RELACIONES ENTRE CLASES

CONTENIDO:

- **Conceptos y Tipos.**
- **Cardinalidad de las relaciones.**
- **Relaciones de Asociación.**
- **Colecciones de Objetos.**

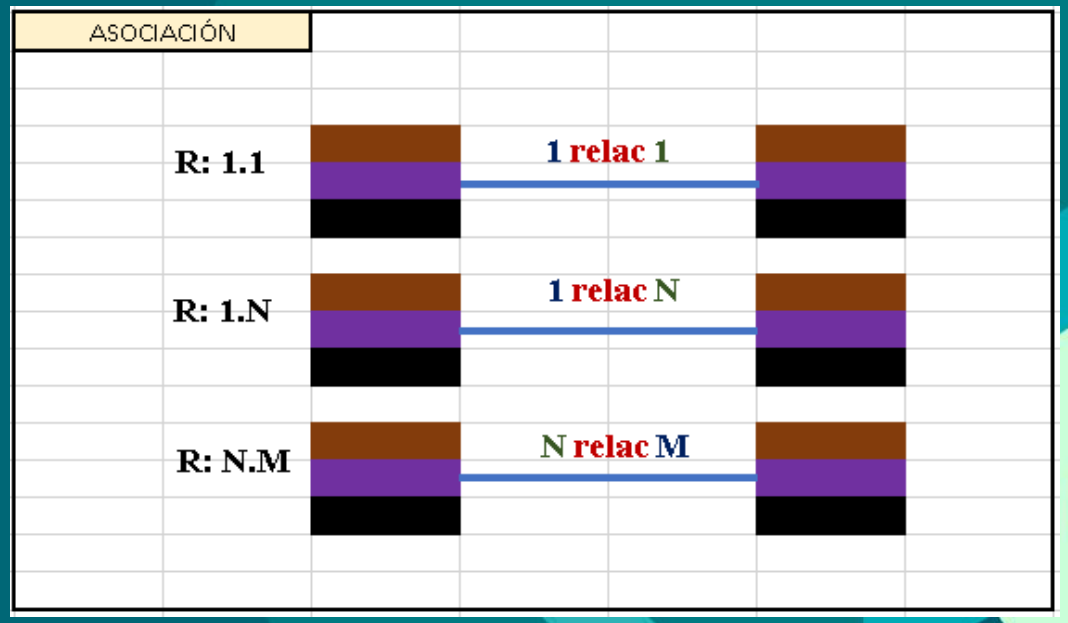
DIAGRAMA DE CLASES

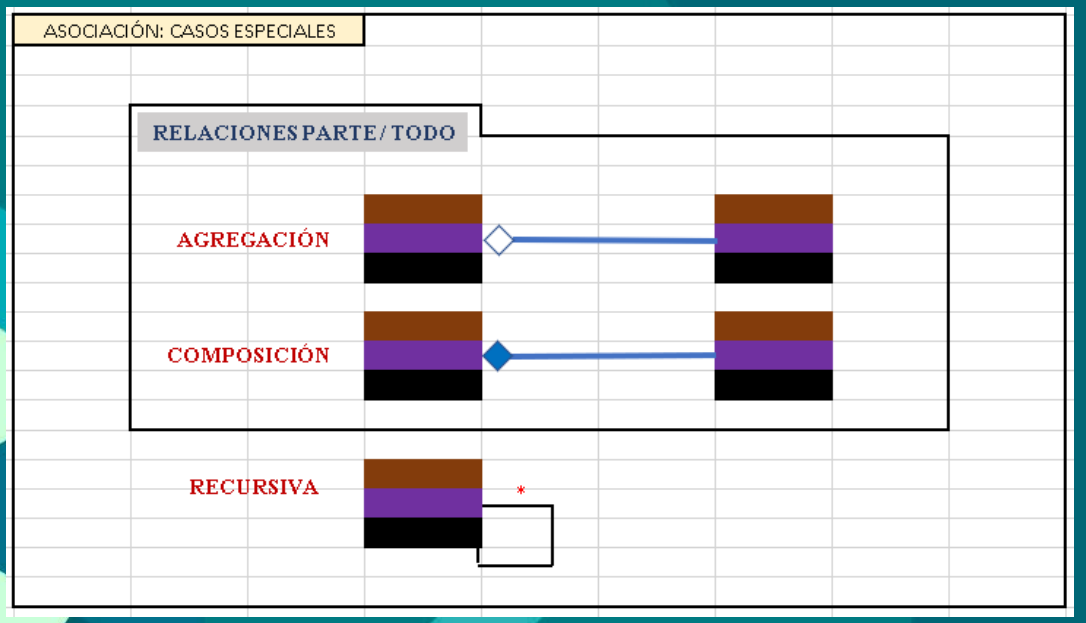
Los diagramas de clase son el componente principal del modelado orientado a objetos. Se utilizan para mostrar los diferentes objetos de un sistema, atributos, operaciones y las relaciones que se manifiestan entre ellos.

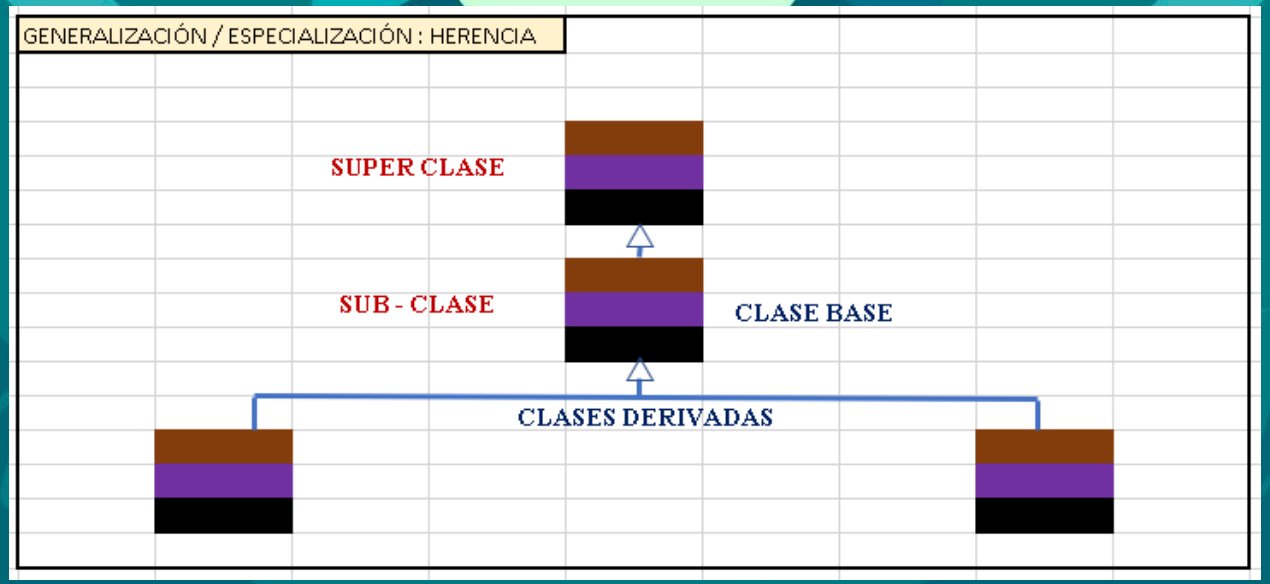
RELACIONES ENTRE CLASES

- Las relaciones nos indican como se comunican las instancias de las clases entre sí.
- Los mensajes navegan entre las relaciones de clases/objetos. Esto es posible gracias a las relaciones establecidas.
- Las relaciones entre las clases son de diferentes tipos. Estos tipos le confieren carácter y naturaleza a la relación.

RELACIONES ENTRE CLASES: TIPOS







RELACIONES ENTRE CLASES: CARDINALIDAD

- También conocida como: multiplicidad.
- Determina cuántos objetos de cada clase intervienen en una relación.
- Especifica el número de instancias de una clase que se relaciona con otro número de instancias de la otra clase.
- Cada relación tiene dos multiplicidades (una en cada extremo de la relación).

RELACIONES ENTRE CLASES: CARDINALIDAD

CARDINALIDAD / MULTIPLICIDAD:

- MUESTRA EL NÚMERO DE OBJETOS QUE PARTICIPAN EN LA ASOCIACIÓN.
- INDICA SI UNA ASOCIACIÓN ES OBLIGATORIA O NO.

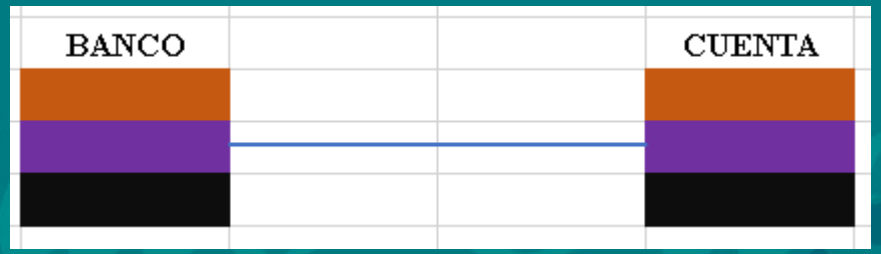
MULTIPLICIDAD	SIGNIFICADO
1	1 y sólo 1
0.1	0 ó 1 (Asociación Opcional)
N.M	Un valor entre N y M
*	Varios
0.*	0 ó muchos
1.*	1 ó muchos
2.4	Rando Específico

RELACIONES DE ASOCIACIÓN

- SE ESTABLECE ESTA RELACIÓN CUANDO DOS CLASES ESTÁN CONECTADAS ENTRE SÍ DE ALGUNA MANERA.
- 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.
- LOS OBJETOS SE PUEDEN CREAR O ELIMINAR DE FORMA INDEPENDIENTE.

EJEMPLO

UNA ASOCIACIÓN DE "CUENTA DE REGISTROS BANCARIOS", SE PUEDE MOSTRAR DE LA SIGUIENTE MANERA.



ASOCIACIÓN: CONCEPTOS

- EL VÍNCULO ES UNA CONEXIÓN ENTRE DOS OBJETOS.
- LOS DOS OBJETOS RELACIONADOS PUEDEN SER DE LA MISMA O DISTINTA CLASE.
- A VISIBILIDAD GRÁFICA, SE REPRESENTA MEDIANTE UNA LÍNEA QUE UNA AMBAS CLASES.
- UNA INSTANCIA DE LA CLASE DEBE CONOCER DE LA OTRA PARA QUE EL TRABAJO PUEDA SER REALIZADO.

EJEMPLO: RELACIONES DE ASOCIACIÓN: 1 A 1

- UN PROFESOR DIRIGE 0 Ó 1 SECCIÓN.
- UNA SECCIÓN ES DIRIGIDA POR 1 PROFESOR.


```
//Solución 1.
class CProfesor {
    CSeccion* Cseccion;

};
```

```
//Solución 2.
class CSeccion {
    CProfesor* Cprofesor;

};
```

EJEMPLO: RELACIONES DE ASOCIACIÓN: 1 A 1

```cpp
#ifndef AUTHOR_H
#define AUTHOR_H

#include<string>

class CAuthor {

private:
    std::string name;
    std::string surname;
    std::string country;

public:
    CAuthor(std::string name, std::string surname, std::string country) {

        this->name = name;
        this->surname = surname;
        this->country = country;
    }
    ~CAuthor() {
    }
    //Métodos de acceso.
    std::string getName() {
        return name;
    }
    std::string getSurname() {
        return surname;
    }
    std::string getCountry() {
        return country;
    }
};
#endif
```

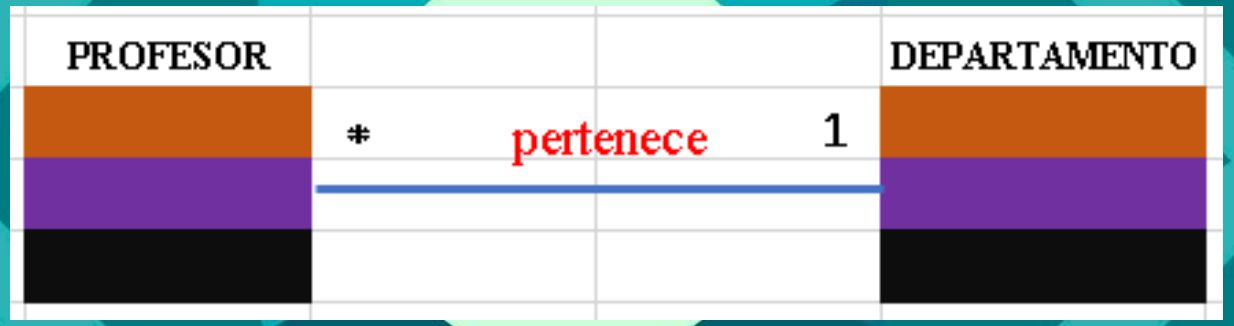
```cpp
#ifndef BOOK_H
#define BOOK_H

#include"CAuthor.h"
#include<iostream>
#include<string>

class CBook {

private:
    std::string title;
    CAuthor* author;

public:
    CBook(std::string title, CAuthor* new_author) {

        this->title = title;
        this->author = new_author;
    }
    ~CBook() {
    }
    void PrintData() {

        std::cout << "\nBook Data: " << std::endl;
        std::cout << "=============" << std::endl;
        std::cout << "Title: " << title << std::endl;
        std::cout << "=============" << std::endl;
        std::cout << "Author:" << std::endl;
        std::cout << "=============" << std::endl;
        std::cout << "Name: " << author->getName() << std::endl;
        std::cout << "Surname: " << author->getSurname() << std::endl;
        std::cout << "Country: " << author->getCountry() << std::endl;
    }
};
#endif
```

EJEMPLO: RELACIONES DE ASOCIACIÓN: 1 A 1

```
#include<iostream>
#include<conio.h>

#include"CAuthor.h"
#include"CBook.h"

void main(){

    CAuthor* author = new CAuthor("Gonzalo", "Quedena", "Peru");

    CBook* book = new CBook("Los hechizeros", author);

    book->PrintData();


    _getch();
}
```

EJEMPLO: RELACIONES DE ASOCIACIÓN: N A 1

- UN PROFESOR PERTENECE A UN DEPARTAMENTO
- UN DEPARTAMENTO TIENE MUCHOS PROFESORES.

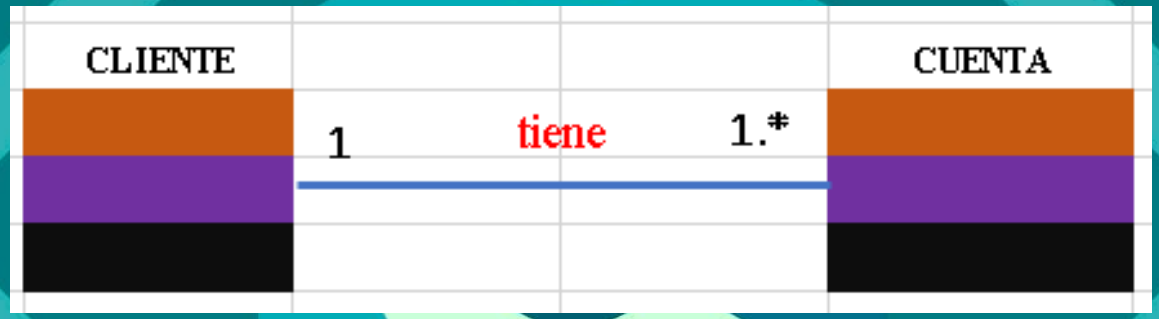



```
//Solución 1
class CTeacher {
    CDepartment* department;
};
```

```
//Solución 2.
class CDepartment {
    //Primera forma.
    CTeacher** teacher;
    //Segunda forma.
    std::vector<CTeacher*>techers;
};
```

EJEMPLO: RELACIONES DE ASOCIACIÓN: 1 A N

- UN CLIENTE TIENE 1 O MÁS CUENTAS.
- UNA CUENTA PERTENECE A UN CLIENTE.




```
//Solución 1.
class CClient {
    /*Forma 1.*/
    CAccount* account;
    /*Forma 2.*/
    std::vector<CAccount*>accounts;
};
```

```
//Solución 2.
class CAccount {
    //...
};
```

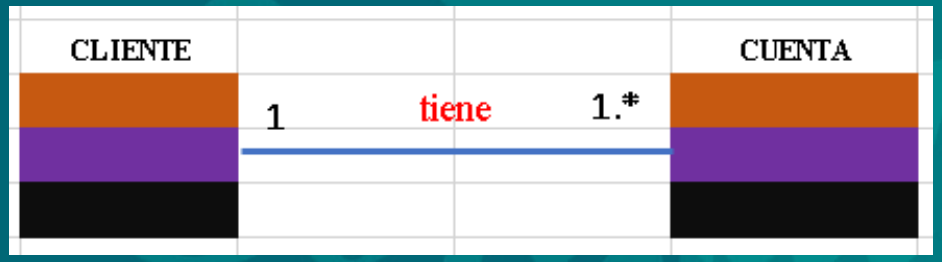
COLECCIONES DE OBJETOS

- UNO DE LOS PROBLEMAS A NIVEL DE IMPLEMENTACIÓN SE DAN EN LAS RELACIONES DE 1 A N, ES DECIR, LA FORMA EN CÓMO SE DEBE IMPLEMENTAR.

- EXISTEN DOS FORMAS DE IMPLEMENTACIÓN:
  - UTILIZAR ARREGLOS TIPADOS.
  - UTILIZAR CONTENEDORES DE OBJETOS.

EJEMPLO

- UN CLIENTE TIENE 1 O MÁS CUENTAS.
- UNA CUENTA PERTENECE A UN CLIENTE.




```
//USO DE ARREGLOS.
class CClient {
    /*Forma 1.*/
    CAccount* account;
};
```

```
//USO DE CONTENEDORES.
class CClient {
   /*Forma 2.*/
   std::vector<CAccount*>cuentas;
};
```

EJEMPLO CON ARREGLOS!

¡HACER EL EJEMPLO DE MADRE E HIJOS!

¡ENTREMOS AL VISUAL Y GRABEN LA EXPLICACIÓN!