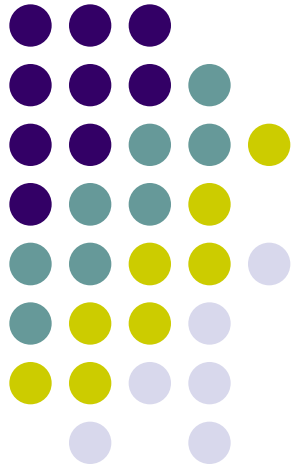
Listas Enlazadas

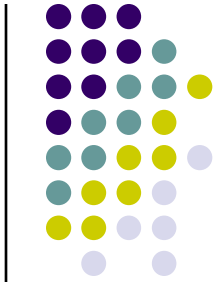
Estructura de Datos

Jhonny Felípez Andrade

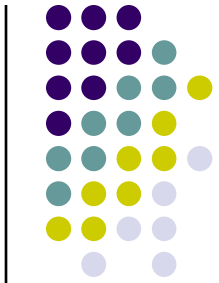
jrfelizamigo@yahoo.es



Contenido



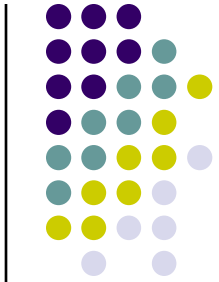
- Listas Simples.
- Listas Doblemente Enlazadas.
- Iteradores.
- Listas Múltiples.



Listas Enlazadas

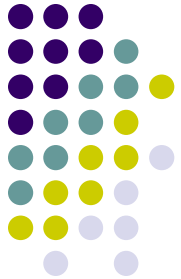
LISTAS SIMPLES

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.

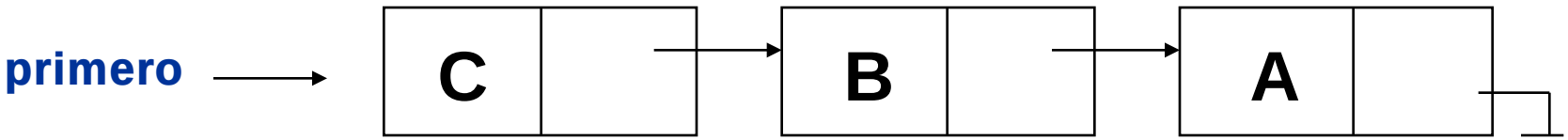
Listas Enlazadas



- Las listas enlazadas es un mecanismo adecuado para utilizar en muchas 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

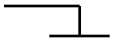
Lista enlazada de tres elementos



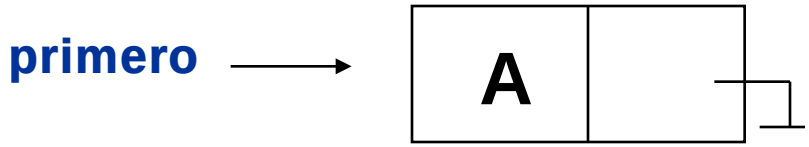
Inserción

Lista Vacía

primero =



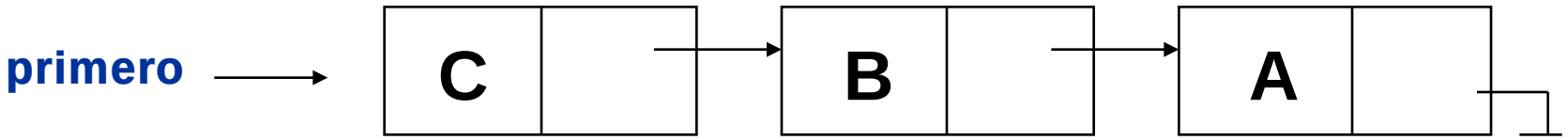
Lista de un solo elemento



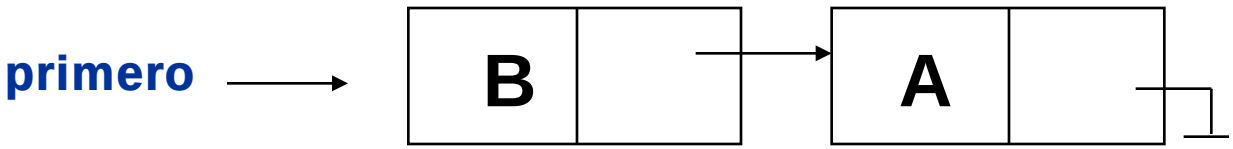
Lista de dos elementos

Eliminación

Lista de tres elementos



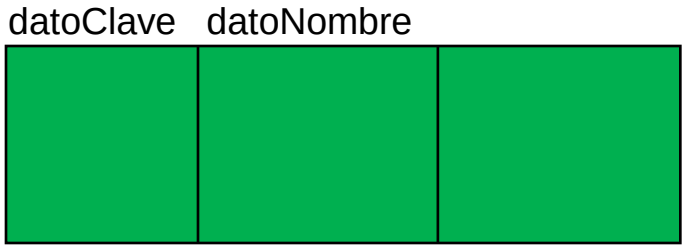
Lista de dos elementos

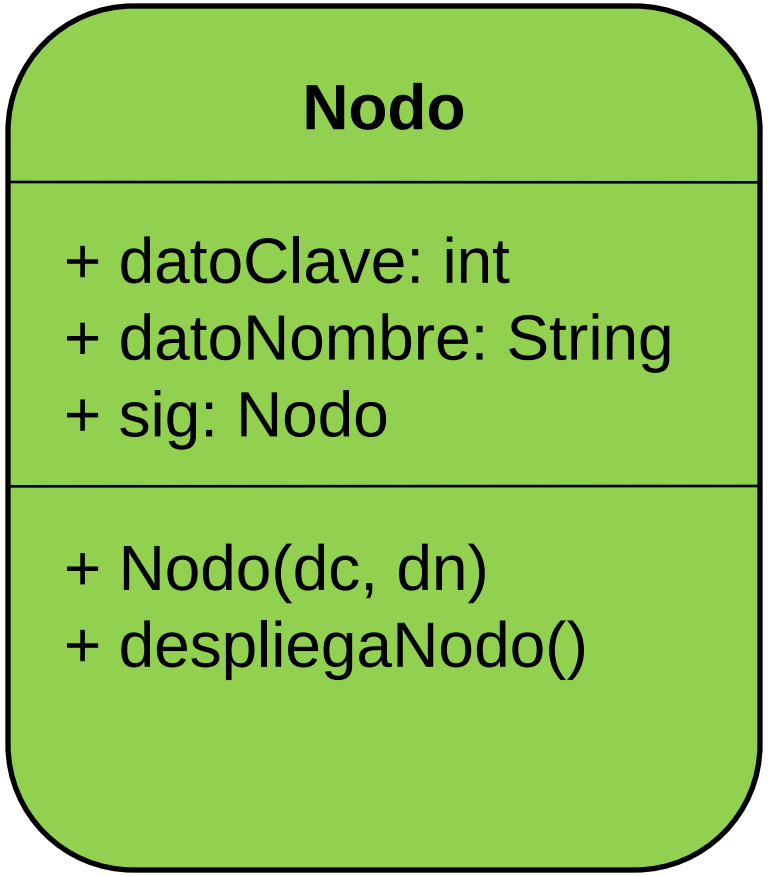


Nodo

Estructura del Nodo:

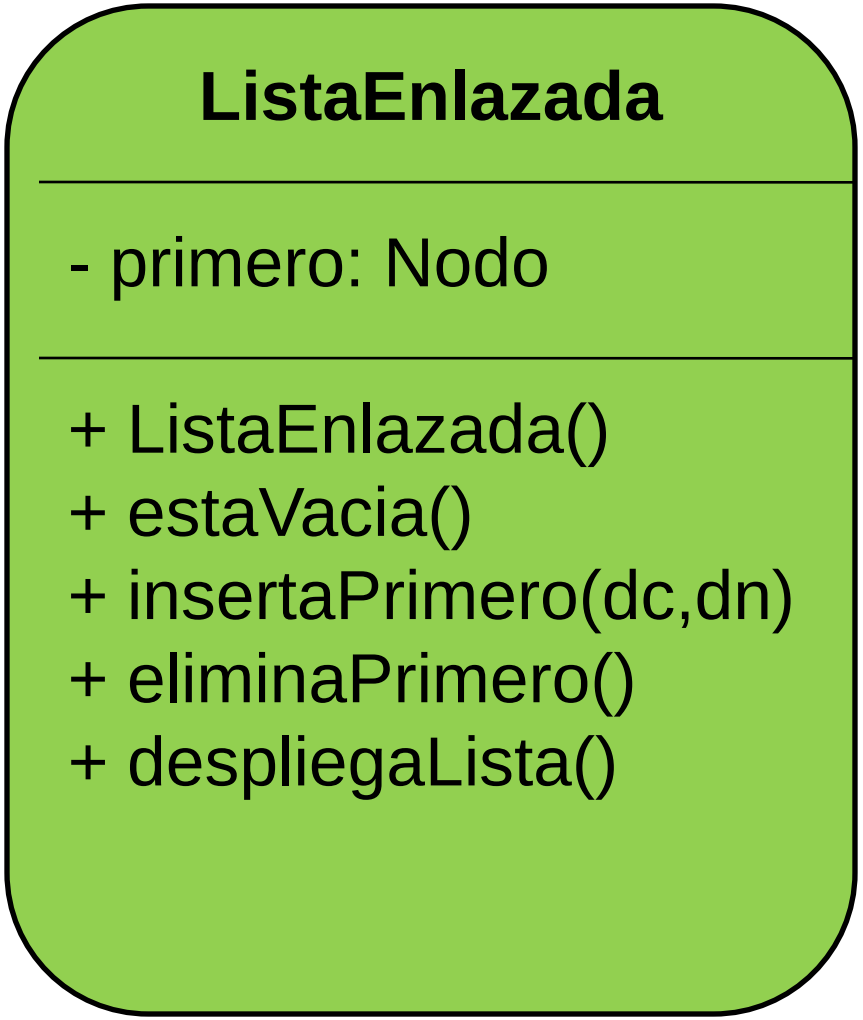
dato sig





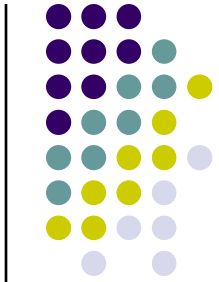

```
public int datoClave;
public String datoNombre;
public Nodo sig;
public Nodo(int dc, String dn) {
   datoClave = dc;
   datoNombre = dn;
}
public void despliegaNodo() {
  System.out.print("{" + datoClave + ", " +
datoNombre + "} ");
}
```

Lista Enlazada




```
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;
   }
}
```

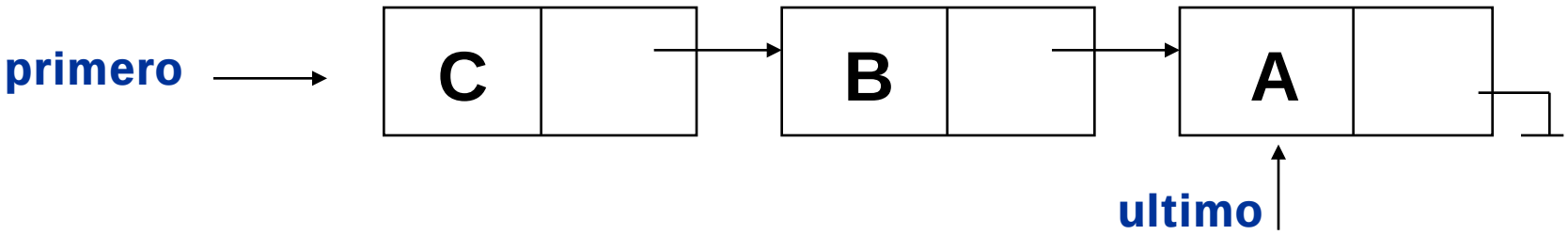
Ejercicio



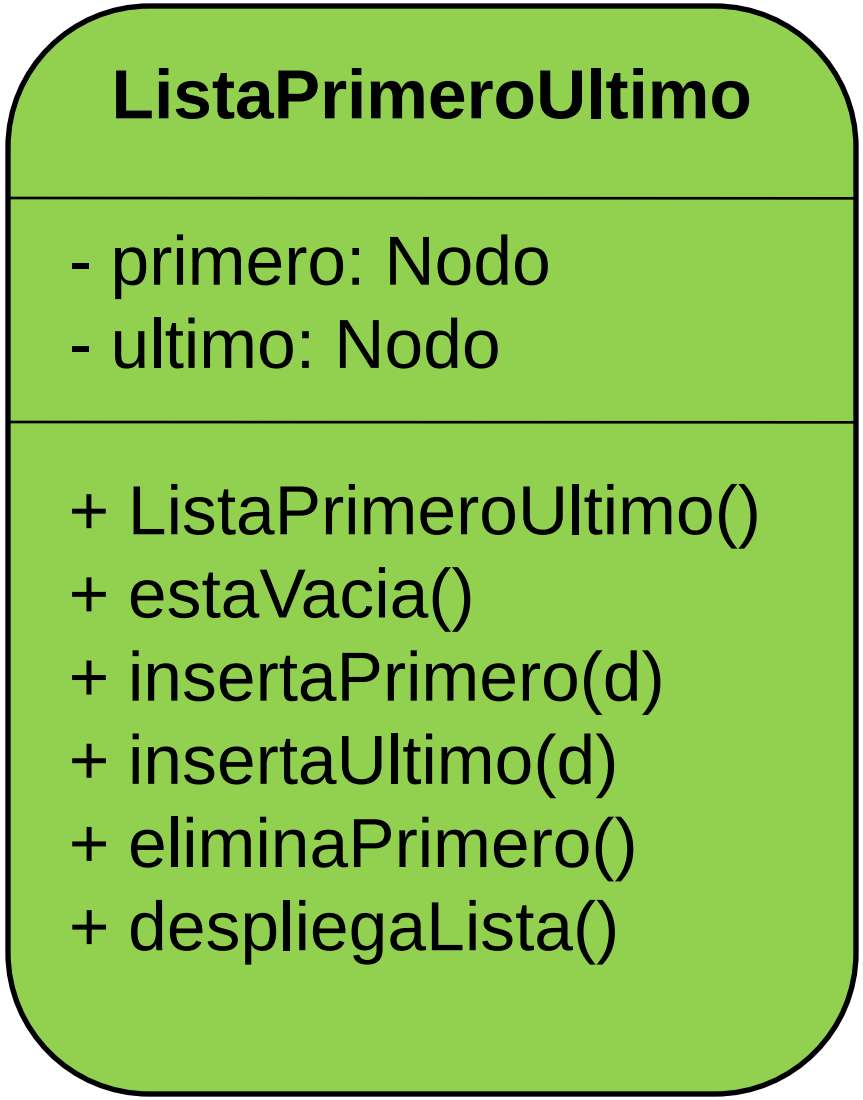
- Agregar a la clase ListaEnlazada el método insertaDespues(dc, dc1, dn1), el cual busca un nodo con la clave dc e inserta un nuevo (dc1, dn1) a continuación de éste nodo.

Lista enlazada con dos extremos

Lista con tres elementos

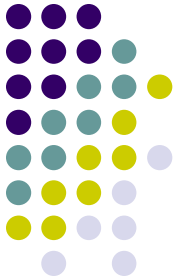


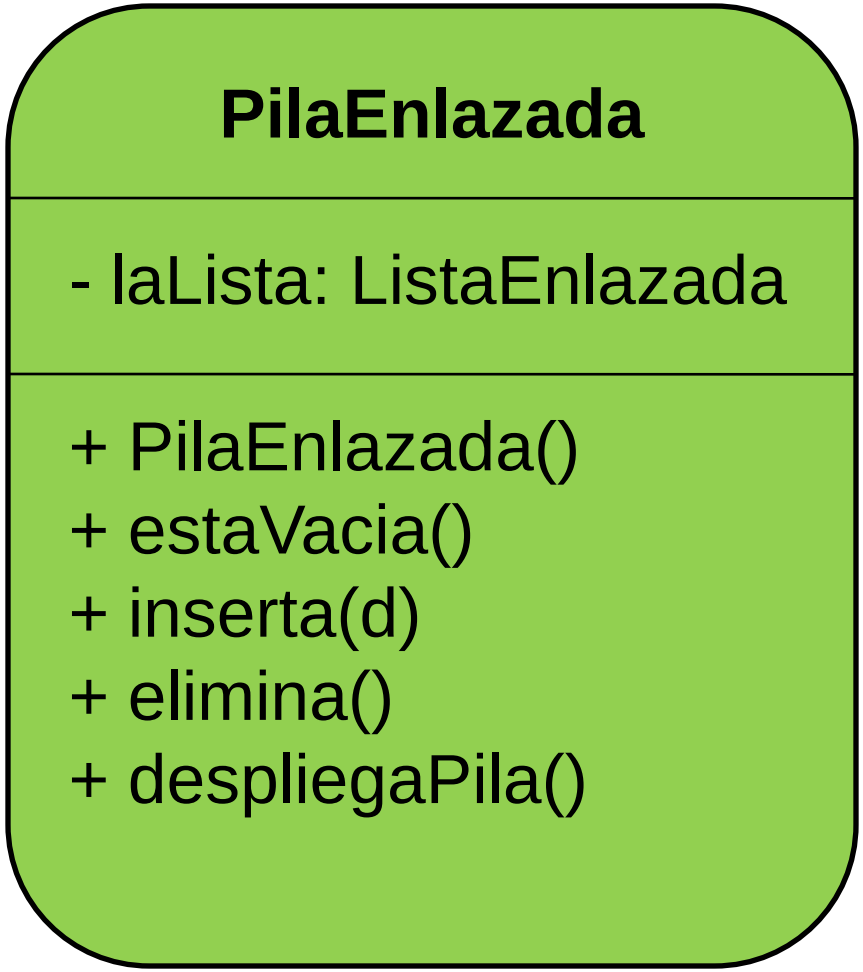
Lista con dos Extremos




```
public ListaPrimeroUltimo() {
   primero = null;   ultimo=null;
}
public boolean estaVacia() {
   return (primero == null);
}
public void insertaPrimero(int d) {
   Nodo nuevo = new Nodo(d);
   if (estaVacia())  ultimo = nuevo;
   nuevo.sig = primero;
   primero = nuevo;
}
public void insertaUltimo(int d) {
   Nodo nuevo = new Nodo(d);
   if (estaVacia())  primero = nuevo;
   else   ultimo.sig = nuevo;
   ultimo = nuevo;
}
public int eliminaPrimero() {
   int temp = primero.dato;
   if (primero.sig == null) ultimo = null;
   primero = primero.sig;
   return temp;
}
```

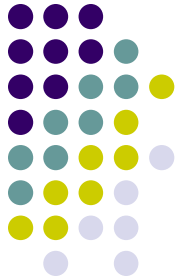
Pila Enlazada






```
public PilaEnlazada() {
   laLista = new ListaEnlazada();
}
public boolean estaVacia() {
   return (laLista.estaVacia());
}
public void inserta(int d) {
   laLista.insertaPrimero(d);
}
public int elimina() {
   return laLista.eliminaPrimero();
}
public void despliegaPila() {
   System.out.print("Pila (arriba-->abajo): ");
   laLista.despliegaLista();
}
```

Cola Enlazada

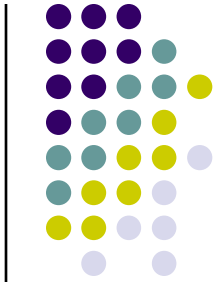



```
public PilaEnlazada() {
   laLista = new ListaPrimeroUltimo();
}
public boolean estaVacia() {
   return (laLista.estaVacia());
}
public void inserta(int d) {
   laLista.insertaUltimo(d);
}
public int elimina() {
   return laLista.eliminaPrimero();
}
public void despliegaCola() {
   System.out.print("Cola (frente-->final): ");
   laLista.despliegaLista();
}
```

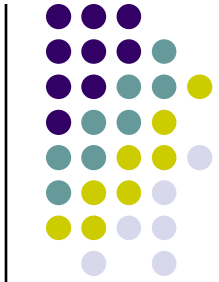
Lista Ordenada


```
public ListaOrdenada() {
   primero = null;
}
public boolean estaVacia() {
   return (primero==null);
}
public void inserta(int d) {
 Nodo nuevo = new Nodo(d);
 Nodo anterior = null;
 Nodo actual = primero;
 while (actual != null && clave > actual.dato) {
       anterior = actual;
       actual = actual.sig;
 }
 if (anterior == null) primero = nuevo;
 else anterior.sig = nuevo;
 nuevo.sig = actual;
}
public Nodo elimina() {
   Nodo temp = primero;
   primero = primero.sig;
   return temp;
}
```

Eficiencia de la listas ordenadas



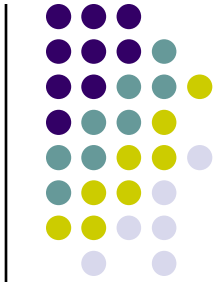
- Un lista ordenada puede ser utilizada eficientemente para ordenar números.
  - Para ello inserte los elementos del arreglo no ordenado a una lista ordenada (ellos se ordenaran automáticamente en la lista). Luego elimine los elementos de la lista llevando nuevamente los elementos al arreglo. El arreglo estará ordenado.
- Como se realiza N*2 copias. La complejidad será log(N).



Listas Enlazadas

LISTAS DOBLEMENTE ENLAZADAS

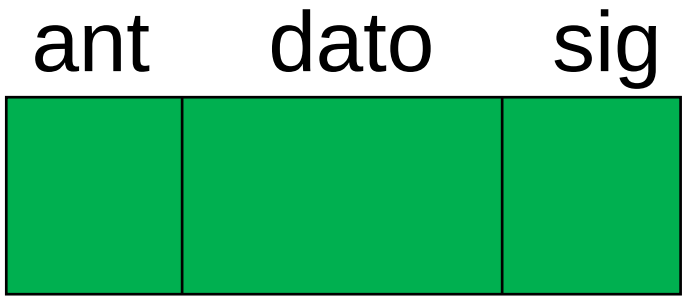
Consideraciones

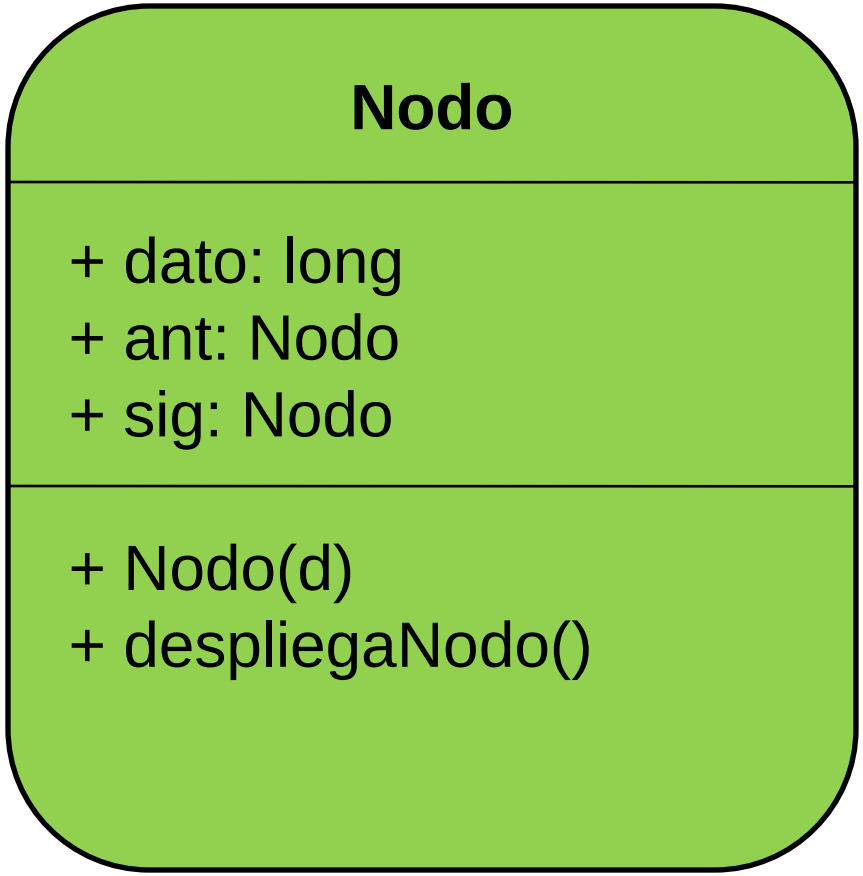


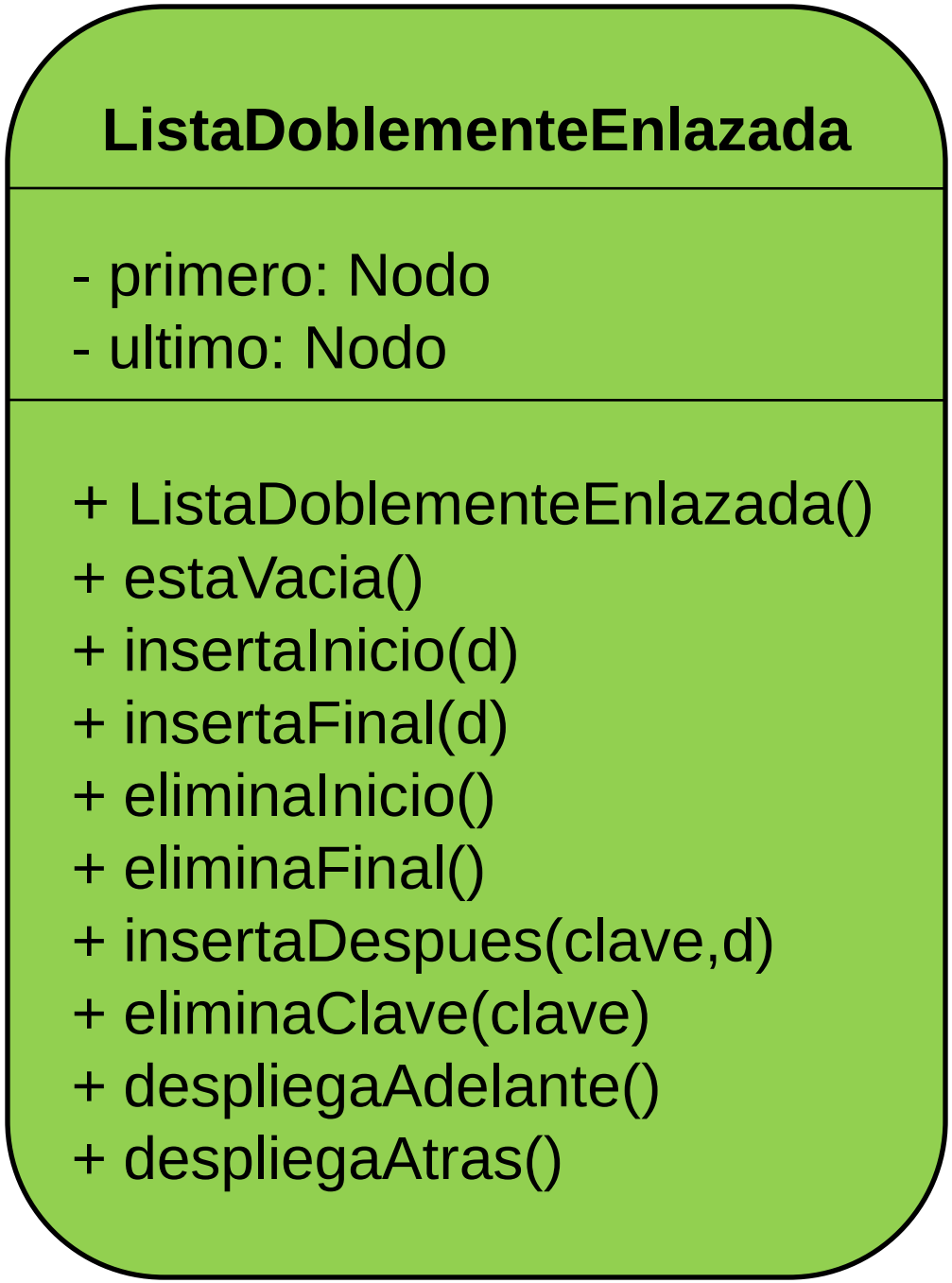
- El problema de las listas enlazadas simples es que no se pueden recorrer hacia atrás.
- Las listas dobles nos levantan el anterior problema.

Nodo

Estructura del Nodo:





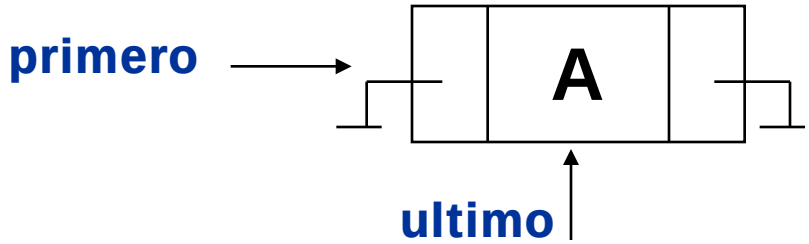



```
public ListaDoblementeEnlazada() {
  primero = null;
  ultimo = null;
}
public boolean estaVacia() {
  return (primero == null);
}
public void insertaInicio(int d) {
  Nodo nuevo = new Nodo(d);
  if (estaVacia()) ultimo = nuevo;
  else primero.ant = nuevo;
  nuevo.sig = primero;
  primero = nuevo;
}
public Nodo eliminaInicio() {
  Nodo temp = primero;
  if (primero.sig == null) ultimo = null;
  else primero.sig.ant = null;
  primero = primero.sig;
  return temp;
}
```

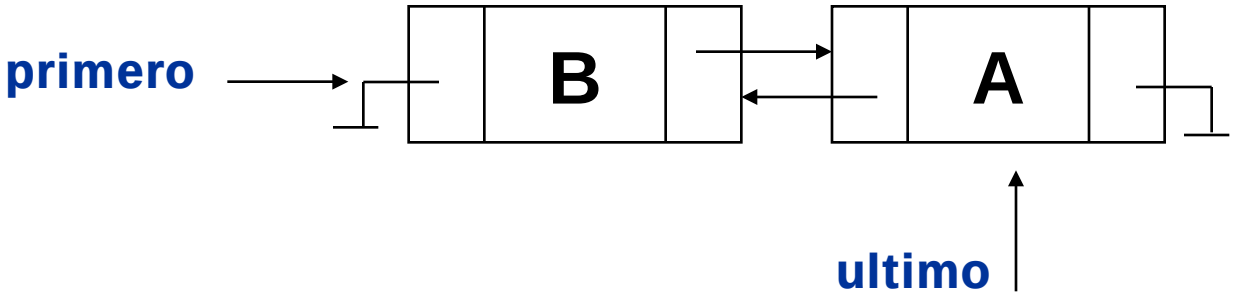
Inserción

Lista Vacía

Lista de un solo elemento

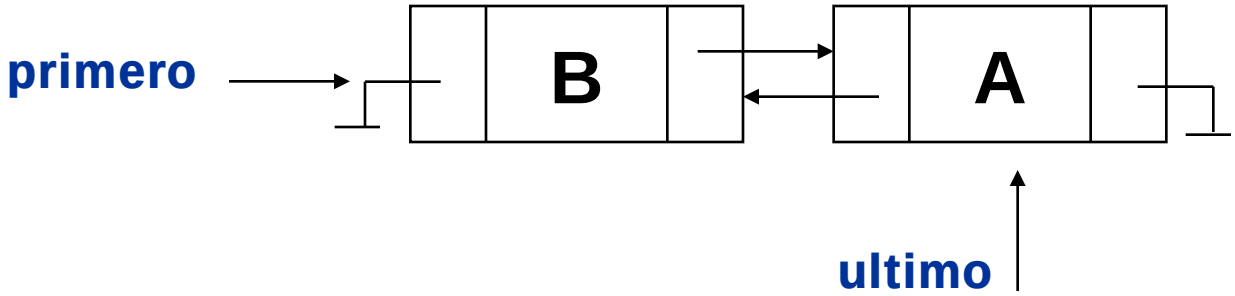


Lista de dos elementos

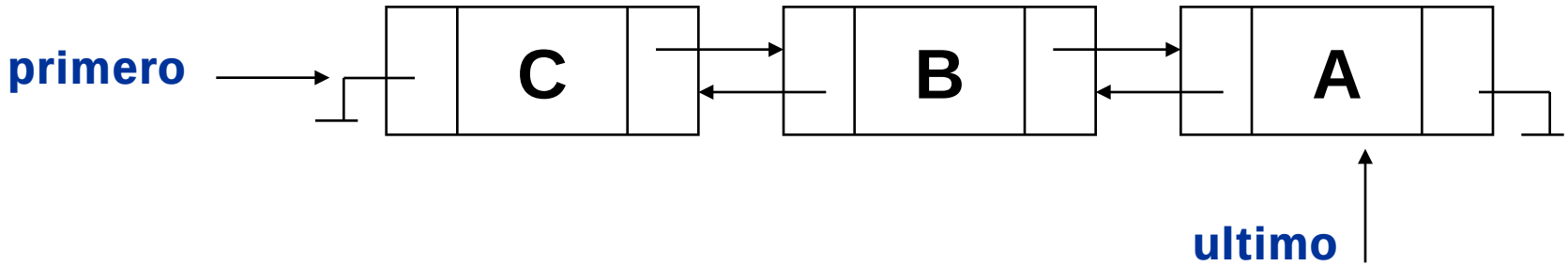


Inserción

Lista de dos elementos

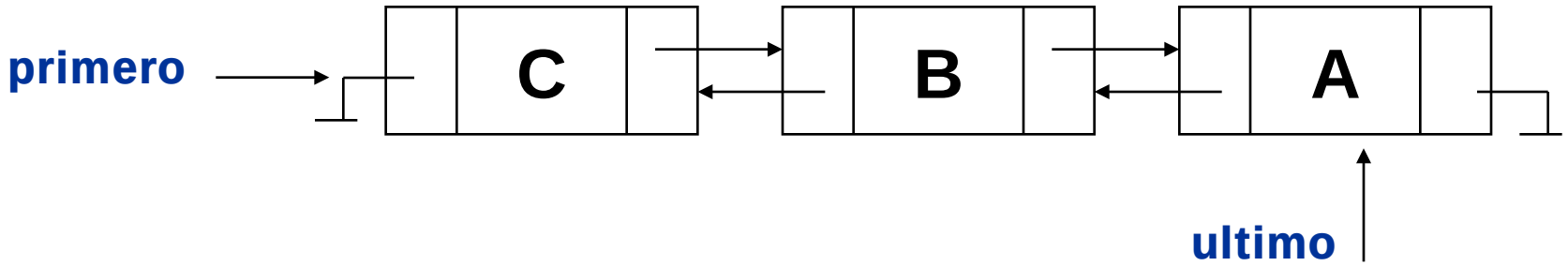


Lista de tres elementos

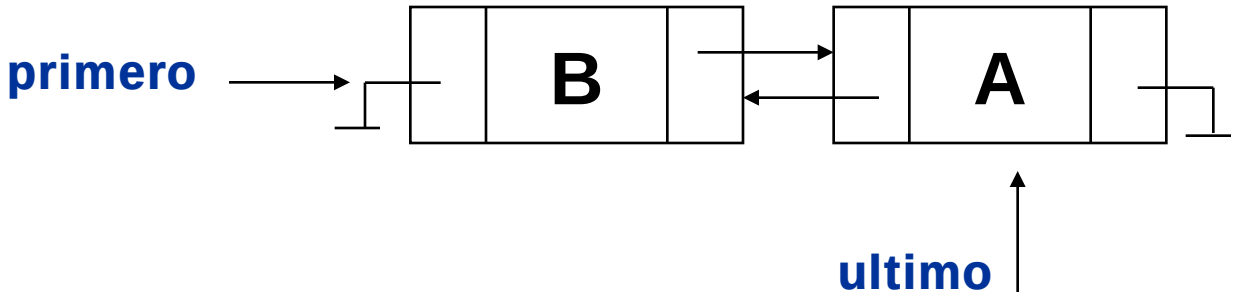


Eliminación

Lista de tres elementos

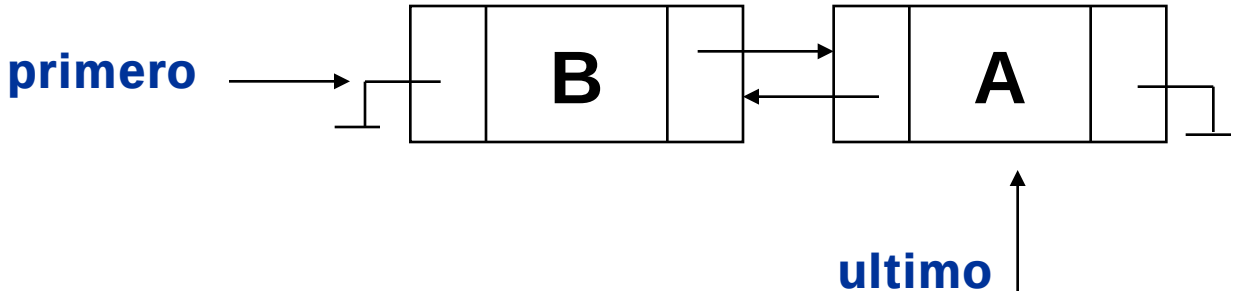


Lista de dos elementos

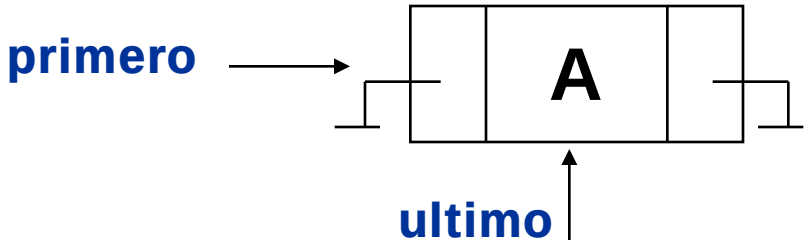


Eliminación

Lista de dos elementos



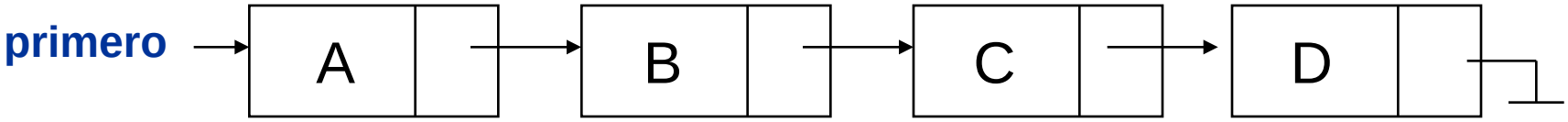
Lista de un solo elemento



Lista Vacía

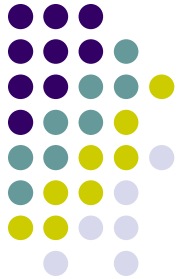
primero = ultimo =

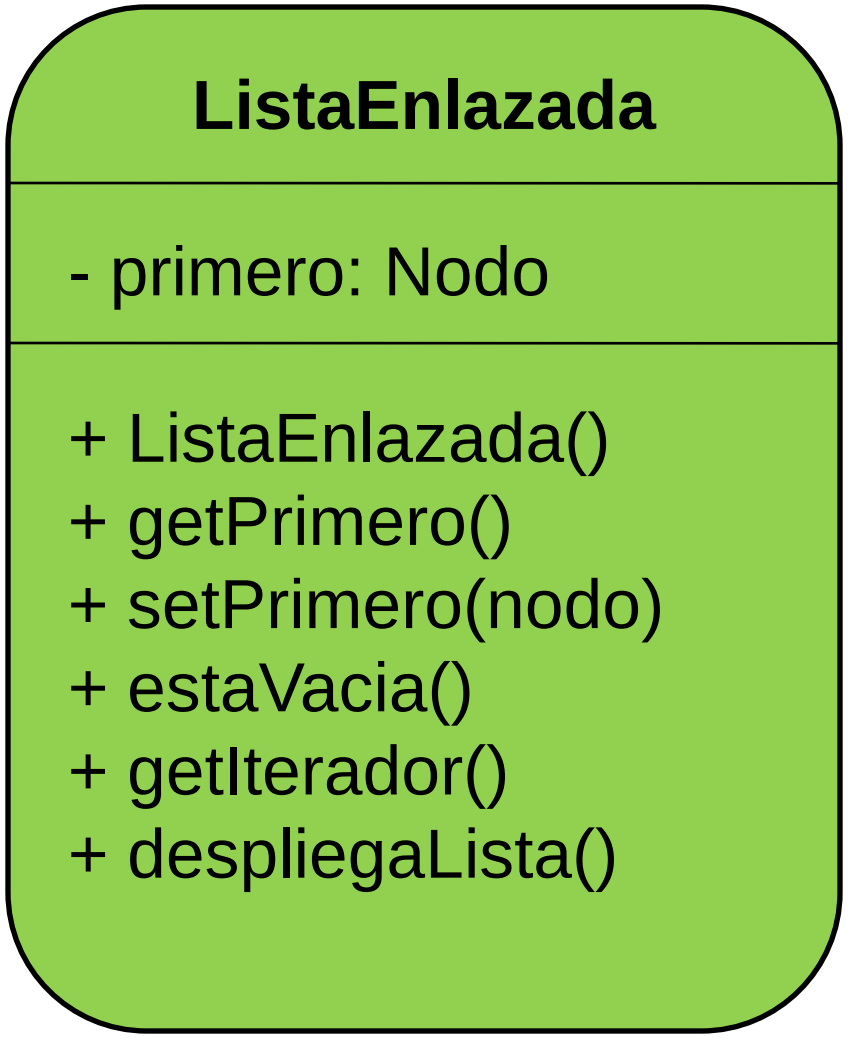
Iteradores

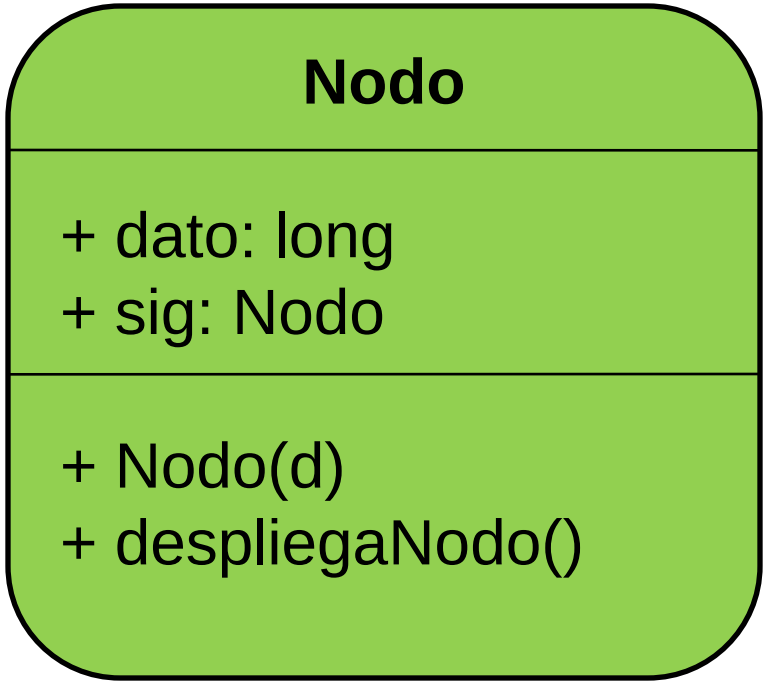


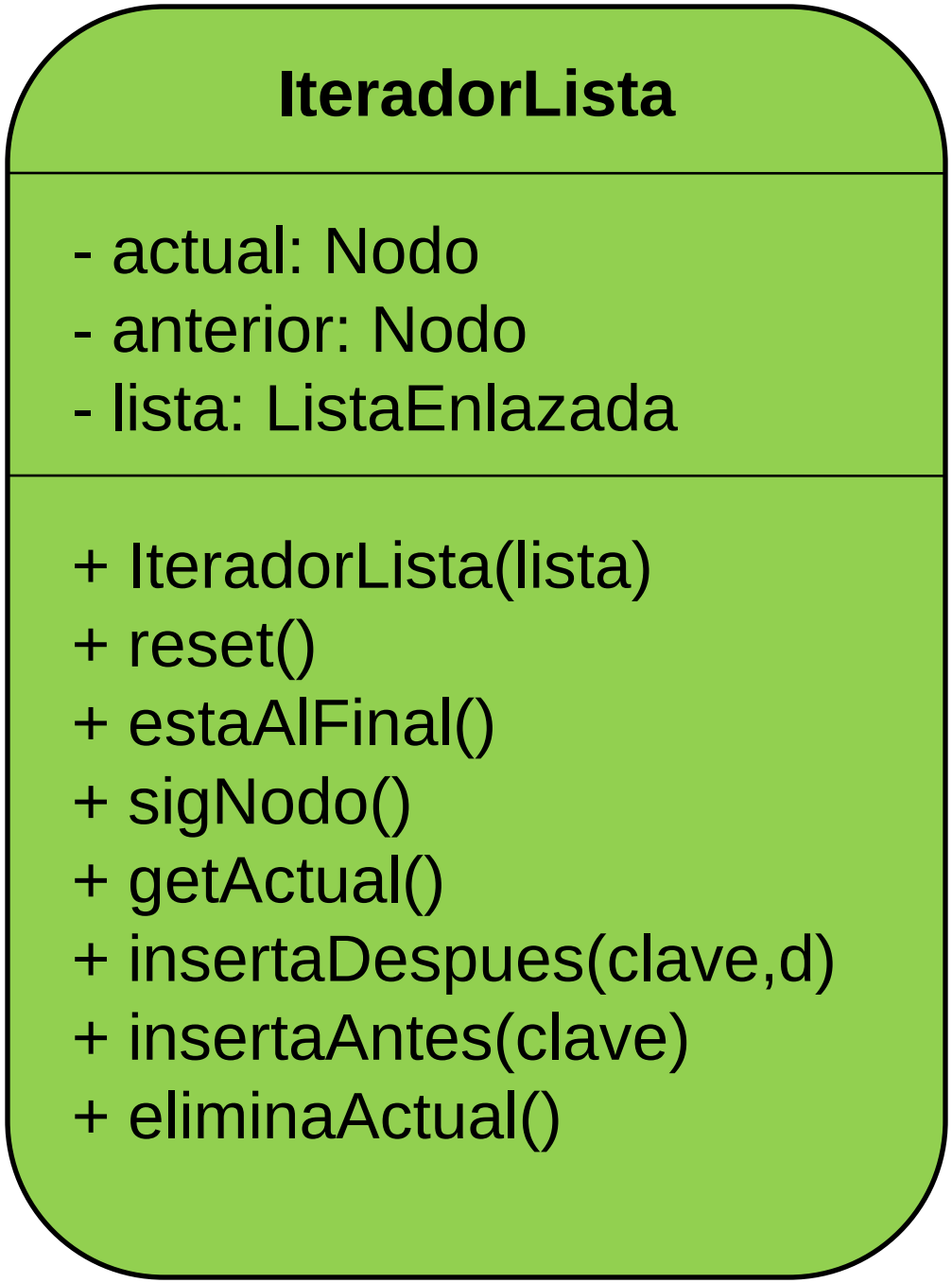
- ¿Cómo ubicar uno o más nodos que tienen ciertas características?
  - Solución 1: Recorrer toda la lista y visitar uno a uno los nodos.
  - Solución 2: Permitir al usuario crear referencias a estos nodos. Cada referencia es un objeto de nombre iterador. Puede crearse varios iteradores sobre una lista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

Iterador (cont.)

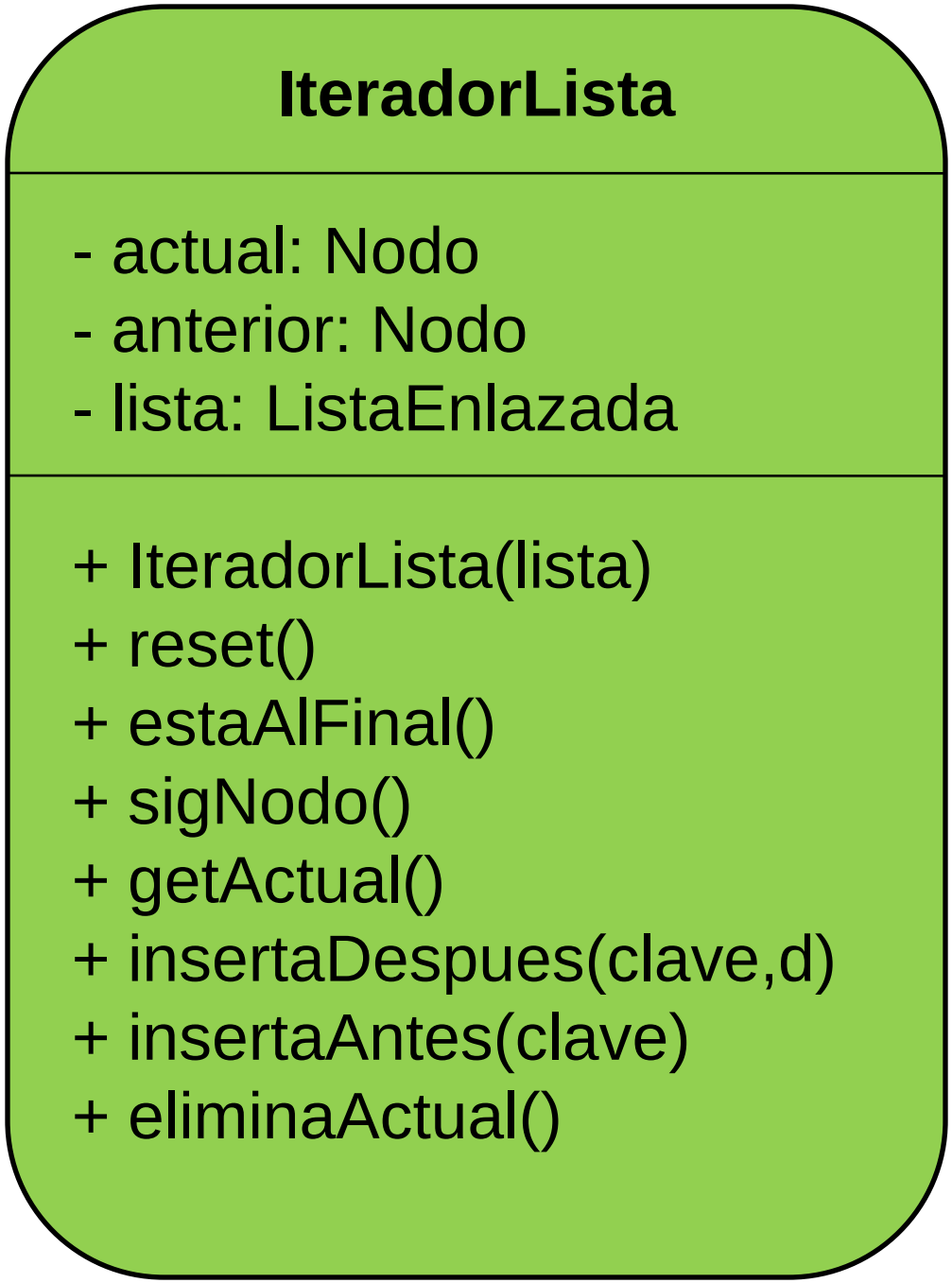









```
public IteradorLista(ListaEnlazada l)
  lista = l;
  reset();
}
public void reset() {
  actual = lista.getPrimero();
  anterior = null;
}
public boolean estaAlFinal() {
  return (actual.sig == null);
}
public void sigNodo() {
  anterior = actual;
  actual = actual.sig;
}
public Nodo getActual() {
  return actual;
}
```




```
public void insertaDespues(int d) {
  Nodo nuevoNodo = new Nodo(d);
  if (lista.estaVacia()) {
    lista.setPrimero(nuevoNodo);
    actual = nuevoNodo;
  } else {
    nuevoNodo.sig = actual.sig;
    actual.sig = nuevoNodo;
    sigNodo();
  }
}
public int eliminaActual() {
  int value = actual.dato;
  if (anterior == null) {
    lista.setPrimero(actual.sig);
    reset();
  } else {
    anterior.sig = actual.sig;
    if (estaAlFinal())  reset();
    else actual = actual.sig;
  }
  return value;
}
```

AppIterador.java

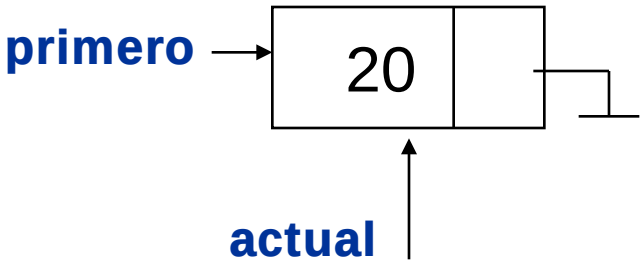
ListaEnlazada.java

primero =

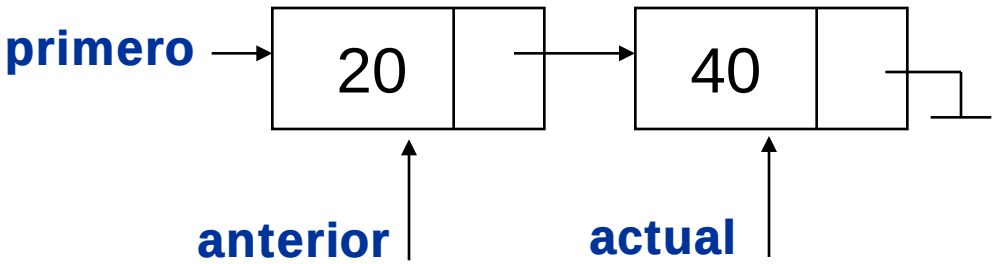
IteradorLista.java

actual = anterior =

insertaDespues(20)

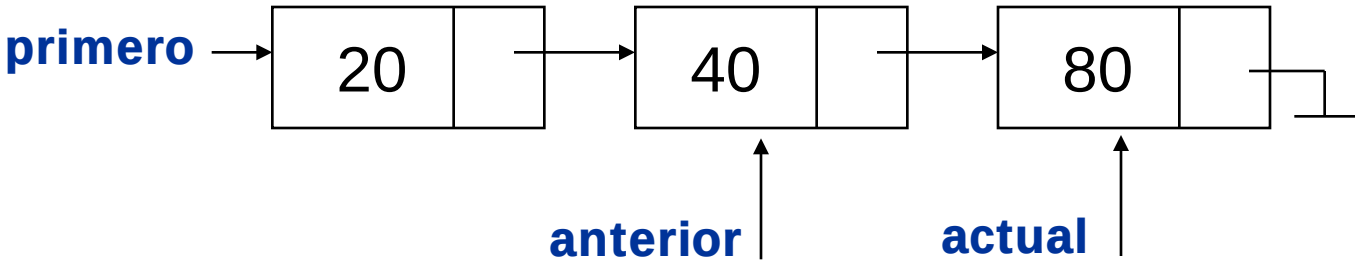


insertaDespues(40)

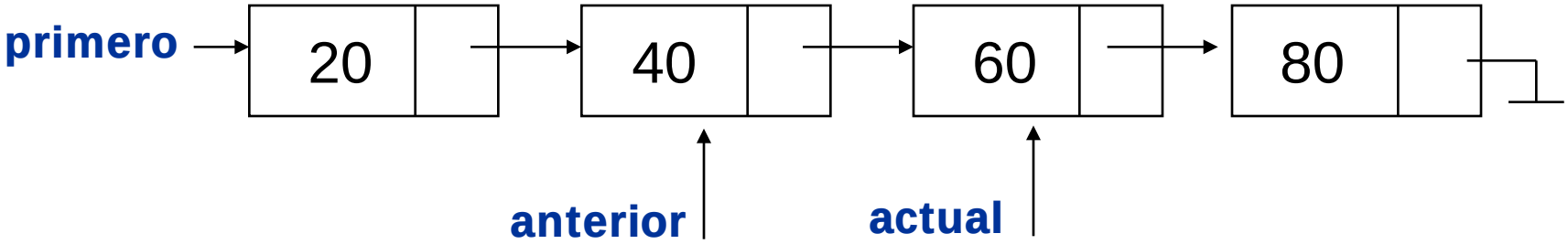


AppIterador.java (cont.)

insertaDespues(80)

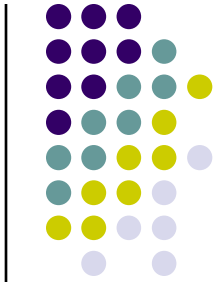


insertaAntes(60)



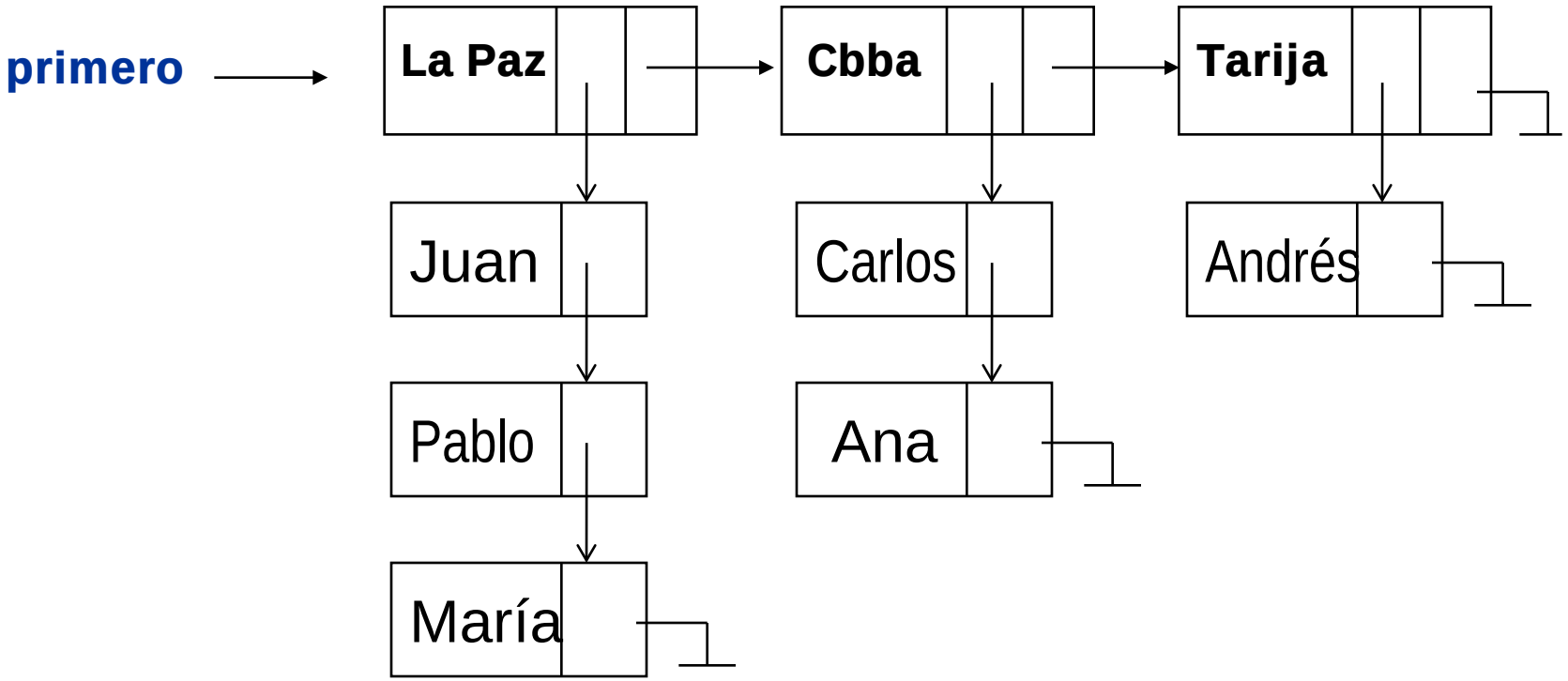
reset: **actual = primero**

Ejercicio



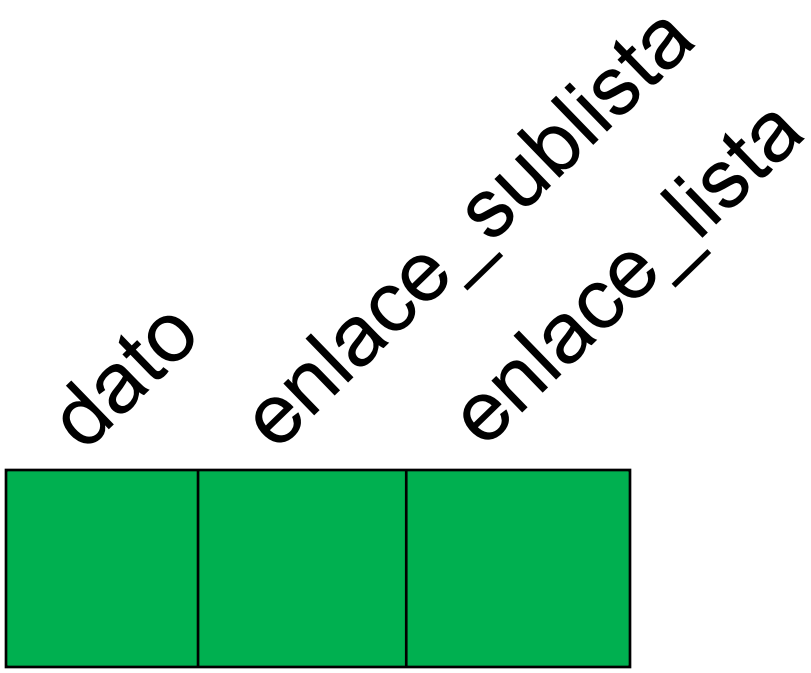
- Agregar a la clase IteradorLista los métodos:
  - busca(d), el cual retorna el nodo dada la clave d.
  - reemplaza(d,d1), el cual reemplaza la clave de los nodos que tienen ‘d’ por ‘d1’.

Listas Múltiples



Listas Múltiples (cont.)

Estructura del Nodo_L:



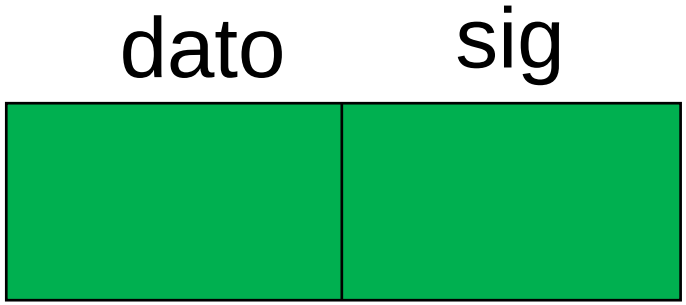
Nodo_L

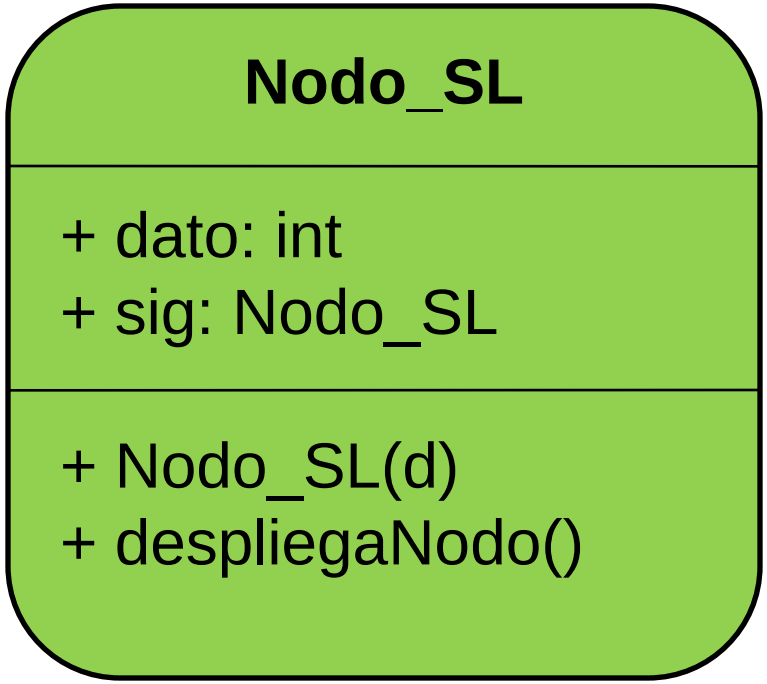
+ dato: int
+ enlace_sublista:Nodo_SL
+ enlace_lista:Nodo_L

+ Nodo_L(d)
+ despliegaNodo()

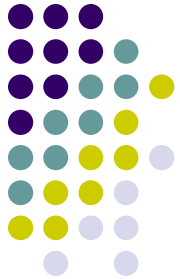
Listas Múltiples (cont.)

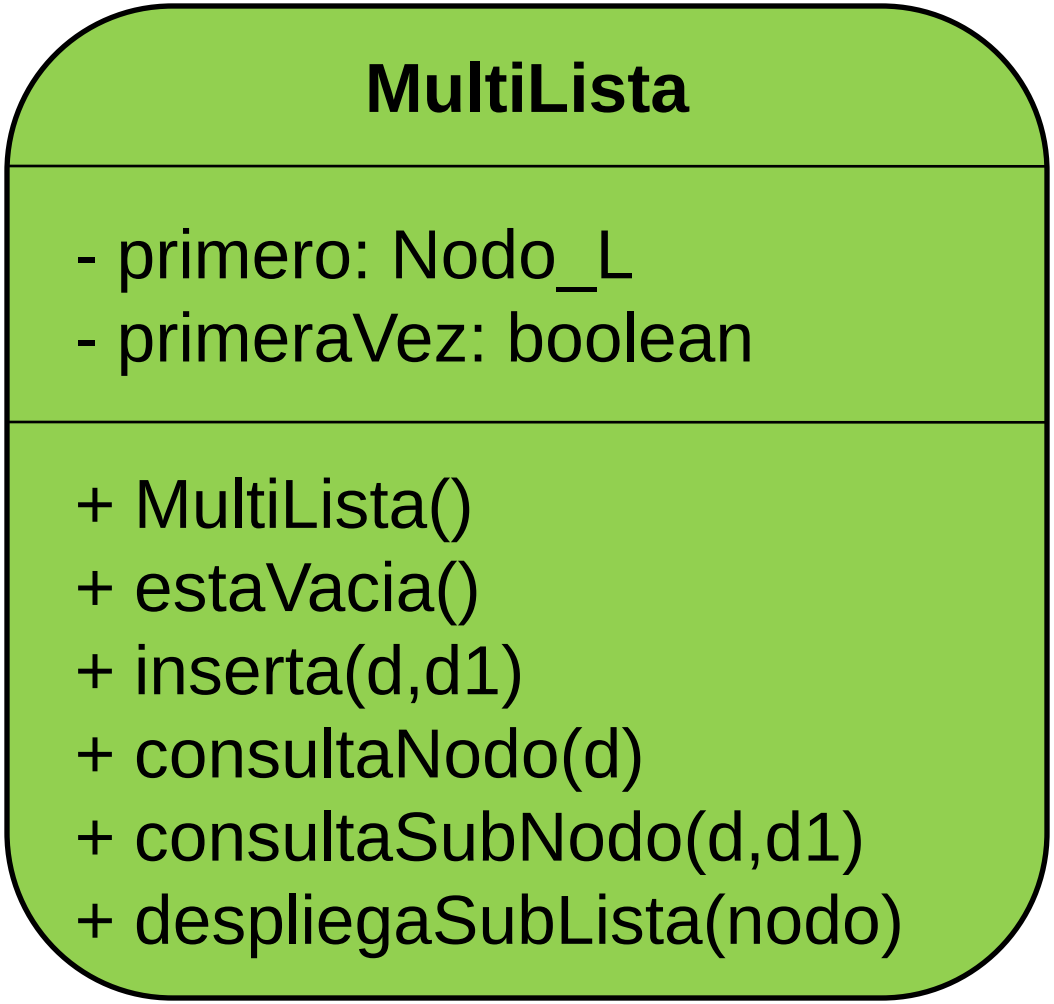
Estructura del Nodo_SL:

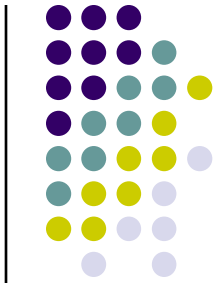




Listas Múltiples (cont.)







Listas Enlazadas

GRACIAS