Listas Simples

// Insertar nodo en la primera posición

    public void addFirst (int element){

    if(this.first == null){

        this.first == new Node(element);   // el if queda sobrando

    }else{

        Node temple = new Node(element);

        temple.next = this.first;

        this.first = temple;

    }

}

// Eliminar nodos

    public void deletefirst (){

    if (this.first == null){

    return;

    }else{

    this.first = this.first.next;

    }

}

// Segunda manera de eliminar al inicio

    public void deletefirst (){

    if (this.first == null){

    return;

    }else{

    Node temp = this.first;

    this.first = temp.next;

    temp.next = null;

    }

}

// ------------------------>

new Object;  //Cada vez que hay un new, se pide un espacio en la memoria, en este caso no tiene referencia

Esta memoria se llama el HEAP, hay otra que se llama el Stack (o = 0x00)

Object o = new Object(); // Se pide una nueva casilla de espacio en el HEAD

Object p = o; // Agrega otra referencia al espacio de memoria, igual que o

Las referencias declarados en el se elimina al momento de cerrar las llaves, pero el espacio en el HEAD se concerva

GARBAGE COLLECTOR

    Va al HEAD y libera los espacios al no tener referencia; corre cuando java lo considere

O = null  // Elimina la referencia de O del Stack, pero si otro apunta al bloque, no elimina el espacio de memoria en el

        HEAD

// Eliminar al final

    public void deletelast (){

    if (this.first != null) {

        if (this.first.next == null){

        this.first = null;

        } else {

            Node temp = this.first;

            while (temp.next.next != null){

                temp.next = null;

            }

        }temp.next = null;

    }

}

//------------->

Listas doble

    - Se puede navegar hacia atrás

    - Cada nodo tiene dos referencias, el next y el previews (prev)

// Insertar elemento en lista doble

class node{

    node next;

    node prev;

    int data;

}

public node (int e){

    this.data = e;

    this.next = null;

    this.prev = null;

}

    void add first (int element){

    if (this.first == null){

        this.first = new Node (element);

    }else{

        Node n = new Node (element);

        n.next = this.first;

        this.first.prev = n;

        this.first = n;

// Node n = new node (element);

// n.next = this.first;

// this.first = n;

// this.first.next.prev = this.frst;

    }

}

// void devuelve vacio