Nombre y apellido alumno: Gomez Hertler Lisandro DNI: 46461232

Nombre y Apellido alumno: Firmapaz Gabriel Andrés DNI: 43205785

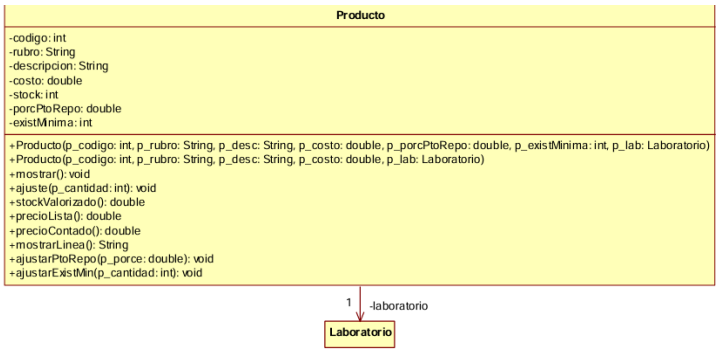
Profesores: Yanina Medina, Andrea Airaldi y Matías Mascazzini

Grupo de Laboratorio: Grupo 2 TP: 3

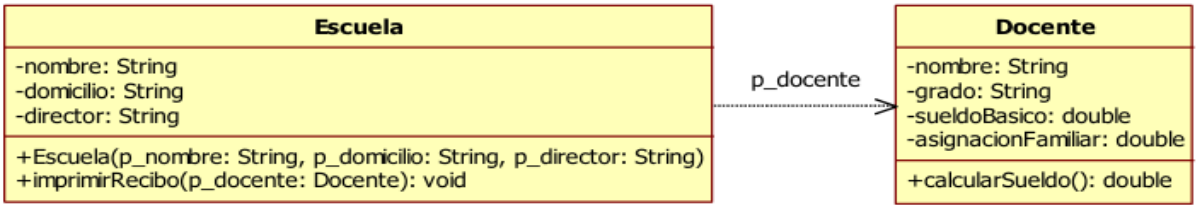
Fecha de entrega: 9/9/24

Conocimiento entre Objetos: Definen una relación entre un objeto y sus atributos. Constituyen la estructura de la clase/objeto. Las variables de instancia acompañan al objeto desde su creación hasta que muere.

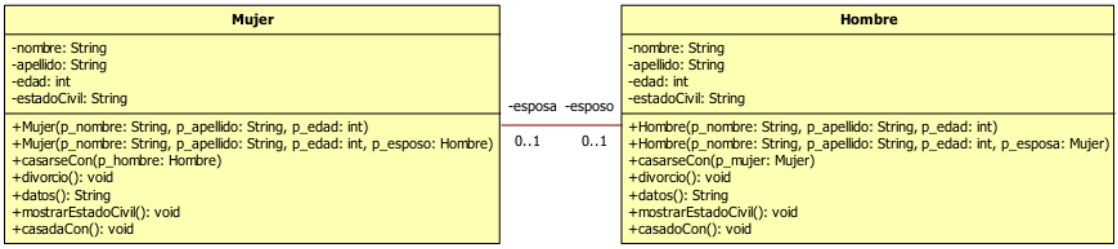
Por Variable de Instancia: En variables de instancia vimos en el ejemplo del punto 1 (Laboratorio) que instanciando un objeto Laboratorio dentro de la clase Producto, este objeto pasa a ser un atributo de la clase Producto. Por lo tanto, el objeto Producto conoce al objeto Laboratorio hasta que terminamos de ejecutar la clase Producto.



Por Parámetro: En el ejercicio 6 encontramos en el método imprimirRecibo() el uso del parámetro docente, parámetro que se utiliza solamente dentro de tal método (siendo este el receptor) y por lo tanto, dura todo el tiempo que dura el mismo método. Se recibe el parámetro para poder imprimir el recibo con los datos que contiene un objeto Docente, ej: nombre y tipos de sueldo.



Variable Temporal: En el caso de variable temporal encontramos un ejemplo en el ejercicio 10, donde dentro del método divorcio() debimos crear una variable temporal "consorte", en donde almacenamos temporalmente al objeto Esposa (desde el punto de vista del divorcio del esposo), para ejecutar el método divorcio() y divorciar también al objeto Esposa del Esposo.



Colaboradores: Los colaboradores son objetos o clases que interactúan entre sí para lograr una funcionalidad conjunta. Un objeto colaborador suele referirse a otro objeto dentro de una clase, ya sea por composición, agregación o dependencia. En este caso colaboran las clases Empleado y EmpleadoConJefe.

```
ublic class EmpleadoConJefe {
    private long cuil;
    private String apellido;
    private String nombre;
    private double sueldoBasico;
    private Calendar fechaIngreso;
    private EmpleadoConJefe jefe; // Referencia al jefe del empleado, si lo tiene
```

Ventajas de el ocultamiento de la información:

El encapsulamiento u ocultación de la información se implementa en los lenguajes de POO a través de la visibilidad o acceso a las variables miembro o atributos y a los métodos. La visibilidad es precisamente la propiedad que define si un atributo u operación de un objeto es accesible desde fuera del propio objeto. La encapsulación es un mecanismo para reunir datos y métodos dentro de una estructura ocultando la implementación del objeto, es decir, impidiendo el acceso a los datos por cualquier medio que no sean los servicios propuestos. La encapsulación permite, por tanto, garantizar la integridad de los datos contenidos en el objeto. Por lo tanto, si queremos proteger la información contra modificaciones inesperadas, debemos recurrir al principio de encapsulación. El desarrollador tiene control sobre cómo y cuándo se accede o modifica el estado interno de un objeto por ejemplo: los Observadores (getters) y Mutadores (setters).

Los atributos privados no pueden ser modificados directamente desde fuera de la clase. Por Ejemplo: puedeExtraer(p_importe: double): Boolean y extraccion(p_importe: double): void.

```
private boolean puedeExtraer(double p_importe) {
    return p_importe <= (this.getSaldo() + this.getLimiteDescubierto());
}
```

```
private void extraccion(double p_importe) {
    this.setSaldo(this.getSaldo() - p_importe);
}
```