**UML**

**Clase:** se representa con un **rectángulo** con el **nombre** de la clase,  se define utilizando la palabra clave class, seguida del nombre de la clase y un bloque de código que contiene los atributos y métodos.

**Atributos:**  se representan como **variables** dentro del **rectángulo** de la clase y tienen distinto **modo de acceso** + público ,  - privado o # protected

**Métodos** se representan como **funciones** dentro del **rectángulo** de la clase, son funciones asociadas a una clase. saludar() y obtenerEdad().

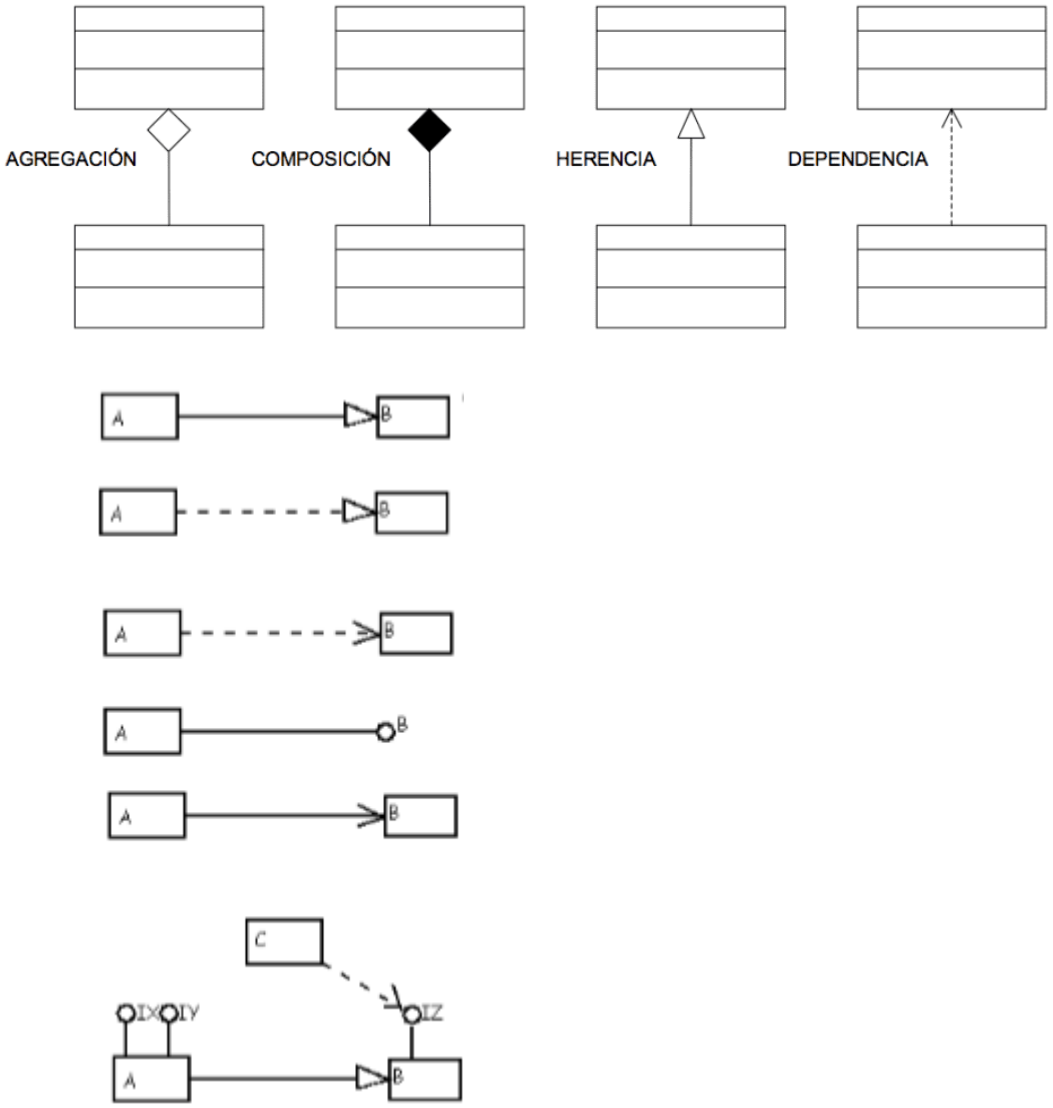
```
public class Persona {
   // Atributos
    private String nombre;
    private int edad;

  // constructor
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }

   // Métodos
    public void saludar() {
        System.out.println("¡Hola! Mi nombre es " + nombre);
    }

    public int obtenerEdad() {
        return edad;
    }
}
```

**Relaciones entre Clases**: se representan con **líneas** continuas o discontinuas que apunta desde la subclase hacia la superclase.




```
public class Empleado extends Persona {
    private double salario;

    public Empleado(String nombre, int edad, double salario) {
        super(nombre, edad);
        this.salario = salario;
    }

    public double obtenerSalario() {
        return salario;
    }}
```

## Reglas clave para la sobrescritura

1. El método sobrescrito debe tener los mismos parámetros que el método base. Si no, se considera una sobrecarga (overload).
2. El nivel de acceso del método sobrescrito no puede ser más restrictivo que el declarado en el método base.
3. Una subclase dentro del mismo paquete puede sobrescribir cualquier método que no esté declarado como private o final.
4. Una subclase fuera del paquete puede sobrescribir solo métodos public o protected que no estén marcados como final.
5. El método sobrescrito puede lanzar cualquier excepción de tiempo de ejecución.
6. No se puede sobrescribir un método marcado como final o static.

## Java.time

Java.time más usados : LocalDate LocalTime LocalDateTime Instant Duration Period

se instancia con : now of parse
se compara con : isBefore, isAfter