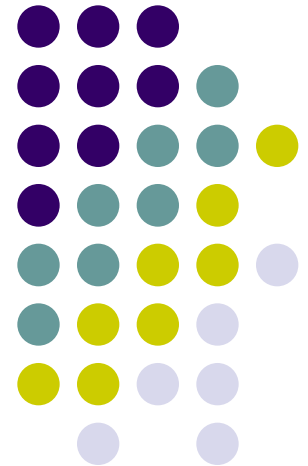


# Listas Enlazadas

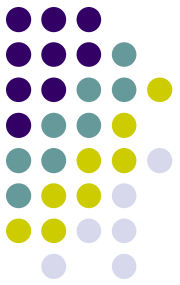
---

Estructura de Datos

Jhonny Felípez Andrade

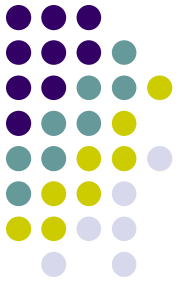


[jrfelizamigo@yahoo.es](mailto:jrfelizamigo@yahoo.es)



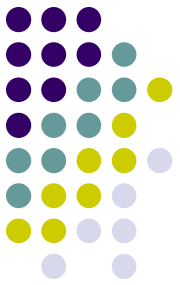
# Contenido

- Listas Simples.
- Listas Doblemente Enlazadas.
- Iteradores.
- Listas Múltiples.



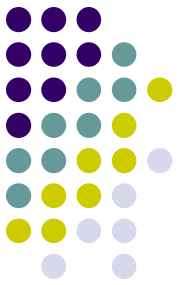
Listas Enlazadas

# LISTAS SIMPLES



# Consideraciones

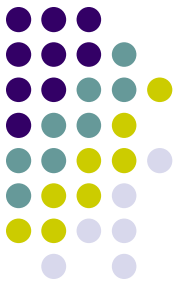
- Desventajas de los arreglos:
  - En un **arreglo no ordenado**, la **búsqueda** es lenta.
  - En un **arreglo ordenado** la **inserción** es lenta.
  - En **ambos** la **eliminación** es lenta.
  - Además, el **tamaño** del arreglo **no puede ser modificado** después de ser creado.
- Las listas enlazadas resuelven algunos de estos problemas.



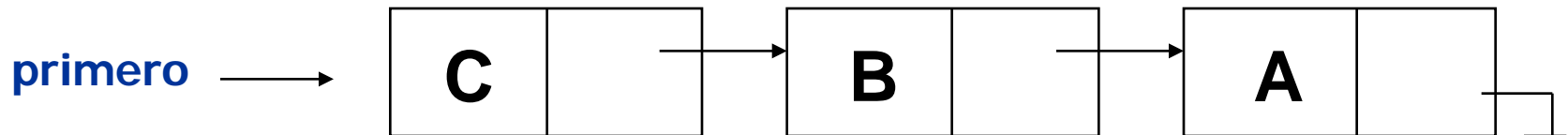
# Listas Enlazadas

- Las listas enlazadas es un mecanismo adecuado para utilizar en muchos tipos de **base de datos** de propósito general.
- Reemplaza al arreglo como la **base** de las otras estructuras de almacenamiento tales como las pilas y colas.

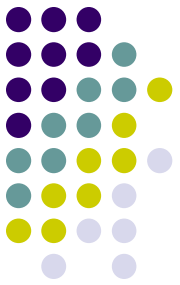
# Ejemplo



Lista enlazada de tres elementos

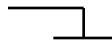


# Inserción



Lista Vacía

**primero** =



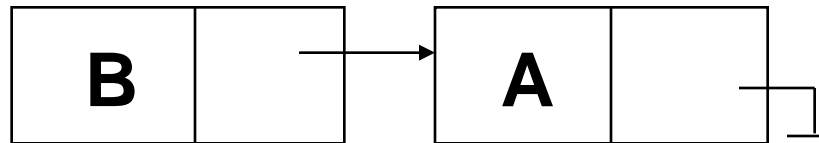
Lista de un solo elemento

**primero** →

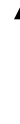


Lista de dos elementos

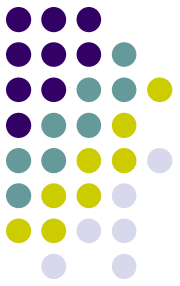
**primero** →



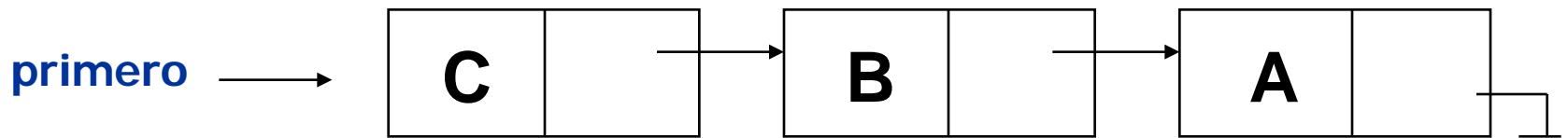
**ultimo** ↑



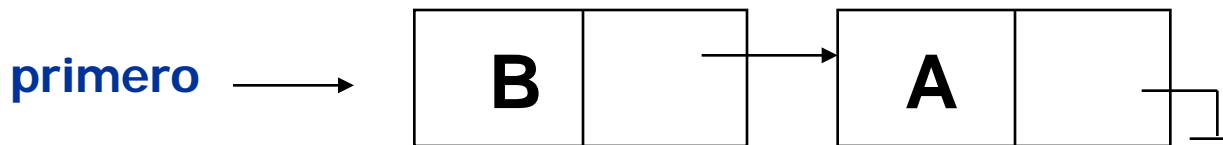
# Eliminación



Lista de tres elementos

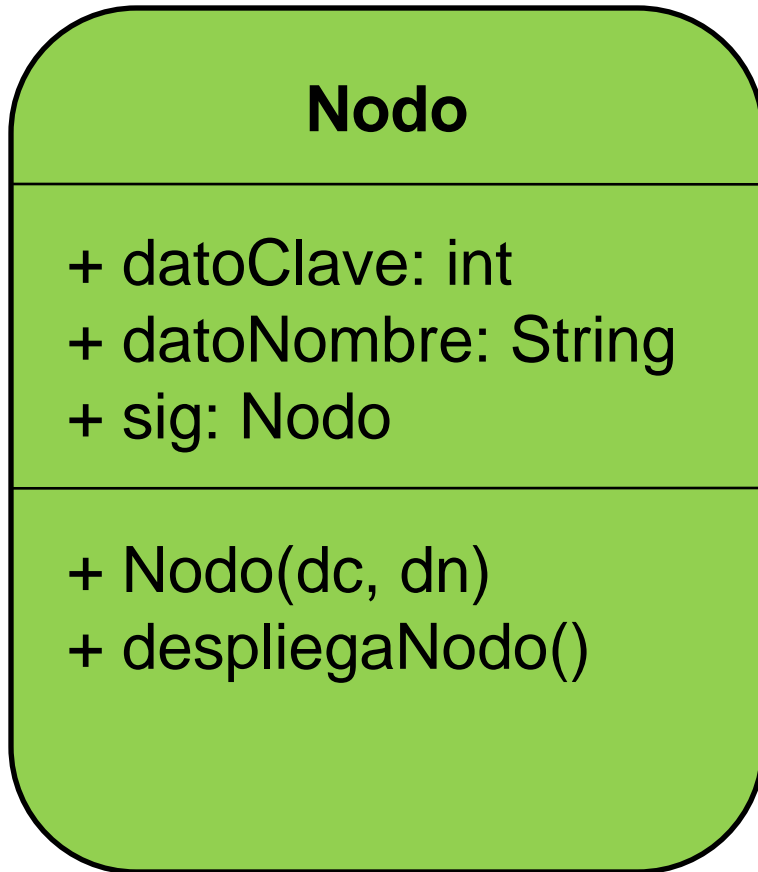


Lista de dos elementos



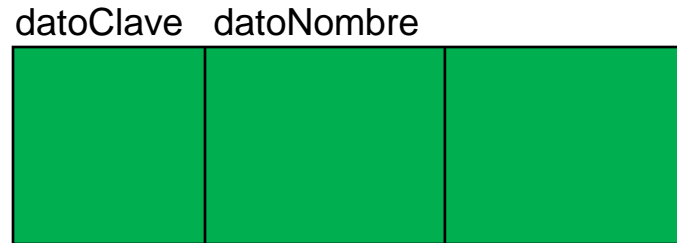


# Nodo

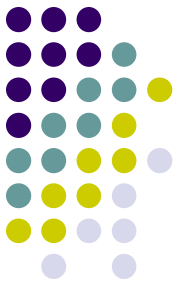


## Estructura del Nodo:

dato sig



```
public int datoClave;
public String datoNombre;
public Nodo sig;
public Nodo(int dc, String dn) {
    datoClave = dc;
    datoNombre = dn;
}
public void despliegaNodo() {
    System.out.print("{ " + datoClave + ", " +
        datoNombre + " } ");
}
```



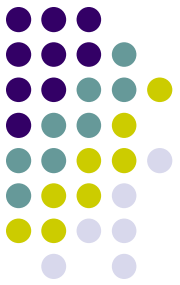
# Lista Enlazada

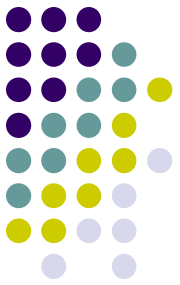
## ListaEnlazada

- primero: Nodo

+ ListaEnlazada()  
+ estaVacia()  
+ insertaPrimero(dc,dn)  
+ eliminaPrimero()  
+ despliegaLista()

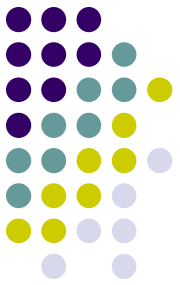
```
public ListaEnlazada() {  
    primero = null;  
}  
  
public boolean estaVacia() {  
    return (primero == null);  
}  
  
public void insertaPrimero(int dc, String dn)  
{  
    Nodo nuevo = new Nodo(dc, dn);  
    nuevo.sig = primero;  
    primero = nuevo;  
}  
  
public Nodo eliminaPrimero() {  
    Nodo temp = primero;  
    primero = primero.sig;  
    return temp;  
}  
  
public void despliegaLista() {  
    Nodo actual = primero;  
    while (actual != null) {  
        actual.despliegaNodo();  
        actual = actual.sig;  
    }  
}
```





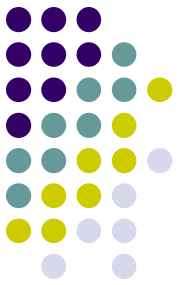
# Ejercicio

- Agregar a la clase ListaEnlazada el método `insertaDespues(dc, dc1, dn1)`, el cual busca un nodo con la clave `dc` e inserta un nuevo `(dc1, dn1)` a continuación de éste nodo.



Listas Enlazadas

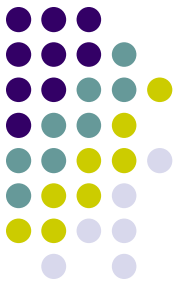
# **LISTAS DOBLEMENTE ENLAZADAS**



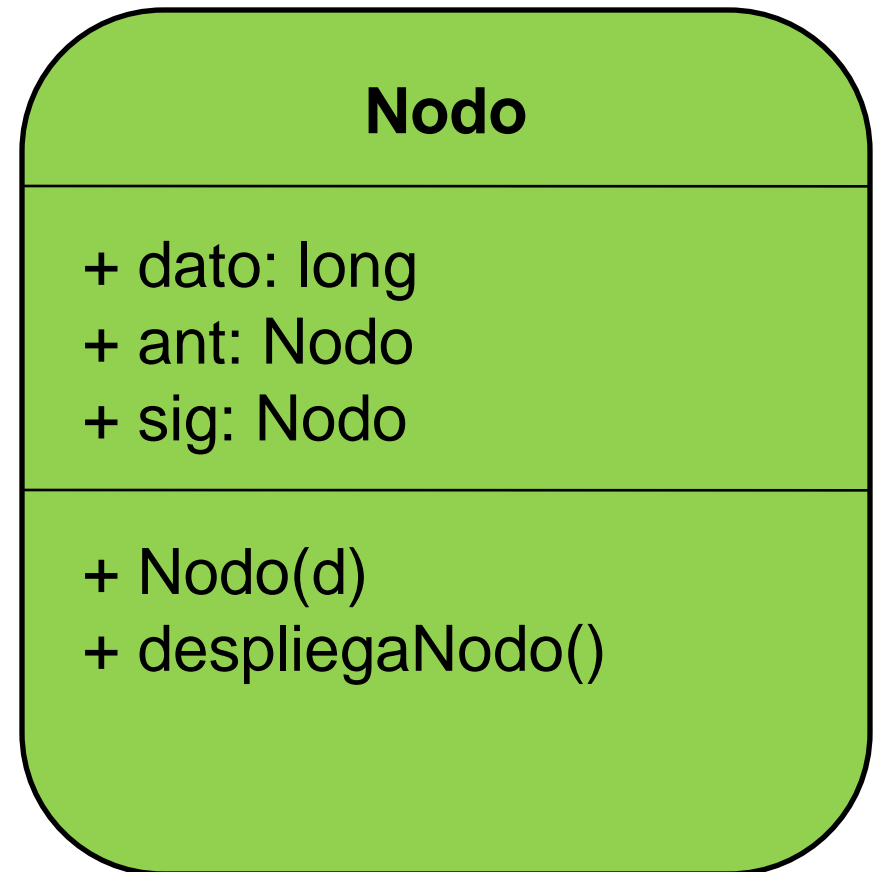
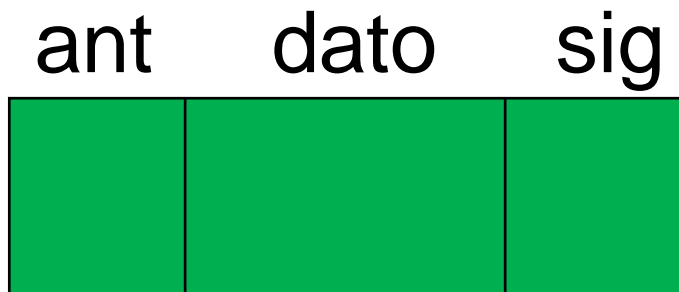
# Consideraciones

- El problema de las listas enlazadas simples es que no se pueden recorrer hacia atrás.
- Las listas dobles nos levantan el anterior problema.

# Nodo



Estructura del Nodo:



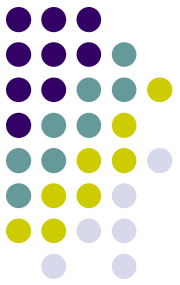
# ListaDoblementeEnlazada

- primero: Nodo
- ultimo: Nodo

- + ListaDoblementeEnlazada()
- + estaVacia()
- + insertaInicio(d)
- + insertaFinal(d)
- + eliminaInicio()
- + eliminaFinal()
- + insertaDespues(clave,d)
- + eliminaClave(clave)
- + despliegaAdelante()
- + despliegaAtras()



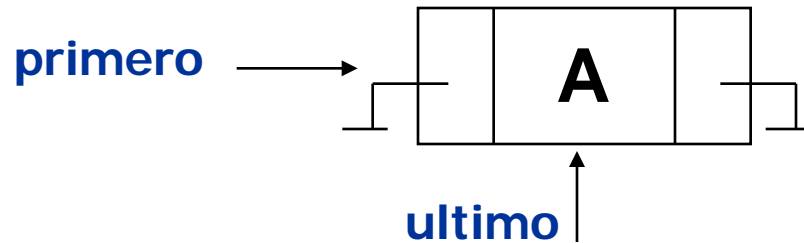
# Inserción



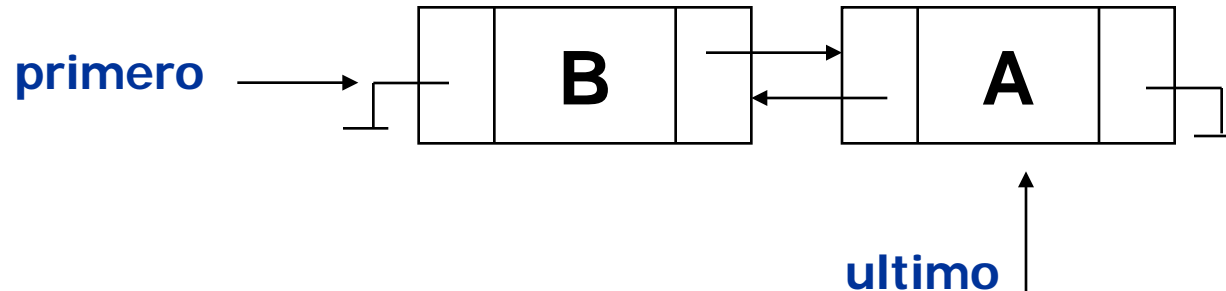
Lista Vacía

**primero = ultimo =** 

Lista de un solo elemento



Lista de dos elementos

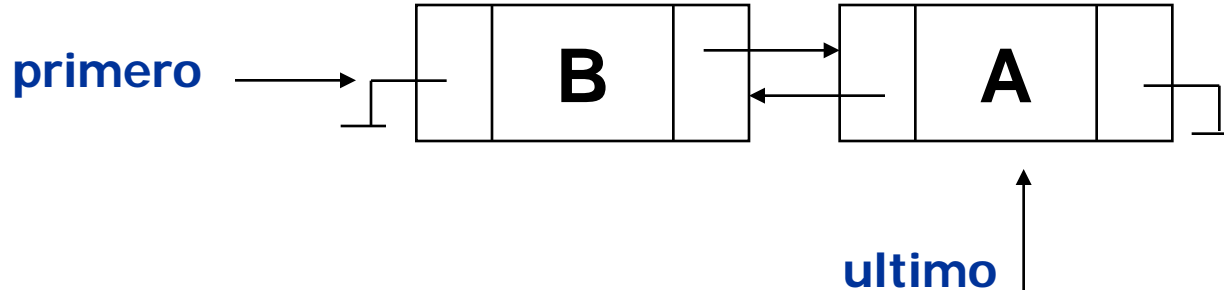




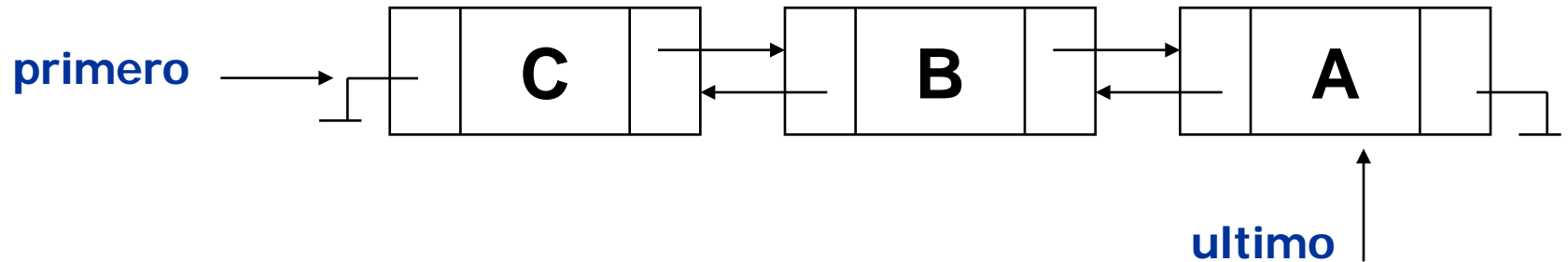
# Inserción



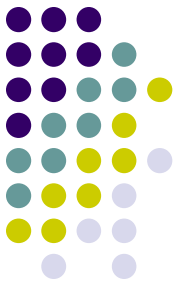
Lista de dos elementos



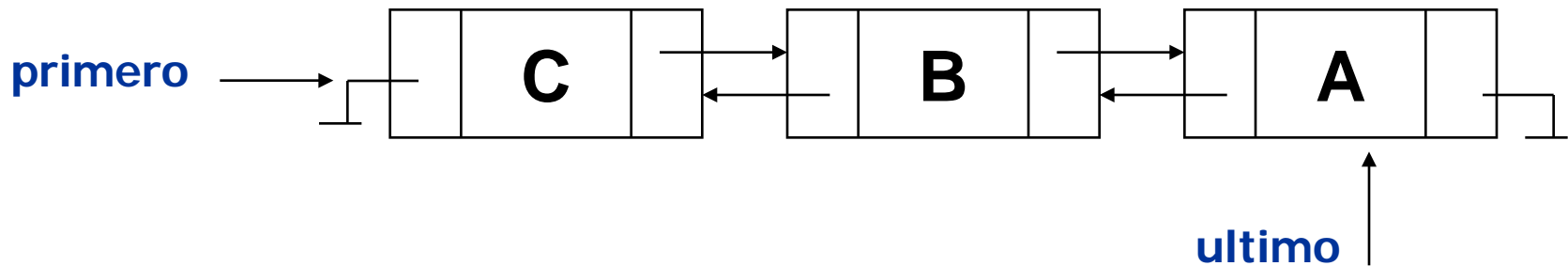
Lista de tres elementos



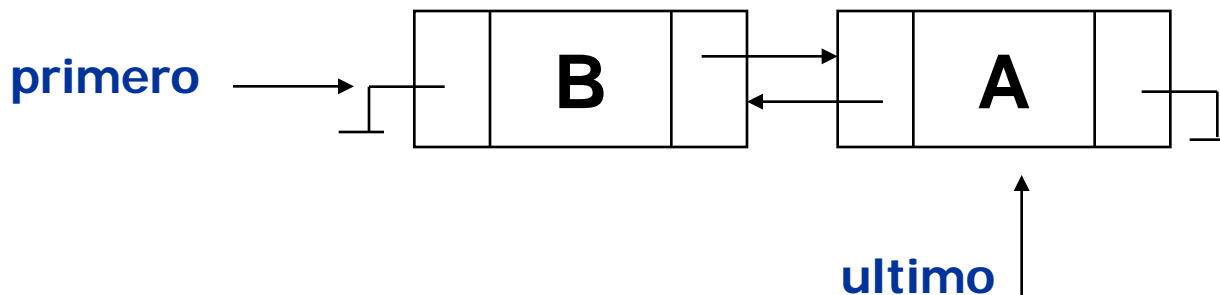
# Eliminación



Lista de tres elementos



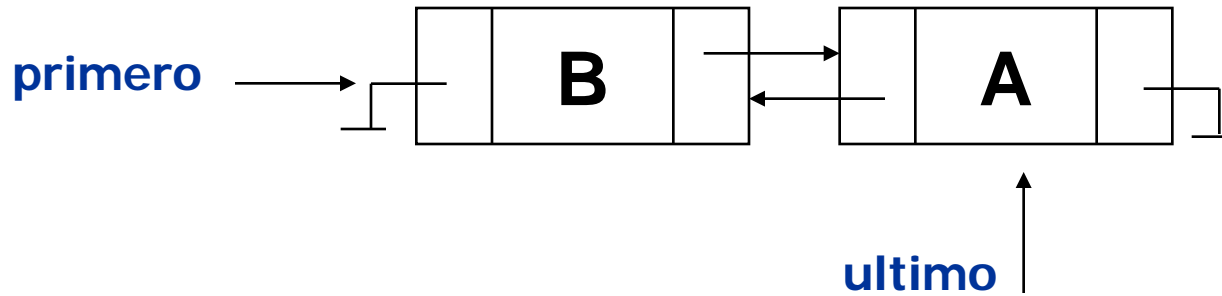
Lista de dos elementos



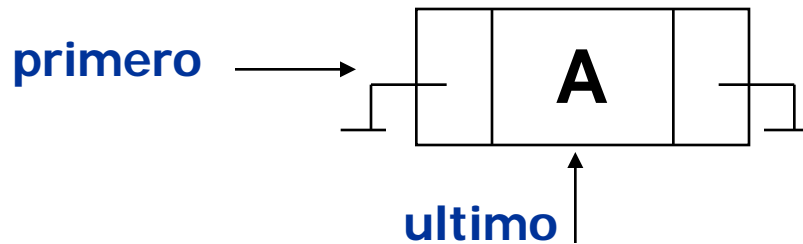
# Eliminación



Lista de dos elementos



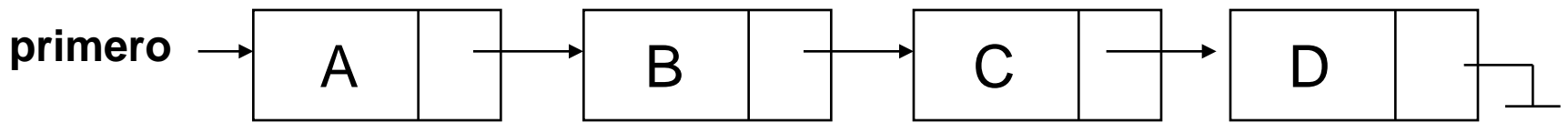
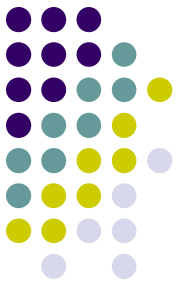
Lista de un solo elemento



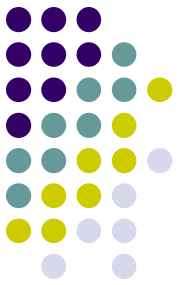
Lista Vacía

**primero = ultimo =** 

# Iteradores



- ¿Cómo ubicar uno o más nodos que tienen ciertas características?
  - **Solución 1:** Recorrer toda la lista y visitar uno a uno los nodos.
  - **Solución 2:** Permitir al usuario crear **referencias** a estos nodos. Cada referencia es un objeto de nombre **iterador**. Puede crearse varios iteradores sobre una lista enlazada.



# Iterador (cont.)

## ListaEnlazada

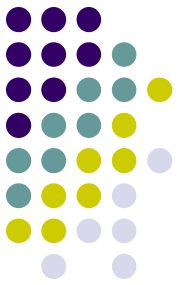
- primero: Nodo

- + ListaEnlazada()
- + getPrimero()
- + setPrimero(nodo)
- + estaVacia()
- + getIterador()
- + despliegaLista()

## Nodo

- + dato: long
- + sig: Nodo

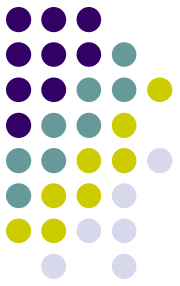
- + Nodo(d)
- + despliegaNodo()



## IteradorLista

- actual: Nodo
  - anterior: Nodo
  - lista: ListaEnlazada
- 
- + IteradorLista(lista)
  - + reset()
  - + estaAlFinal()
  - + sigNodo()
  - + getActual()
  - + insertaDespues(clave,d)
  - + insertaAntes(clave)
  - + eliminaActual()

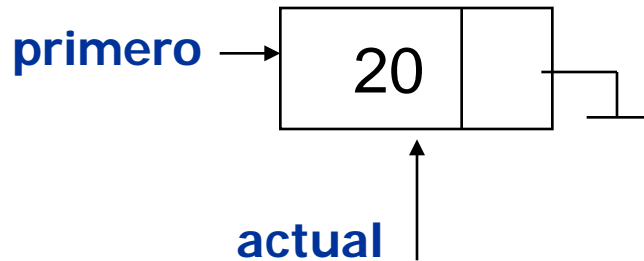
# Applterador.java



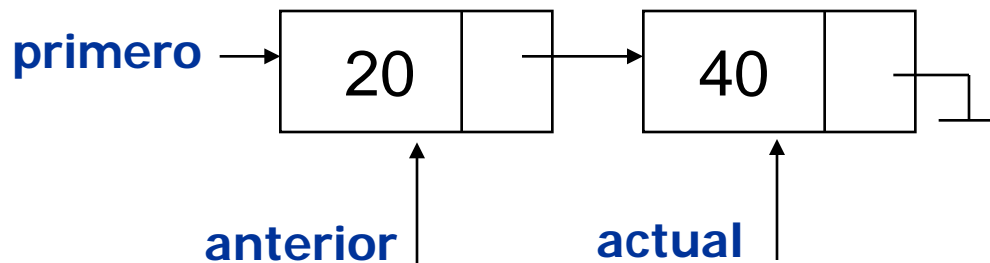
ListaEnlazada.java

**primero** = 

insertaDespues(20)



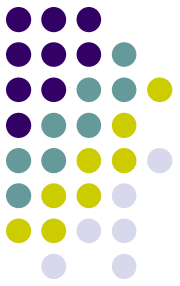
insertaDespues(40)



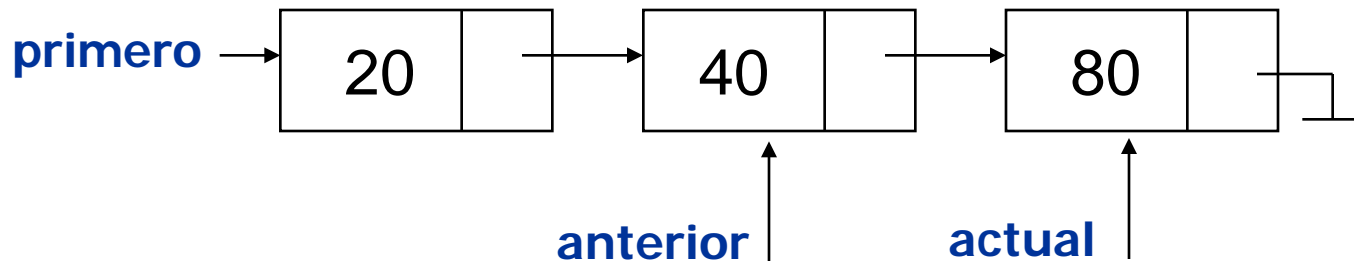
IteradorLista.java

**actual** = **anterior** = 

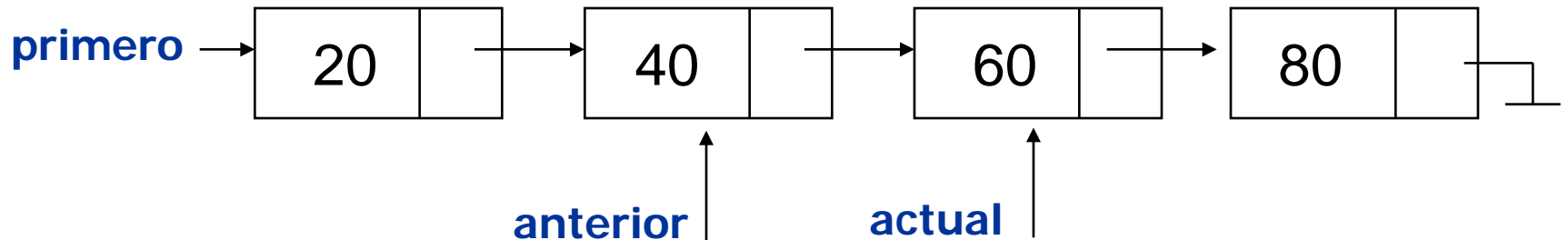
# Applterador.java (cont.)



insertaDespues(80)



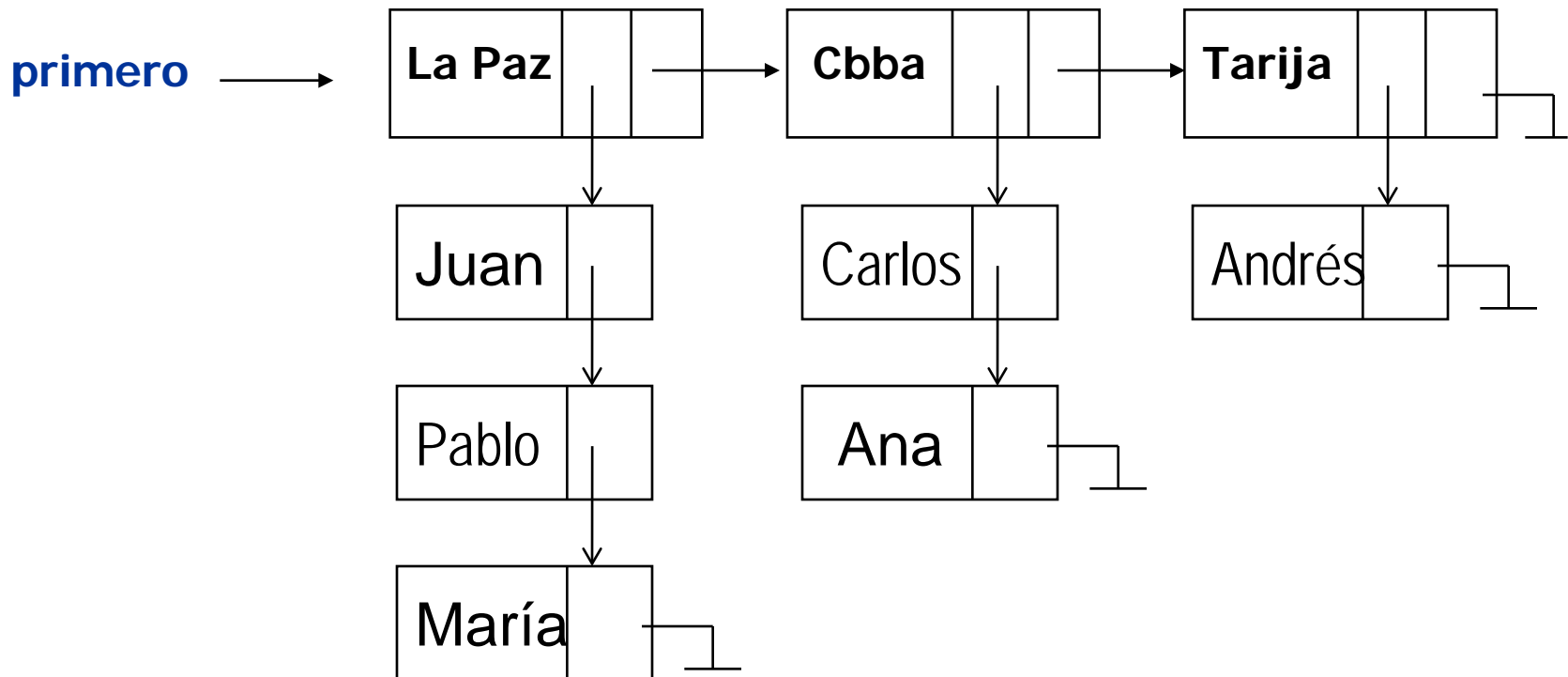
insertaAntes(60)

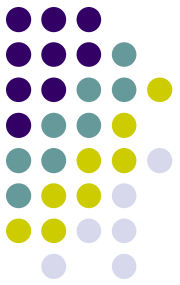


**reset:**      **actual = primero**



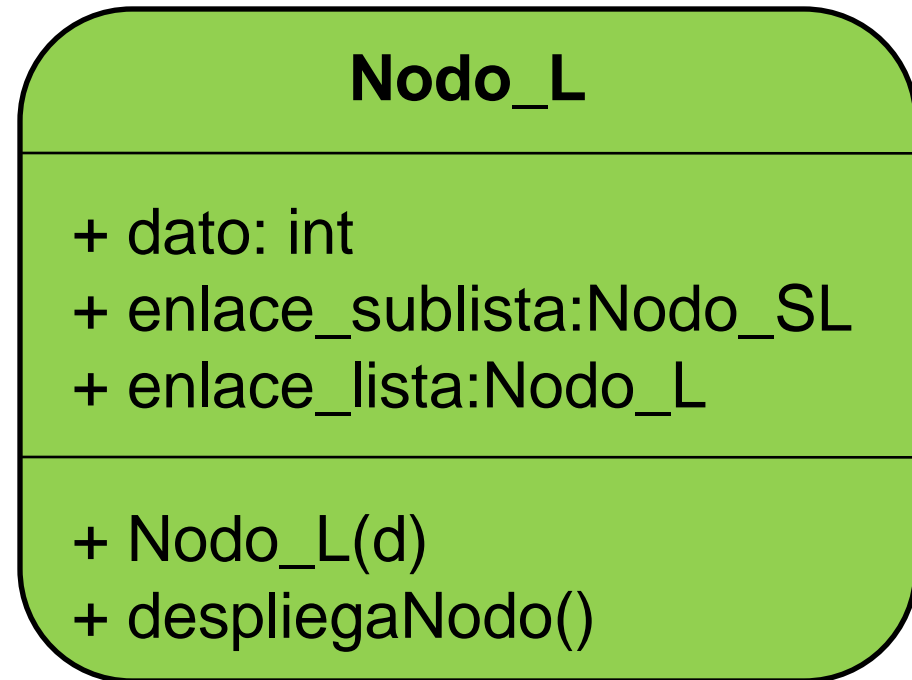
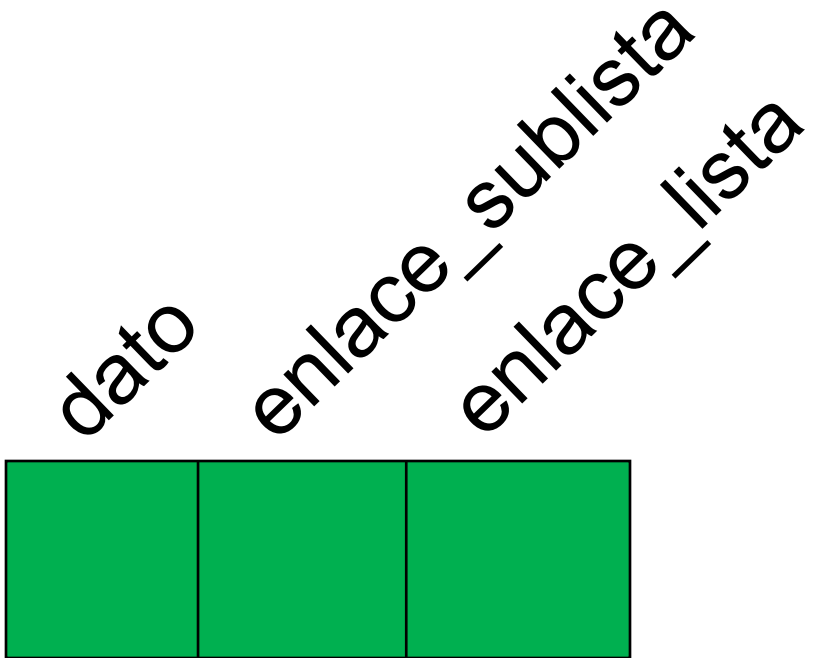
# Listas Múltiples

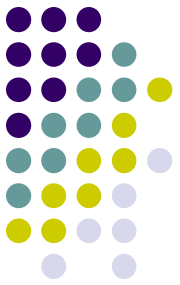




# Listas Múltiples (cont.)

Estructura del Nodo\_L:

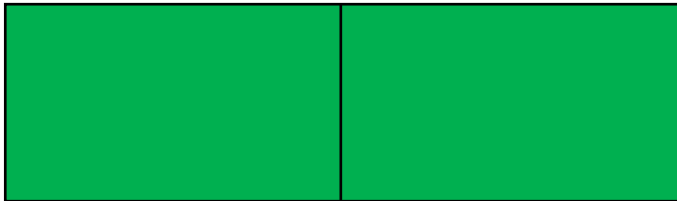




# Listas Múltiples (cont.)

Estructura del Nodo\_SL:

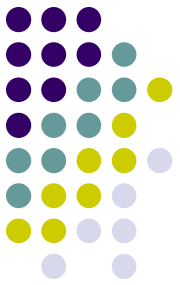
dato      sig



**Nodo\_SL**

+ dato: int  
+ sig: Nodo\_SL

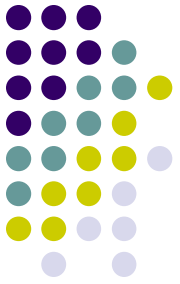
+ Nodo\_SL(d)  
+ despliegaNodo()



# Listas Múltiples (cont.)

## MultiLista

- primero: Nodo\_L
  - primeraVez: boolean
- 
- + MultiLista()
  - + estaVacia()
  - + inserta(d,d1)
  - + consultaNodo(d)
  - + consultaSubNodo(d,d1)
  - + despliegaSubLista(nodo)



Listas Enlazadas

**GRACIAS**