Curso de Desarrollo en Lenguaje Python para Inteligencia Artificial (Málaga)

M.374.001.001

29 de marzo 2021 09:30-13:30

Modulo 2 – Tema 2

Spiros Michalakopoulos

Gobierno de España · Ministerio de Industria, Comercio y Turismo · EOI Escuela de organización industrial

Unión Europea · Fondo Social Europeo · El FSE invierte en tu futuro

polo de contenidos digitales

Ayuntamiento de Málaga

Módulo 2: Programación orientada a objetos (POO) en Python

Desarrollo en Lenguaje Python para Inteligencia Artificial

Año de realización: 2021

PROFESOR
Spiros Michalakopoulos

spiros.eoi@gmail.com
https://www.linkedin.com/in/spiros-michalakopoulos

www.eoi.es

Índice

Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Tema 2

Encapsulación, propiedades. Composición y agregación.

Lunes 29 de marzo

1. Prueba módulo 1
2. Ejercicio 1F – métodos especiales
3. Encapsulación, propiedades
4. Composición, agregación

Encapsulación (1/2)

Protección de datos

Uno de los objetivos que tiene la programación orientada a objetos es proteger los datos de acceso o usos no controlados, y esto es lo que se conoce como **encapsulación** (o **encapsulamiento**). Hay dos tipos de datos (atributos):

- **Públicos**: accesibles sin control
- **Privados**: accesibles por métodos (get y set)

Encapsulación (2/2)

Evitar imprevistos

De esta forma el usuario de la clase puede **obviar la implementación** de los métodos y propiedades para concentrarse solo en cómo usarlos. Por otro lado se evita que el usuario pueda cambiar su estado de maneras **imprevistas** e **incontroladas**.

La **encapsulación** no solo puede realizarse sobre los atributos de la clase, sino también **sobre los métodos**.

Atributos, métodos y propiedades

¿Qué son las propiedades de una clase?

Propiedades son un tipo especial de atributos, con un comportamiento ligeramente diferente en su implementación en Python.

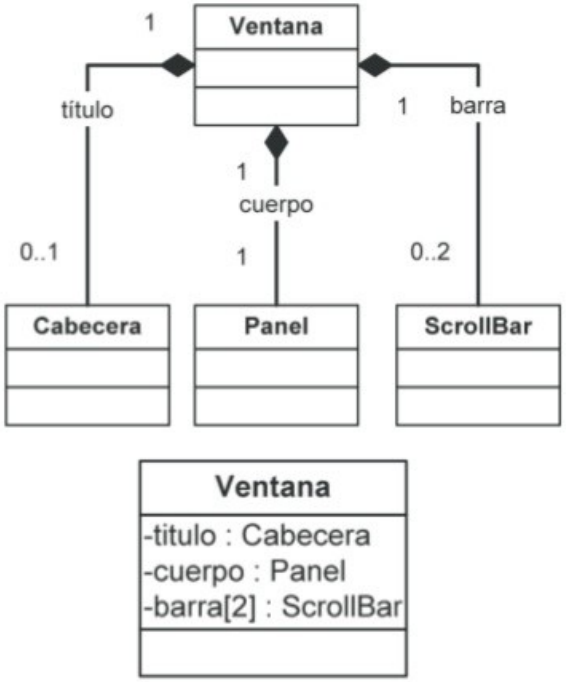
Composición (1/2)

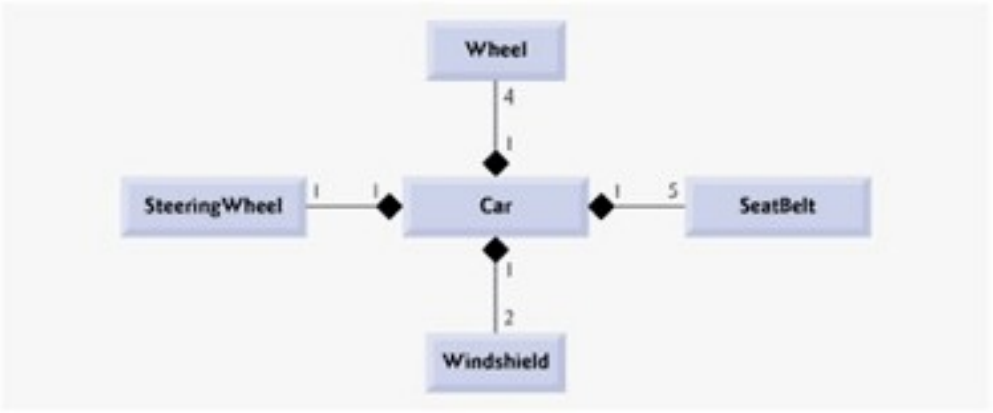
Clases con otras clases como atributos

La **composición** consiste en la creación de nuevas clases a partir de otras clases ya existentes que actúan como elementos compositores de la nueva. Las clases existentes serán atributos de la nueva clase.

La **composición** significa que entre las dos clases existe una relación del tipo "**tiene un**", ("**has-a**").

Composición (2/2)





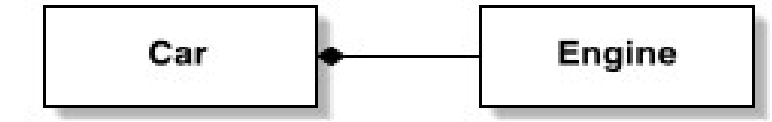
Agregación

Una variante de “has-a”

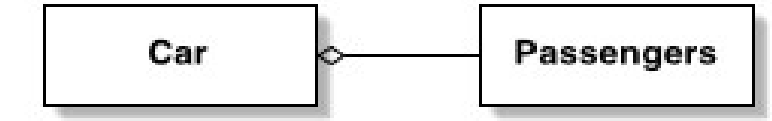
La **agregación** es una variante de la relación “**tiene un**”. Se puede dar cuando una clase es una colección o un contenedor de otras clases, pero a su vez, el tiempo de vida de las clases contenidas no tienen una dependencia fuerte del tiempo de vida de la clase contenedora. Es decir, el contenido de la clase contenedora no se destruye automáticamente cuando desaparece dicha clase.

Composición vs. Agregación

La diferencia con ejemplos



Composition: every car has an engine.



Aggregation: cars may have passengers, they come and go

- **Composición**: Cuando se elimina el contenedor, el contenido también es eliminado. Ejemplo, si eliminamos una universidad, eliminamos igualmente sus departamentos.
- **Agregación**: Cuando el contenedor es eliminado, el contenido usualmente no es destruido. Ejemplo, un profesor tiene estudiantes, cuando el profesor muere, los estudiantes no mueren con el.