

Estructuras avanzadas de datos



Estructuras dinámica de datos vs Estructuras estáticas

- ▶ Estructuras estáticas

```
int []  numeros = new int[20];
```

- ▶ ¿Qué cantidad de memoria ocupa?
- ▶ ¿Cómo se pueden poner más elementos de los que se definen inicialmente?
- ▶ ¿Qué pasa si se quiere eliminar un elemento?

Estructuras dinámica de datos vs Estructuras estáticas

- Problema de las estructuras estáticas
 - Siempre ocupan la misma cantidad de memoria
 - no se hace uso efectivo de la memoria.
- Solución: estructuras dinámicas
 - Se usa exactamente la memoria que se necesita → se gana eficiencia
 - Se complican los algoritmos de manipulación de las estructuras

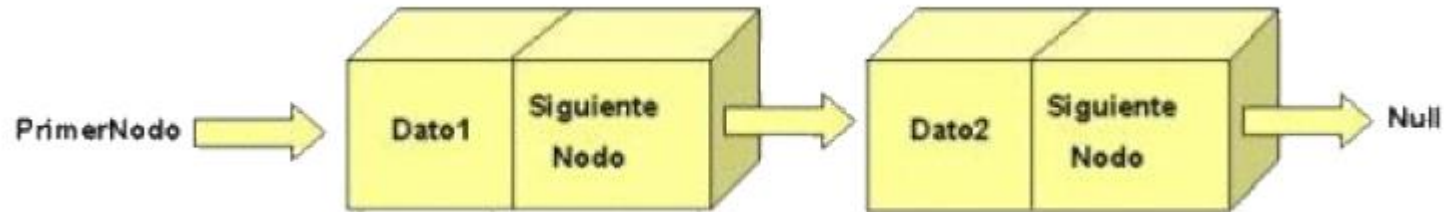
Referencias

- ▶ Puntero (referencia en Java): dirección de memoria donde se encuentra el dato.
 - Ejemplo conocido: objetos
- ▶ Aplicado a estructura de datos:
 - Nuevo elemento: se crea referencia
 - Borrar elemento: referencia a nulo y se libera memoria
- ▶ Conclusión: la memoria utilizada se ajusta al tamaño de la estructura.
- ▶ Ventaja de Java: no hay que liberar la memoria a mano -> Garbage Collector

Lista enlazada – concepto

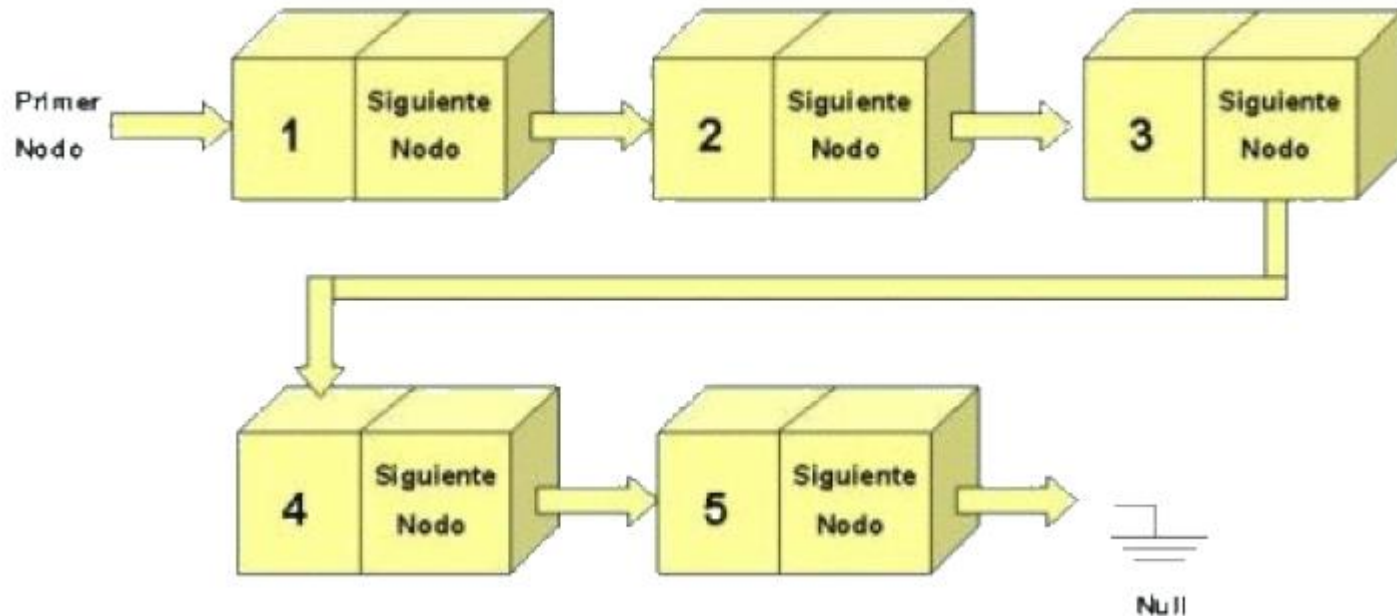
- ▶ Conjunto de elementos colocados uno detrás de otro (a priori, igual que los arrays)
- ▶ Diferencias con los arrays
 - No se almacenan de forma contigua en la memoria.
 - Cada elemento sabe dónde está el siguiente.
 - El último elemento debe indicar que no tiene siguiente.
 - Existe una forma para acceder al primer elemento, de éste al siguiente y así sucesivamente hasta el último.
 - El número de elementos que se pueden tener depende de la memoria.

Lista enlazada – implementación



- ▶ Una lista consta de una serie de nodos (su estructura se define en una clase: Nodo)
 - Cada nodo contiene:
 - Dato: información que se almacena en la estructura.
 - Referencia: enlace al siguiente elemento.
- ▶ Existirá una referencia al primer nodo para acceder a la lista. Esta referencia será nulo si la lista está vacía.
- ▶ La referencia del último nodo será nulo para indicar que no hay siguiente.

Lista enlazada – ejemplo



¿Cómo se puede crear
una lista de Strings?
¿o de objetos de la
clase Persona?


```
class Nodo {  
    int dato;  
    Nodo siguienteNodo;  
  
    public Nodo(int numero){  
        dato=numero;  
        siguienteNodo=null;  
    }  
}
```

Lista enlazada genérica

▶ Se necesitará:

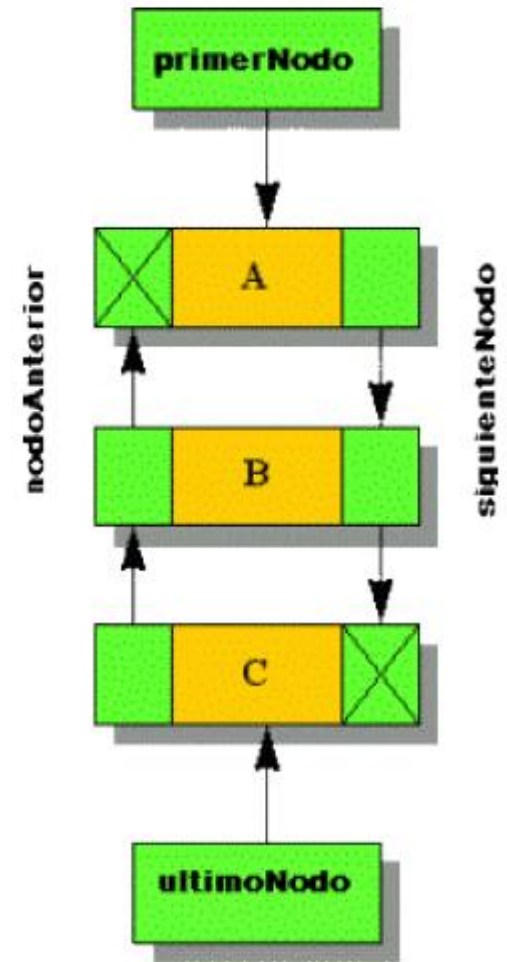
- Clase Nodo donde el atributo que guarda la información es de la clase Object.
- Clase Lista desde donde se accederá al primer nodo y desde dónde realizar operaciones sobre la lista.

▶ Operaciones:

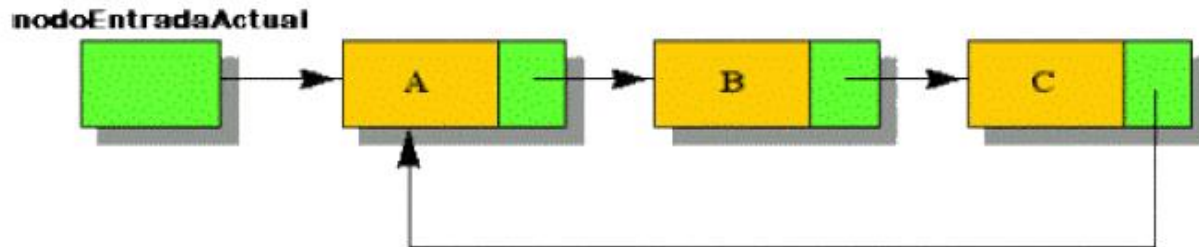
- Insertar elemento
 - Eliminar elemento
 - Actualizar elemento
 - Vaciar lista
 - Consultar un elemento
 - Consultar todos los elementos
- 

Lista doblemente enlazada

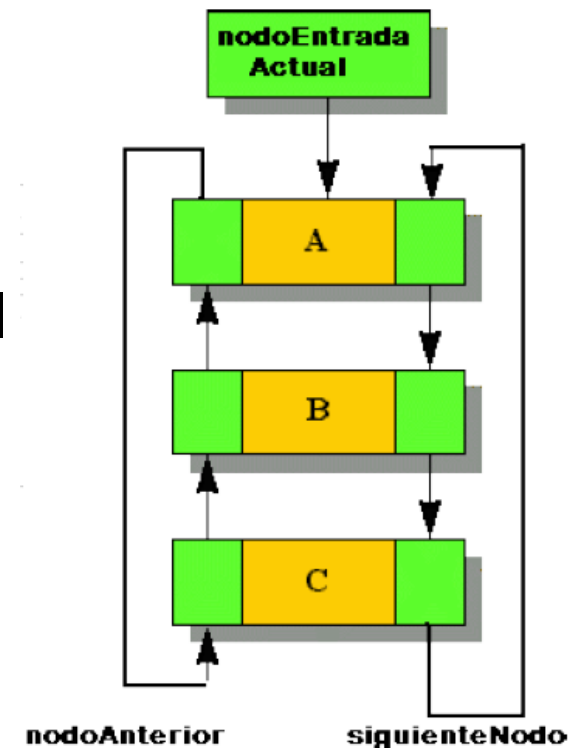
- ▶ Contiene tres partes diferenciadas en cada nodo:
 - Información
 - Referencia al siguiente
 - Referencia al anterior
- ▶ El último nodo, tendrá valor nulo en su referencia al siguiente.
- ▶ El primer nodo, tendrá valor nulo en su referencia al anterior.



Listas circulares enlazadas



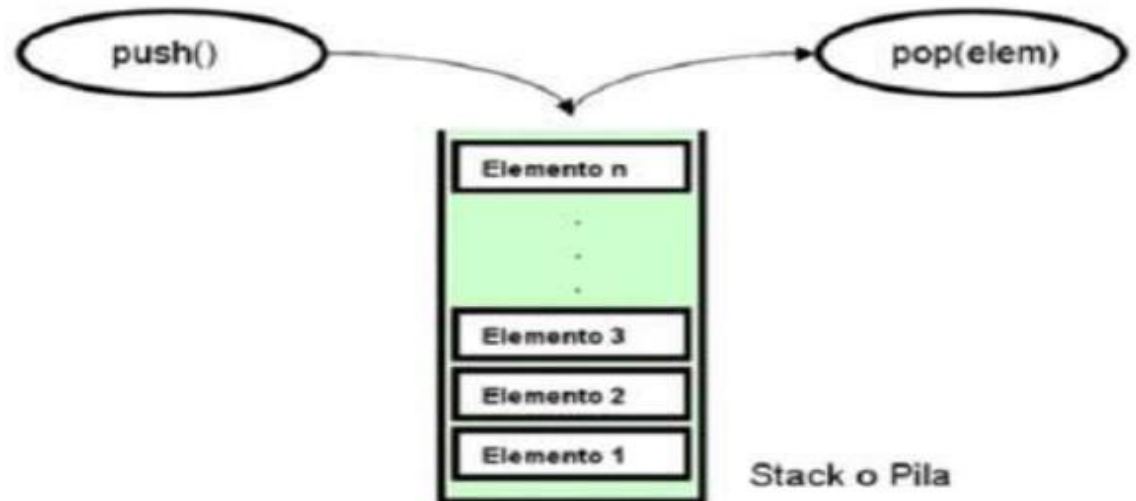
- ▶ **Lista circular enlazada:** es como una lista enlazada simple, pero la referencia al nodo siguiente del último elemento es el primer elemento.
- ▶ **Lista circular doblemente enlazada:** es como una lista doblemente enlazada, pero la referencia al nodo siguiente del último elemento es el primer elemento y la referencia al nodo anterior del último es el primero.



Pila

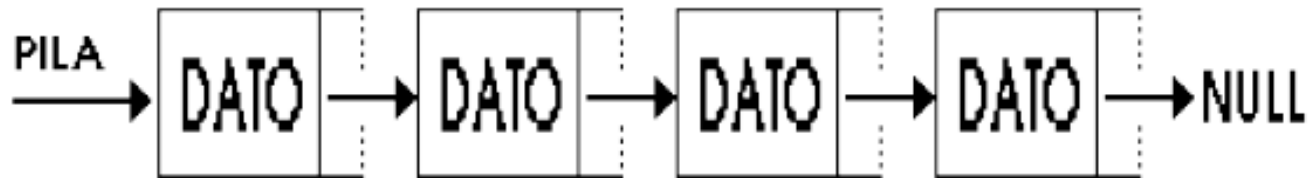
► Características:

- Solo se puede insertar por un extremo: **cima** o **cabecera**
- La operación añadir se suele llamar **push**
- Solo se puede eliminar por un extremo: **cima** o **cabecera**
- La operación eliminar se suele llamar **pop**
- Otras operación que se puede necesitar es saber si está o no vacía



Pila – implementación

- ▶ **Con vector:** índice de la última posición o del primer sitio disponible. Tamaño máximo predefinido. Se necesita una forma para saber si caben más.



- ▶ **Con lista enlazada:** al añadir un elemento, éste pasa a ser el primero y tiene como elemento siguiente el que antes era primero. Solo se podrá eliminar el primer elemento y el segundo pasará a ser el primero.

Cola

► Características:

- Los datos se almacenan por orden de llegada
- Solo se puede añadir por un extremo, el final de la cola
- Solo se puede eliminar por un extremo, el principio de la cola, también llamado cima o cabecera
- Otras operación que se puede necesitar es saber si está o no vacía

Cola – Implementación

► Implementación:

- **Con vector:** dos índices, uno para última y otra para el último. Tamaño máximo predefinido. Se necesita una forma para saber si caben más.
- **Con lista enlazada:** al añadir un elemento, éste pasa a ser el último e irá después del que antes era el último. Solo se podrá eliminar el primer elemento y el segundo pasará a ser el primero.