Universidad Tecnológica de Durango Tecnologías de la Información Programación Orientada a Objetos

Actividades

*“Evidencias de Actividades y Tareas”*

Alumnos:

* Vargas Pérez Cristian Arturo

3°B T.I BIS

Docente:

* Ing. Dagoberto Fiscal Gurrola, M.T.I.

Julio 2024

*Tabla de ilustraciones*

[Ilustración 1 – Métodos, clases y objetos 3](#_bookmark0)

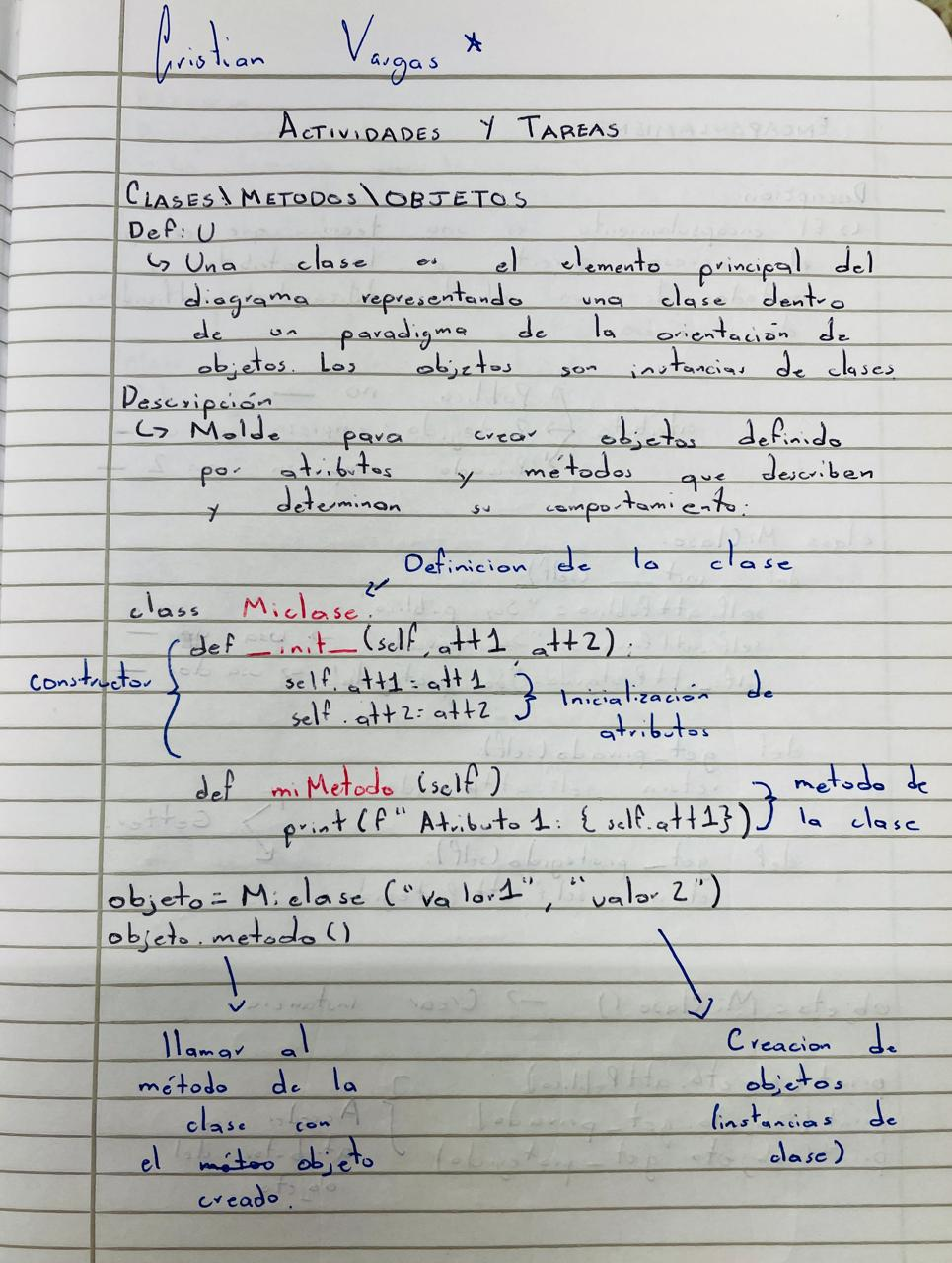
[Ilustración 2 – Encapsulamiento……………………………………………………. 4](#_bookmark1) [Ilustración 3 - Herencia 5](#_bookmark2)

[Ilustración 4 – Clase Abstracta y polimorfismo 6](#_bookmark3)

[Ilustración 5 - Sobreescritura de métodos y Casting 7](#_bookmark4)

[Ilustración 6 - Sintaxis métodos estáticos y no estáticos](#_bookmark9) ..8

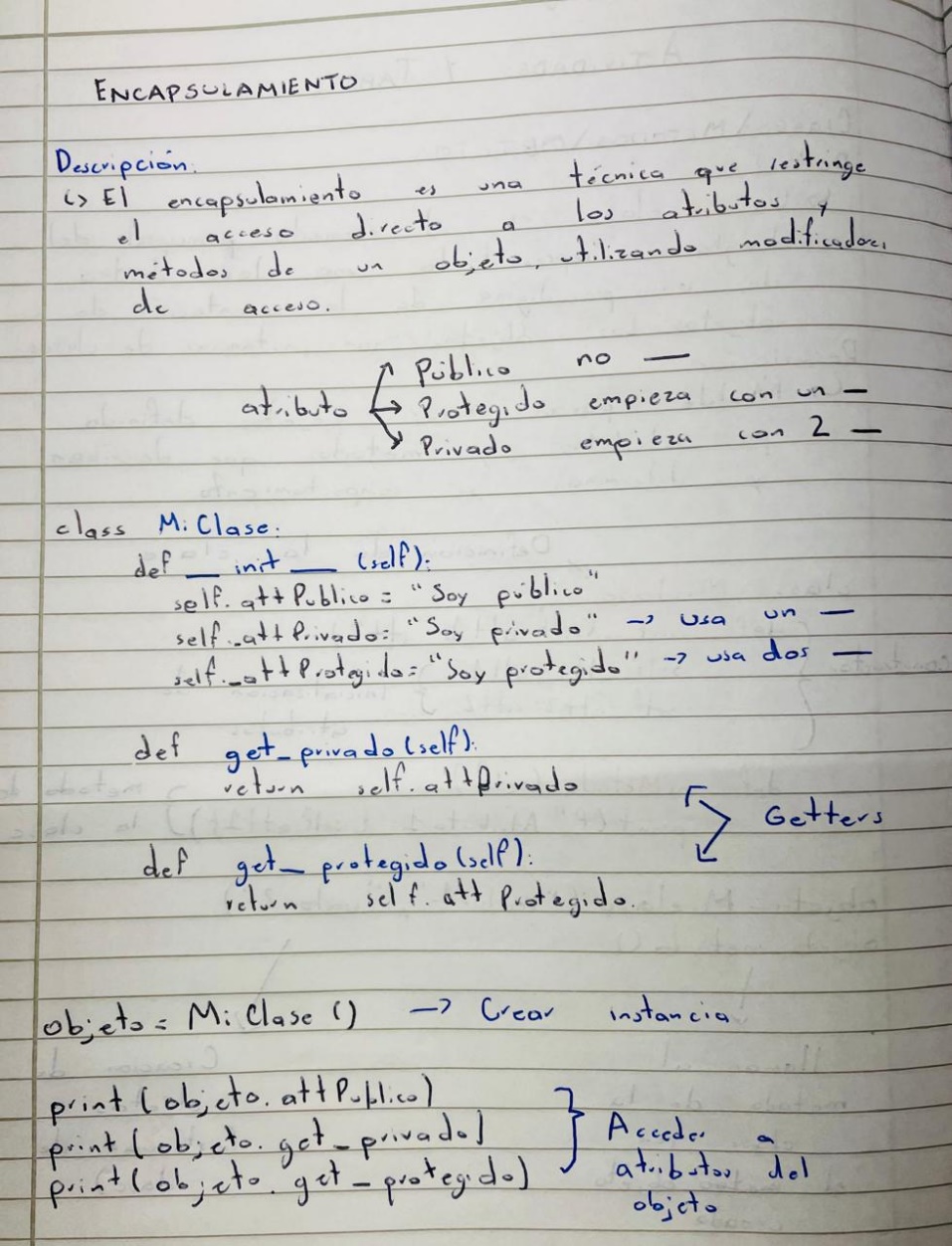
Programación Orientada a Objetos P á g i n a | **2**



*Ilustración 1 – Métodos, Clases Y Objetos*

En la imagen anterior se muestra la descripción y ejemplos de sintaxis de Clases, Métodos y Objetos, se presenta un ejemplo de una clase llamada MiClase y se hace uso de objetos y métodos(objeto, miMetodo). También se indican detalladamente las partes de la clase.

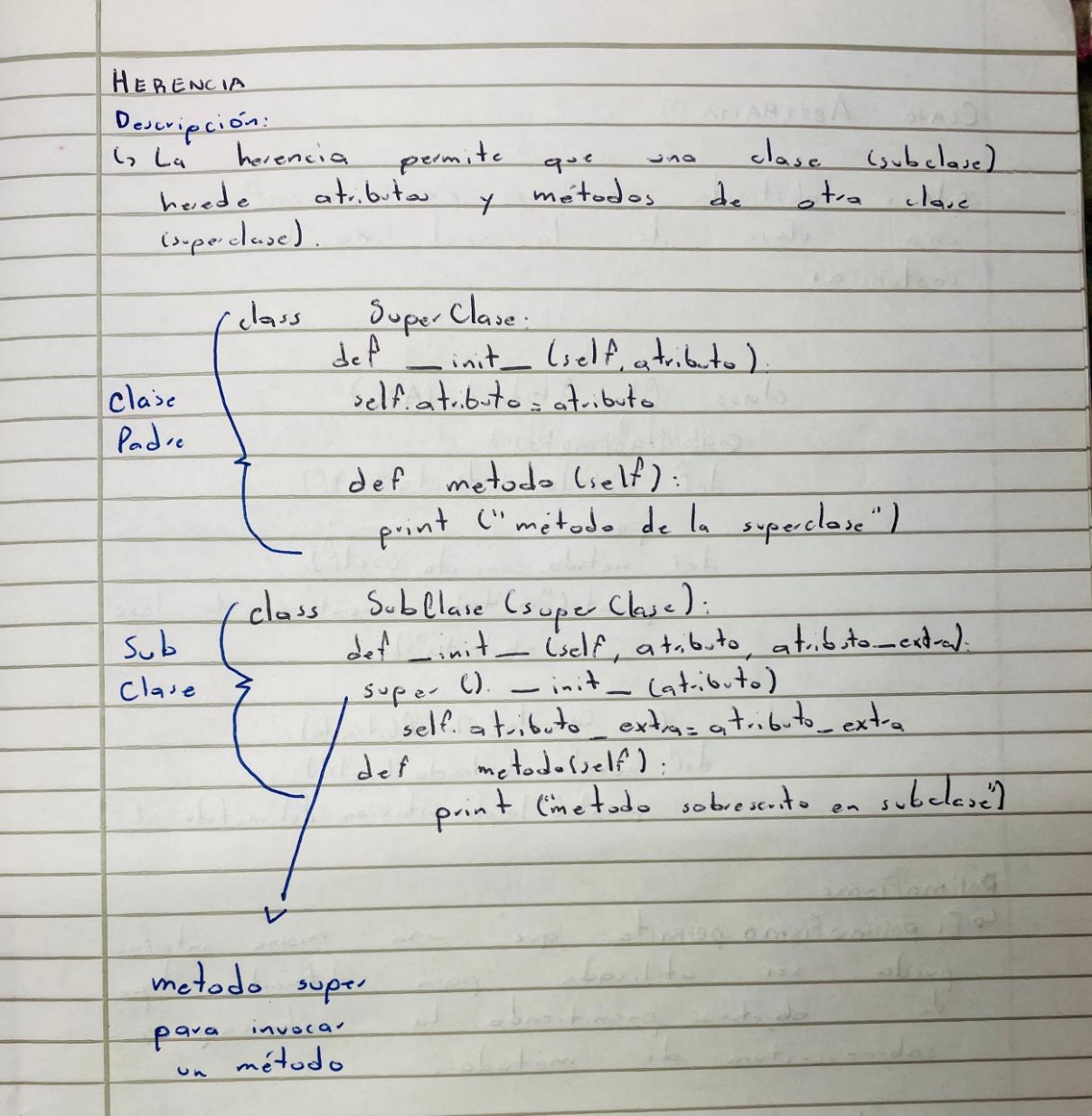
Programación Orientada a Objetos P á g i n a | **3**



*Ilustración 2 - Encapsulamiento*

En la imagen anterior se muestra la descripción y ejemplos de sintaxis de Encapsulamiento, se habla de atributos privados, protegidos y públicos se presenta un ejemplo de una clase llamada MiClase y se hace uso de dichos atibutos.

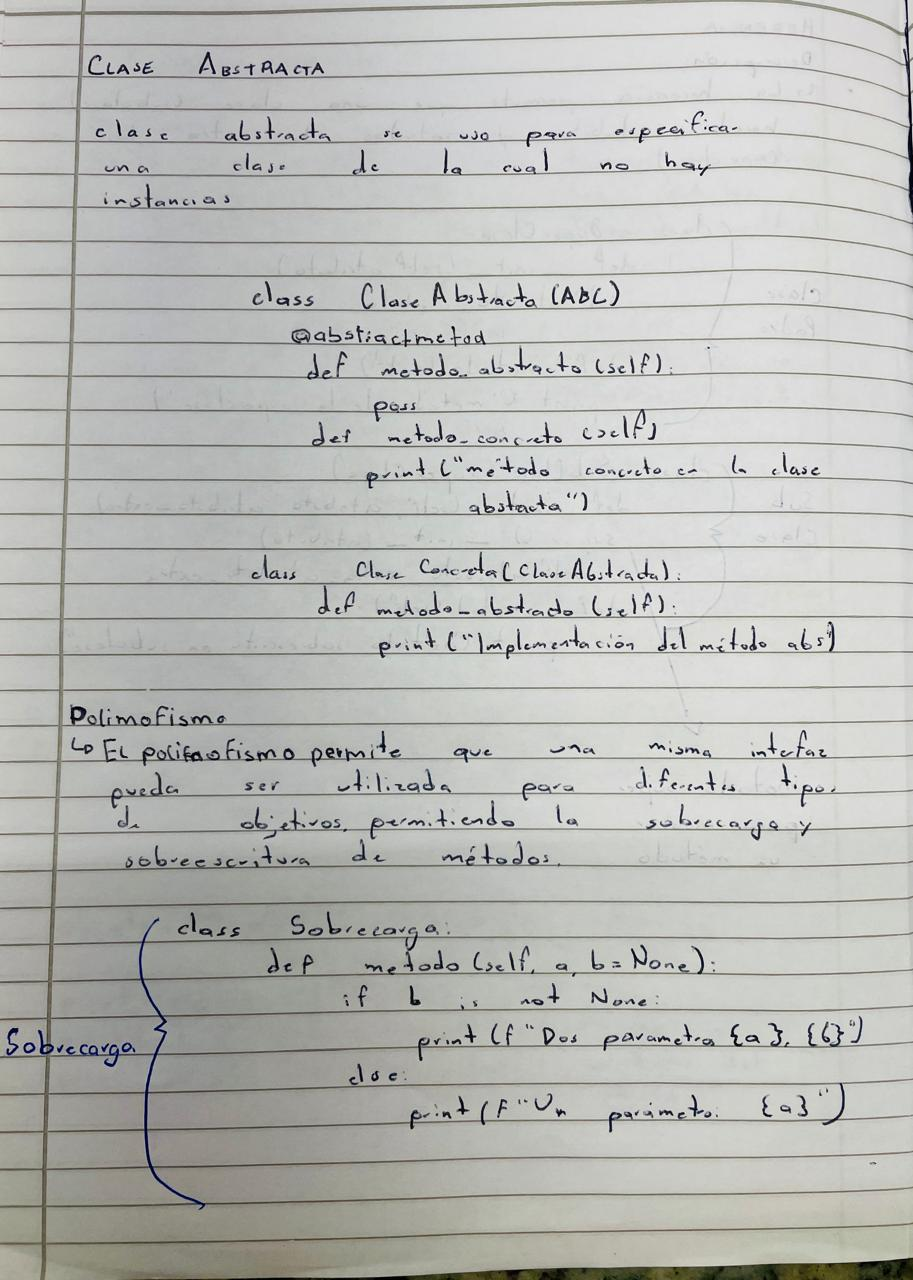
Programación Orientada a Objetos P á g i n a | **4**



*Ilustración 3 - Herencia*

En la imagen anterior se muestra la evidencia de las características y descripción de la herencia, se hace uso de la clase Padre “SuperClase” y la clase Hijo “SubClase”. Se indica cual es cada clase y se da énfasis en el método super, que es fundamental en el uso de la herencia.

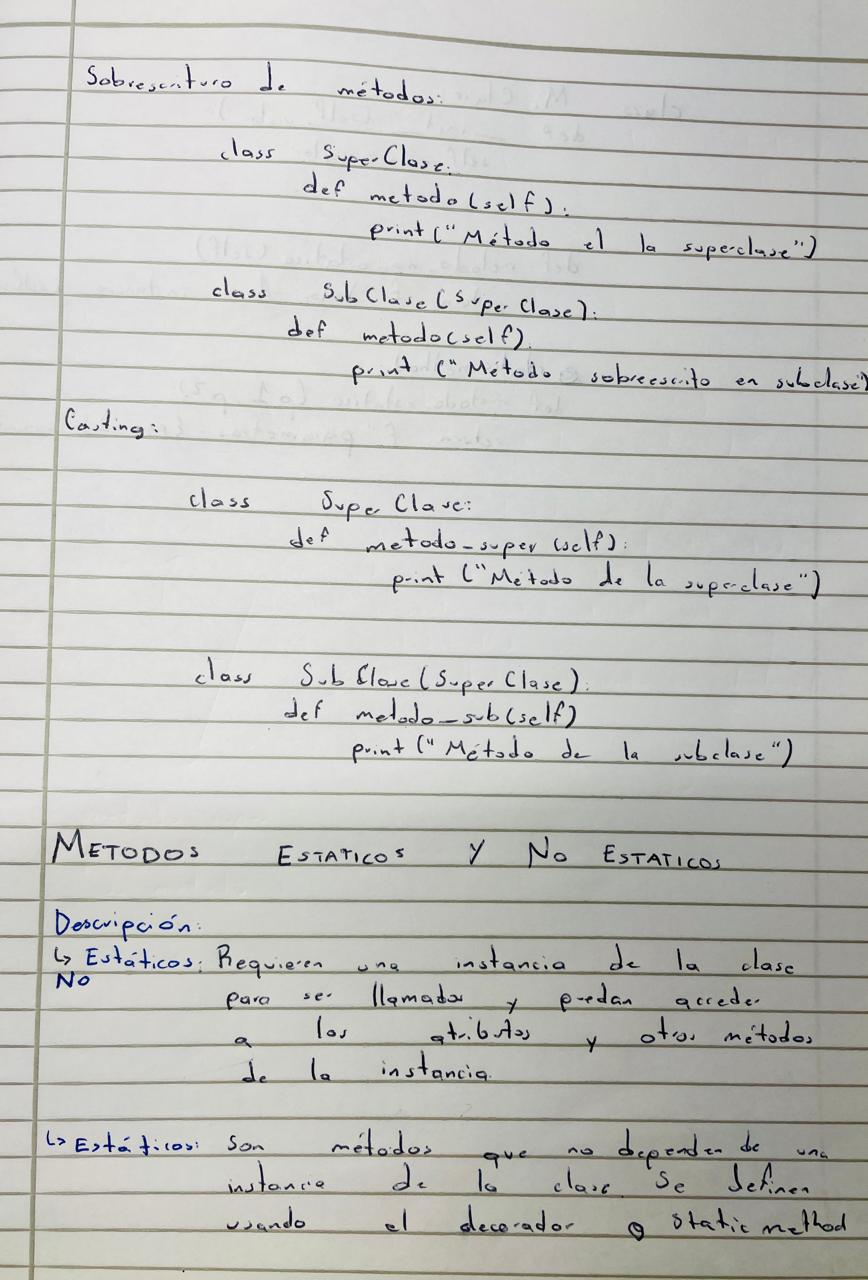
Programación Orientada a Objetos P á g i n a | **5**



*Ilustración 4 – Clase Abstracta y polimorfismo*

En la imagen anterior se muestra la descripción de la clase abstracta haciendo uso de la palabra reservada “pass”, y la subClase “Clase Concreta” , también se introduce al polimorfismo con un ejemplo de “Sobrecarga”.

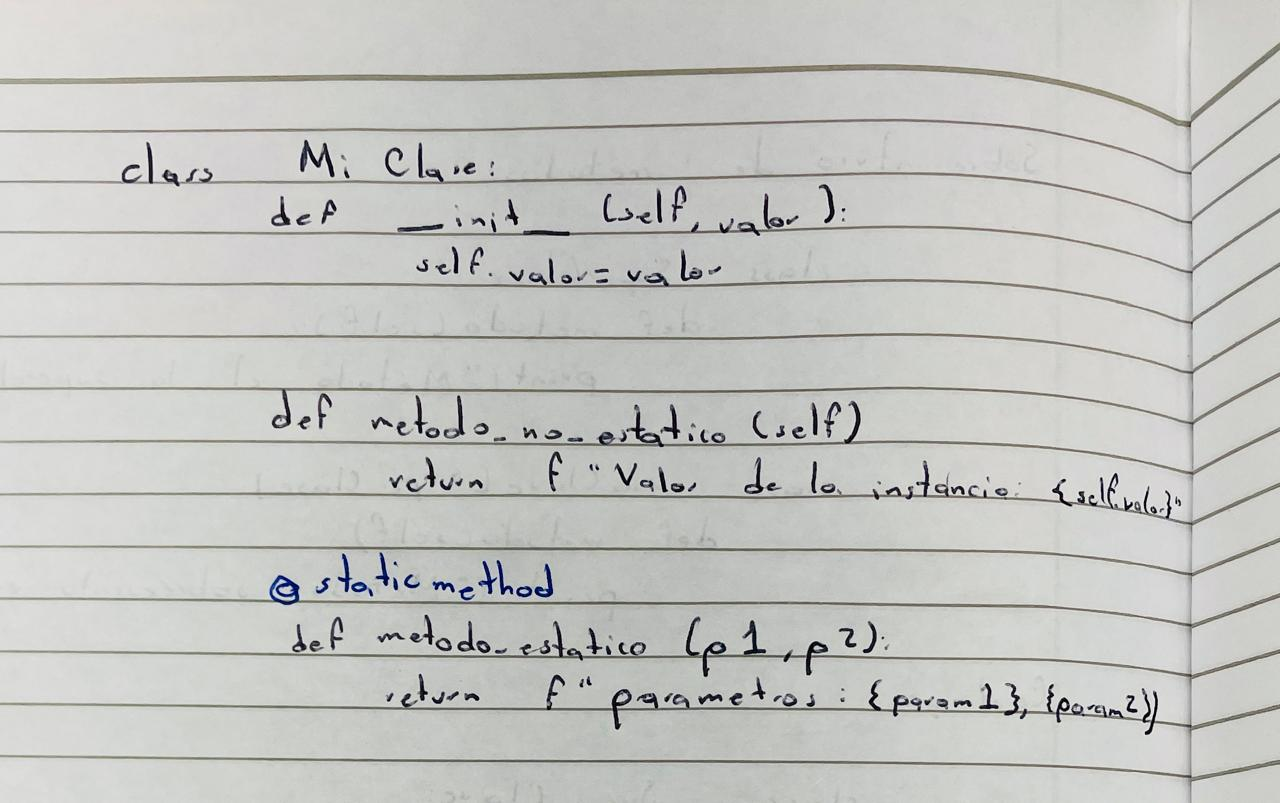
Programación Orientada a Objetos P á g i n a | **6**



*Ilustración 5 –* *Sobreescritura de métodos y Casting*

En la imagen anterior podemos ver una breve explicación de los conceptos de métodos estáticos y no estáticos, además se muestra la sintaxis de sobreescritura de métodos y la de Casting.

Programación Orientada a Objetos P á g i n a | 7



*Ilustración 6 -* *Sintaxis métodos estáticos y no estáticos*

En la imagen podemos observar la sintaxis de los métodos estáticos y no estáticos dentro de la clase “MiClase”, y se recalca la importancia del decorador “Staticmethod”. Continuando el tema de la ilustración anterior.

Programación Orientada a Objetos P á g i n a | 8

# Retroalimentación

La comprensión de los conceptos de clases, métodos y objetos es fundamental para cualquier programador que trabaje con POO. Las clases proporcionan una estructura organizada para definir y crear objetos, lo que facilita la reutilización y el mantenimiento del código. Los métodos, tanto estáticos como no estáticos, permiten modularizar las funcionalidades y definir claramente qué acciones pueden realizar los objetos. La práctica de crear y manipular objetos instancias de clases fortalece la habilidad para diseñar sistemas complejos de manera efectiva.

El encapsulamiento es un principio clave que mejora la seguridad y la integridad del software. Al controlar el acceso a los datos y ocultar los detalles internos, se pueden evitar modificaciones no deseadas y errores difíciles de rastrear. Este enfoque no solo protege los datos, sino que también hace que el código sea más fácil de mantener y entender. Implementar correctamente el encapsulamiento mediante modificadores de acceso y métodos de acceso (getters y setters) es una habilidad esencial para garantizar la robustez y la fiabilidad del software.

La herencia y el polimorfismo son herramientas poderosas para construir sistemas flexibles y extensibles. La herencia permite la creación de nuevas clases basadas en clases existentes, lo que promueve la reutilización del código y la creación de jerarquías lógicas. El polimorfismo, por su parte, permite que diferentes clases se comporten de manera intercambiable, lo que facilita la implementación de interfaces comunes y el manejo de comportamientos dinámicos. Comprender y aplicar estos conceptos permite a los desarrolladores diseñar sistemas más elegantes y adaptables, capaces de evolucionar con las necesidades cambiantes de los usuarios y del negocio.

Programación Orientada a Objetos P á g i n a | **9**