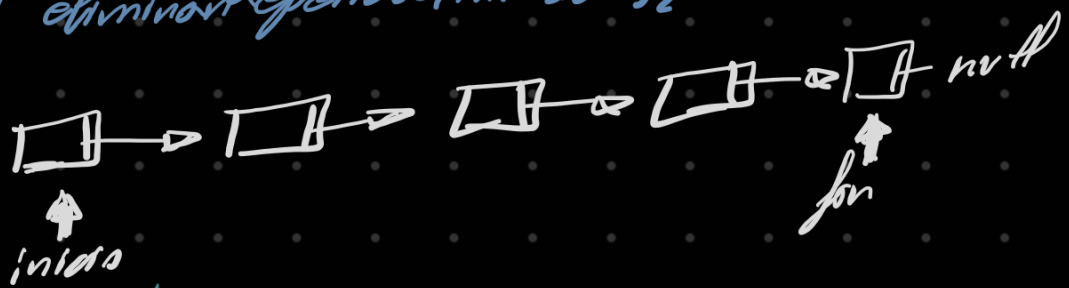
Ejercicio 1 de listas.

```
void eliminarRepetidos(int dato){
```



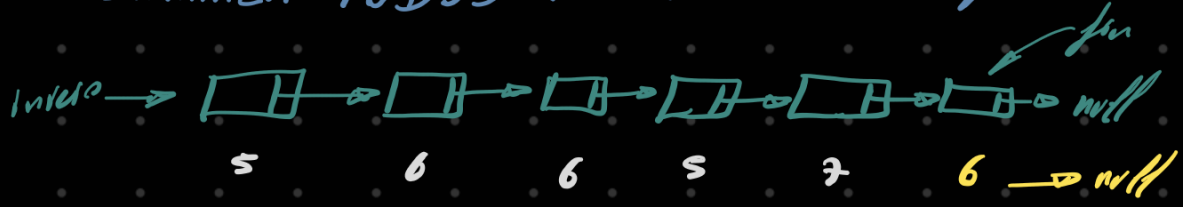

```
Nodo actual = inicio;
Nodo fin = null;
while(actual!=null && actual.getDato()!=dato){
    anterior=actual;
    actual = actual.getSiguiente();

}
if(actual!=null){ //Se encuentra el primer dato
actual = actual.getSiguiente();
while(actual!=null){
    if(actual.getDato()==dato){
        if(actual==fin){
            fin = anterior; //Se borra el el último
        }else{
            anterior.setSiguiente(actual.getSig());
        }
    }
    Aquí anterior no avanza.
//Mientras no eliminemos anterior es el actual y actual avanza
anterior=actual;
actual = actual.getSiguiente();
}
```

Aquí no hace falta comprobar el primero

Pero si comprobamos el último.

Intentar siguiendo el ejercicio anterior haz que se eliminen TODOS los elementos repetidos.




```
Nodo dato = new Nodo (inicio.getDato(), inicio.getSiguiente());

while ( dato != null){
    anterior = dato;
    actual = dato.getSiguiente;

    while (actual != null){
        if( dato.getDato() == actual.getDato()){
            anterior.setSiguiente ( actual.getSiguiente());
            if( actual == fin){
                fin = anterior;
            }
        } else { anterior = actual; // No avanza anterior si se elimina.
        }
    actual = actual.getSiguiente();
    }

    dato = dato.getSiguiente,
}
```

Ejercicio 2 Solo disponible Nodo Lista vacia()

Constructor de Nodo
.getSiguiente()
.setSiguiente()
.getDato()
.setDato()

Solo se recorre una vez

void triplicar(){

```
while(actual != null){
    for(int i=0; i<2; i++){
        Nodo nuevo = new Nodo(actual.getDato(), actual.getSiguiente())
        actual = actual.setSiguiente(nuevo);
        actual = actual.getSiguiente();
        numElementos++;
        if(fin == actual){
            fin = nuevo; // El fin se actualiza las ultimas repeticiones
        }
    }
    actual = actual.getSiguiente();
}
```