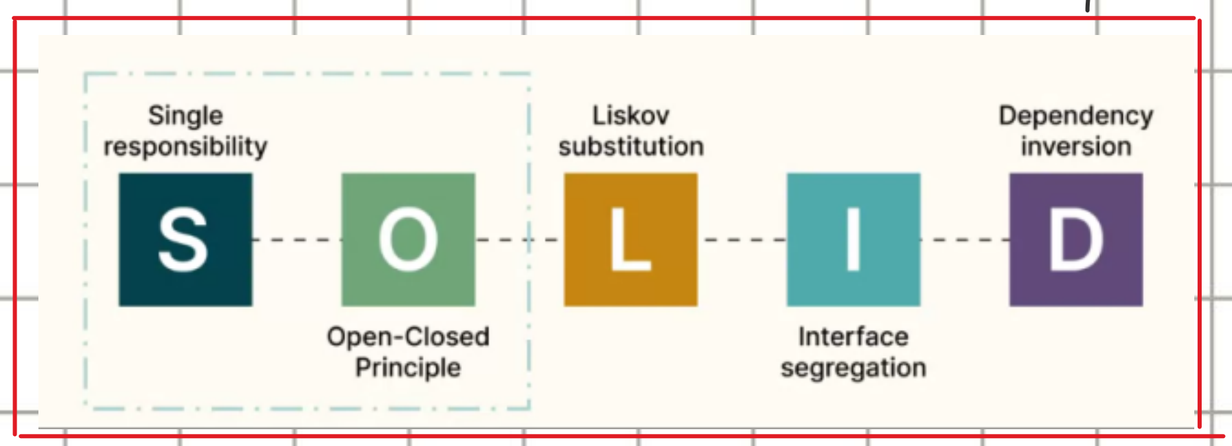
POO - TDA - UML

POO es un **paradigma de programación** en el cual **pensamos todo como objetos** partiendo desde objetos simples que interactúan entre sí y se complejizan para resolver un problema.

Clases una clase es una "plantilla" que define las **propiedades** (**atributos**) y **comportamientos** (**métodos**) **de un objeto**, es decir, de una instancia de sí misma.

La idea es **ordenar el código en clases** haciendo que **cada una respete su razón de existir** (**responsabilidad**) sin exponerse a otras clases (romper el **encapsulamiento**) y respetando los Principios



Responsabilidad Las clases que creamos tienen que responder fácilmente qué hacen.

Por ejemplo una clase "calculadora" hace cálculos matemáticos. si a la hora de explicarlo decimos que hace cálculos matemáticos, se comunica con el usuario, muestra un menú... es más de una clase.

Encapsulamiento Cada clase tiene que saber resolver sus propios problemas.

Si tengo una clase calculadora no tiene sentido que le pida el primer número, el segundo y por fuera lo sume porque la responsabilidad de la calculadora es sumar.

Capacidad de extender el código

Nuestro código tiene que permitir que agreguemos funcionalidades sin modificar lo que ya existe y no está relacionado

Por ejemplo, si ahora mi calculadora permite cálculos de raíces se debería poder resolver agregando un método nuevo, no debería ser necesario modificar las sumas, restas u otros métodos.

Ejercicio Crear una clase Persona con los atributos:

- Nombre (valor default Juan)
- Edad (valor default 20)

Además crear un método para la siguiente funcionalidad:

- mostrar el nombre y la edad de la persona

Usen lo visto de Poo para resolver este problema.

Main.java | Persona.java

```java
public class Persona {  2 usages

    private String nombre;  2 usages
    private Integer edad;  2 usages

    /** constructor ...*/
    public Persona(String nombre, Integer edad) {...}

    /**
     * Muestra por pantalla los datos de la persona
     */
    public void mostrarDatos() {  1 usage
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }
}
```

Constructor un constructor es un método que se llama al instanciar una clase para preparar a la misma para ser usada, por ejemplo inicializando los atributos del objeto.

Estos cuentan con algunas particularidades:

- No tienen tipo de retorno
- Llevan el mismo nombre que la clase
- Si no lo definimos se usa uno por defecto llamado "constructor de oficio" el cual es transparente para el desarrollador.

```
public Casa(String direccion, Integer ambientes) {
    this.direccion = direccion;
    this.ambientes = ambientes;
    antiguedad = 0;
}
```

Puede existir más de uno.

Ejercicio utilizando lo visto en constructores actualicen la clase Persona para que se guarden los datos de nombre y edad que el usuario le envie por consola.

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Ingrese su nombre: ");
        String nombre = sc.nextLine();

        System.out.println("Ingrese su edad: ");
        Integer edad = sc.nextInt();

        Persona persona = new Persona(nombre, edad);
        persona.mostrarDatos();
```

```
public class Persona {  2 usages
    private String nombre;  2 usages
    private Integer edad;  2 usages

    public Persona(String nombre, Integer edad) {  1 u
        this.nombre = nombre;
        this.edad = edad;
    }

    public void mostrarDatos() {  1 usage
        System.out.println("Nombre: " + nombre);
        System.out.println("Edad: " + edad);
    }
}
```

Resumen

Atributos Agregación de características/datos

Métodos Agregación de comportamiento.

Constructor Métodos para instanciar clases

Sobrecarga Múltiples versiones de un método que comparte nombre y difieren en retorno o cantidad de parámetros.

TDA Es un mecanismo de descripción de alto nivel que, al implementarse, genera una clase.

Es una estructura de datos definida por las operaciones que puede hacer y no por cómo hacerlas.

Fases

Diseño Independiente de lenguaje e implementación.

Pienso <u>qué</u> tiene que hacer ya sea definiendo firmas o con un UML.

Implementación Dependiente del lenguaje.

Pienso en <u>cómo</u> cumplir con lo comprometido en la fase de diseño.

Uso Llamado a los métodos disponibles en el TDA teniendo en cuenta sus pre y post condiciones.

TDA - Número complejo

Dado que queremos realizar cuentas básicas con números complejos necesitamos diseñar, implementar y utilizar un TDA para números complejos.

Nombre del TDA: Complejo

Dominio: Complejo = (real, imaginario), con real, imaginario $\in \mathbb{R}$

Operaciones

Complejo: $\mathbb{R} \times \mathbb{R} \longrightarrow$ Complejo: Construye un número complejo en base a dos números que conforman la parte real e imaginaria respectivamente.

Sumar: Complejo x Complejo $\longrightarrow$ Complejo.
Suma dos números complejos y retorna el resultado

Restar: Complejo x Complejo $\longrightarrow$ Complejo.
Restar dos números complejos y retorna el resultado

Módulo: Complejo $\longrightarrow \mathbb{R}$
Devuelve un valor real calculando como la raíz cuadrada dela suma de los cuadrados de la parte real y la parte imaginaria del número complejo.

UML

significa lenguaje de modelado unificado y sirve como documentación del código.

Si tenemos varias clases tenemos que hacer un UML.

El mismo puede ser: esquelético, es decir, que tiene solo cómo se relacionan las clases entre sí o más completo incluyendo atributos y métodos de cada clase.

Elementos

Clase

<Nombre de la clase>
<Atributos>
<Métodos>

Relaciones básicas

→	Herencia
→	Asociación

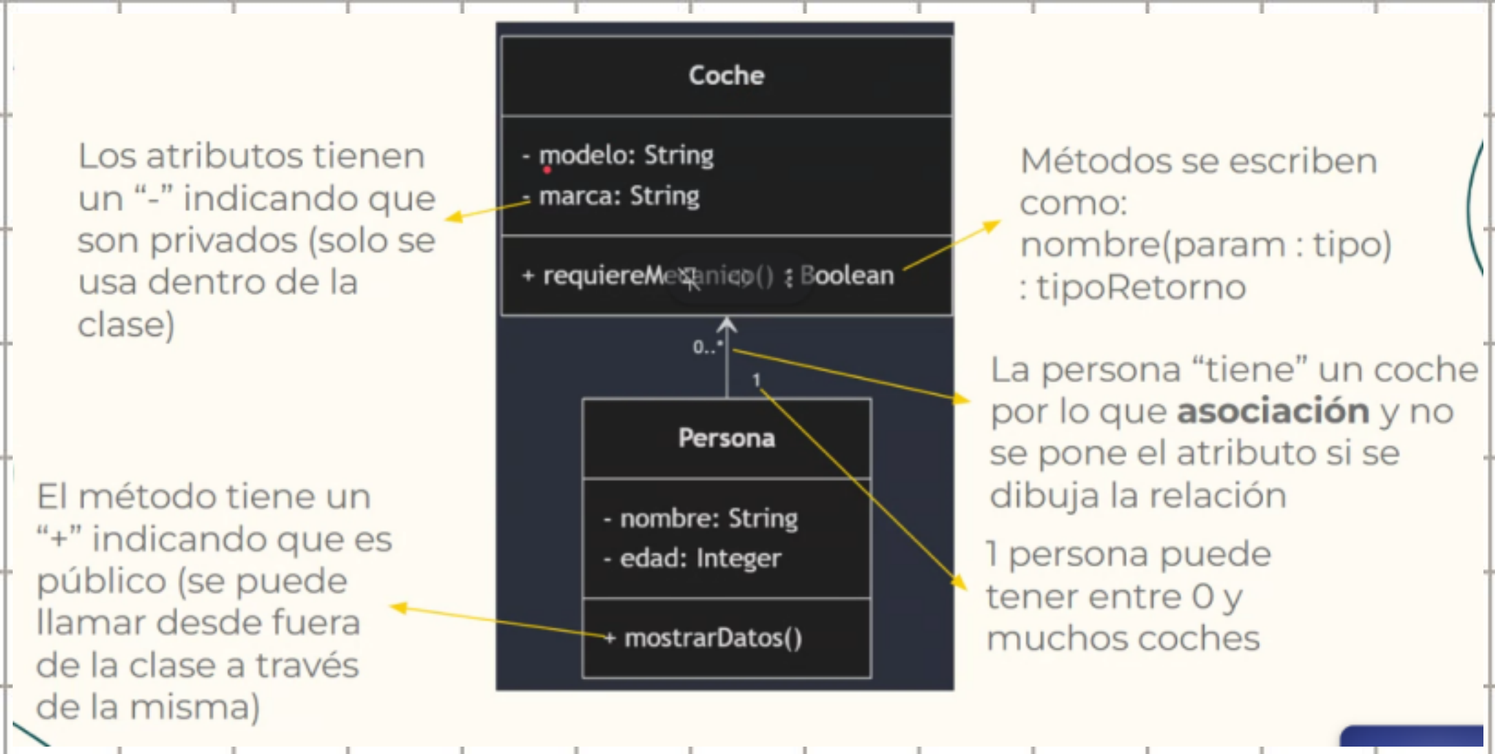
Visibilidad

+	Público
-	Privado
#	Protegido

Multiplicidad

1	Exactamente 1
0..1	Cero a uno
0..*	Cero a muchos
*	Muchos

Ejemplo



Ejercicio integrador

Realizar un programa que nos permita cargar notas de los alumnos de la materia y calcular si aprueban o no.

Las notas deberán ser un tipo de dato positivo que vaya del 1 al 10
Se deberá permitir cargar un alumno con sus notas y consultar si el mismo está aprobado.
Realice el UML.