UML (Unified Modeling Language)

El lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el Object Management Group (OMG).

Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Es importante remarcar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo.

Se puede aplicar en el desarrollo de software gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional, Rational Unified Process o RUP), pero no especifica en sí mismo qué metodología o proceso usar.

UNIDAD 1: Fundamentos de la Programación Orientada a Objetos

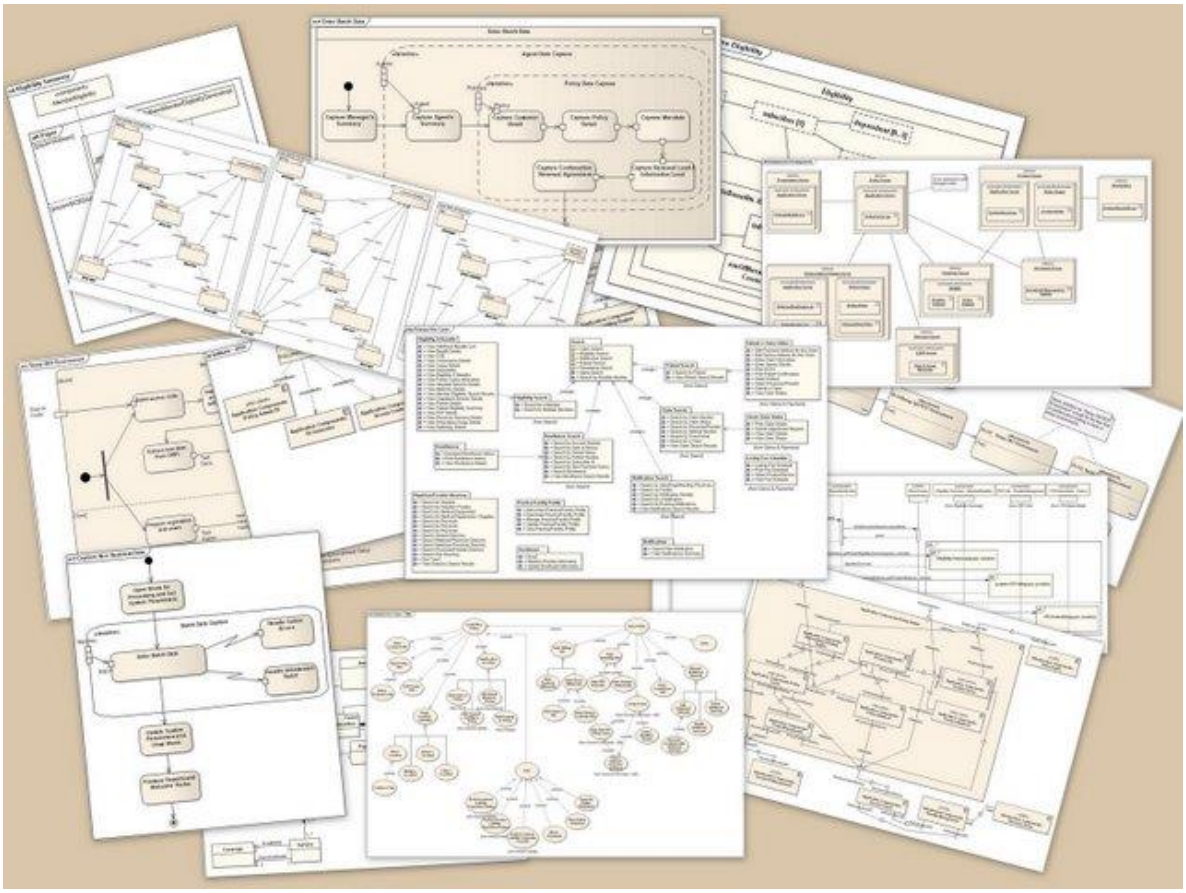
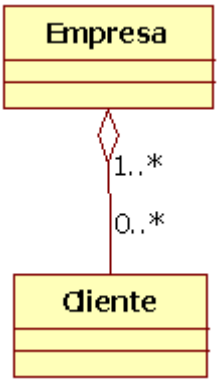


Diagrama de clases en Lenguaje unificado de Modelado (UML) es un tipo de diagrama de estructura estática que describe la estructura de un sistema mostrando las clases del sistema, sus atributos, operaciones (o métodos), y las relaciones entre los objetos.

Agregación

La agregación es un tipo de asociación que indica que una clase es parte de otra clase (composición débil). Los componentes pueden ser compartidos por varios compuestos (de la misma asociación de agregación o de varias asociaciones de agregación distintas). La destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición. La agregación se representa en UML mediante un diamante de color blanco colocado en el extremo en el que está la clase que representa el “todo”.

Veamos un ejemplo de agregación:

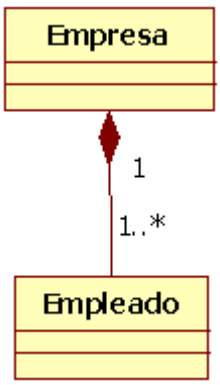


- Tenemos una clase Empresa.
- Tenemos una clase Cliente.
- Una empresa agrupa a varios clientes.

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. De esta forma, los componentes no pueden ser compartidos por varios objetos compuestos. La supresión del objeto compuesto conlleva la supresión de los componentes.
El símbolo de composición es un diamante de color negro colocado en el extremo en el que está la clase que representa el “todo” (Compuesto).

Veamos un ejemplo de composición:



• Tenemos una clase Empresa.
• Un objeto Empresa está a su vez compuesto por uno o varios objetos del tipo empleado.
• El tiempo de vida de los objetos Empleado depende del tiempo de vida de Empresa, ya que si no existe una Empresa no pueden existir sus empleados.

Diferencias entre Composición y Agregación

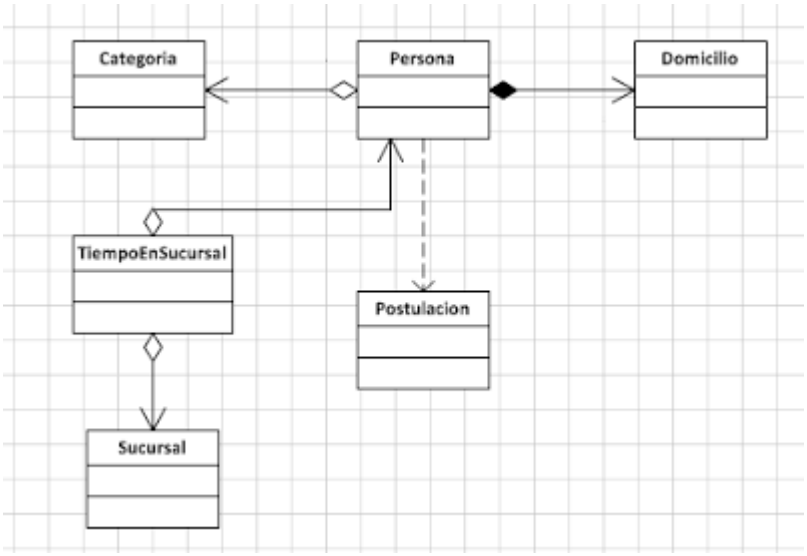
La siguiente tabla intenta resumir algunas diferencias entre agregación y composición.

	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

Ejemplo de Asociaciones

Diagrama de Clases

UNIDAD 1: Fundamentos de la Programación Orientada a Objetos



Codigo Java

```
class Persona
{
    //Relación de composición: hago el new dentro de la clase
    Domicilio dom = new Domicilio();

    //Relación de agregación: no hago el new, ya viene desde afuera resuelto.
    Categoria cat;
    public Persona(Categoria c)
    {
        cat = c;
    }

    //Atributos de la clase
    public string Apellido { get; set; }
    public string Nombre { get; set; }
    public int Id { get; set; }

    //Relación de dependencia: no forma parte de la clase, es utilizada para hacer alguna de sus operaciones
    public void Postularse()
    {
        Postulacion p = new Postulacion();
        p.Enviar();
    }

}
```

```
class Categoria
{
    public int Id { get; set; }
    public string IdCategoria { get; set; }
}
class Domicilio
{
    public int PersonaId { get; set; }
    public string Calle { get; set; }
    public int Numero { get; set; }
}
class Postulacion
{
    public int Id { get; set; }
    public void Enviar() { }

}
class Sucursal
{
    public int Id { get; set; }
    public string NombreSucursal { get; set; }
}
class TiempoEnSucursal
{
    //Atributos relacionales
    public Sucursal Sucursal { get; set; }
    public Persona Persona { get; set; }

    //Atributos de la clase
    public DateTime FechaInico { get; set; }
    public DateTime FechaFin { get; set; }
}
```

Fuentes:

https://www.seas.es/blog/informatica/agregacion-vs-composicion-en-diagramas-de-clases-uml/

http://alumnosdc.blogspot.com.co/2013/05/agregacion-composicion-y-dependencia.html