El futuro digital es de todos

MinTIC

Mision TIC 2022

Semana 4

John Anderson Gómez Múnera

UNIVERSIDAD DE ANTIOQUIA
1803
Facultad de Ingeniería

Temario

- Programación orientada a objetos
- Clases
- Herencia y polimorfismo
- Arreglos de dos dimensiones

Programación +++ Orientada a Objetos

Una clase es un tipo que tiene asociado las operaciones que se pueden ejecutar con objetos de esa clase.

¿Qué debe hacer una clase?

Debe cumplir con un propósito especifico, es decir, capturar una idea. Cuando más de una idea es encapsulada en una clase, se pierden posibilidades de reusó

Elementos y características de las clases en la programación orientada a objetos

- Atributos: son los datos propios de cada objeto perteneciente a la clase.
- Constructor: crea una instancia de la clase e inicializa sus atributos.
- Métodos: cada método es un subprograma; son las acciones que puede realizar un objeto perteneciente a la clase.
- Encapsulamiento: los atributos solo se podrán acceder a través de métodos.
- Polimorfismo: puede haber más de un método con el mismo nombre.
- Herencia: se definen clases derivadas de otra clase, las cuales podrán ejecutar los métodos definidos para la clase base.

Instanciar

instance

/ˈɪnst(ə)ns/

noun

an example or single occurrence of something.
"a serious instance of corruption"

Similar: example | occasion | occurrence | case | representative case

verb

cite (a fact, case, etc.) as an example.
"I instanced Bob as someone whose commitment had certainly got things done"

Similar: cite | quote | refer to | make reference to | mention | allude to

Traducir instance al Español

noun

1. ejemplo
2. instancia

verb

1. citar como ejemplo
2. poner por caso

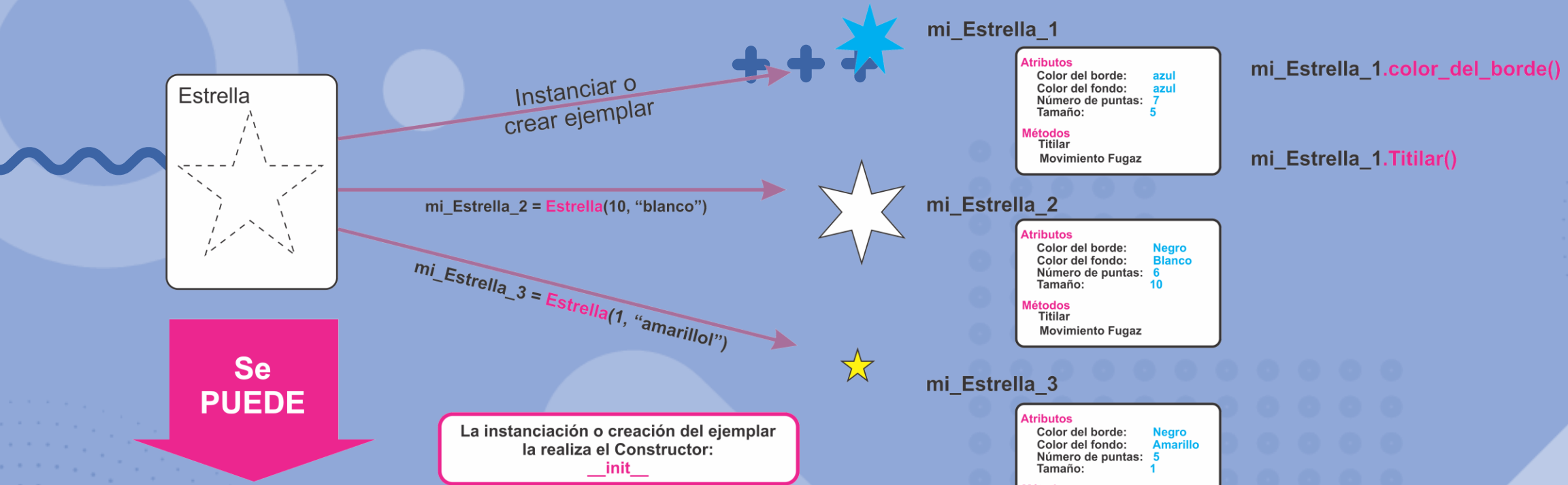
Clase

Instanciar
(Crear Ejemplar)

Objeto

Class Estrella:

mi_Estrella_1 = Estrella(5, “azul”)



mi_Estrella_1

Atributos	
Color del borde:	azul
Color del fondo:	azul
Número de puntas:	7
Tamaño:	5
Métodos	
Titilar	
Movimiento Fugaz	

mi_Estrella_1.color_del_borde()

mi_Estrella_1.Titilar()

mi_Estrella_2

Atributos	
Color del borde:	Negro
Color del fondo:	Blanco
Número de puntas:	6
Tamaño:	10
Métodos	
Titilar	
Movimiento Fugaz	

mi_Estrella_3

Atributos	
Color del borde:	Negro
Color del fondo:	Amarillo
Número de puntas:	5
Tamaño:	1
Métodos	
Titilar	
Movimiento Fugaz	

Se PUEDE

Crear nuevas clases a partir de la inicial con herencia

Dichas clases pueden tener:
Atributos y
métodos específicos

La instanciación o creación del ejemplar la realiza el Constructor: __init__

NO PUEDE

Crear nuevos métodos
Crear nuevos atributos
Crear otros objetos
Tener atributos no asignados
Realizar métodos no asignados

Los datos que pertenecen a una clase, por lo general se definen privados, y se conocen como los atributos de esa clase.

Las operaciones que pueden realizar los objetos de la clase son en realidad subprogramas, los cuales seguiremos llamando métodos y que pueden ser privados o públicos.

Cada método es una función. El constructor no es un método. El constructor es la función que se ejecuta cuando se defina una variable como una instancia de la clase vector

El encapsulamiento

Hace referencia al ocultamiento de los estado internos de una clase al exterior.

Dicho de otra manera, encapsular consiste en hacer que los atributos o métodos internos a una clase no se puedan acceder ni modificar desde fuera, sino que tan solo el propio objeto pueda acceder a ellos.

```python
class Perro:
    # Atributo de clase
    especie = 'mamífero'

    # El método __init__ es llamado al crear el objeto
    def __init__(self, nombre, raza):
        print(f"Creando perro {nombre}, {raza}")

        # Atributos de instancia
        self.nombre = nombre
        self.raza = raza

    def ladra(self):
        print("Guau")

    def camina(self, pasos):
        print(f"Caminando {pasos} pasos")
```

Visibilidad de los atributos y métodos

Una clase revela sus métodos y atributos a otras clases usando visibilidad (encapsulamiento) Hay cuatro tipos diferentes de visibilidad que se pueden aplicar a los atributos y métodos de una clase (Publica “+”, Protegida, de Paquete y Privada “-”).

Python	Objetivo
	Pública: accesible para todos
__	Privada: Solo podrá ser accedida o modificada por la misma clase
_	Protegida: Establece que solo puede ser accedido por esa clase y sus sub-clases

https://quizizz.com/join?gc=55651514

Atributos intrínsecos

- __name__ nombre de la clase
- __module__ el modulo (o libreria) desde la cual es cargada
- __dict__ un diccionario (un set de pares clave.valor) contiene todos los atributos

(inclusión de métodos)

- __doc__ muestra el docstring.

Para Objetos:

- __class__ El nombre de la clase del objeto
- __dict__ Un diccionario que contiene todos los atributos del objeto.

Atributos de clase

En Python las clases también pueden tener atributos; estos se denominan variables de clase (a diferencia de las variables de instancia o atributos). En Python las variables definidas dentro del ámbito de la clase, pero fuera de cualquier método están ligadas a la clase y no a ninguna instancia, por lo que son variables de clase.

Herencia

La **herencia** es un proceso mediante el cual se puede crear una clase **hija** que hereda de una clase **padre**, compartiendo sus métodos y atributos. Además de ello, una clase hija puede sobreescribir los métodos o atributos, o incluso definir unos nuevos.

```python
# Definimos una clase padre
class Animal:
    pass

# Creamos una clase hija que hereda de la padre
class Perro(Animal):
    pass
```

Ejercicio

Escribir un programa que defina una superclase vehículo, identifique atributos y métodos. La clase debe tener por lo menos un método que despliegue información. Crear a su vez Clases múltiples heredadas para los tipos espacios en los que actúan. Cree a su vez Clases heredadas para diferenciar algunos de ellos en un terreno especifico. Todas las clases deben contar con método para desplegar información.

Polimorfismo

El término polimorfismo tiene origen en las palabras poly (muchos) y morfo (formas), y aplicado a la programación hace referencia a que los objetos pueden tomar diferentes formas.

Objetos de diferentes clases pueden ser accedidos utilizando el mismo interfaz, mostrando un comportamiento distinto (tomando diferentes formas) según cómo sean accedidos

Al ser un lenguaje con tipado dinámico y permitir duck typing, en Python no es necesario que los objetos compartan un interfaz, simplemente basta con que tengan los métodos que se quieren llamar.

Interfaz

Una interfaz es un conjunto de funciones, métodos o atributos con nombres específicos. Una interfaz es un contrato entre el programador que realiza una clase y el que la utiliza, puede consistir en uno solo o varios métodos o atributos.

Polimorfismo

```python
class Animal:
    def hablar(self):
        pass
```

```python
class Perro(Animal):
    def hablar(self):
        print("Guau!")

class Gato(Animal):
    def hablar(self):
        print("Miau!")
```

```python
for animal in Perro(), Gato():
    animal.hablar()

# Guau!
# Miau!
```

Duck Typing, Sobre carga y Polimorfismo

Esto del ejemplo en Python no es posible principalmente porque ya deberías saberlo.. "La dinamicidad" de python causa un conflicto al no saber que tipo de variables serian en el caso del ejemplo "a" y "b". Podrías meter un string con un entero y Python se quebraría... Cómo ves el ejemplo anterior era en JAVA donde se especifica el tipo de parámetro al momento de crear el método (INT).

En conclusión en Python:

- "**El mismo nombre de Métodos en *diferentes clases*" SI**!. (Polimorfismo)
- "**El mismo nombre de Métodos en la *misma clase*" NO**! (Sobrecarga)

https://quizizz.com/join?gc=40836794