UNIVERSIDAD DE ANTIOQUIA

1 8 0 3

LÓGICA Y REPRESENTACIÓN I

PROGRAMA DE INGENIERÍA DE SISTEMAS

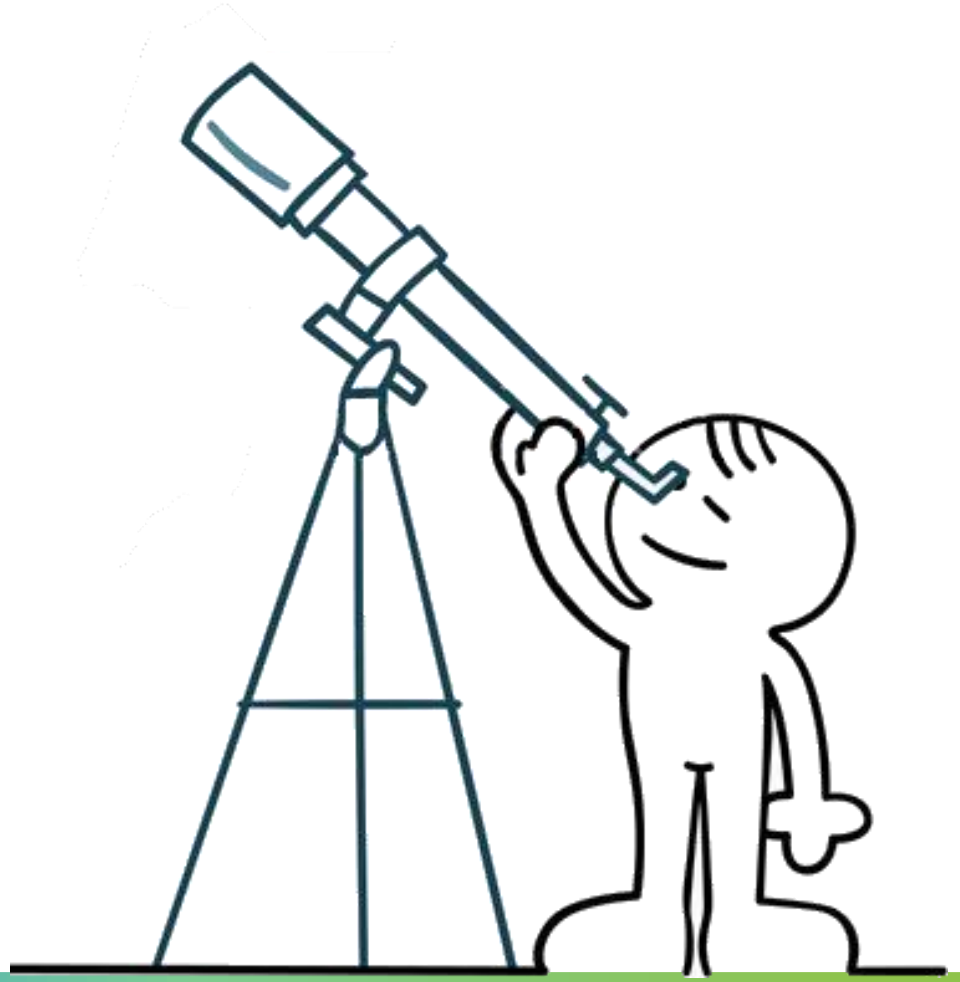
Material desarrollado por **Carlos Andrés Mera Banguero**
https://github.com/carlosmera20/

CONTENIDO

UNIVERSIDAD DE ANTIOQUIA

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

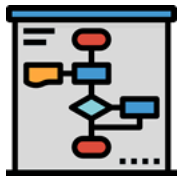
Los Paradigmas de Programación

UNIVERSIDAD DE ANTIOQUIA

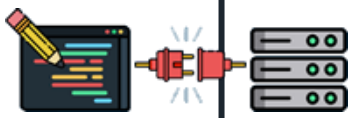
Paradigmas de Programación

Los paradigmas de programación interpretan el mundo de una determinada manera y sobre esa visión establecen reglas y guías para resolver un problema determinado usando un lenguaje de programación

Programación Estructurada
Establece que la solución de un problema se puede definir a partir de tres estructuras de control: secuenciales, de selección y de iteración.



Programación Orientada a Objetos
Define al mundo como un conjunto de objetos (con datos y comportamiento) que interactúan entre sí para la solución de un problema.



Programación Funcional
La solución de un problema se puede descomponer en pequeñas tareas que se implementan en funciones y que, como en matemáticas, se pueden llamar de manera recursiva para solucionar un problema

Programación Reactiva
Considera que el mundo no es más que un flujo de datos asincrónicos observable y que reacciona frente a ciertos cambios

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

UNIVERSIDAD DE ANTIOQUIA

DEFINICIÓN

La P.O.O. modela las entidades que intervienen en el problema 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.

PERSONA

Atributos

```
nombre
correo
descripción
password
```

Los atributos definen el estado (o datos) de los objetos

Métodos

```
editar_perfil
Cambiar_password
autenticarse
```

Los métodos definen el comportamiento de los objetos

Programación Orientada a Objetos

UNIVERSIDAD DE ANTIOQUIA

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

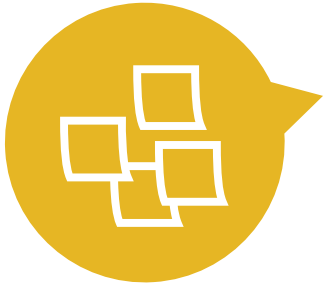
Permite proteger la información de los objetos de manipulaciones no autorizadas ocultando incluso los detalles de la implementación.

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.



¿Qué es un Objeto?

UN OBJETO ES ...

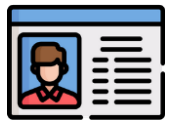
CUALQUIER COSA!

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIVERSIDAD DE ANTIOQUIA

- **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.

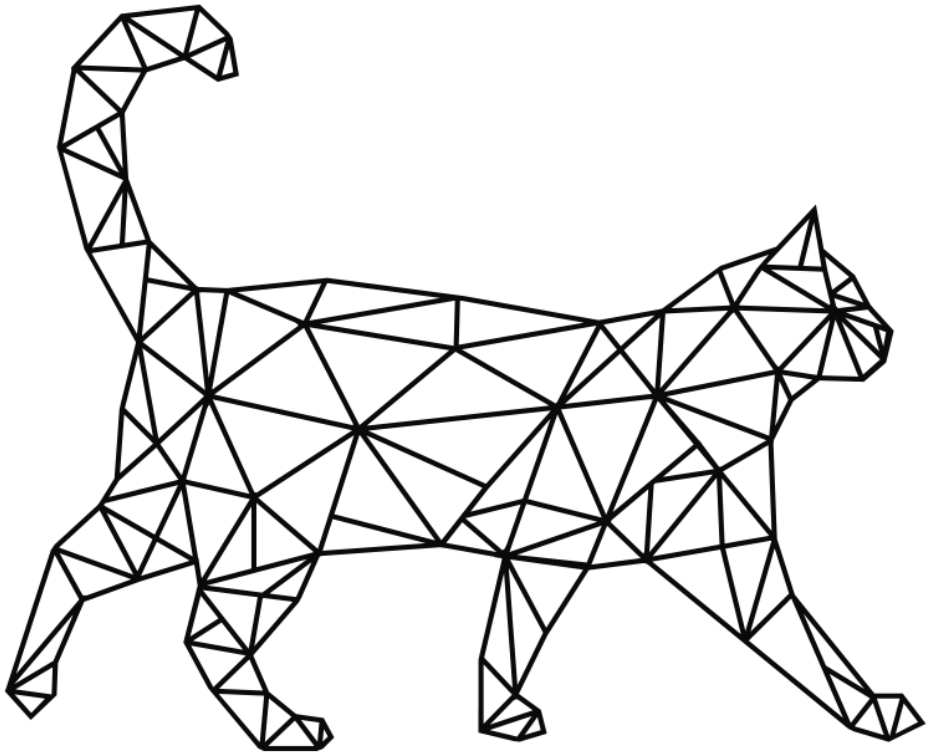
UNIVERSIDAD DE ANTIOQUIA

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

UNIVERSIDAD DE ANTIOQUIA
1803



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

¿Qué es una Clase?

UNIVERSIDAD DE ANTIOQUIA

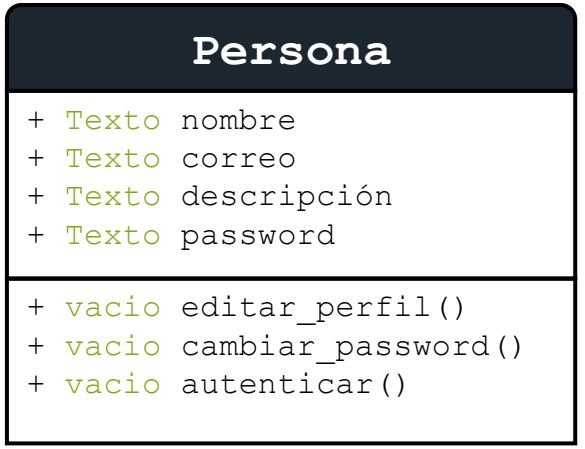
DEFINICIÓN DE CLASE:

CLASE

Una clase es una plantilla para la creación de objetos. Así , una clase define cuáles son los atributos y el comportamiento que tendrán todos los objetos que se creen a partir de ella.

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)



UNIVERSIDAD DE ANTIOQUIA

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! 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.

UNIVERSIDAD DE ANTIOQUIA

REPRESENTACIÓN DE CLASES Y OBJETOS

TENGA EN CUENTA:

- *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`.

UNIVERSIDAD DE ANTIOQUIA

SEUDOCÓDIGO DE UNA CLASE:

Para definir una clase se usa la palabra reservada `clase` seguida del nombre en notación PascalCase.

Estos son los atributos de la clase. Cada uno debe tener un tipo de dato asociado.

Esta es la firma de un método, contiene la visibilidad, el tipo de dato de retorno, el nombre del método en notación camelCase y la lista de parámetros

Este es el cuerpo del algoritmo. El paso a paso de las instrucciones que deben realizarse al ejecutar el método

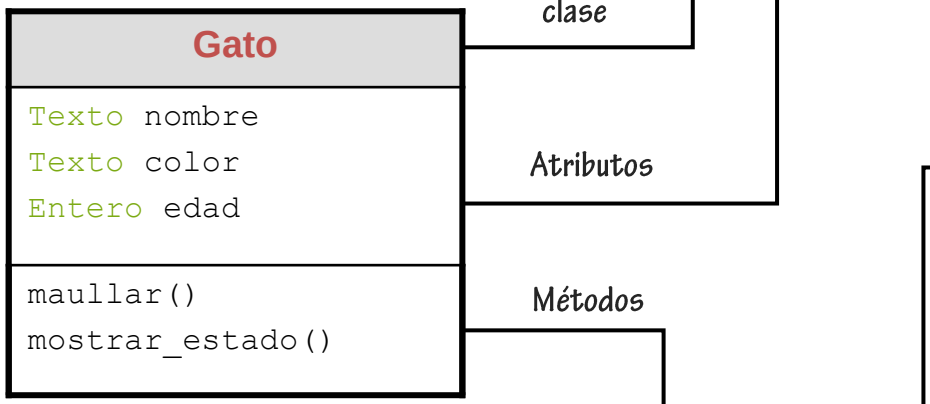
Esta es la declaración de las variables locales que se usan dentro del método.

```
clase NombreDeLaClase

    <Tipo_Dato> atr_1, atr_2, atr_3, ...

    publico <Tipo_Dato> nombre_metodo(<Lista_De_Parametros>)

        Tipo_Dato var1, var2, var3

        <Instrucción 1>
        <Instrucción 2>
        ...
    Fin_Metodo

Fin_Clase
```

PROGRAMACIÓN ORIENTADA A OBJETOS

UNIVERSIDAD DE ANTIOQUIA

SEUDOCÓDIGO DE UNA CLASE:




```
clase Gato
    #Atributos
    Texto nombre
    Texto color
    Entero edad

    #Este método hace que gato maúlle
    publico vacio maullar()
      Escribir ("Miau, Miau, Miau")
    Fin_Metodo

    #Este método hace que el gato muestre sus datos
    publico vacio mostrar_estado()
      Escribir("Mi nombre es ", nombre)
      Escribir("Soy de color ", color)
      Escribir("Tengo ", edad, " años")
    Fin_Metodo
Fin_Clase
```

Retomemos un Ejemplo

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.

UNIVERSIDAD DE ANTIOQUIA

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

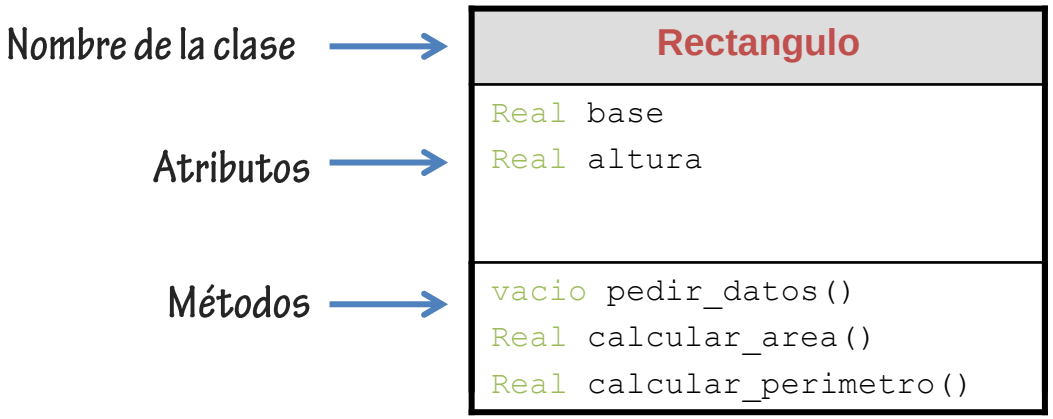
REQUERIMIENTOS FUNCIONALES

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 rectángulo ▪ El 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

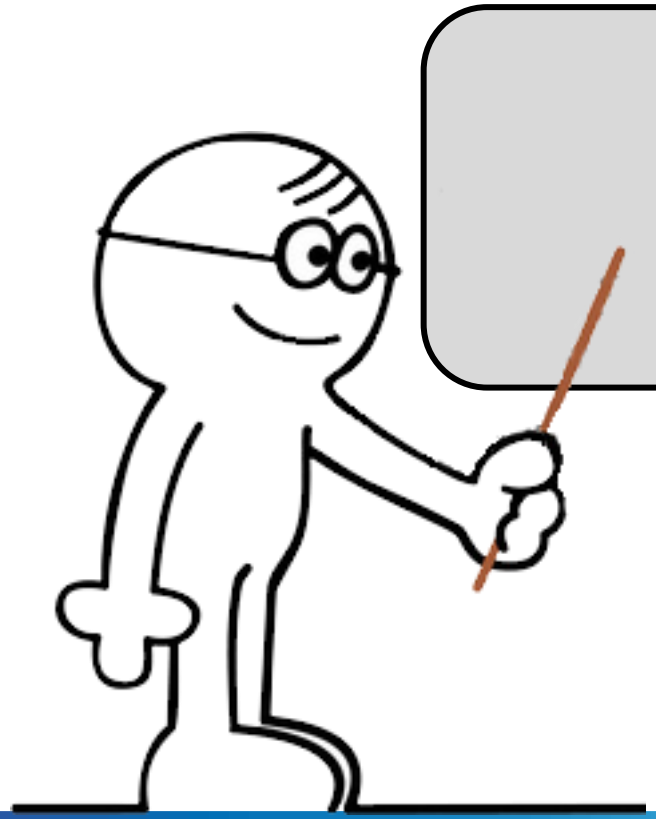
DISEÑO DE LA SOLUCIÓN



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
```

RECORDANDO ...



ATRIBUTOS Y MÉTODOS

UNIVERSIDAD DE ANTIOQUIA

PROGRAMACIÓN ORIENTADA A OBJETOS

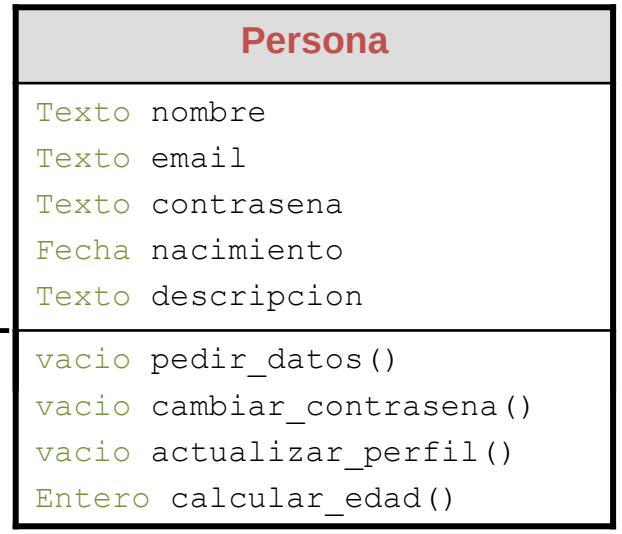
- La P.O.O. modela las entidades que intervienen en el problema como objetos, identificando cuáles son sus atributos (o características) y sus comportamientos (o métodos).

ATRIBUTOS

Son las características que definen la estructura interna de los objetos de un problema. Es decir, son los datos que, de acuerdo con el problema, se deben almacenar del objeto.

MÉTODOS

Los métodos definen el comportamiento de los objetos. Permiten que un objeto se comunique con otro objeto solicitándole cierta información o pidiéndole que ejecute una determinada tarea.



DECLARACIÓN DE ATRIBUTOS

UNIVERSIDAD DE ANTIOQUIA

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.

Identificador
Todo atributo tiene un identificador único

```
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

UNIVERSIDAD DE ANTIOQUIA

```
clase NombreDeLaClase

  #Declaración de los atributos
  ...


#Declaración de los métodos
publico <Tipo_Dato> calcular_xyz(<Tipo_Dato> param_1, …)

    //Cuerpo del método

  Fin_Metodo

Fin_Clase
```

Modificador de Acceso
Controla quien puede acceder al método: publico, privado, protegido

Identificador
Nombre del método que inicia con un verbo en infinitivo

Tipo de Retorno
Todo método, no constructor, debe indicar el tipo de dato que retorna: vacio, Entero, Cadena, Real, etc.

Lista de Parámetros
Son las entradas, separadas por comas, que requiere el método para realizar su trabajo

GENERALIDADES SOBRE LOS MÉTODOS

UNIVERSIDAD DE ANTIOQUIA

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.)

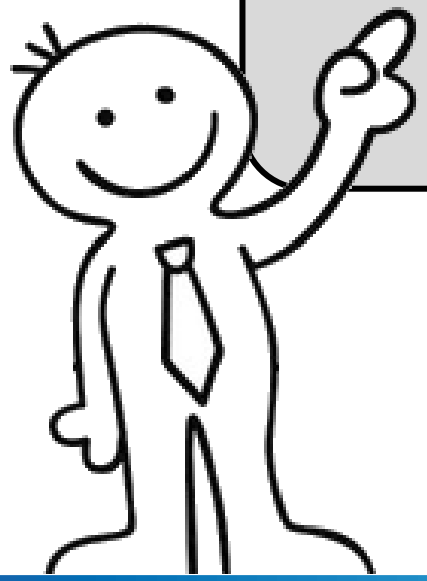
No puede haber dos parámetros con el mismo nombre en un mismo método. Se produciría ambigüedad

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

UNIVERSIDAD DE ANTIOQUIA

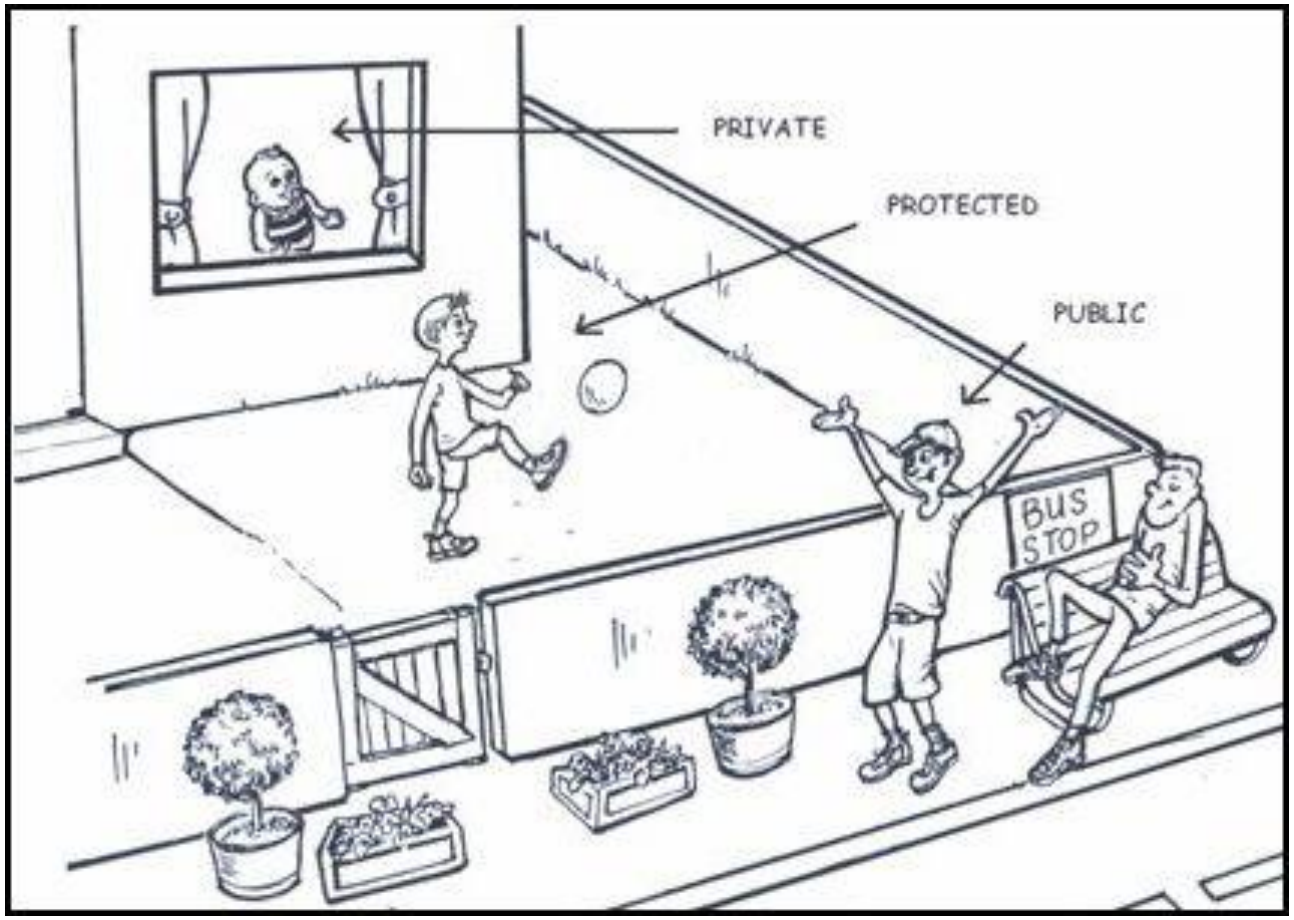
Los **Modificadores de Acceso** nos ayudan a limitar qué clases pueden acceder o modificar los atributos y métodos de un objeto de otra clase. Esto nos permite Encapsular y Ocultar la información de los objetos a diferentes niveles. Los modificadores básicos de acceso son tres:

- 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

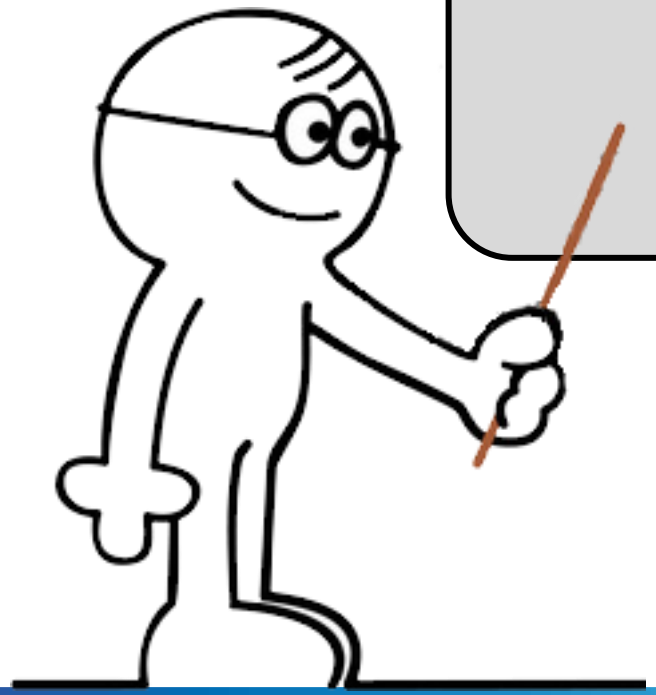
UNIVERSIDAD DE ANTIOQUIA
1803



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

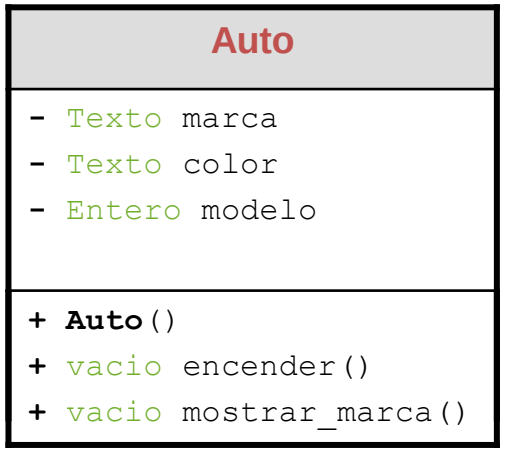
UNIVERSIDAD DE ANTIOQUIA

MÉTODO CONSTRUCTOR

Un constructor es un método que se ejecuta de manera automática al instanciar un objeto de una clase. El constructor tiene como finalidad la inicialización de las variables de la instancia, además que posibilita ejecutar algunos de los métodos de la clase.

MÉTODO CONSTRUCTOR

UNIVERSIDAD DE ANTIOQUIA



¿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()
      marca = "Pepito"
      color = "rojo"
      modelo = 0
    Fin_Metodo

    publico vacio encender( )
      Escribir("El auto ha sido encendido")
    Fin_Metodo

    publico vacio mostrar_marca( )
      Escribir("Marca: ", marca)
    Fin_Metodo

Fin_Clase
```

Métodos Accesores, Modificadores y Analizadores

MÉTODOS ACCESORES Y MODIFICADORES

UNIVERSIDAD DE ANTIOQUIA

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 marca
Fin_Metodo
```

```
#Este método retorna la marca del auto
publico Entero get_modelo( )
  Retornar 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)
  this.marca = marca
Fin_Metodo
```

```
#Este método cambia el color del auto
publico vacio set_modelo(Texto color)
  this.modelo = modelo
Fin_Metodo
```

MÉTODOS ANALIZADORES

UNIVERSIDAD DE ANTIOQUIA

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
```

UNIVERSIDAD DE ANTIOQUIA

1 8 0 3

LÓGICA Y REPRESENTACIÓN I

PROGRAMA DE INGENIERÍA DE SISTEMAS

Material desarrollado por **Carlos Andrés Mera Banguero**
https://github.com/carlosmera20/