El futuro digital es de todos

MinTIC

Hechos QUE CONECTAN

Tutoría temática

Relaciones entre clases – Asociación, Agregación, y Composición.

Por equipo de tutores Misión TIC – UIS

Inicio: 8:05 PM

Universidad Industrial de Santander

UIS

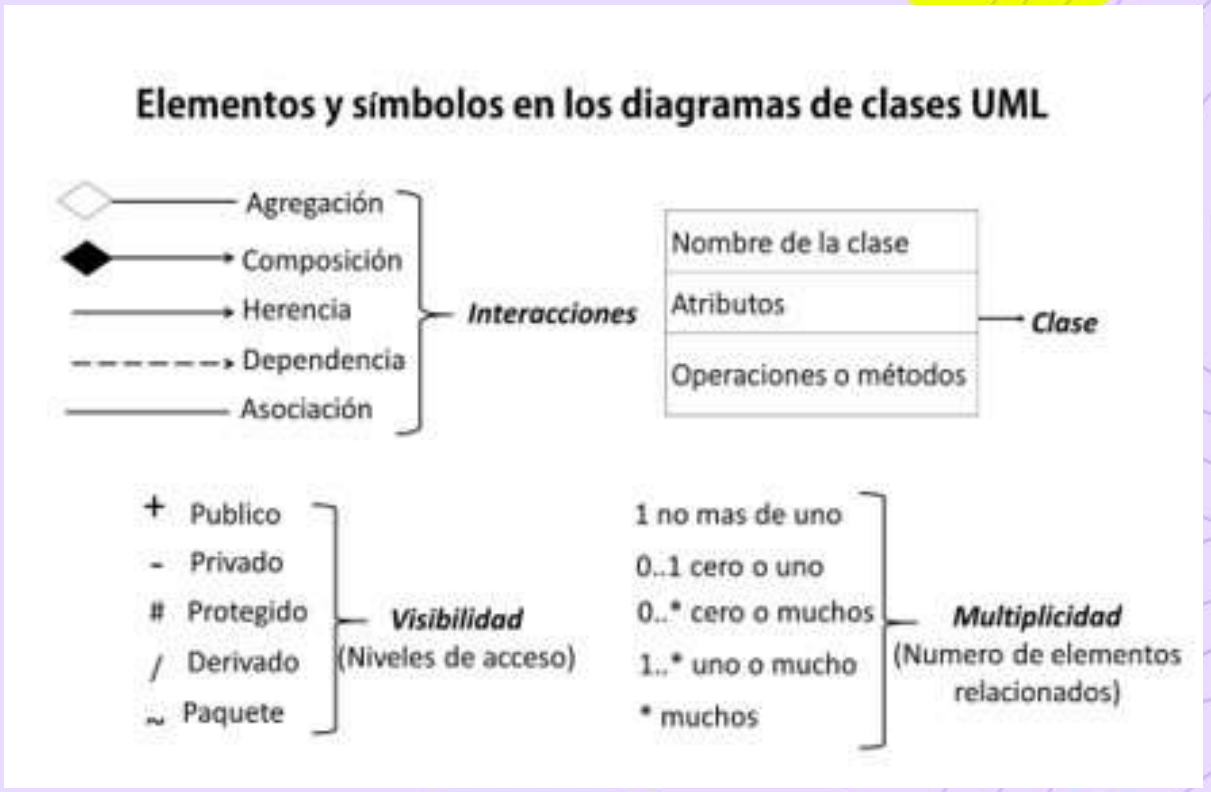
Misión TIC 2022

Temario

- Diagramas estructurales.
- Relaciones entre clases UML.
- Tipos de relaciones.
- Relación de asociación, agregación y composición.
- Asociación directa o bidireccional.
- Diferencias entre agregación y composición.
- Navegabilidad.
- Ejercicio práctico de asociación, agregación y composición.

Diagramas

- Cuando modelizamos un sistema usando UML, lo hacemos mediante la creación de diferentes diagramas, los cuales están divididos en dos grandes categorías: estructurales y de comportamiento. En el Ciclo 2 abordaremos los diagramas estructurales



Diagramas Estructurales

Un diagrama estructural, es una visualización de cómo un sistema está compuesto, es decir, muestra las partes que existen y cómo se estructuran. Estos diagramas son los más usados cuando se habla de arquitectura y existen diversos tipos. Para mencionar algunos están:

- Diagrama de paquetes.
- Diagrama de despliego (deployment).
- **Diagrama de clases.**
- Entre otros.

Seguramente el más conocido es el diagrama de clase, que muestra cómo las clases de nuestro sistema se relacionan entre ellas. El cual abordaremos en esta presentación.

Tipos de Relaciones

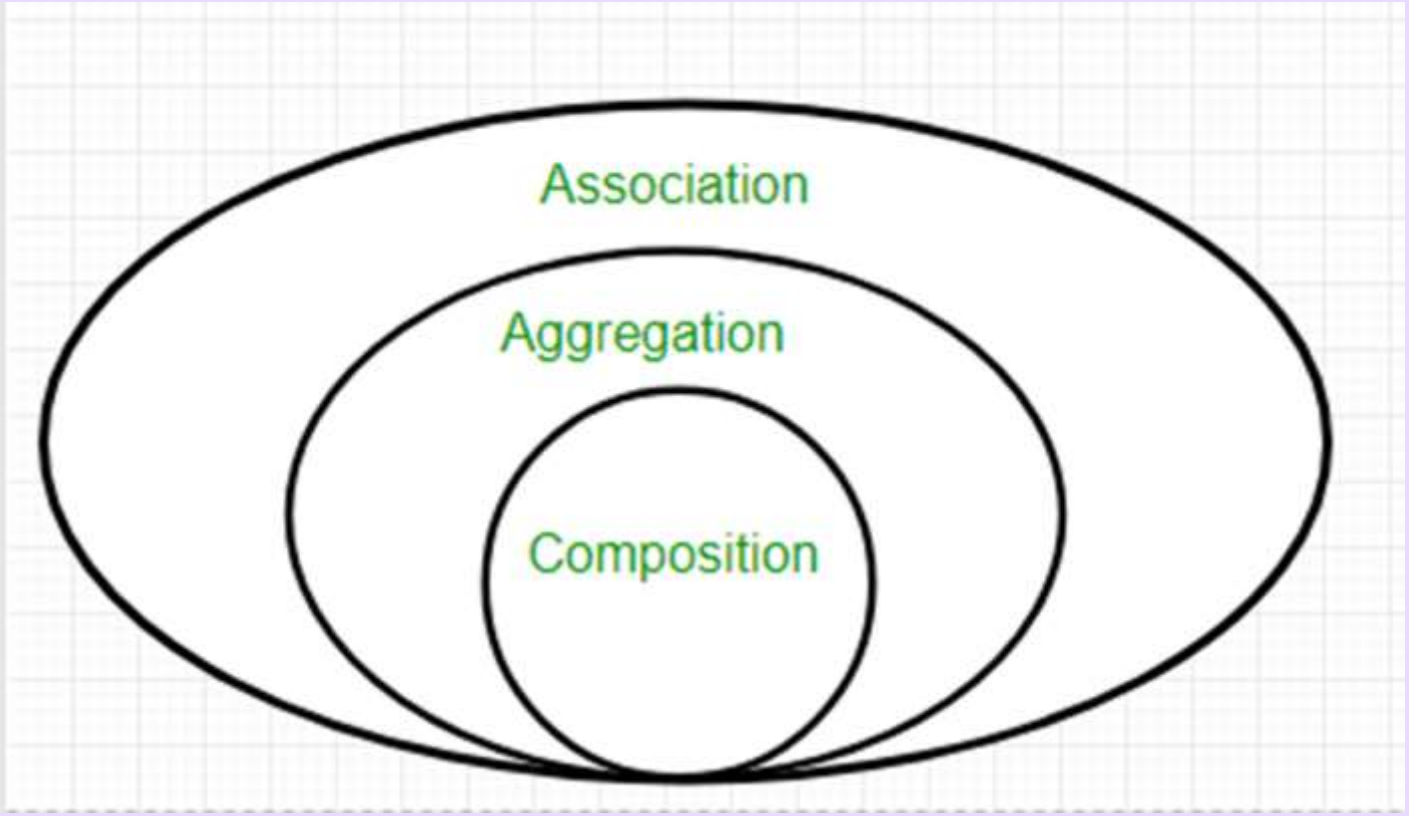
Entre los diferentes tipos de relaciones que hay entre clases, se abordará el tipo de relación de asociación y sus divisiones:

- **Asociación:** Línea entera. Representa un vínculo entre dos clases.
- **Agregación:** Línea entera con un diamante blanco al final.
- **Composición:** Línea entera con diamante negro al final.

Relación de asociación, agregación y composición.

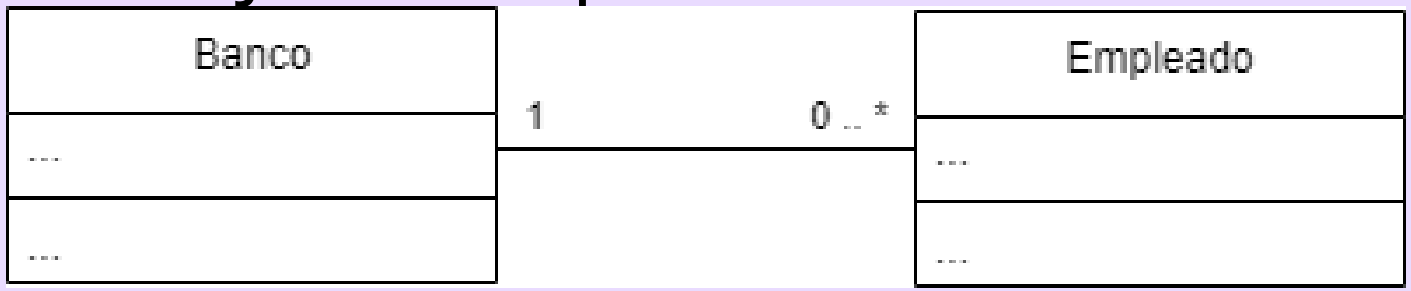
Asociación	Agregación	Composición
		
flecha direccionada:	**flecha de diamante vacía**	**flecha de diamante llena**
Cuando un objeto de una clase trabaja con otro objeto de otra clase por un periodo prolongado de tiempo.	Cuando un objeto forma parte de otra clase, pero sigue teniendo una existencia por sí misma.	Cuando un objeto de una clase hace parte de otro objeto de otra clase y se compone de ese objeto.

Relación de asociación, agregación y composición.



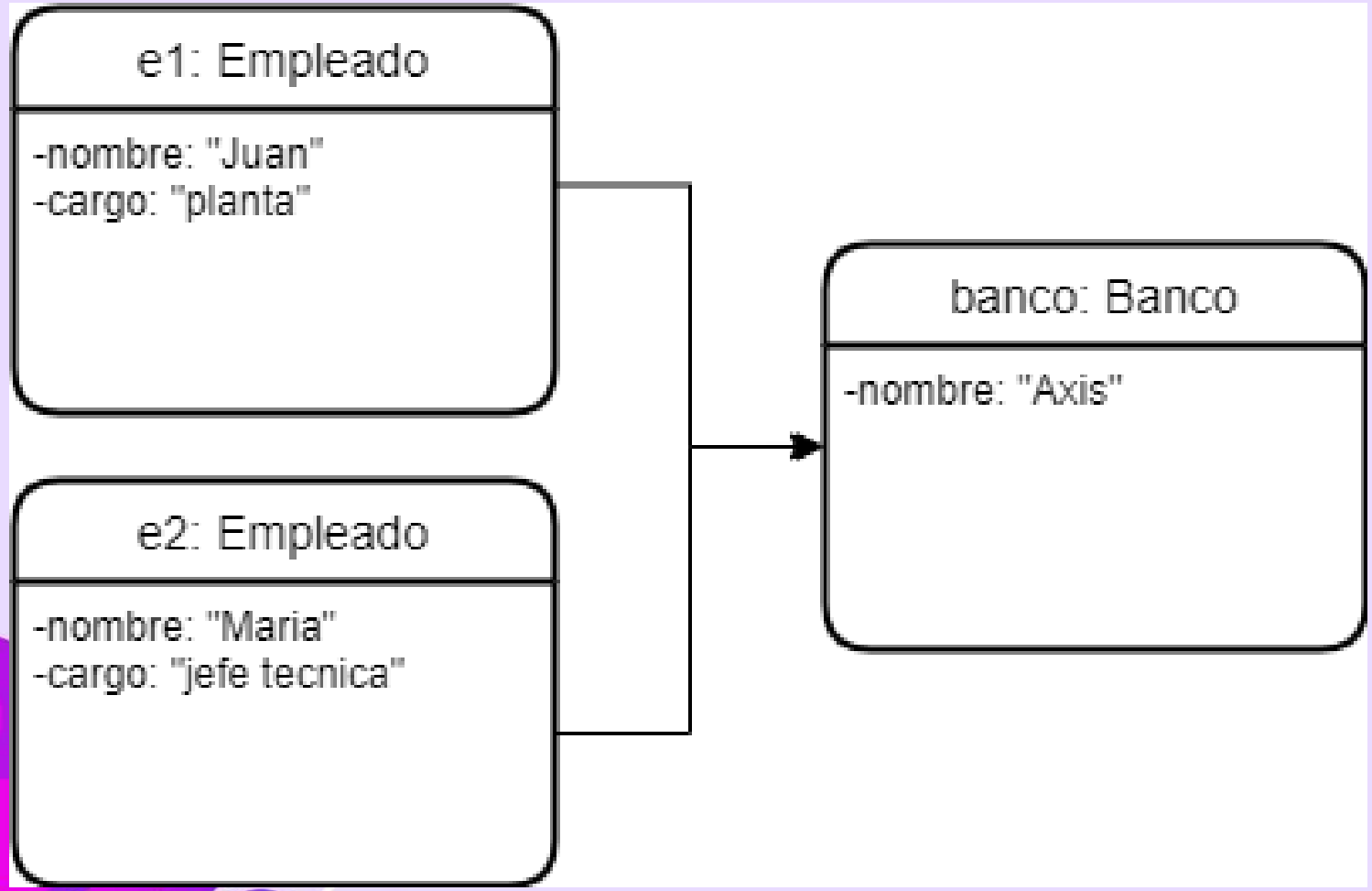
Asociación

La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre sí, para alcanzar una meta. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.



- Ejemplo: Un banco puede tener cero o muchos empleados vinculados, pero un empleado puede tener solo un banco.

Asociación


e1: Empleado
-nombre: "Juan"
-cargo: "planta"
e2: Empleado
-nombre: "Maria"
-cargo: "jefe tecnica"
banco: Banco
-nombre: "Axis"

Asociación

```java
class Empleado{
    private String nombre;
    private Banco banco;

    public Empleado(String nombre) {
        this.nombre = nombre;
    }
    public String obtenerNombreEmpleado() {
        return this.nombre;
    }
    public void vincularBanco(Banco banco){
        this.banco = banco;
    }
    public String obtenerNombreBanco(){
        return this.banco.obtenerNombreBanco();
    }
}
```

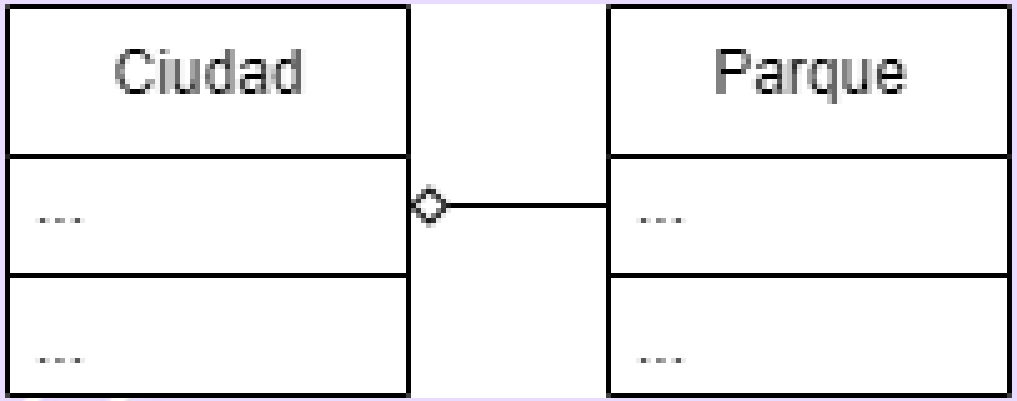
```java
class Banco{
    private String nombre;

    public Banco(String nombre) {
        this.nombre = nombre;
    }
    public String obtenerNombreBanco() {
        return this.nombre;
    }
}
```

Asociación


```
public class Asociacion{
    public static void main(String[] args) {
        Banco banco = new Banco("Axis");
        Empleado empleado = new Empleado("Andrea");
        empleado.vincularBanco(banco);
        System.out.println("El/la empleado/a "+empleado.obtenerNombreEmpleado()+
                " pertenece al banco "+empleado.obtenerNombreBanco());

    }
}
```

Agregación

La agregación es un tipo de asociación que indica que una clase es parte de otra clase (composición débil). La destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición.



Ejemplo:
Un objeto de tipo Ciudad tiene una lista de objetos de tipo Parque, una ciudad tiene un número de parques. La cardinalidad del extremo que lleva el rombo es siempre uno.

Agregación

```java
class Ciudad{
    private String nombre;
    private List<Parque> parques;

    public Ciudad(String nombre, ArrayList<Parque> parques) {
        this.nombre = nombre;
        this.parques = parques;
    }
    public void agregarParque(Parque nuevoParque){
        this.parques.add(nuevoParque);
    }

    public List<Parque> listaParques(){
        return this.parques;
    }
}
```

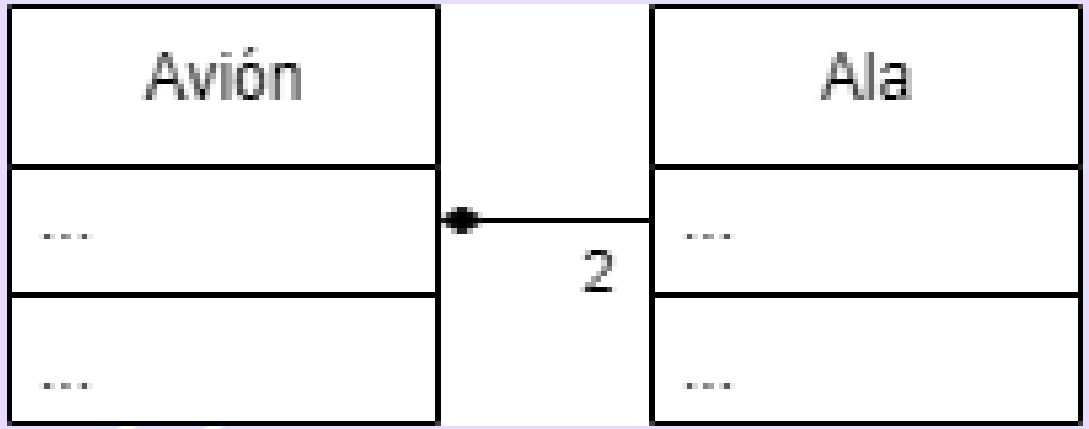
```java
class Parque{
    private String nombre;
    private double area;

    public Parque(String nombre, double area) {
        this.nombre = nombre;
        this.area = area;
    }
    public String getNombre() {
        return nombre;
    }
    public double getArea() {
        return area;
    }
}
```

Agregación

```java
public class Agregacion{
    public static void main(String[] args) {
        Parque parque1 =
                new Parque("""
                        Parque Central Simon Bolivar
                        """, 2000.40);
        Parque parque2 =
                new Parque("""
                        Jardín Botánico de Bogotá Jose Celestino Mutis
                        """, 1850.2);
        Parque parque3 =
                new Parque("""
                        Parque Metropolitano Timiza
                        """, 1580.23);
        Ciudad ciudad = new Ciudad("Bogotá DC", (ArrayList<Parque>) Arrays.asList(parque1));
        ciudad.agregarParque(parque2);
        ciudad.agregarParque(parque3);
        for(Parque parque : ciudad.listaParques()){
            System.out.println(parque.getNombre()+" "+parque.getArea());
        }
    }
}
```

Composición

Composición es una forma fuerte de composición donde la vida de la clase contenida debe coincidir con la vida de la clase contenedor. Los componentes constituyen una parte del objeto compuesto. La supresión del objeto compuesto conlleva la supresión de los componentes.



Ejemplo:
El avión tiene sentido por sí solo, pero está claro que está compuesto de 2 alas, esta relación es de mucha fuerza, mucho más que el caso de los parques.

Composición

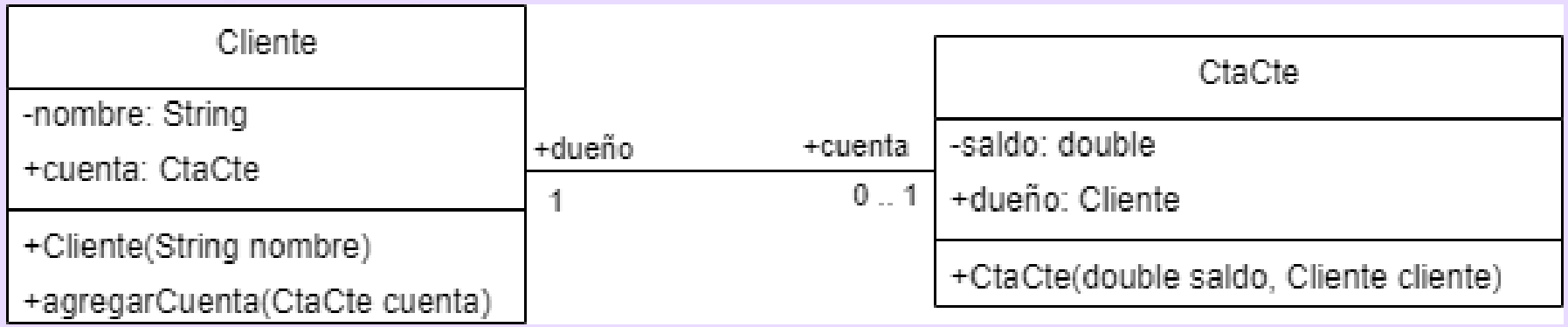
```java
class Avion{
    private String REF;
    private double peso;
    private final List<Ala> alas;
    public Avion(String REF, double peso) {
        this.REF = REF;
        this.peso = peso;
        Ala alalRef222l = new Ala("REF:222_left",12);
        Ala ala2Ref222r = new Ala("Ref:222_right",12);
        this.alas = (ArrayList<Ala>) Arrays.asList(alalRef222l,ala2Ref222r);
    }
    public String getREF() {...3 lines }
    public double getPeso() {...3 lines }
    public List<Ala> getAlas() {...3 lines }
    public String getStringAlas(){
        String infoAlas = "";
        for(Ala ala : alas){
            infoAlas += (ala.getREF() + " Longitud: "+
                    ala.getLongitud() +"\n");
        }
        return infoAlas;
    }
}
```

```java
class Ala {
    private String REF;
    private double longitud;

    public Ala(String REF, double longitud) {
        this.REF = REF;
        this.longitud = longitud;
    }
    public String getREF() {
        return REF;
    }
    public double getLongitud() {
        return longitud;
    }
}
```

Composición

```
public class Composicion{
    public static void main(String[] args) {

        Avion avionR24 = new Avion("Aerolinea AAA - REF:%24",124.2);
        System.out.println(avionR24.getREF()+", peso igual a: "+
                avionR24.getPeso()+
                "\n"+
                avionR24.getStringAlas()
                );
    }
}
```

Asociación bidireccional




```java
public class Cliente{
    private String nombre;
    public CtaCte cuenta;

    public Cliente(String nombre){
        this.nombre = nombre;
    }
    public void agregarCuenta(CtaCte cuenta){
        this.cuenta = cuenta;
    }
}
```

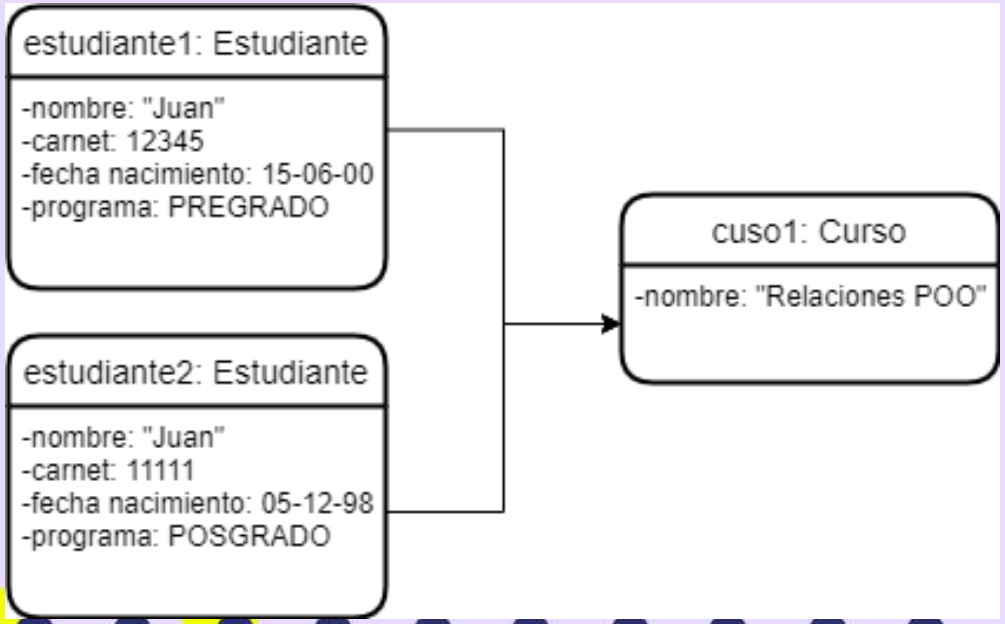
```java
public class CtaCte {
    private double saldo;
    public Cliente dueño;

    public CtaCte(double saldo, Cliente cliente){
        this.saldo = saldo;
        this.dueño = cliente;
    }
}
```

Diferencia entre agregación y composición

	Agregación	Composición
Varias asociaciones comparten los componentes	Sí	No
Destrucción de los componentes al destruir el compuesto	No	Sí
Cardinalidad a nivel de compuesto	Cualquiera	0..1 ó 1
Representación	Rombo transparente	Rombo negro

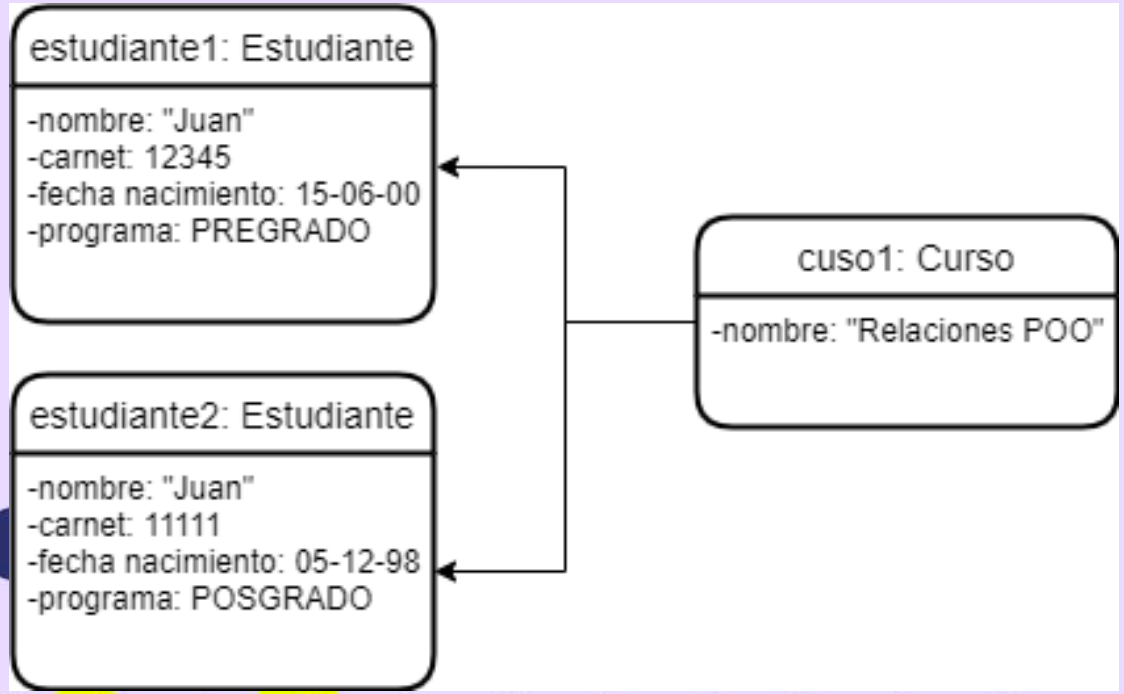
Navegabilidad

- Es como las instancias de una clase pueden acceder las unas a las otras
- En este caso la navegabilidad con la dirección de la flecha nos indica que las instancias de estudiantes pueden acceder a la instancia de curso que están tomando, pero el curso no sabe que estudiantes lo toman.

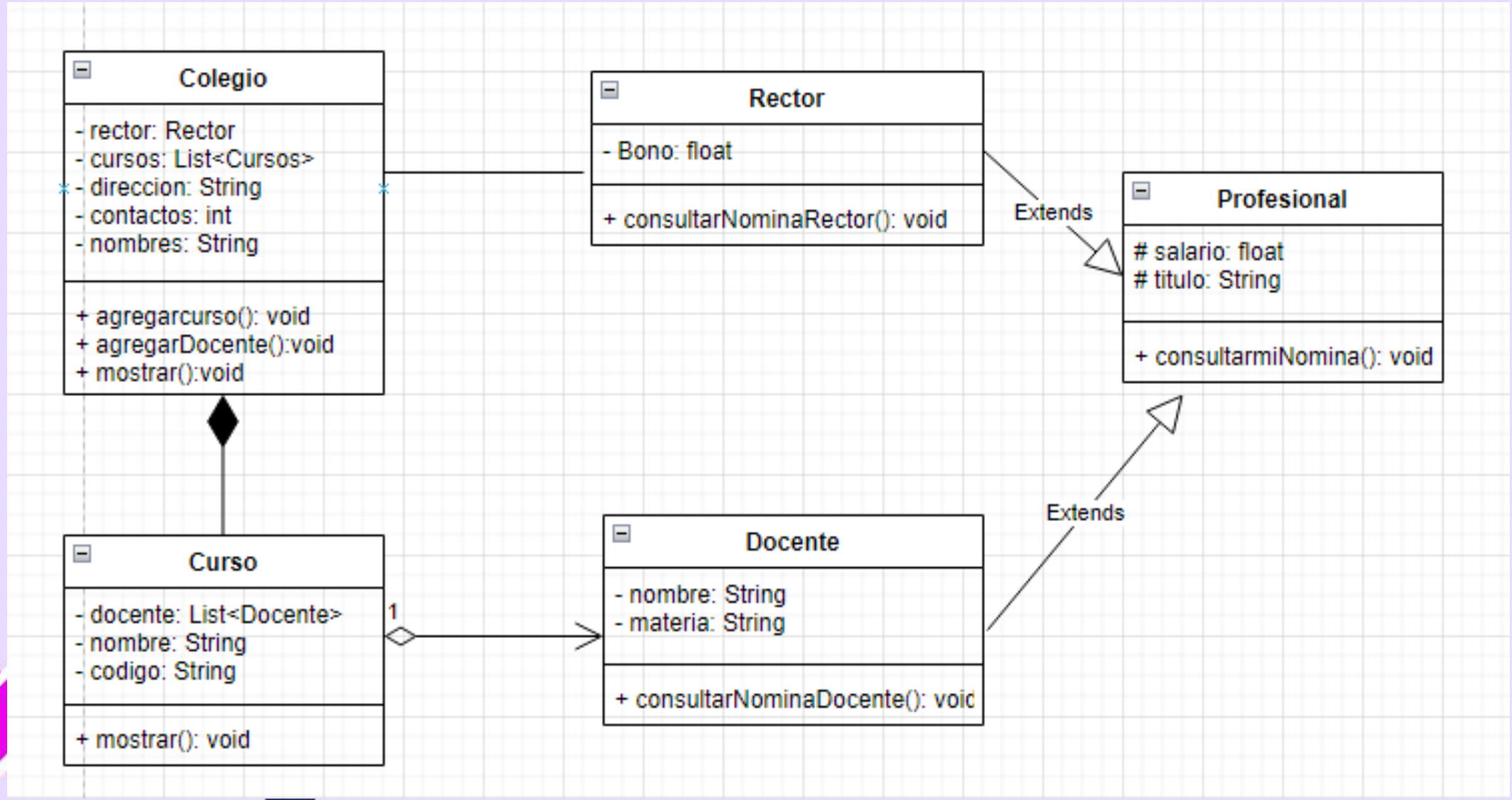


Navegabilidad

- En este caso la navegabilidad con la dirección de la flecha nos indica que la instancia de curso puede acceder a los estudiantes que están tomando el curso, pero los estudiantes no saben que curso toman.



Ejercicio práctico de asociación, agregación y composición.


Colegio
- rector: Rector
- cursos: List<Cursos>
- direccion: String
- contactos: int
- nombres: String
+ agregarcurso(): void
+ agregarDocente():void
+ mostrar():void
Rector
- Bono: float
+ consultarNominaRector(): void
Extends
Profesional
salario: float
titulo: String
+ consultarmiNomina(): void
Curso
- docente: List<Docente>
- nombre: String
- codigo: String
+ mostrar(): void
1
Docente
- nombre: String
- materia: String
+ consultarNominaDocente(): void
Extends