlazada?
- ¿Qué otras tareas faltarían implementar en el ejemplo desarrollado?
- ¿Cómo construiríamos una lista circular? ¿Qué opciones hay?
- ¿podría implementarse una lista DE con una sola referencia al elemento inicial?

EJERCICIOS DE APLICACIÓN

A poner en práctica lo aprendido

EJERCICIOS DE APLICACIÓN

- Crear un método que permita:
  - Eliminar todos los nodos de una lista DE
  - Buscar un elemento dentro de la lista DE
  - Contar cuántos nodos existen
  - Recorrer la lista en orden inverso
- Crear un menú de opciones que implemente todos los métodos revisados.