Programación Orientada a Objetos

Dra. Maricela Claudia Bravo Contreras

mari_clau_18@hotmail.com

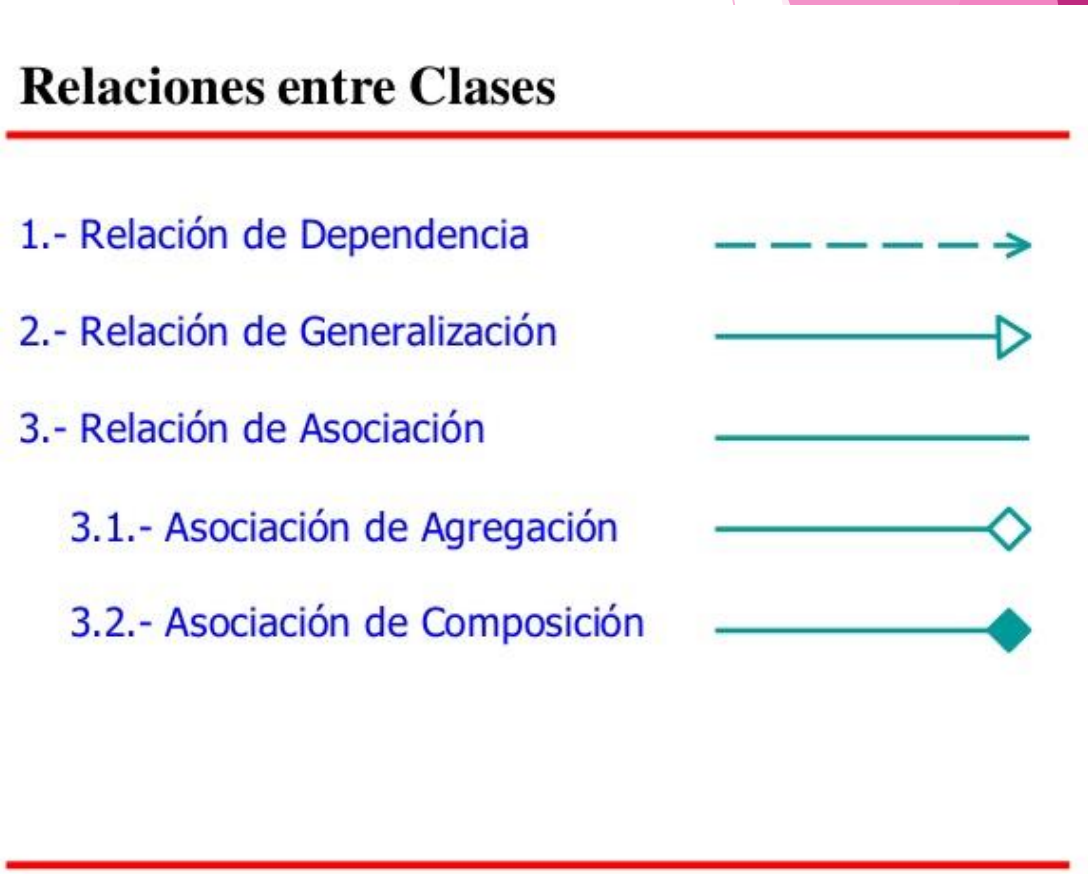
Relaciones de Agregación y Composición

Relaciones entre clases y objetos

- Las clases y objetos no pueden existir aislados y en consecuencia existirán relaciones entre ellos.
- Al igual que ocurre con los conceptos, entes del mundo real, que entre ellos hay relaciones de distinto tipo.
- Las relaciones se expresan frecuentemente utilizando verbos o frases con verbo del lenguaje natural, tales como *vive-en, participa-en, trabaja-para, está compuesto de*.
- Por ejemplo:
  - Periodista *participa* en Fiesta
  - Gerente *paga* la Factura
  - Semáforo *controla* Trafico
- Las relaciones entre clase pueden indicar alguna forma de compartir, así como algún tipo de conexión semántica.

Tipos de Relaciones

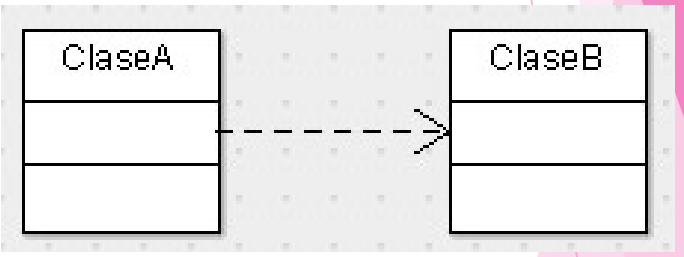
- Los tipos de relaciones entre clases son:
  - Asociación
  - Agregación
  - Composición
  - Dependencias



Dependencia

Es una relación de uso entre dos clases (una usa a la otra).

Esta relación es la más básica entre clases y comparada con los demás tipos de relación, la más débil.



Interpretación

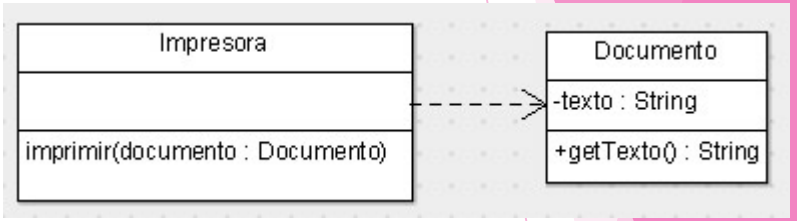
- La ClaseA usa a la ClaseB.
- La ClaseA depende de la ClaseB.
- Dada la dependencia, todo cambio en la ClaseB podrá afectar a la ClaseA.
- La ClaseA conoce la existencia de la ClaseB pero la ClaseB desconoce que existe la ClaseA.

En la practica este tipo de relación se interpreta como que la *ClaseA* hace uso de la *ClaseB* ya sea instanciandola directamente, o bien, recibiéndola como parámetro de entrada en uno de sus métodos.

Dependencia

```
/* Clase Documento */
class Documento
{
    private String texto;

    public Documento(String texto)
  {
        this.texto = texto;
    }

    public String getTexto()
   {
        return this.texto;
    }
}
```

```
/* Clase Impresora */

class Impresora

{

    public Impresora()

{

    }

    public void imprimir(Documento
documento) {

        String texto = documento.getTexto();

        System.out.println(texto);

    }

}
```




```
Documento miDocumento = new
Documento("Hello World!");

Impresora miImpresora = new Impresora();

miImpresora.imprimir(miDocumento);
```

Asociación

La asociación se podría definir como el momento en que dos objetos se unen para trabajar juntos y así, alcanzar una meta.

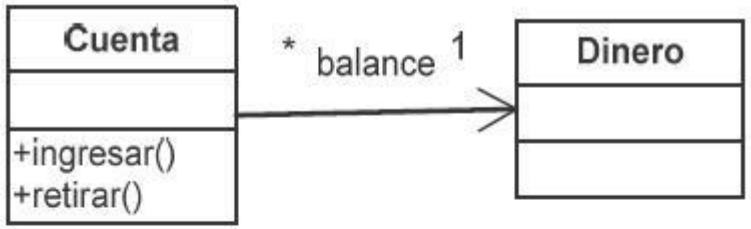
Un punto a tomar muy en cuenta es que ambos objetos son independientes entre sí.

Para validar la asociación, la frase *"Usa un"*, debe tener sentido:

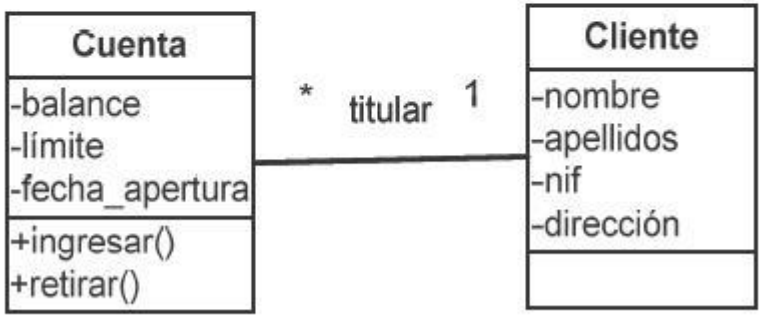
El ingeniero *usa* una computadora

El cliente *usa* tarjeta de crédito.

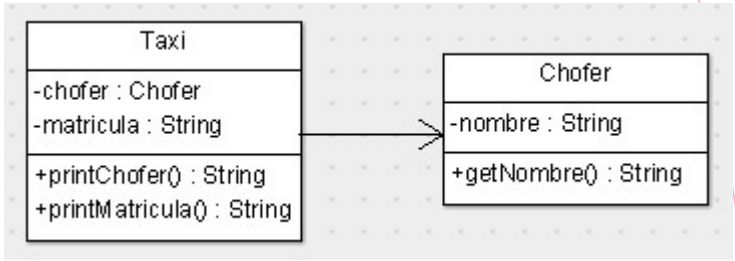
Ejemplos de Asociación en UML



Asociación unidireccional



Asociación bidireccional

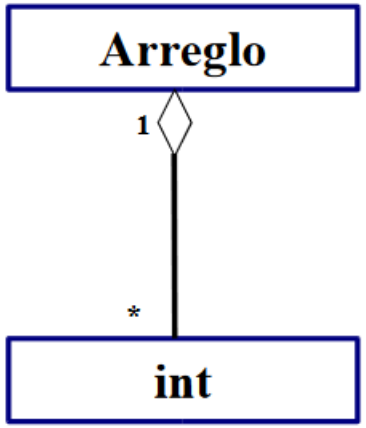


Ejemplo de Asociación
El cliente *usa* tarjeta de crédito

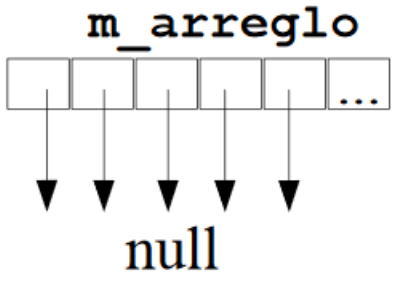
```
public class Cliente
{
    private int id;
    private String nombre;
    private String apellido;
    private CreditCard tarjeta;
    public Cliente() {    }
    public void setTarjeta(CreditCard tarjeta) {
        this.tarjeta = tarjeta;
    }
}
```

Agregación

- Relación todo/partes en donde una clase representa el todo y otra una de sus partes
- Se realiza declarando referencias dentro de los atributos (rombo vacío en UML)




```
Class Arreglo
{
    int m_arreglo[];
    int m_tamanio;
    public Arreglo(int tamanio)
     {
        m_tamanio=tamanio;
        m_arreglo=new int[m_tamanio]
     }     // Agregación simple
    public agregarElemento()
     }
        ...........
     {
}
```



Composición

- En caso contrario, la composición es un tipo de relación **dependiente** en dónde un objeto más complejo es conformado por objetos más pequeños.
- En esta situación, la frase "*Tiene un*", debe tener sentido:
- **El auto** ***tiene*** **llantas**
- **La portátil** ***tiene*** **un teclado.**

La portátil *tiene un* teclado

```
public class Laptop
{
    private String marca;
    private String modelo;
    private String serviceTag;
    private KeyBoard keyBoard = new KeyBoard();

    public Laptop() {
        //Lo que sea que el constructor haga
    }
}
```

```
public class Principal
{

    private Teclado teclado = new Teclado(); // Ejemplo de
                                             //relación de composición

   public static void main(String [] argc)
    {
          Principal p = new Principal();

     // Aquí se cachan excepciones que pueda generar el menu.
     try
      {
           p.pruebaConMenu();
      }
     catch(Exception ex)
      {
         ex.printStackTrace();
      }
}
```