**¿Por qué aprender Programación Orientada a Objetos?**

* **Vas a programar más rápido**. Tener un análisis previo de lo que estás realizando te ayudará a generar código mucho más veloz
* **Dejas de ser Programador Jr**. Podrás responder preguntas como ¿Qué es encapsulamiento?, ¿Qué es Abstracción?, ¿Qué es Herencia?, ¿Qué es Polimorfismo? en futuras entrevistas de trabajo
* **Dejar de Copiar y Pegar Código**.

La programación orientada a objetos tiene cuatro características principales:

**Encapsulamiento**. Quiere decir que oculta datos mediante código.

**Abstracción**. Es como se pueden representar los objetos en modo de código.

**Herencia**. Es donde una clase nueva se crea a partir de una clase existente.

**Polimorfismo**. Se refiere a la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos.

En este curso, los pasos a seguir será.

Analisis

Plasmar

Programar

La mayoría solo aprende a hacer esto en un lenguajes de programación,a quí se tiene una variabilidad.

# ¿Qué resuelve la Programación Orientada a Objetos?

La programación Orientada a Objetos nace de los problemas creados por la programación estructurada y nos ayuda a resolver cierto problemas como:

* Código muy largo: A medida que un sistema va creciendo y se hace más robusta el código generado se vuelve muy extenso haciéndose difícil de leer, depurar, mantener.
* Si algo falla, todo se rompe: Ya que con la programación estructurada el código se ejecuta secuencialmente al momento de que una de esas líneas fallara todo lo demás deja de funcionar.
* Difícil de mantener.

# Paradigma Orientado a Objetos

La **Programación Orientada a Objetos** viene de una filosofía o forma de pensar que es la **Orientación a Objetos** y esto surge a partir de los problemas que necesitamos plasmar en código.

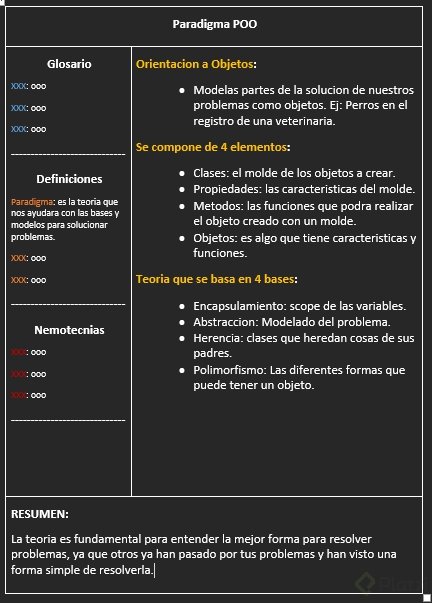
Es analizar un problema en forma de objetos para después llevarlo a código, eso es la **Orientación a Objetos**.

Un **paradigma** es una teoría que suministra la base y modelo para resolver problemas. La paradigma de Programación Orientada a Objetos se compone de 4 elementos:

* Clases
* Propiedades
* Métodos
* Objetos

Y 4 Pilares:

* Encapsulamiento
* Abstracción
* Herencia
* Polimorfismo



# Lenguajes Orientados a Objetos

**Correcciones:  
En el minuto 5:06. El navegador NO interpreta el código PHP. Lo hace el servidor**  
**Visual studio code y visual studio no es lo mismo.**  
**Visual Studio Code, el editor de código multiplataforma de Microsoft, es uno de los preferidos por muchos desarrolladores mientras que Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET**

Algunos de los lenguajes de programación Orientados a Objetos son:

* **Java**:  
  – Orientado a Objetos naturalmente  
  – Es muy útilizado en Android  
  – Y es usado del lado del servidor o Server Side
* **PHP**  
  – Lenguaje interpretado  
  – Pensado para la Web
* **Python**  
  – Diseñado para ser fácil de usar  
  – Múltiples usos: Web, Server Side, Análisis de Datos, Machine Learning, etc
* **Javascript**  
  – Lenguaje interpretado  
  – Orientado a Objetos pero basado en prototipos  
  – Pensado para la Web
* C#
* Ruby
* Kotlin

# Diagramas de Modelado

**OMT**: Object Modeling Techniques. Es una metodología para el análisis orientado a objetos.

**UML**: Unified Modeling Language o Lenguaje de Modelado Unificado. Tomó las bases y técnicas de OMT unificándolas. Tenemos más opciones de diagramas como lo son Clases, Casos de Uso, Objetos, Actividades, Iteración, Estados, Implementación.

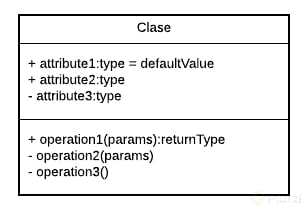
# Qué es UML

Como ya viste UML significa Unified Modeling Language el cual es un lenguaje estándar de modelado de sistemas orientados a objetos.



Esto significa que tendremos una manera gráfica de representar una situación, justo como hemos venido viendo. A continuación te voy a presentar los elementos que puedes utilizar para hacer estas representaciones.

Las **clases** se representan así:



En la parte superior se colocan los atributos o propiedades, y debajo las operaciones de la clase. Notarás que el primer caracter con el que empiezan es un símbolo. Este denotará la visibilidad del atributo o método, esto es un término que tiene que ver con Encapsulamiento y veremos más adelante a detalle.

Estos son los niveles de **visibilidad** que puedes tener:

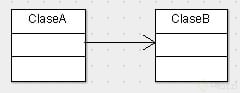
**-** private  
**+** public  
**#** protected  
**~** default

Una forma de representar las relaciones que tendrá un elemento con otro es a través de las flechas en UML, y aquí tenemos varios tipos, estos son los más comunes:

## Asociación



Como su nombre lo dice, notarás que cada vez que esté referenciada este tipo de flecha significará que ese elemento contiene al otro en su definición. La flecha apuntará hacia la dependencia.

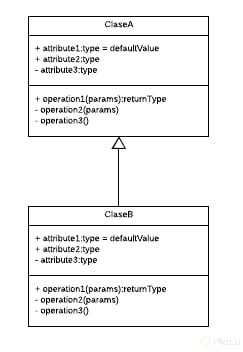


Con esto vemos que la ClaseA está asociada y depende de la ClaseB.

## Herencia



Siempre que veamos este tipo de flecha se estará expresando la herencia.  
La dirección de la flecha irá desde el hijo hasta el padre.

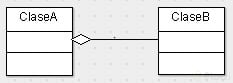


Con esto vemos que la ClaseB hereda de la ClaseA

## Agregación



Este se parece a la asociación en que un elemento dependerá del otro, pero en este caso será: Un elemento dependerá de muchos otros. Aquí tomamos como referencia la multiplicidad del elemento. Lo que comúnmente conocerías en Bases de Datos como Relaciones uno a muchos.

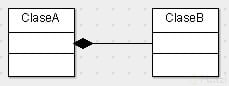


Con esto decimos que la ClaseA contiene varios elementos de la ClaseB. Estos últimos son comúnmente representados con listas o colecciones de datos.

## Composición



Este es similar al anterior solo que su relación es totalmente compenetrada de tal modo que conceptualmente una de estas clases no podría vivir si no existiera la otra.



Con esto terminamos nuestro primer módulo. Vamos al siguiente para entender cómo podemos hacer un análisis y utilizar estos elementos para construir nuestro diagrama de clases de Uber.

<https://www.youtube.com/watch?v=Z0yLerU0g-Q>

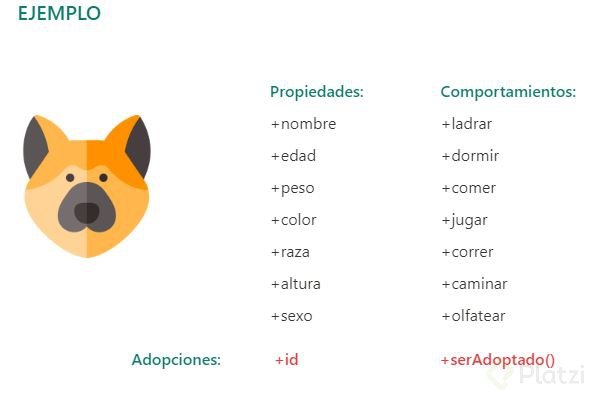
# Objetos

Los Objetos son aquellos que tienen propiedades y comportamientos, también serán sustantivos.

* Pueden ser Físicos o Conceptuales

Las **Propiedades** también pueden llamarse atributos y estos también serán sustantivos. Algunos atributos o propiedades son nombre, tamaño, forma, estado, etc. Son todas las características del objeto.

Los **Comportamientos** serán todas las operaciones que el objeto puede hacer, suelen ser verbos o sustantivos y verbo. Algunos ejemplos pueden ser que el usuario pueda hacer login y logout.



# Abstracción y Clases

Una **Clase** es el modelo por el cual nuestros objetos se van a construir y nos van a permitir generar más objetos.

Analizamos Objetos para crear **Clases**. Las **Clases** son los modelos sobres los cuales construiremos nuestros objetos.

**Abstracción** es cuando separamos los datos de un objeto para generar un molde.

Nota de clase:

En el vídeo anterior Anahi hablo muy bien de lo que es un Objeto y como esta compuesto por atributos o propiedades y a su vez estos tienen acciones o métodos.

Ahora el concepto se aumenta cuando todo esto se ve como una sola cosa llamada Clase.

Una Clase Es el modelo sobre el cual nuestros objetos se construyen.

Es decir si tenemos un objeto llamado perro y este tiene sus atributos que lo describen generalmente y a su vez tiene métodos donde se define las acciones que pueda hacer ese perrito. Una clase me permite generar mas objetos (mas perros) con mismos atributos y métodos pero con resultados diferentes. ej:

Objeto #1 llamado “Rocky”:

tributo\_1: color = marrón

atributo\_2: taman’o = pequen’o

atributo\_3: raza = chiguagua

metodo\_1: ladrar

metodo\_2: comer

metodo\_3: dormir

Objeto #2 llamado "Max"

atributo\_1: color = blanco

atributo\_2: taman’o = grande

atributo\_3: raza = hunky siberiano

metodo\_1: ladrar

metodo\_2: comer

metodo\_3: dormir

Para no repetir esto muchas veces de acuerdo a la cantidad de perros que es mi ejemplo de objeto, la idea es analizar todos estos objetos extraemos todos esos atributos y entonces generamos modelos. Esos modelos se le llaman Clases.

Una Clase son los modelos sobre los cuales construiremos Objetos

A este análisis se le conoce como Abstracción, simplemente consiste en generar un molde en base a esas propiedades y métodos de los objetos, abstraemos todos esos datos para generar dicho molde.

Resumen: Una clase es un molde para generar un objeto y este análisis se llama Abstracción

abstracción: recoger los datos y caracteristicas de un objeto para luego definir con ello una interfaz

# Modularidad

La **modularidad** va muy relacionada con las clases y es un principio de la Programación Orientado a Objetos y va de la mano con el Diseño Modular que significa dividir un sistema en partes pequeñas y estas serán nuestros módulos pudiendo funcionar de manera independiente.

La **modularidad** de nuestro código nos va a permitir

* Reutilizar
* Evitar colapsos
* Hacer nuestro código más mantenible
* Legibilidad
* Resolución rápida de problemas

Una buena práctica es separando las clases en archivos diferentes.

Modular: Dividir un sitema y así crear módulos independientes, lo que permite evitar un colapso masivo en nuestro código y mejorar la legibilidad.

La modularidad va de la mano con el Diseño Modular, este consiste en dividir nuestro sistema en partes pequeñas que funcionen de manera independiente.

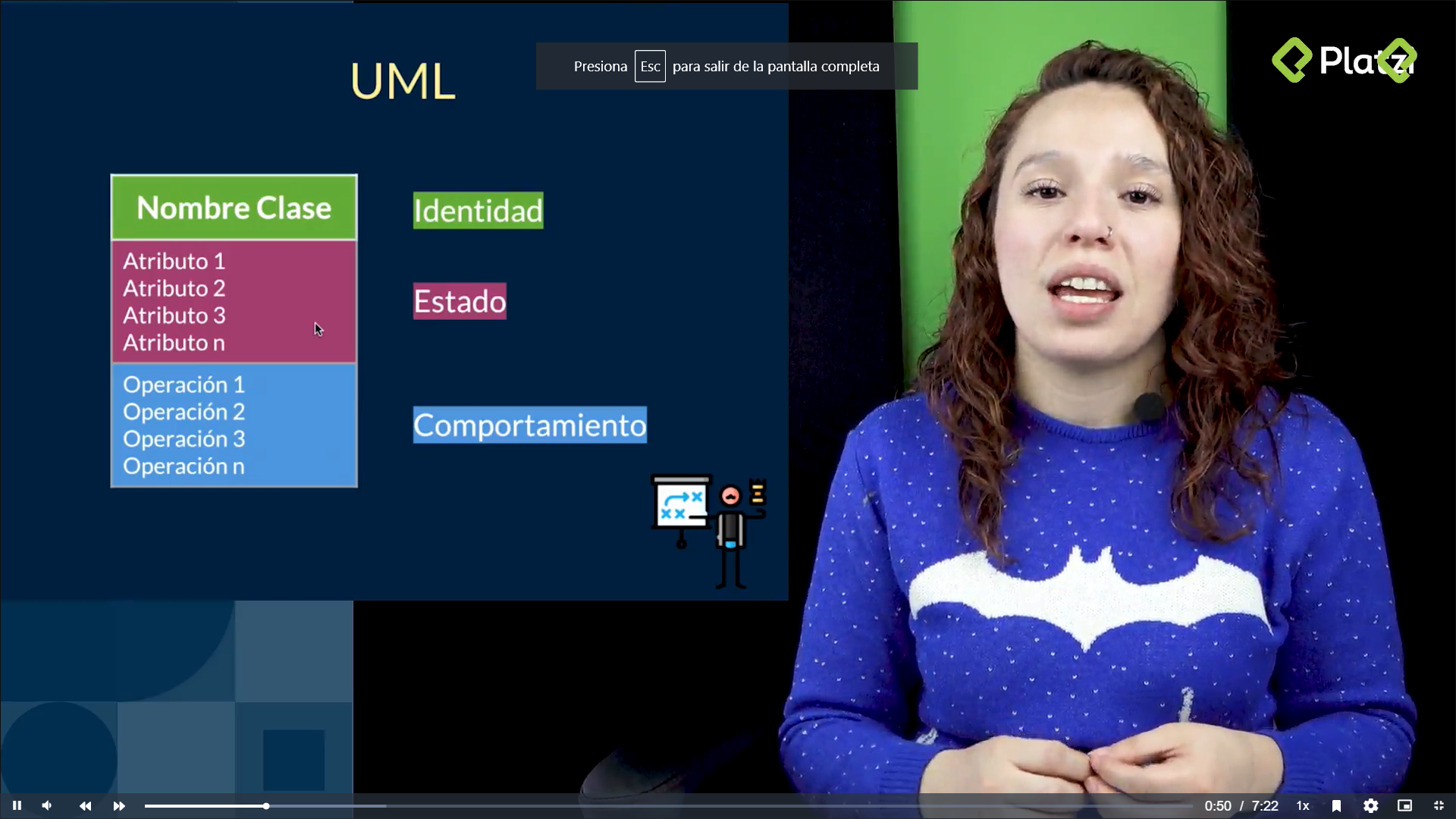


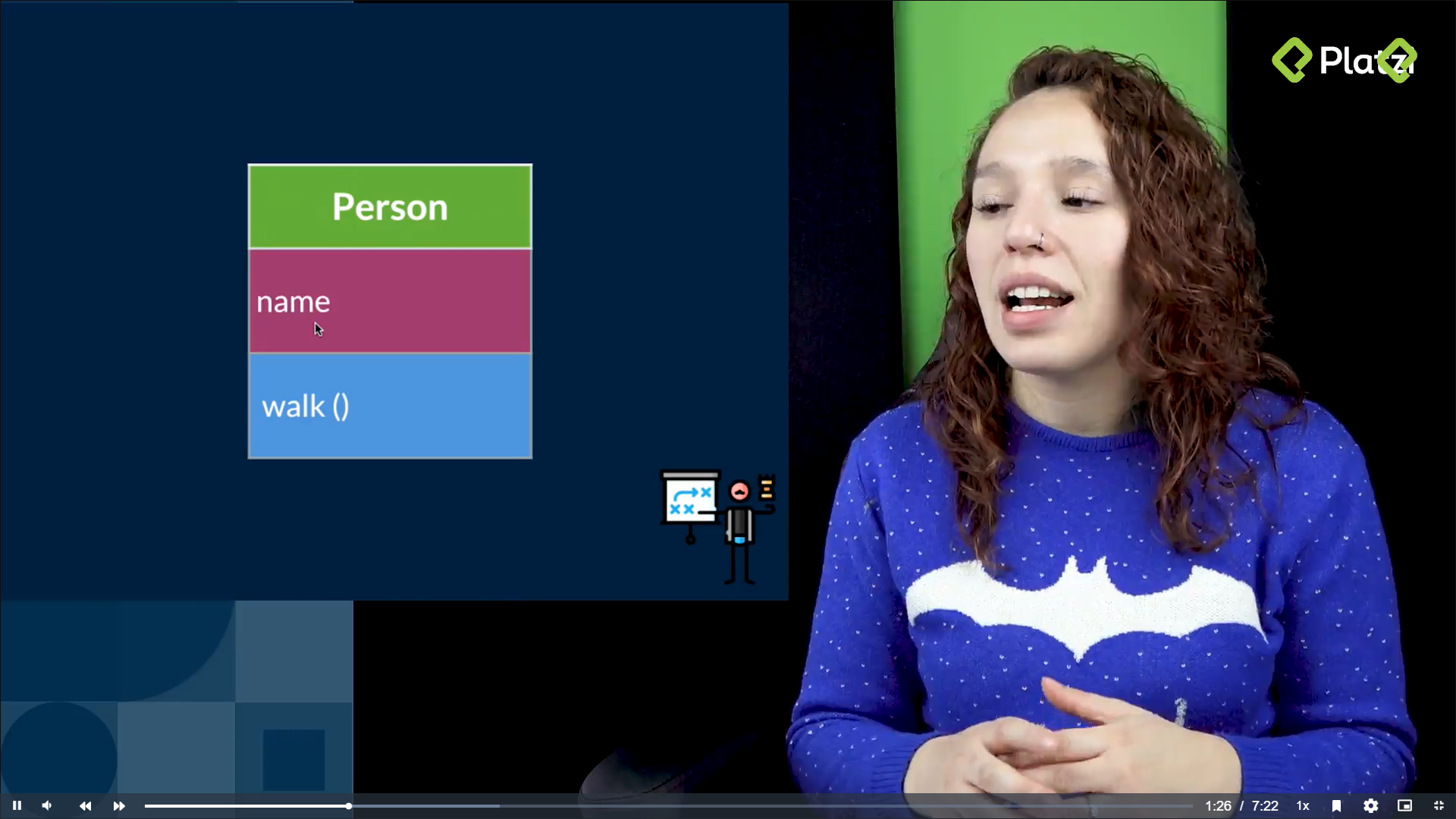
# Analizando Uber en Objetos

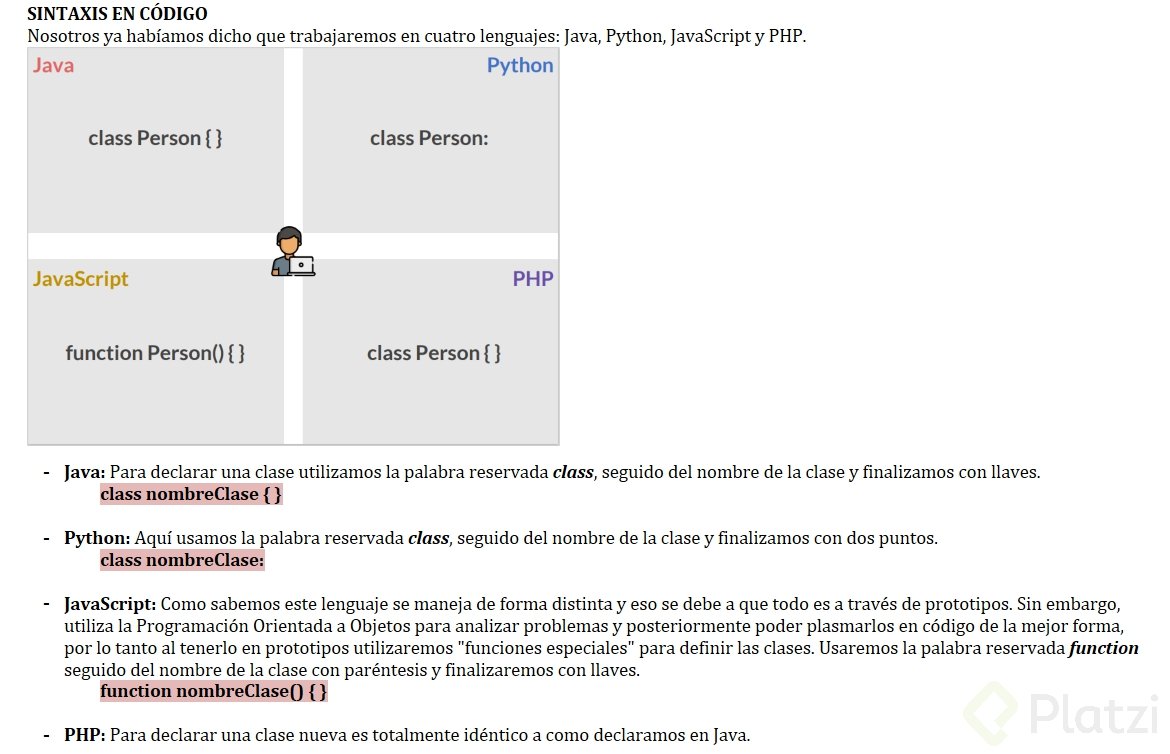
# 

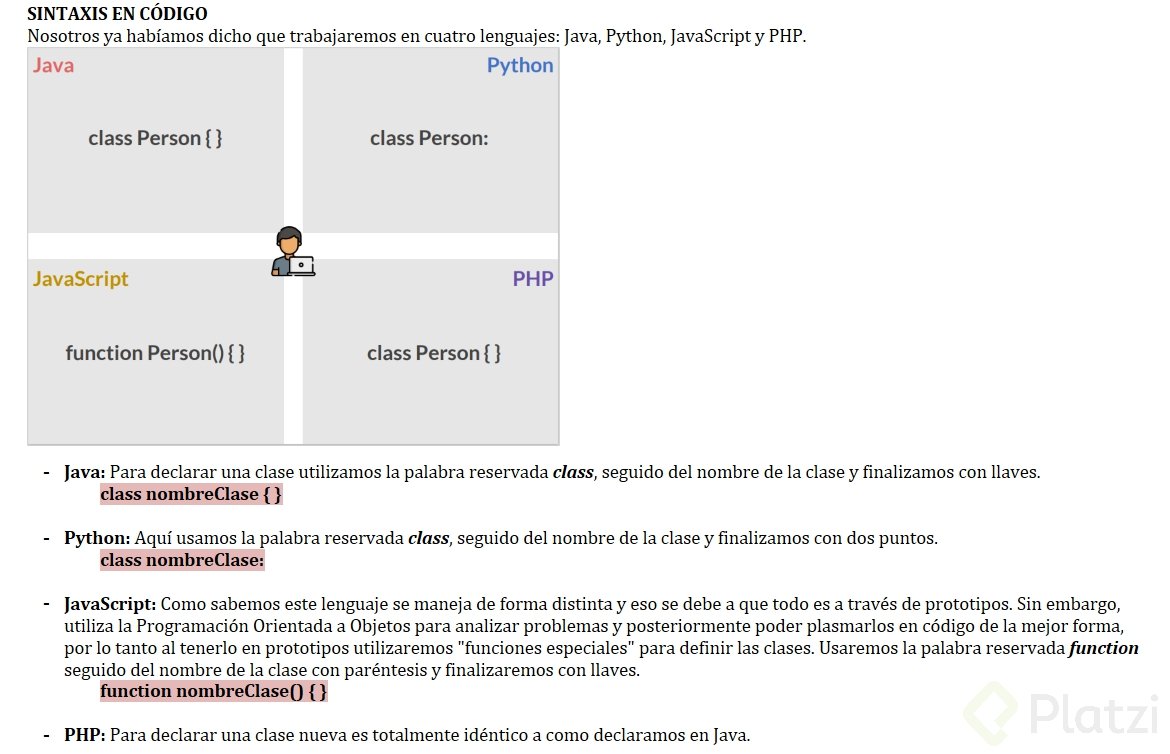
# Preview

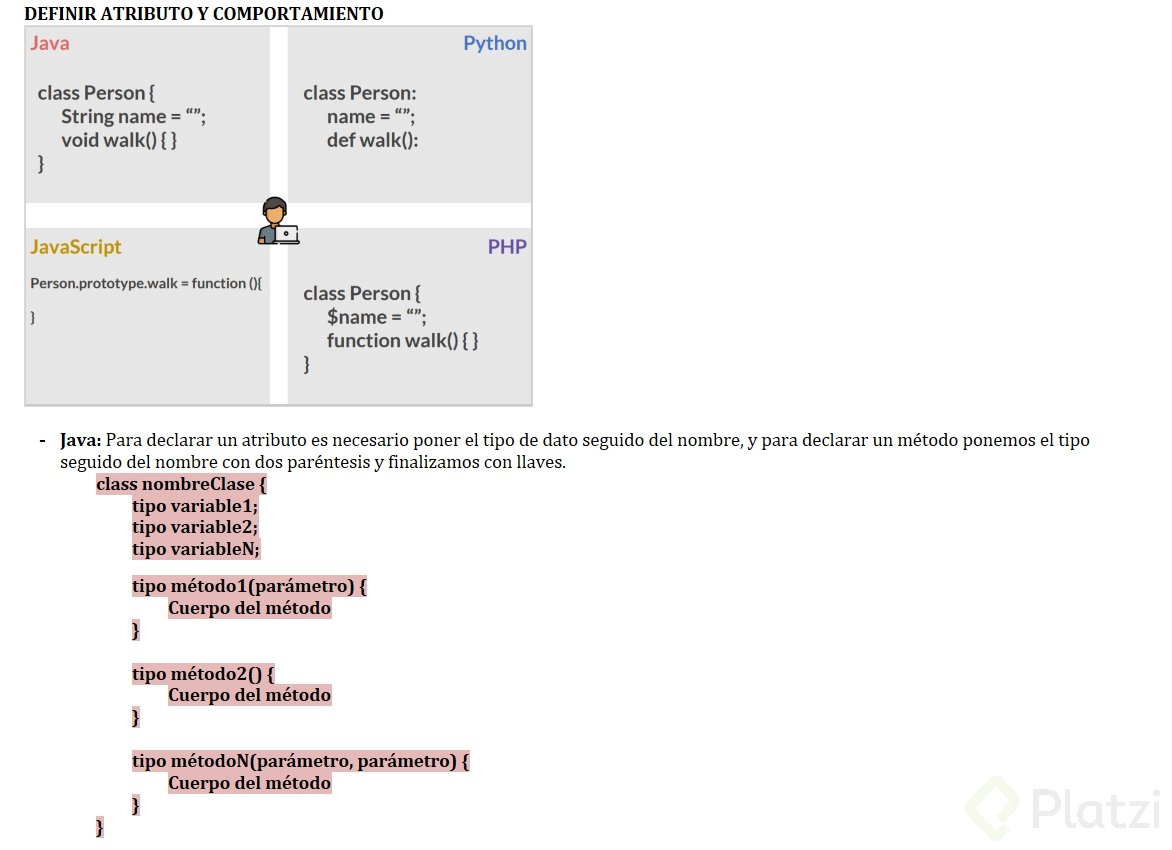
# Clases en UML y su sintaxis en código

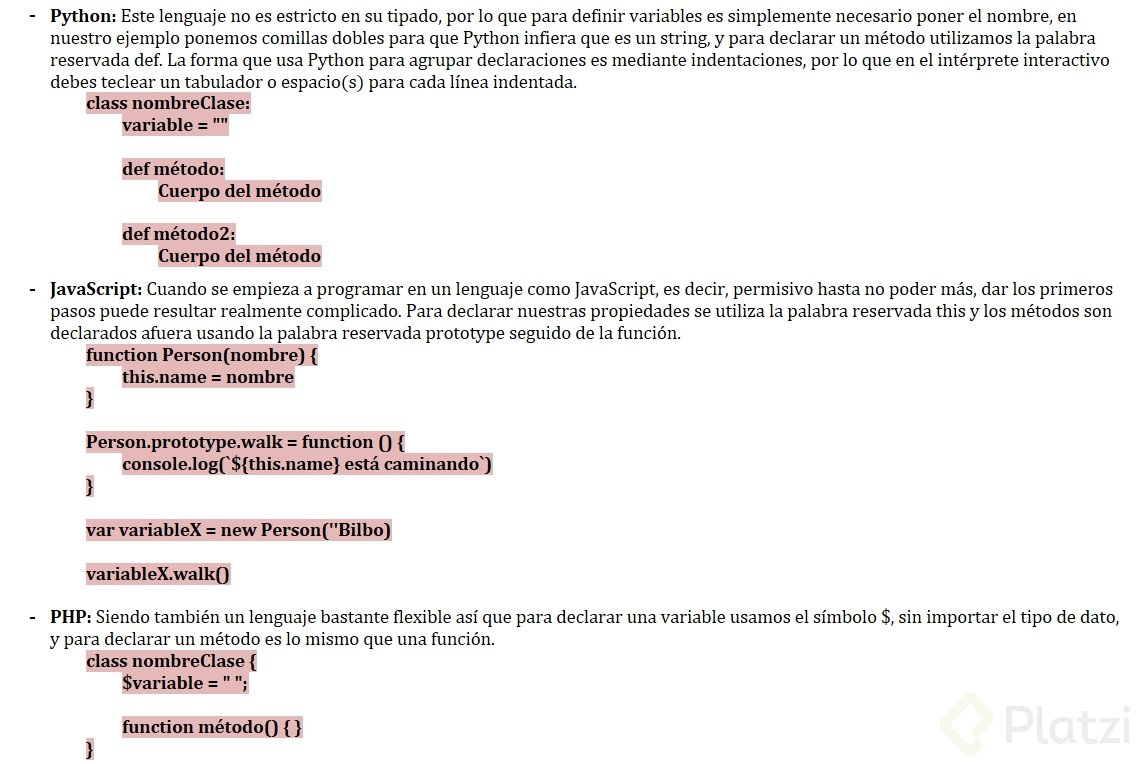












# Modelando nuestros objetos Uber

