Algoritmos

Logro de sesión

- Al finalizar la sesión, el estudiante **utiliza relaciones de asociación entre clases** para la construcción de programas.

Relaciones de Asociación

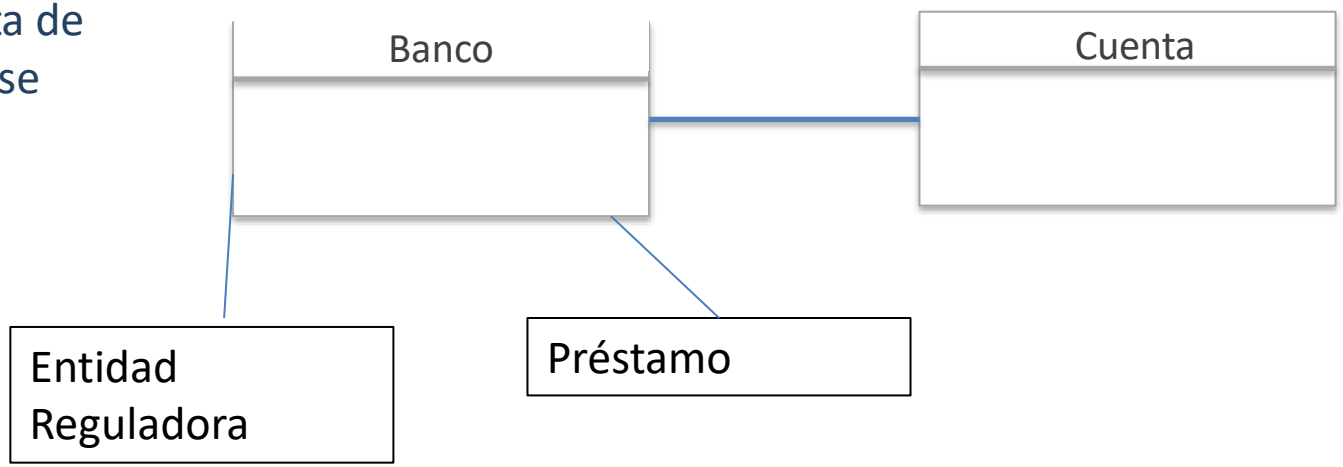
Asociación

Se establece una relación de asociación cuando **dos clases están conectadas entre sí** de alguna manera.

Línea simple

Esta relación se da **cuando los objetos de una clase conocen los objetos de otra clase.** La relación puede ser de uno a uno, de uno a muchos, de muchos a uno o de muchos a muchos. Además, los objetos se pueden crear o eliminar de forma independiente.

Por ejemplo: Una asociación de “cuenta de registros bancarios” se puede mostrar de la siguiente manera.



Asociación

Línea simple

Ejemplo: Empleado y compañía

Ejemplo: el pasajero y la aerolínea pueden estar relacionados de la siguiente manera

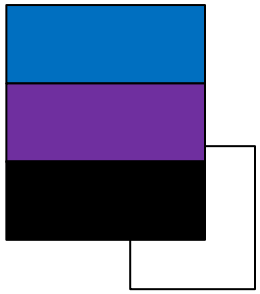
Asociación: Conceptos

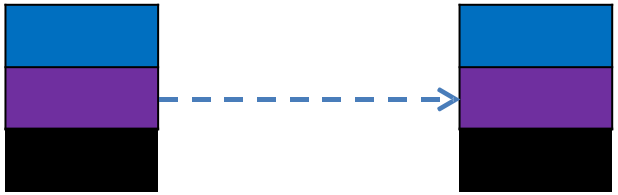
Es una RELACION ESTRUCTURAL que vincula a dos objetos. El vinculo es una CONEXIÓN entre dos objetos

Los dos objetos relacionados pueden ser de la misma o diferente clase.

Gráficamente se representa mediante **una línea** que une las dos clases

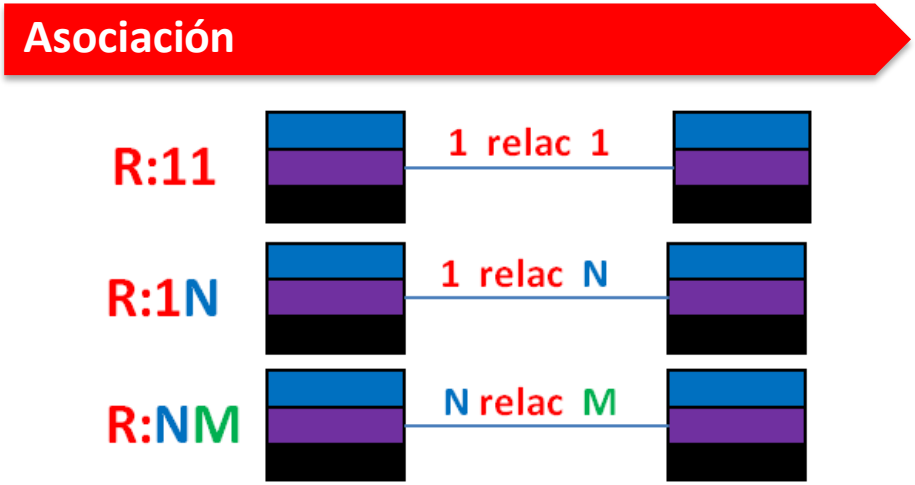
una instancia de una clase debe conocer de la otra para poder realizar su trabajo.





Asociación: Tipos

- Las relaciones de asociación se pueden clasificar en:
  - Relaciones 1 a 1 (R:11)
  - Relaciones de 1 a N y viceversa (R:1N)
  - Relaciones N a M (RNM)
- Este tipo de relaciones indica la cantidad de objetos que participan en la relación.



Relaciones de Asociación: Ejemplo 1

Una asociación entre dos clases indica que los objetos en un extremo de una asociación "reconocen" objetos en el otro extremo y pueden enviar mensajes entre ellos

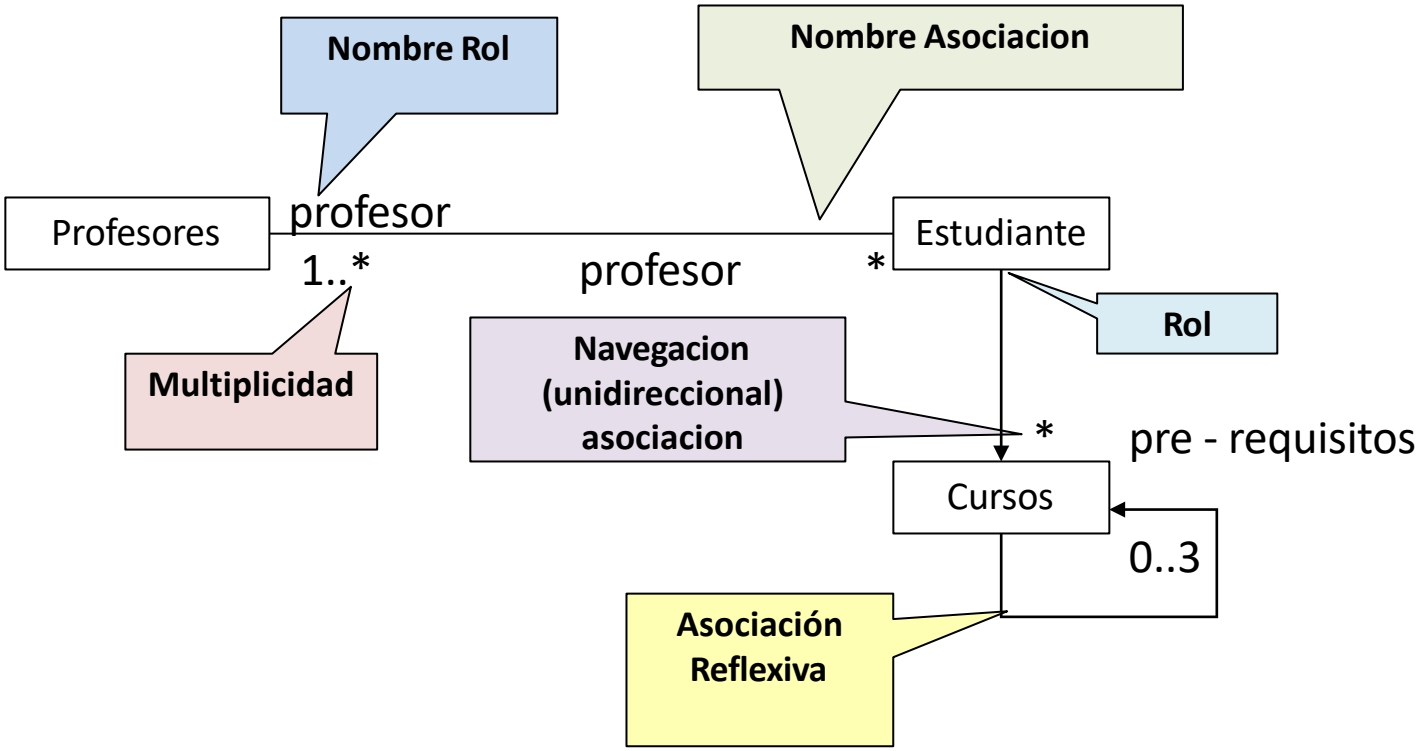
Ejemplo: “Un Empleado trabaja para una compañía”

Relaciones de Asociación (cont.)

- **El nombre** se representa con una **etiqueta** situada al **centro de la línea de asociación** y generalmente usa un verbo o una frase verbal.
- **Un rol** esta al **final de una asociación** donde conecta con una clase. Usualmente es un **sustantivo o frase nominal** y es obligatorio para asociaciones reflexivas

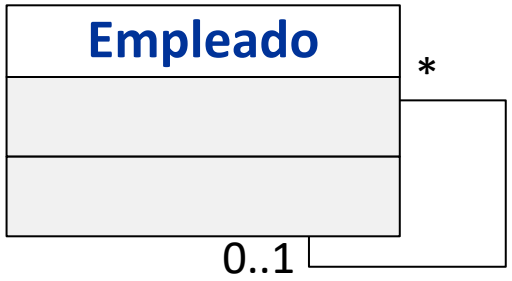
Relaciones de Asociación: Ejemplo 2

Diagrama de Clases de una Institución Educativa Superior



Actividad: Modificar agregando el Sistema de evaluación de un curso

Asociación caso especial: Asoc. Ciclica

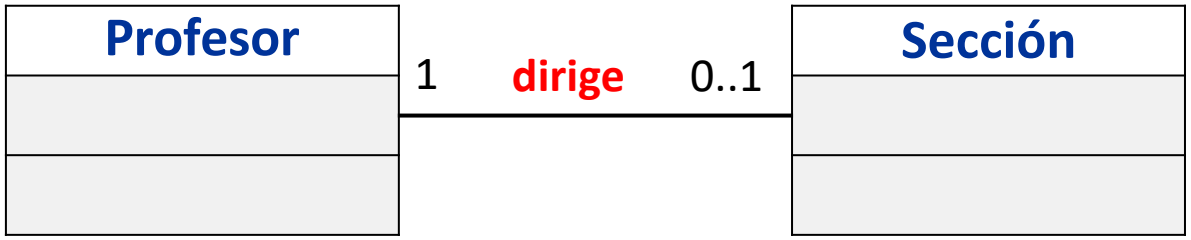


- El empleado puede tener 0 o 1 jefe.
- El empleado(con rol de jefe) puede tener varios subordinados.
- Relación Reflexiva.

```
class Empleado {
  private Empleado jefe;
  private Empleado subordinados[];
  …
}
```

Relaciones de Asociación: 1 a 1

- **Un** Profesor **dirige** 0 ó 1 sección
- **Un** sección **es dirigido** por 1 profesor




```
//SOLUCION 1
class Profesor {
  Seccion* d;
  …
}
```

```
//SOLUCION 2
class Seccion{
  Profesor* p;
  …
}
```

Ejemplo

autor.h

```
#ifndef _AUTOR_H_
#define _AUTOR_H_
#include<string>
class Autor{
  private:
    char* nombre, apellido, pais;
  public:
    Autor(char* n, char* a, char* p){
        nombre = n;
        apellido = a;
        pais = p;
    }
    ~Autor(){}

    //Metodos de servicio
     char* toString(){
      string val = “nom:” + (string)nombre
      + “ape:” + (string)apellido +
      “pais:” + (string)pais;
      char* valret = &*val.begin();
      return (valret);
    }
};
#endif
```

libro.h

```
#ifndef _LIBRO_H_
#define _LIBRO_H_
#include<sstream>
class Libro{
  private:
    char* titulo;
    Autor* autor;
  public:
    Libro(char* t, Autor* a){
        titulo = t;
        autor = a;
    }
    ~Libro(){}
    //Metodos de servicio
     char* toString(){
     string val;
      val+= “titulo:” + (string)titulo;
      val+= (string)(autor->toString());

      char* valret = &*val.begin();
      return (valret);
    }
};
#endif
```

Ejemplo

```
#include<iostream>
#include “autor.h”
#include “libro.h”
using namespace std;

int main() {
   Autor* a = new Autor("Luis", "Joyanes", "España");
   Libro* l = new Libro(“C++”, a);
   //boolean res = a->agregarLibro(l);
   cout<<“Objeto Autor: ” <<a->toString()<<endl;
   cout<<“Objeto Libro: ” <<l->toString()<<endl;
   cin.get();
   return(0);
}
```

Ejemplo 2

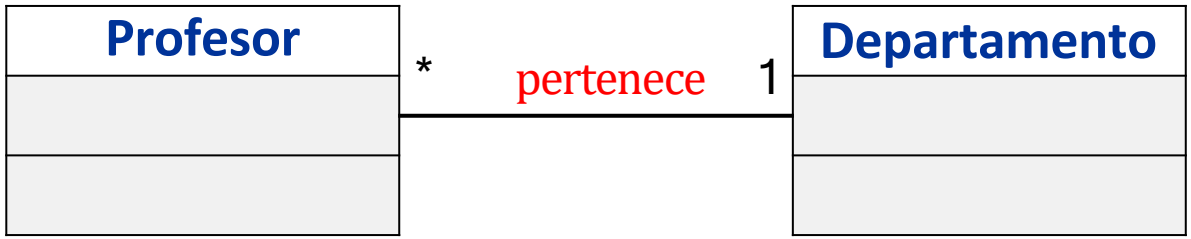
- La Biblioteca de la UPC maneja información de muchos libros y de sus autores; **suponga que**:
  - De cada libro se conoce: **título, editorial, edición y número de páginas**.
  - De los autores se conoce: **su nombre, apellido y país de procedencia.**
  - Un libro es escrito por uno y solo un autor
- Se pide:
  - Elaborar el diagrama de clases.
  - Implementar el diagrama de clases.
  - Implementar los métodos setter/getter

Ejercicio:

- Diagrame e Implemente la relación de asociación que se presenta en la siguiente imagen.

Relaciones de Asociación: N a 1

- Un Profesor **pertenece** a un departamento
- Un Departamento **tiene** muchos profesores

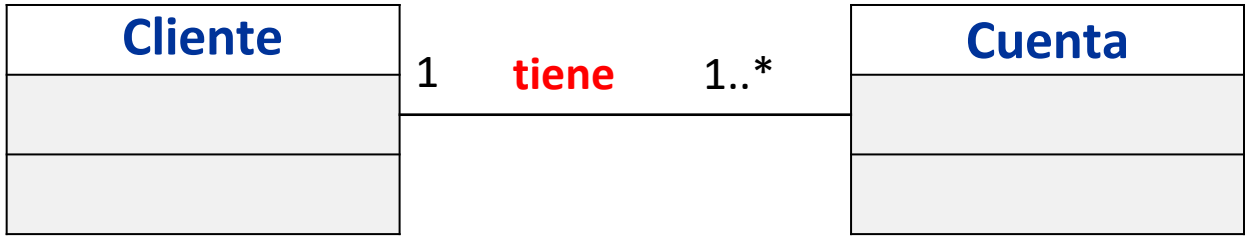



```
//SOLUCION 1
class Profesor {
  Departamento* d;
  …
}
```

```
//SOLUCION 2
class Departamento{
  Profesor** p;
  …
}
```

Relaciones de Asociación: 1 a N

- Un Cliente **tiene** 1 o más cuentas
- Un Cuenta **pertenece** a un cliente




```
//SOLUCION 1
class Cliente {
  Cuenta** cuenta;
  …
}
```

```
//SOLUCION 2
class Cuenta{
  int numCuenta;
  …
}
```

Ejercicio:

- Diagrame e Implemente las relaciones entre clase que se presentan en el siguiente diagrama.

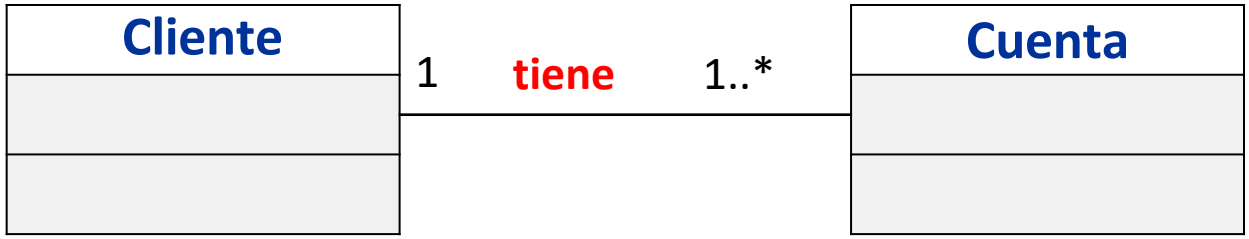
Colecciones de Objetos

Colecciones de Objetos

- Uno de los problemas a nivel de implementación de las relaciones 1 a N es decir la forma en cómo se debe implementar.
- De manera general existen dos formas de implementación:
  - Utilizar Arreglos Tipados
  - Utilizar Contenedores de Objetos (Colecciones de Objetos)

Ejemplo de Implementación

- Un Cliente **tiene** 1 o más cuentas
- Una Cuenta **pertenece** a un cliente




```
//USO DE ARREGLOS
class Cliente {
  Cuenta** cuenta;
  …
}
```

```
//USO DE CONTENEDORES
class Cliente{
  vector<Cuenta*> cuentas
}
```

Ejemplo con arreglos tipados

Ejemplo de Implementación

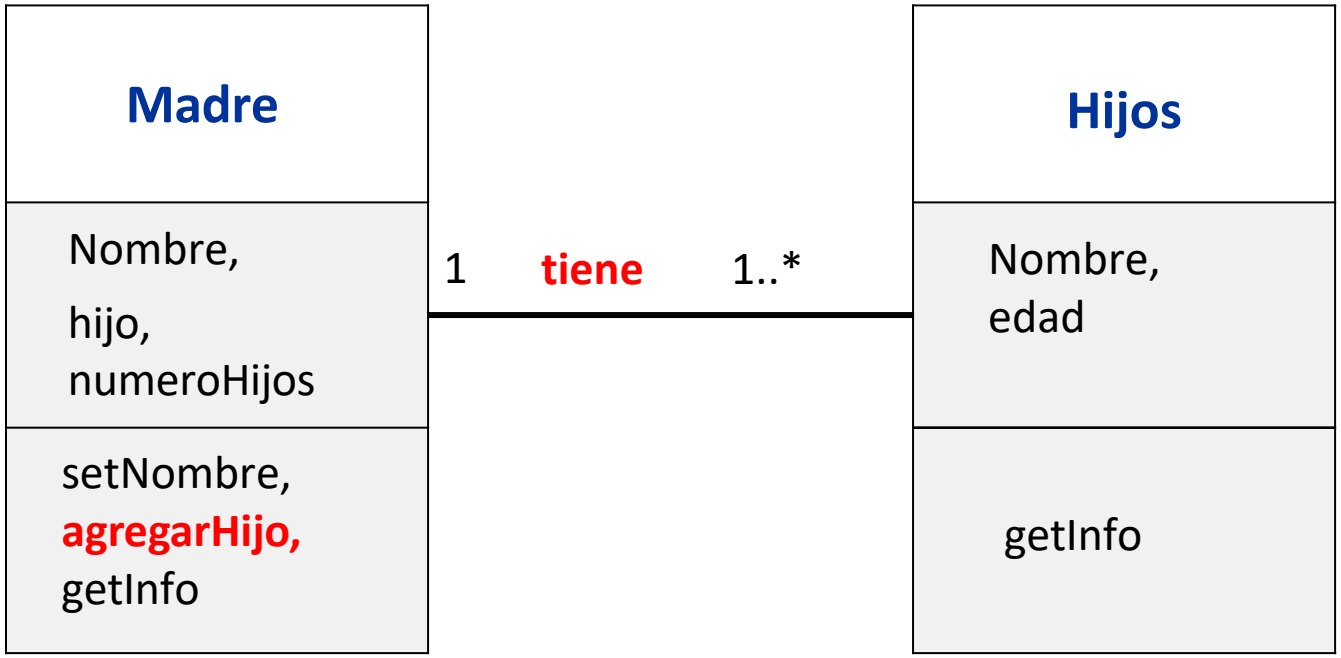
Cree los archivos Hijo. H, Madre.h y TestMadre.cpp donde usará de ejemplo:

Madre: Marge Simpson

Hijo1: Bart Simpson, edad: 8

Hijo2: Lisa Simpson, edad: 7

Hijo3: Maggie Simpson, edad: 1



Ejemplo de Implementación Hijo.h

```
#pragma once
#ifndef _HIJO_H_
#define _HIJO_H_
#include <iostream>
#include <string>
#include <cstring>
using namespace std;
class Hijo {

private:
    //atributos
    char* nombre;
    int edad;
    //constructor
public:
    Hijo(char* n = NULL, int a = 0) :nombre(n), edad(a) {}
    //destructor
    ~Hijo() {}
    //métodos
    string getinfo() {

        string res = "hijos: ";
        res += "nombre: " + (string)(nombre);
        res += "edad: " + to_string(edad);
        return res;
    }
};
#endif
```

Ejemplo de Implementación Madre.h

```h
#pragma once
#ifndef _MADRE_H_
#include <iostream>
#include "Hijo.h"

class Madre {
private:
    //atributos
    char* nombre;
    Hijo** hijos;
    int numeroHijos;
public:
    //constructor
    Madre() {
        nombre = NULL;
        numeroHijos = 0;
        hijos = new Hijo*[numeroHijos];
    }
    //destructor
    ~Madre() {
        if (hijos != NULL)
            delete[] hijos;
    }
    //métodos
    void setNombre(char* nombre) {
        this->nombre = nombre;
    }
    void agregarHijo(Hijo* nuevoHijo) {
        Hijo** hj = new Hijo*[numeroHijos + 1];
        for (int i = 0; i < numeroHijos; i++) {
            hj[i + 1] = hijos[i];
        }
        hj[0] = nuevoHijo;
        numeroHijos++;
        //validación
        if (hijos != NULL)
            delete[] hijos;
        hijos = hj;
    }
    string getInfo() {
        string res = "Información de la Madre \n";
        res += "Nombre: " + (string)(nombre);
        res += "\n";
        string r;
        for (int i = 0; i < numeroHijos; i++) {
            r += "\n" + (string)(hijos[i]->getinfo());
        }
        return (res + r);
    }
};

#endif
```

Ejemplo de Implementación TestMadre.cpp

```cpp
#include <iostream>
#include "Hijo.h"
#include "Madre.h"
#include <conio.h>

using namespace std;

int main() {
    setlocale(LC_ALL, "");
    Hijo* c1 = new Hijo((char*)"Bart Simpson ", 8);
    Hijo* c2 = new Hijo((char*)"Lisa Simpson ", 7);
    Hijo* c3 = new Hijo((char*)"Maggie Simpson ", 1);
    Madre* m = new Madre();

    m->setNombre("Marge Simpson");
    m->agregarHijo(c1);
    m->agregarHijo(c2);
    m->agregarHijo(c3);

    cout << m->getInfo();
    _getch();
    return (0);
}
```

Lo que aparece en consola

```
Información de la Madre
Nombre: Marge Simpson

hijos: nombre: Maggie Simpson edad: 1
hijos: nombre: Lisa Simpson edad: 7
hijos: nombre: Bart Simpson edad: 8
```

Recordar

Para establecer la posición de un cursor:

```
#include <Windows.h>
using namespace System;
Console::SetCursorPosition(x, y);
```

Para usar colores:

```
system("color 9B");
Console::ForegroundColor = ConsoleColor::Red;
Console::BackgroundColor = ConsoleColor::Yellow;
```

Repasemos

Representar por medio de una Diagrama de clases la relación:

Contacto

Lista de Contacto

Repasemos

Representar por medio de una Diagrama de clases la relación:

Contacto
Nombre NúmeroTeléfono email IdInstagram IdTiktok
Contacto() ~Contacto() getNombre() getNumeroTelefono() getEmail() getIdInstagram() getIdTiktok() setNombre() …

Lista de Contacto
Contacto **Lista **Int n //tamaño del arreglo**
ListaContacto() ~ListaContacto() **AgregarContacto()** ObtenerContacto() MostrarContactos() getN(); ¿por qué no hay un setN?