Listas Enlazadas

Definición

Una lista enlazada se puede definir recursivamente así

> *Una lista es una estructura nula, o un **nodo** que contiene un item genérico y una referencia a una lista enlazada.*

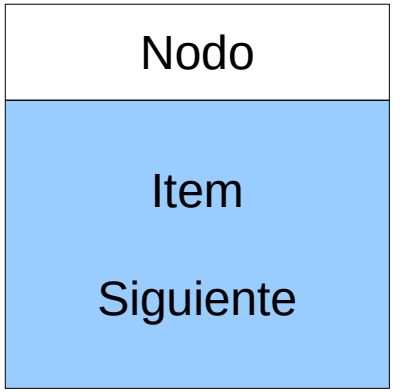
El nodo es un registro* que contiene un item y una referencia a otro nodo.

* Un nodo no se considera ADT porque no tiene métodos, simplemente valores que se acceden directamente desde la clase que lo contenga.

Implementación de la estructura nodo

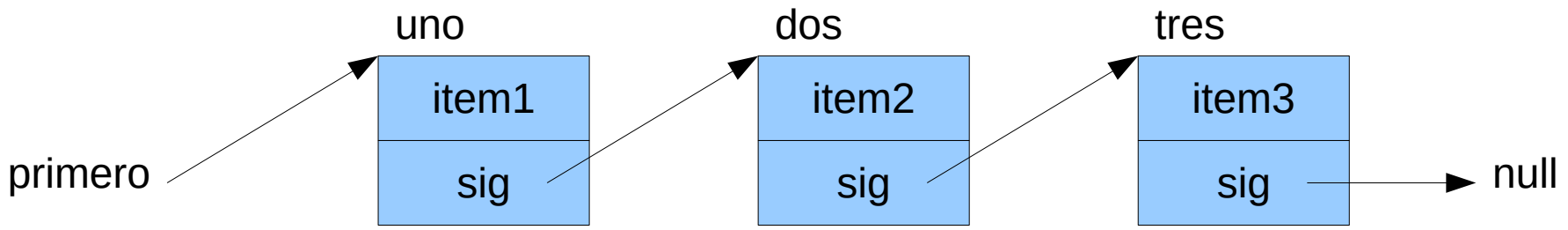
Gráficamente

```
class Nodo<Item> {
   Item item;
   Nodo siguiente;
}
```



Construcción de una lista enlazada

```
Nodo uno = new Nodo();
Nodo dos = new Nodo();
Nodo tres = new Nodo();
primero = uno;   // “cabeza” de la lista
uno.siguiente = dos;
dos.siguiente = tres;
```



Características de la lista enlazada

- No tiene un tamaño predefinido
- Puede crecer/decrecer dinámicamente en tiempo de ejecución
- No hay acceso aleatorio a los elementos. Se debe recorrer la lista desde la cabeza.

Operaciones con listas

- Inserción al comienzo
- Inserción al final
- Remover del comienzo
- Remover del final
- Insertar/Remover de otras posiciones

Recorrido de la lista enlazada

```
for(Nodo x=primero; x!=null; x=x.siguiente) {
    // operar con el nodo x
}
```

Implementación de la pila mediante una lista

- Operación `push` : Agregar un nodo a la cabeza de la lista
- Operación `pop` : Extraer el nodo de la cabeza de la pila
- Operación `isEmpty` : Verificar si la cabeza es nula
- Operación `size` : Contar nodos, o mantener una variable con el conteo.

Implementación de la pila

```
public class PilaConLista<T> implements Iterable<T> {

	private Nodo primero;
	private int n;

	private class Nodo {
		T item;
		Nodo siguiente;
	}

	public PilaConLista() { }

	public void push(T s) {
		Nodo anterior = primero;
		primero = new Nodo();
		primero.item = s;
		primero.siguiente = anterior;
		n++;
	}

	public T pop() {
		T item = primero.item;
		primero = primero.siguiente;
		n--;
		return item;
	}

	public boolean isEmpty() {
		return n==0;
	}

	public int size() {
		return n;
	}
	…
}
```

Características de la implementación

- Contiene cualquier tipo de dato: Genérica
- Tamaño proporcional al número de nodos
- Las operaciones básicas de las principales estructuras (`add`, `push/pop`, `enqueue`,`dequeue`) requieren tiempo constante. Es independiente del tamaño de la estructura.

Iterador de listas

- Se inicializa en el primer elemento de la lista
- La operación next avanza al siguiente nodo de la lista.
- Termina cuando la referencia sea nula.

Implementación del Iterador

```
…

public Iterator<T> iterator() {
	return new IteradorReverso();
}


private class IteradorReverso implements Iterator<T> {

	private Nodo pos=primero;

	@Override
	public boolean hasNext() {
		return pos!=null;
	}

	@Override
	public T next() {
		T item = pos.item;
		pos = pos.siguiente;
		return item;
	}

}
…
```

Implementación de la cola mediante una lista

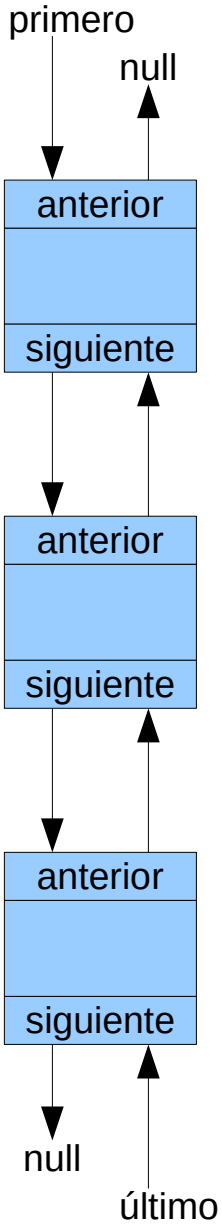
- Operacion `enqueue` : Agregar al final de la lista
- Operación `dequeue` : Remover del comienzo de la lista
- `isEmpty` y `size` : iguales que para la pila

Implementación de la bolsa mediante una lista

- Operación `agregar` : Se insertan los elementos a la cabeza de la lista
- `isEmpty` y `size` : iguales a las anteriores

Listas doblemente enlazadas

- Cada nodo contiene referencias a su predecesor y a su sucesor.
- Se tienen referencias a la cabeza y la cola de la lista.
- Esto permite recorrer la lista en ambos sentidos.



Cola de doble terminación
Double-ended queue (Deque)

- Colección ordenada que permite recorrer los elementos en ambos sentidos.
- Se implementa fácilmente mediante una lista doblemente enlazada
- API

public class Deque**<Item>** implements Iterable<Item>		
	Deque()	// Crear Deque vacío
void	offerLast(Item item)	// Agregar item al final
void	offerFirst(Item item)	// Agregar item al comienzo
Item	pollLast()	// Remover el último
Item	pollFirst()	// Remover el primero

Implementaciones

Java

- Stack
- Queue
- LinkedList
- Deque

Python

- Stack
- Queue
- Bag
- LinkedList