

Objetos y clases

Materia: Programación orientada a objetos – UPSO

Profesor: Carlos Caseres

Conceptos

Una **clase** describe un grupo de objetos que comparten una estructura y un comportamiento en común.

Un **objeto** es una instancia de una clase, de la que toman su estructura y comportamiento.

Conceptos

Ejemplo: queremos modelar un auto.

Características: Es una propiedad distintiva o inherente, que contribuye a que ese objeto sea ese y no otro. Todas las propiedades tienen un valor, que puede ser Una cantidad específica o incluso denotar otro Objeto.

Comportamientos: se define en como actúa y reacciona un objeto, en términos de sus cambios de estado y paso de mensajes.



Conceptos

Ejemplo: queremos modelar un auto.

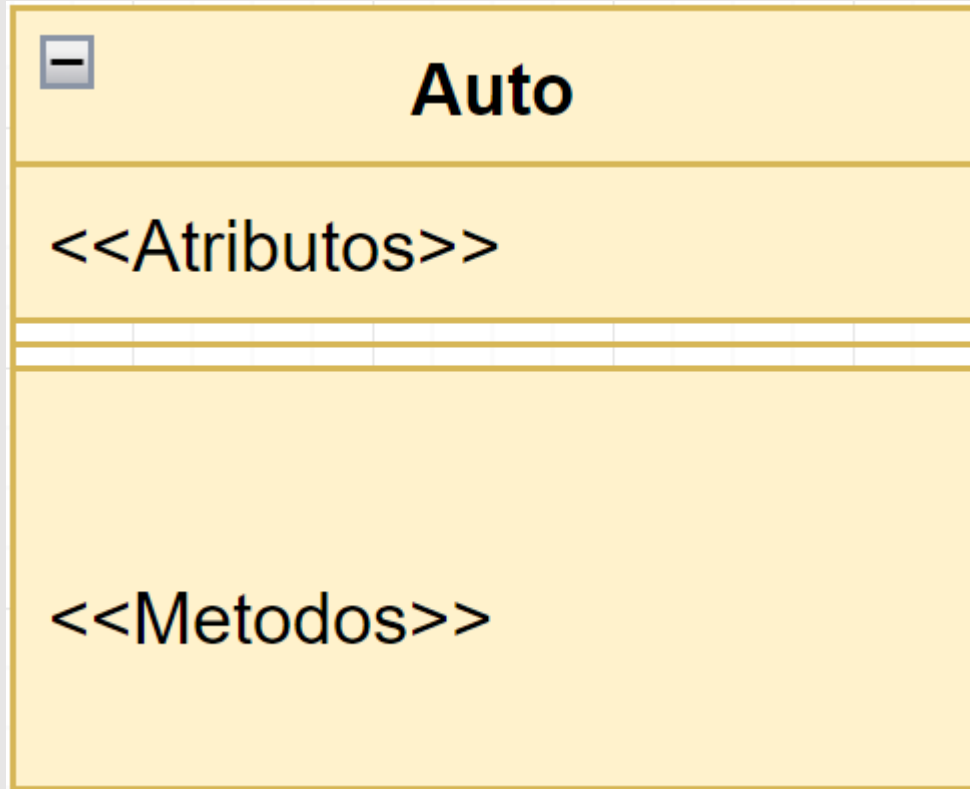
Características: color, cantidad de ruedas, cantidad de puertas, si tiene motor, patente, marca, etc.

Comportamientos: avanzar, retroceder, detenerse, girar a la izquierda, girar a la derecha, encender y apagar motor, etc.



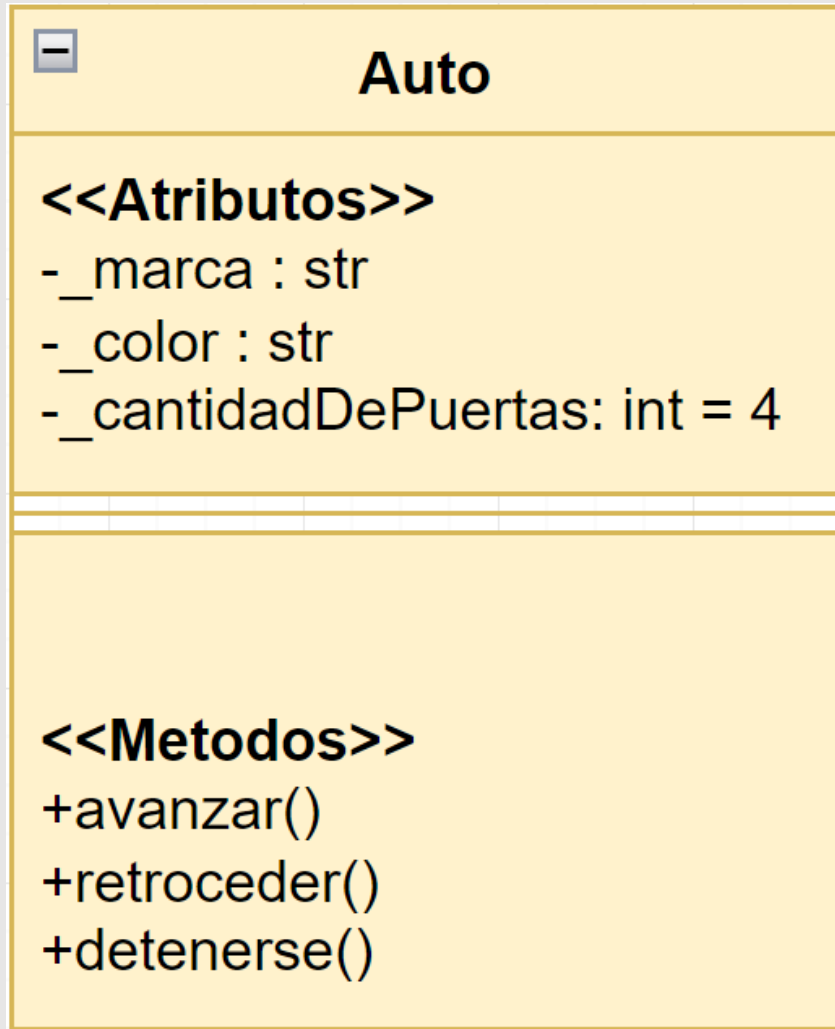
Conceptos

Ejemplo: queremos modelar un auto.




Conceptos

Ejemplo: queremos modelar un auto.



Conceptos

Ejemplo: queremos modelar un auto.

<div></div> <div>Auto</div>
<div data-bbox="84 482 461 529"><<Atributos>></div> <div data-bbox="84 554 784 743"><ul style="list-style-type: none">- _marca : str- _color : str- _cantidadDePuertas: int = 4</div>
<div data-bbox="84 876 532 923"><<Constructor>></div> <div data-bbox="84 948 907 1001"><p>__construct(marca: str, color: str)</p></div> <div data-bbox="84 1019 443 1066"><<Metodos>></div> <div data-bbox="84 1090 410 1280"><ul style="list-style-type: none">+avanzar()+retroceder()+detenerse()</div>



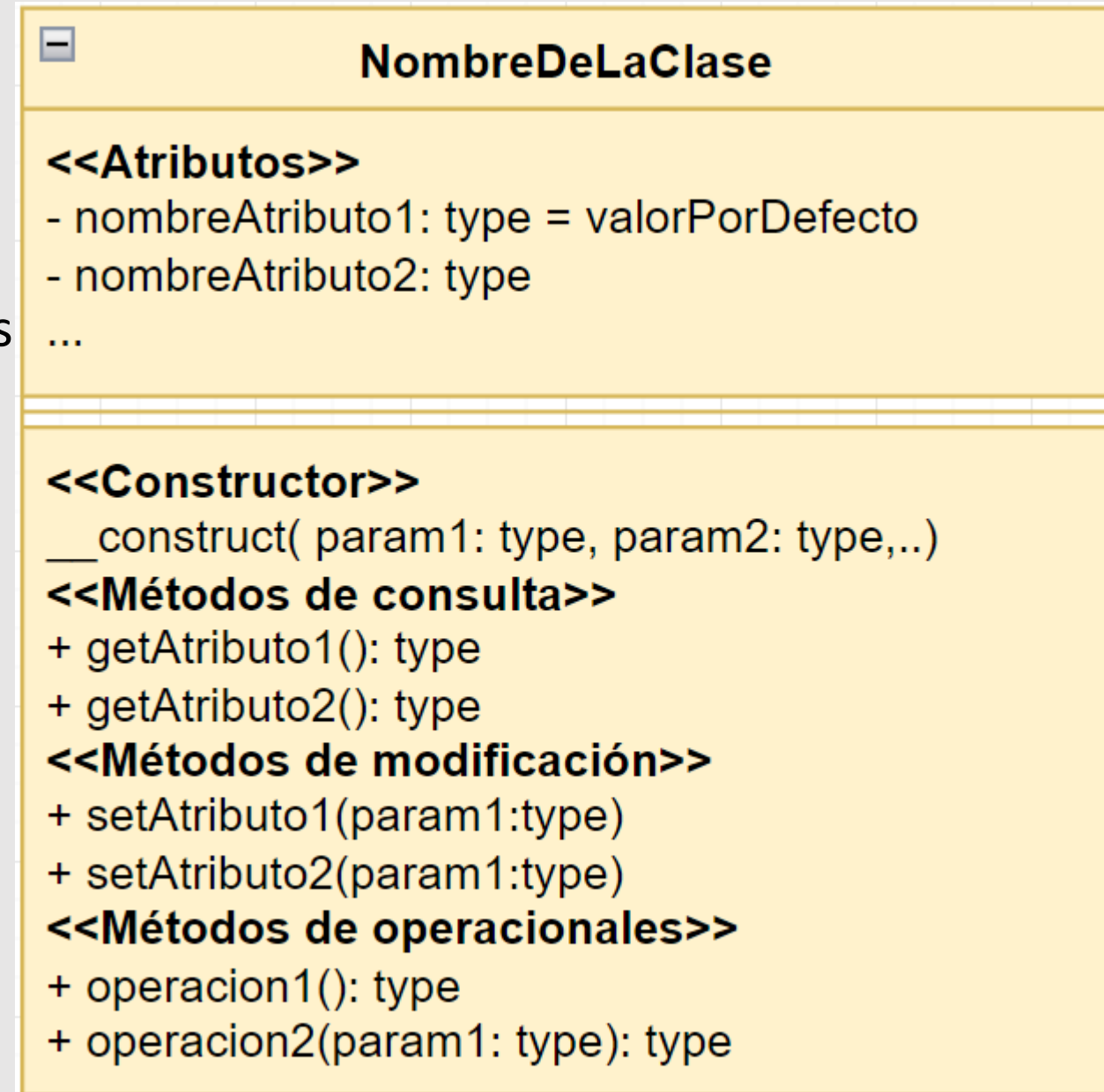
Diseño de clases

Estructura del diagrama de clase

Atributos

Atributos de clase: valor compartido por todos los objetos que son instancias de la clase.

Atributos de instancia: su valor varia en cada objeto



Diseño de clases

Estructura del diagrama de clase

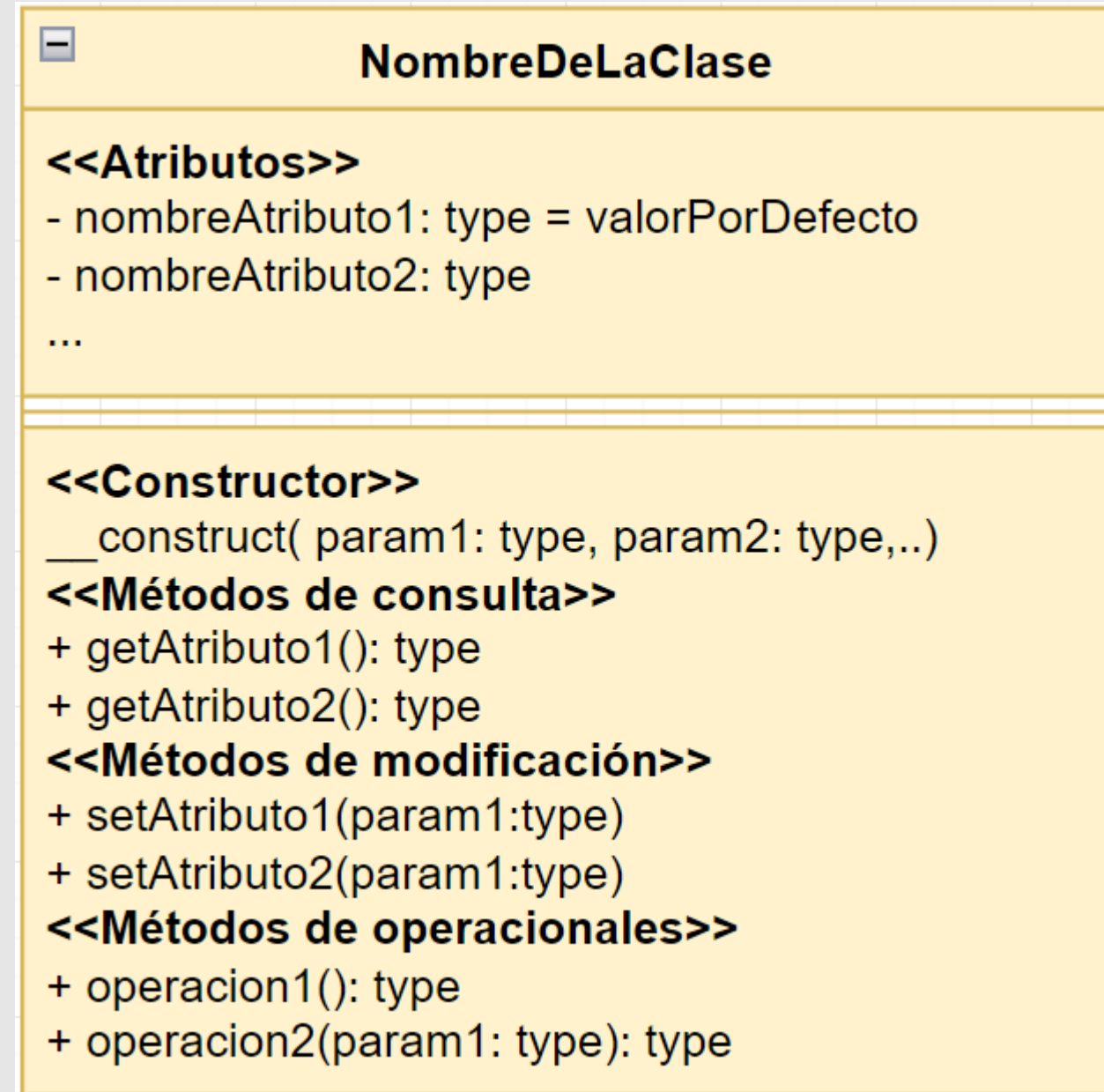
Métodos

Constructor: Para inicializar objetos de una clase (asignación de valores iniciales de los atributos).

de consulta: Retorna los valores del estado (valores de sus atributos).

de modificación: Modifica el estado del objeto (sus atributos).

de operación: Usada generalmente para realizar operaciones distintas a las anteriores. También puede modificar el estado del objeto.



Interfaz e implementación

La interfaz de una clase proporciona su vista externa y enfatiza la abstracción, a la vez que oculta su estructura y los detalles de su comportamiento.

Se compone de todas las operaciones aplicables a instancias de esa clase, pero también puede incluir la declaración de otra clase, constantes, variables, expresiones, según se necesiten.

La implementación de una clase se compone principalmente de la propia implementación de todas las operaciones definidas en la interfaz.

Interfaz e implementación

La interfaz puede dividirse en:

Publica (public): una declaración accesible a todos los clientes.


Protegida (protected): una declaración accesible solo a la propia clase y subclases.

Privada (private): una declaración accesible solo a la propia clase.

Diseño de clases

Estructura del diagrama de clase:

En UML se utiliza una simbología particular para cada tipo de acceso.

 **NombreDeLaClase**

<<Atributos>>

+ atributoPublico
atributoProtegido
- atributoPrivado
atributoDeClase

<<Métodos>>

+ metodoPublico
metodoProtegido
- metodoPrivado
metodoDeClase

Interfaz e implementación

En Python no existe esta distinción de tipos de acceso, por lo que todos los atributos y métodos serán accesible para todos los clientes: no hay nada que haga cumplir el ocultar datos; todo se basa en convención.

Los clientes deben usar los atributos de datos con cuidado, estos pueden romper invariantes que mantienen los métodos si pisan los atributos de datos.

Los clientes pueden añadir sus propios atributos de datos a una instancia sin afectar la validez de sus métodos, siempre y cuando se eviten conflictos de nombres.

Interfaz e implementación

En Python:

Convención:

Nombre del atributo con un _ inicial: indica que el atributo “**debe tratarse como privado**”

Ejemplo:

_atributo1

_atributo2

Interfaz e implementación

Convenciones

Si bien no es obligatorio, comúnmente se establecen ciertos formatos para los identificadores de variables, constantes, funciones, etc (*Naming Conventions*).

1. **PascalCase**: la primera letra del identificador y la primera letra de las siguientes palabras concatenadas están en mayúsculas. **Ejemplo**: MiClase.

2. **camelCase**: la primera letra del identificador está en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula. **Ejemplo**: unaPropiedad.

3. **ALL_CAPS**: todas las letras del identificador se encuentran en mayúsculas y las palabras se separan por un guión bajo `_`. **Ejemplo**: UNA_CONSTANTE.

4. **small_caps**: (ó **snack_case**) todas las letras del identificador se encuentran en minúsculas y las palabras separadas por `_`. **Ejemplo**: una_funcion.

5. **Proper_Case**: como CamelCase, pero cada inicio de palabra separado por un `_`. Se usa muy poco.

Interfaz e implementación

Sugerencias

Nombre de clases: PascalCase

Constantes: ALL_CAPS

Atributos: camelCase (Si son privados o protegidos, comienzan con _)

Métodos: camelCase (Si son privados o protegidos, comienzan con _), los comandos que modifican un atributo comienzan por “set” y los que consultan atributos comienzan con “get”.

Constructor: La sintaxis especifica dependerá del lenguaje, generalmente se define un nombre en particular.

En lenguajes que lo permitan, hasta comprender mejor la diferencia entre `private`, `public` o `protected` es habitual definir atributos privados y métodos públicos.