

LÓGICA Y REPRESENTACIÓN I

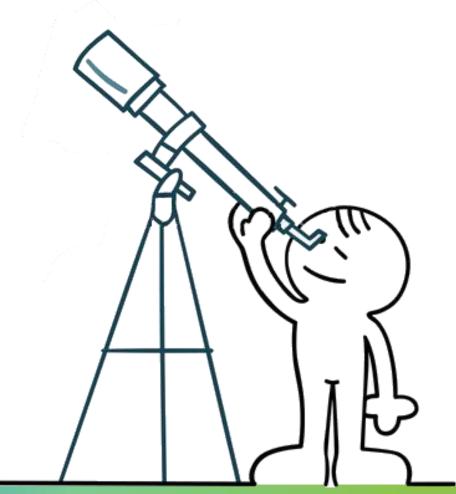
Programa de Ingeniería de Sistemas

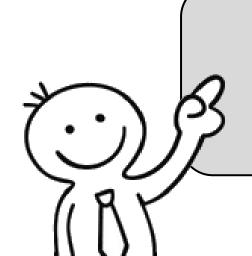
CONTENIDO



Introducción a la P.O.O.

- Los paradigmas de programación
- El paradigma Orientado a Objetos
- Definición de Clase, Objeto, atributos y métodos
- Representación en seudocódigo de una clase
- Modificadores de acceso: visibilidad de atributos y métodos
- Tipos de métodos: constructor, accesor, modificado y analizador

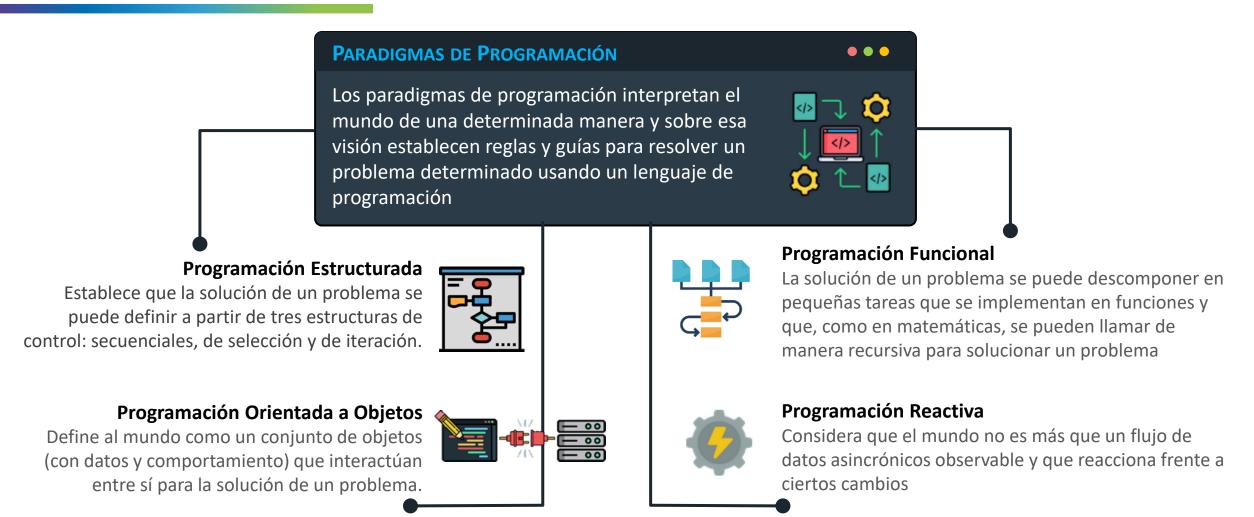




EL PARADIGMA DE PROGRAMACIÓN ORIENTADO POR OBJETOS

LOS PARADIGMAS DE PROGRAMACIÓN





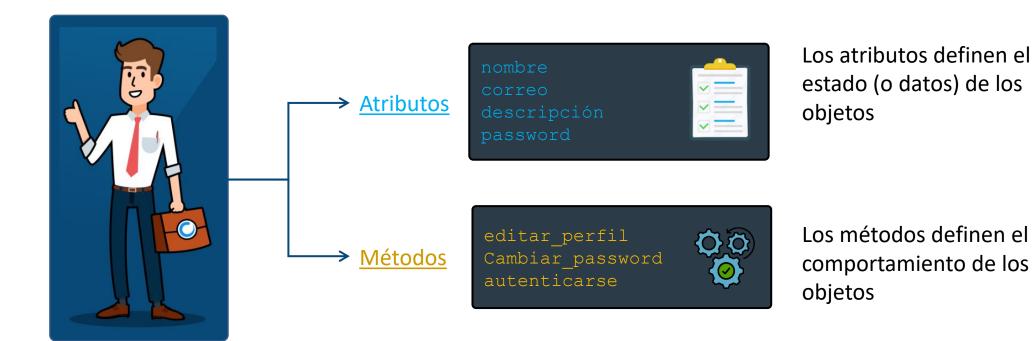
Cada enfoque tiene sus ventajas y desventajas, y cada uno es más apropiado para ciertos problemas o tareas ...

PERSONA



DEFINICIÓN

La P.O.O. modela <u>las entidades que intervienen en el problema</u> como objetos, identificando cuáles son sus atributos (o características) y sus comportamientos (o métodos). Además, modela las relaciones entre los objetos, las cuales definen como interactúan esos objetos para resolver un problema.



Diapositiva # 5



ALGUNAS DE LAS PROPIEDADES

1. Abstracción

Es la capacidad de modelar los objetos del mundo real, manteniendo los elementos importantes (atributos y métodos) e ignorando los detalles menos relevantes





2. Encapsulamiento

Permite proteger la información de los objetos de manipulaciones no autorizadas ocultando incluso los detalles de la implementación.

3. Polimorfismo

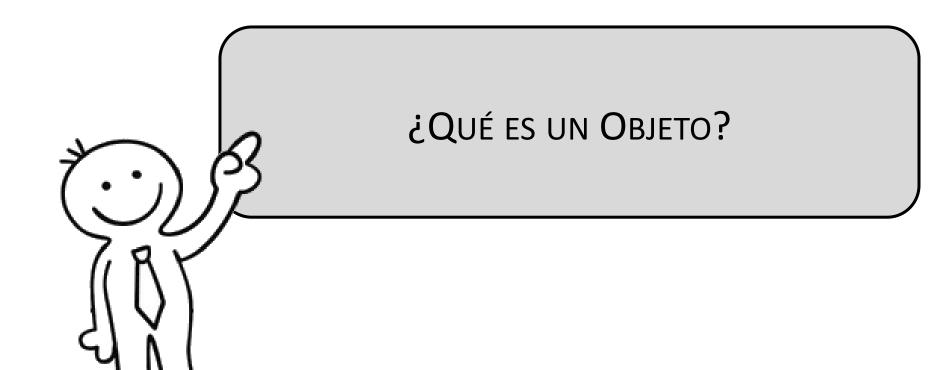
Permite que objetos de una misma clase puedan responder al mismo mensaje de manera diferente.





4. Herencia

Permite que una clase adquiera las propiedades y métodos de otra clase, lo que permite la reutilización de código y la creación de una jerarquía de clases.



PROGRAMACIÓN ORIENTADA A OBJETOS



UN OBJETO ES ...

CUALQUIER COSA!





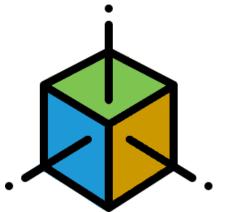
DEFINICIÓN FORMAL DE OBJETO (James Rumbaugh y Grady Booch):

"Un objeto es un concepto, abstracción o cosa tangible con un significado y límites claros en el problema en cuestión."



IDENTIDAD

Cada objeto tiene una identidad única, incluso si su estado es idéntico al de otro objeto.



ESTADO

Es lo que el objeto sabe de sí mismo y está definido por los atributos. El estado de un objeto puede cambiar con el tiempo.



COMPORTAMIENTO

Es lo que el objeto puede hacer. Determina cómo el objeto actúa y reacciona frente a las peticiones de otros objetos.







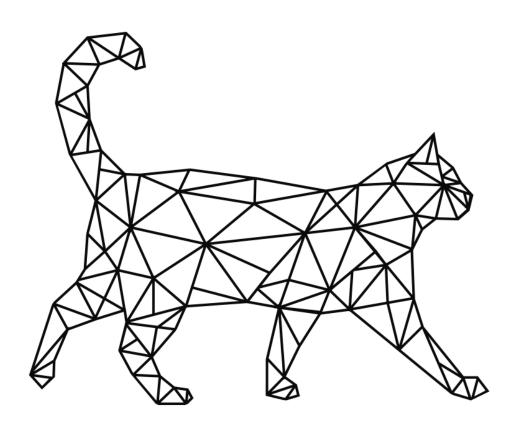
En este caso cada "elemento" en la imagen es un objeto ...



Todos ellos nos evocan un único concepto conocido, ¿cuál es?

PROGRAMACIÓN ORIENTADA A OBJETOS



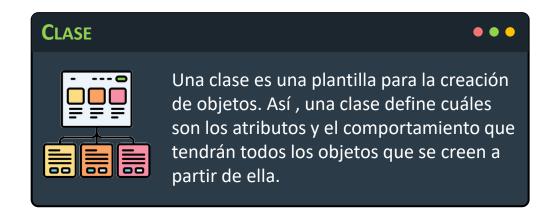


A ESE CONCEPTO GENERAL que agrupa a los objetos lo llamamos CLASE.





DEFINICIÓN DE CLASE:



La clase describe un grupo de objetos en términos de:

- Las características que tienen en común (atributos)
- El comportamiento que es similar a los objetos (métodos)
- La forma como se relacionan los objetos con otros objetos (relaciones)
- La semántica común que tienen en un problema (significan lo mismo)

Persona

- + Texto nombre
- + Texto corre
- + Texto descripción
- + Texto password
- + vacio editar perfil()
- + vacio cambiar password()
- + vacio autenticar()



EJEMPLO DE DEFINICIÓN DE CLASES

- Diseñe una clase que represente a los gatos, que en el problema que se está solucionando tienen dos comportamientos:
 - 1. Los gatos maúllan, es decir, muestran en la pantalla las palabras "Miau! Miau!"
 - 2. Los gatos son capaces de mostrar su estado, es decir, son capaces de decirnos como se llaman, que edad tienen y de qué color son.





REPRESENTACIÓN DE CLASES Y OBJETOS



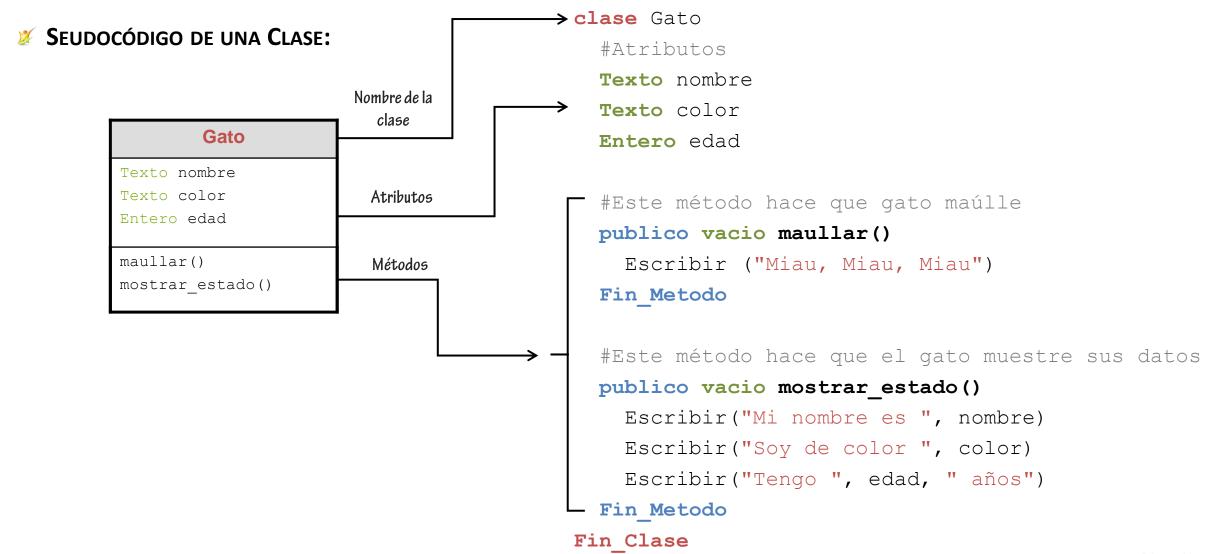
- Nombre de la clase: la primera letra de cada palabra va en mayúscula y se escribe en singular. Es decir, usaremos la notación PascalCase para definir el nombre de las clases.
- Atributos: al ser variables, cada atributo tiene un tipo de dato y para definir su nombre usaremos el estilo snake case.
- Métodos: estos inician con un verbo en infinitivo, el cual denota la acción que puede ejecutar el objeto de la clase. Para el nombre de los métodos usaremos el estilo snake case.

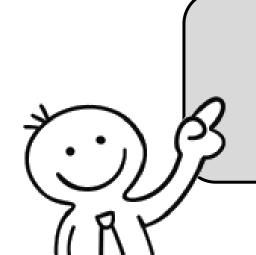


SEUDOCÓDIGO DE UNA CLASE:

Estos son los atributos de la clase. Para definir una clase se usa la palabra Cada uno debe tener un tipo de clase NombreDeLaClase reservada clase seguida del nombre dato asociado. en notación PascalCase. <Tipo Dato> atr 1, atr 2, atr 3, ... publico <Tipo Dato> nombre metodo(<Lista De Parametros>) Esta es la firma de un método, contiene la visibilidad, el tipo de dato de retorno, el nombre del método en notación Tipo Dato var1, var2, var3 camelCase y la lista de parámetros <Instrucción 1> Este es el cuerpo del algoritmo. El paso Esta es la declaración de las <Instrucción 2> a paso de las instrucciones que deben variables locales que se usan realizarse al ejecutar el método dentro del método. Fin Metodo Fin Clase







RETOMEMOS UN EJEMPLO

PROGRAMACIÓN ORIENTADA A OBJETOS







EJEMPLO: Como el profesor de geometría quiere ayudar a sus estudiantes de una manera práctica con una app para la clase, le ha pedido a usted que para empezar desarrolle un programa que le permita calcular el área y el perímetro de un rectángulo.



ANÁLISIS DEL PROBLEMA

Problema: El profesor de geometría quiere ayudar a sus estudiantes de una manera práctica con una app para la clase, por eso le ha pedido a usted que desarrolle un programa que le permita a los estudiantes calcular el área y el perímetro de un rectángulo

CLIENTE Y USUARIO



ENTIDADES DEL MUNDO

- .
- .



REQUERIMIENTOS FUNCIONALES



ENTRADAS, SALIDAS Y EL PROCESO

- .





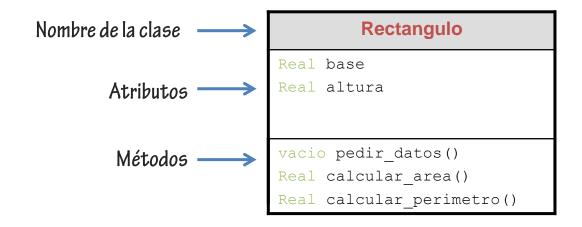
ANÁLISIS DEL PROBLEMA

Cliente	El profesor de geometría
Usuario	Los estudiantes de geometría
Requerimientos Funcionales	El sistema debe permitir: Calcular el perímetro de un rectángulo Calcular el área de un rectángulo
Entidades del Mundo	■ El Rectángulo, del que se requieren la base y la altura
Entradas	Base y altura del rectángulo, ambos son números reales
Salidas	El área del rectánguloEl perímetro del rectángulo
Proceso	 Se solicita al usuario la base y la altura del rectángulo Se calcula el área con la fórmula: area = base * altura Se calcula el perímetro con la fórmula: perimetro = 2*base + 2*altura Se muestra en la pantalla el área y el perímetro calculados

PROGRAMACIÓN ORIENTADA A OBJETOS



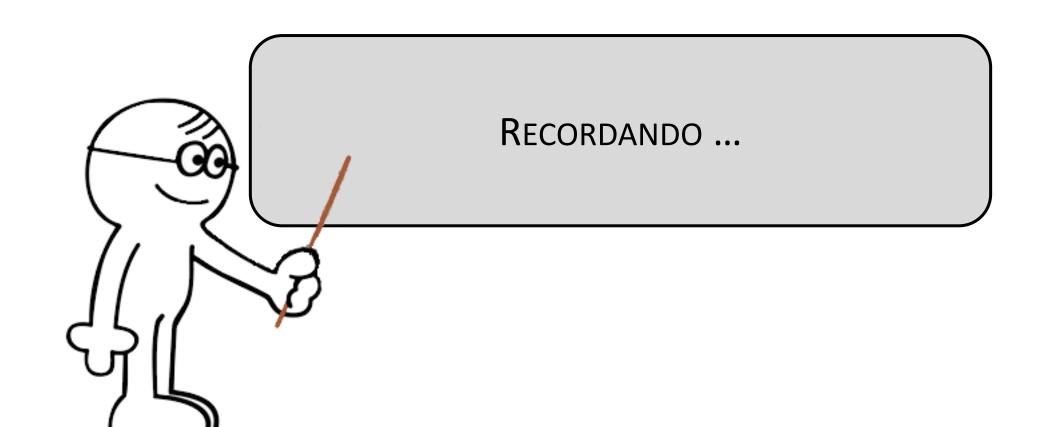
DISEÑO DE LA SOLUCIÓN





DISEÑO DE LA SOLUCIÓN

```
clase Rectangulo
 #Atributos de la clase
 Real base, altura
 #Este método pide la base y altura del rectángulo
publico vacio pedir_datos()
    Escribir ("Ingrese la base y la altura del rectángulo: ")
   Leer (base, altura)
 Fin metodo
#Este método calcula el área del rectángulo
publico Real calcular_area()
  Real area
  area = base * altura
  Retornar area
 Fin metodo
#Este método calcula el perímetro del rectángulo
publico Real calcular perimetro()
  Real perimetro
   perimetro = 2*base + 2*altura
  Retornar perimetro
 Fin metodo
Fin Clase
```



ATRIBUTOS Y MÉTODOS



PROGRAMACIÓN ORIENTADA A OBJETOS

La P.O.O. modela <u>las entidades que intervienen en el problema</u> como objetos, identificando cuáles son sus atributos (o características) y sus comportamientos (o métodos).



DECLARACIÓN DE ATRIBUTOS





Modificador de Acceso

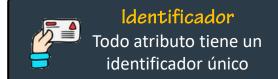
Controla quien puede acceder a la información del atributo: público, privado, protegido



Tipo de Dato

Todo atributo debe estar declaro bajo un Tipo de Dato: Entero,
Cadena, Real, etc.

clase NombreDeLaClase



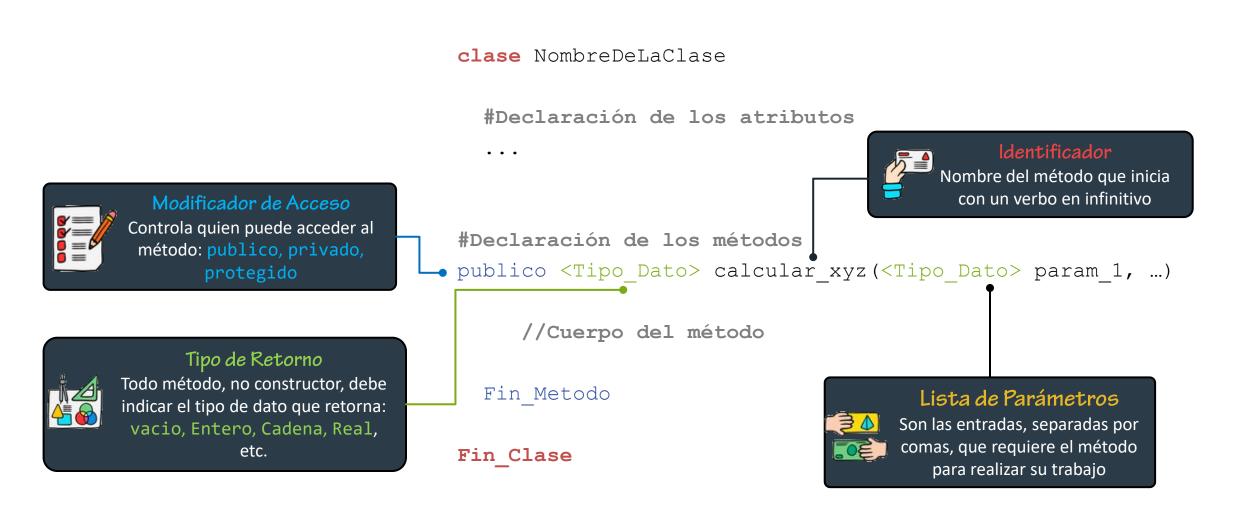
```
#Declaración de los atributos
publico <Tipo_Dato> atr_1
publico <Tipo_Dato> atr_2
```

```
#Declaración de los métodos ...
publico vacio metodo_xyz( ... )
   ...
Fin_Metodo
```

Fin Clase

DECLARACIÓN DE MÉTODOS





GENERALIDADES SOBRE LOS MÉTODOS



El nombre de un método inicia con un verbo en infinitivo, siguiendo la notación snake_case. Por ejemplo, calcular valor a pagar.



Los parámetros podrán ser de cualquier tipo (primitivos, objetos, arreglos, etc.)



No puede haber dos parámetros con el mismo nombre en un mismo método. Se produciría ambigüedad





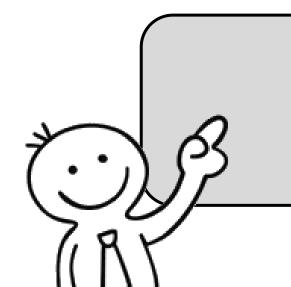
Un método puede tener cualquier cantidad de parámetros. Se trata de una decisión del programador.



No está permitido que el nombre de una variable local, declarada en el cuerpo del método, coincida con el nombre de un parámetro



Si el nombre de algún parámetro coincide con el de un atributo, éste será ocultado por el parámetro, en el método. Para poder acceder al atributo desde el método se debe hacer uso del operador de autorreferencia self, que verás un poco más adelante



Modificadores de Acceso

MODIFICADORES DE ACCESO



Los **Modificadores de Acceso** nos ayudan a limitar qué clases pueden acceder o modificar los atributos y métodos de un objeto de otra clase. Esto nos permite <u>Encapsular</u> y <u>Ocultar</u> la información de los objetos a diferentes niveles. Los modificadores básicos de acceso son tres:

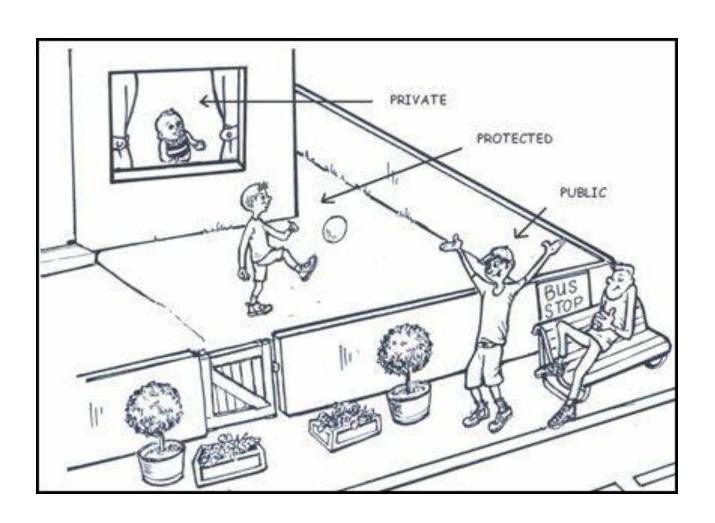
- √ Publico (+): indica que atributo o método de la clase puede ser accedido desde cualquier otra clase, cualquiera que ella sea. En el caso de que un atributo sea público el valor de ese atributo puede ser modificado desde otra clase.
- √ Privado (-): señala que el atributo o método sólo puede ser accedido por la propia clase. Un método o atributo privado no se puede acceder desde ninguna otra clase.
- √ Protegido (#): determina que el atributo o método el elemento es accesible desde la propia clase, desde sus subclases, y desde clases del mismo paquete.



Nota: estos modificadores no existen en todos los lenguajes de programación. Por ejemplo, en Python estos modificadores no existen, se emulan usando doble guion bajo para datos privados (__) y usando un guion bajo (_) para datos protegidos

MODIFICADORES DE ACCESO

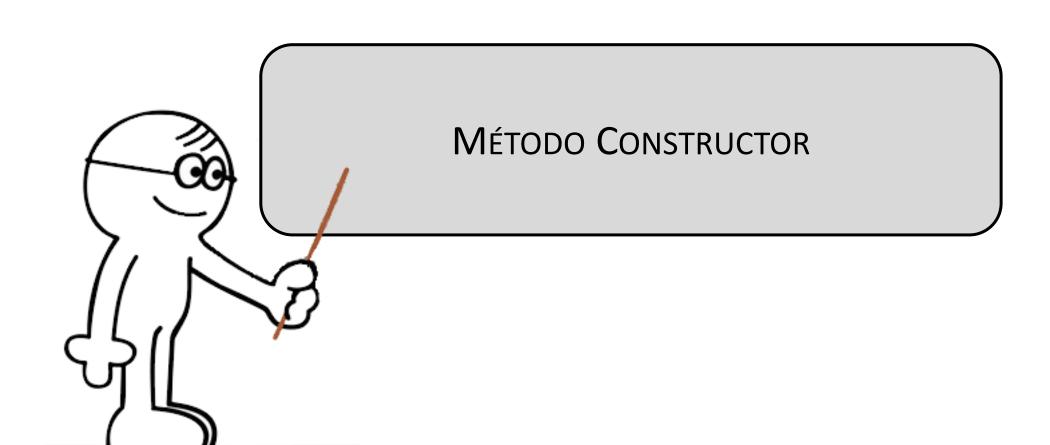




Access Modifiers

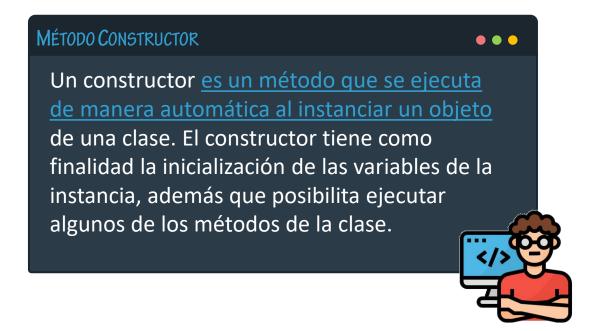


- The public modifier is the simplest one. A component set as public can be accessed anywhere.
- A private component, on the other hand, can only be accessed within its class.
- A protected component stands in the middle. It can be accessed within its class and all derived classes.



MÉTODO CONSTRUCTOR





MÉTODO CONSTRUCTOR



Auto

- Texto marca
- Texto color
- Entero modelo
- + Auto()
- + vacio encender()
- + vacio mostrar_marca()

¿Cómo creamos un objeto de esta clase?

```
publico clase Auto
    #ATRIBUTOS de la clase
    privado Texto marca
    privado Texto color
    privado Entero modelo
   #Método constructor
   Auto()
     self.marca = "Pepito"
     self.color = "rojo"
     self.modelo = 0
   Fin Metodo
   publico vacio encender( )
     Escribir ("El auto ha sido encendido")
   Fin Metodo
   publico vacio mostrar marca( )
     Escribir("Marca: ", self.marca)
   Fin Metodo
Fin Clase
```

MÉTODOS ACCESORES, MODIFICADORES Y ANALIZADORES

MÉTODOS ACCESORES Y MODIFICADORES



MÉTODOS ACCESORES Y MODIFICADORES:

✓ Métodos Consultores o Accesores: son métodos que devuelven el valor almacenado en un atributo que es privado en un objeto, sin modificar su valor. A este tipo de método se le denomina getter.

```
#Este método retorna la marca del auto
publico Texto get_marca()
   Retornar self.marca
Fin Metodo
```

```
#Este método retorna la marca del auto
publico Entero get_modelo()
   Retornar self.modelo
Fin Metodo
```

✓ Métodos Modificadores: son aquellos métodos que modifican el valor de un atributo que es privado en el objeto. Este tipo de métodos son los que llamamos setters.

```
#Este método cambia la marca del auto
publico vacio set_marca(Texto marca)
  self.marca = marca
Fin_Metodo
```

```
#Este método cambia el color del auto
publico vacio set_modelo(Texto color)
  self.modelo = modelo
Fin Metodo
```

MÉTODOS ANALIZADORES



MÉTODOS ANALIZADORES:

✓ Métodos Analizadores: son aquellos que implementan las reglas del negocio o transforman los datos del objeto en un resultado específico.

```
/*Este método retorna la marca del auto*/
publico vacio encender()
    Escribir("El auto ha sido encendido")
Fin_Metodo
```



LÓGICA Y REPRESENTACIÓN I

Programa de Ingeniería de Sistemas