# Introducción a la Programación Orientada a Objetos

Luis José Sánchez González

# 1. ¿QUÉ ES LA POO?

La programación orientada a objetos es una forma de programar que se basa en la utilización de **objetos**.

# 2. ¿QUÉ SON LOS OBJETOS?

Un objeto en términos de POO no se diferencia mucho de lo que conocemos como un objeto en la vida real.

Pensemos por ejemplo en un coche. Nuestro coche sería un objeto concreto de la vida real, igual que el coche del vecino, o el coche de un compañero de trabajo, o un deportivo que vimos por la calle el fin de semana pasado...

Todos esos coches serían objetos concretos que podemos ver y tocar. Usando la terminología de la POO diríamos que son **instancias**.

# 3. CLASES

Tanto mi coche como el coche del vecino tienen algo en común, ambos son coches. En este caso mi coche y el coche del vecino serían **instancias** y coche (a secas) sería una **clase**. La palabra coche define algo genérico, es una abstracción, no es un coche concreto sino que hace referencia a algo que tiene una serie de propiedades como matrícula, marca, modelo, color, etc. Este conjunto de propiedades se denominan **atributos**.

Podríamos esquematizar este ejemplo de la siguiente manera:

```
definición de la clase coche

        atributos
                matricula
                marca
                modelo
                color
                numero_de_plazas

fin de la definición de coche

crea mi_coche como instancia de la clase coche
crea coche_de_vecino como instancia de la clase coche
crea furgo1 como instancia de la clase coche
crea furgo2 como instancia de la clase coche
crea tartana como instancia de la clase coche
```

**Ejercicios**

1. ¿Cuáles serían los atributos de la clase piloto de fórmula 1? ¿se te ocurren algunas instancias de esta clase? Define la clase y crea algunas instancias, todo ello en pseudocódigo, igual que hemos hecho en el ejemplo anterior.

2. A continuación tienes una lista en la que están mezcladas las clases con las instancias. Di cuáles son las clases, cuáles las instancias y a qué clase pertenece cada una de estas instancias:

   goofy, gardfiel, perro, caballo, tom, silvestre, rocinante, milu, snoopy, gato, pluto, bucefalo, pegaso, ayudante_de_santa_claus, laika

3. ¿Cuáles serían los atributos de la clase gato? Defínelos utilizando pseudocódigo. Crea las instancias de la clase gato que aparecen en el ejercicio anterior.

4. Piensa en la liga de baloncesto, ¿qué 5 clases se te ocurren para representar 5 elementos distintos que intervengan en la liga?

# 4. MÉTODOS

¿De qué nos sirven los objetos si no podemos hacer nada con ellos? Un coche lo podemos arrancar, parar, aparcar, podemos hacer que suene el claxon, podemos llevarlo a reparar... Un gato puede comer, dormir, maullar, ronronear...

A las acciones que se pueden realizar con las instancias se llaman **métodos**. Una parte importante de la definición de una clase consiste en la definición de dichos métodos.

Vamos a ampliar nuestro primer ejemplo con algunos métodos:

```
definición de la clase coche

        atributos
                matricula :             cadena de caracteres
                marca :                 cadena de caracteres
                modelo :                cadena de caracteres
                color :                 cadena de caracteres
                numero_de_plazas : número entero

        métodos
                arranca
                para
                anda
                pita
                gira

fin de la definición de coche

eleonor = coche.crear     #crea eleonor como instancia de la clase coche
eleonor.arranca
eleonor.pita
eleonor.anda(200)
eleonor.gira(45)
eleonor.anda(100)
eleonor.para
```

Hemos definido la clase coche especificando los atributos (indicando también el tipo al que pertenece cada uno de esos atributos) y hemos especificado los métodos de la clase. Lógicamente, si queremos que un programa funcione bien deberemos definir bien qué es lo que hace cada método, cosa que no hemos hecho en este ejemplo.

A continuación se ha creado una instancia de la clase coche con nombre **eleonor**. Hemos arrancado el coche, ha pitado, ha recorrido 200 metros, después ha girado 45 grados, ha continuado otros 100 metros y por último ha parado el motor.

A **aplicar un método** también se le suele llamar **mandar un mensaje**. Si tenemos eleonor.arranca podemos decir que le estamos aplicando el método arranca a la instancia eleonor, o también que estamos mandando un mensaje a la instancia eleonor diciendo que arranque.

Como te habrás dado cuenta ya, en POO debemos centrar nuestra atención en la definición de las clases. El programa en sí saldrá solo si tenemos las clases bien definidas.

**Ejercicios**

5. Crea en pseudocódigo la clase **caballo**, incluyendo los atributos y los nombres de algunos de los métodos. Crea también varias instancias (puedes utilizar los ejemplos del ejercicio 2) y realiza algunas acciones con esas instancias, es decir, aplícale los métodos a las instancias.

6. Crea en pseudocódigo la clase **alumno**. ¿Sería **nombre** uno de los atributos de la clase? Razona tu respuesta.

7. Crea en pseudocódigo la clase **pájaro**. Crea varias instancias y aplícales algunos métodos.

Vamos a volver sobre uno de los ejercicios anteriores, concretamente sobre aquel que nos pedía crear la clase **gato**. Vamos a crear esta clase con los atributos y métodos correspondientes.

```
definición de la clase gato

        atributos
                raza :      cadena de caracteres
                color :     cadena de caracteres
                sexo :     "macho" o "hembra"
                permitir el acceso al atributo sexo

        métodos
                maullar
                    muestra "Miau miau"
                fin maullar

                ronronear
                    muestra "mrrrrrr"
                fin ronronear

                comer (comida)
                    si comida == "pescado"
                            muestra "Hmmm, gracias"
                        si no
                            muestra "Lo siento, yo sólo como pescado"
                    fin si
                fin comer

                pelear(contrincante)
                    si @sexo == "hembra"
                            muestra "No me gusta pelear"
                        si no
                            si contrincante.sexo == "hembra"
                                    muestra "No peleo contra gatitas"
                                si no
                                    muestra "Ven aquí que te vas a enterar"
                            fin si
                    fin si
                fin pelear

fin de la definición de gato
```

```
garfield = gato.crear
garfield.sexo = "macho"
garfield.maullar
garfield.ronronear
garfield.comer("tortilla de patatas")
garfield.comer("pescado")
tom = gato.crear
garfield.sexo = "macho"
tom.comer("sopa de verduras")
lisa = gato.crear
lisa.ronronear
lisa.sexo = "hembra"

# concierto de gatos
garfield.maullar
tom.maullar
lisa.maullar

# vamos a ver si se pelean
garfiel.pelear(lisa)
lisa.pelear(tom)
tom.pelear(garfield)
```

Hemos tenido que acceder directamente al atributo **sexo** para indicar si el gato que creamos es macho o hembra y para ello hemos tenido que permitir previamente el acceso a este atributo.

Normalmente esto resulta poco elegante. Estaría mejor indicar el sexo del gato en el momento de crearlo, para ello definiremos el **constructor** de la clase gato que es el método **crear**.

```
definición de la clase gato

        atributos
                raza :      cadena de caracteres
                color :     cadena de caracteres
                sexo :      "macho" o "hembra"

        métodos

                crear (genero)
                      @sexo = genero
                fin crear

                de_que_sexo
                      devuelve @sexo
                fin_de_que_sexo

                # El resto de métodos se dejaría igual


fin de la definición de gato

garfield = gato.crear("macho")
tom = gato.crear("macho")
lisa = gato.crear("hembra")
```

```
muestra "Garfield es "
muestra garfield.de_que_sexo
muestra " y Lisa es "
muestra lisa.de_que_sexo
```

**Ejercicios**

8. Crea en pseudocódigo la clase **fracción**. Obviamente, los atributos serán **numerador** y **denominador**. Y algunos de los métodos pueden ser invertir, simplificar, multiplicar por un número, dividir entre un número, etc.

9. Crea la clase **ventana**, no es necesario definir los métodos, sólo escribir los nombres. Piensa en las propiedades y en el comportamiento de una ventana de cualquier programa.