Clases, objetos e instancias. Estado y comportamiento

La programación orientada a objetos intenta representar la realidad. El paradigma estructurado, en cambio, lidia con estructura de datos y rutinas o funciones que operan sobre esos datos. Por su lado, la programación orientada a objetos "habla" en los mismos términos que hablamos las personas. Entiende a las entidades de software como "cosas" (objetos) que "realizan" acciones.

La programación orientada a objetos tiene cuatro pilares fundamentales que iremos viendo con el tiempo:

1. Abstracción
2. Encapsulamiento
3. Herencia
4. Polimorfismo

Cada uno de estos conceptos, por más que parezcan complejos, no son nada más que características concretas que vemos en las cosas que nos rodean.

Abstracción: las clases

La abstracción consiste en capturar la esencia de una cosa. Es decir, mirar lo que tenemos a nuestro alrededor y hacer algún tipo de modelo mental de ello: **¿qué características tiene esta cosa? ¿Qué hace? ¿Para qué sirve? ¿Cambia con el tiempo?

Gracias a nuestros modelos mentales de las cosas es que las podemos reconocer. Cuando nos sentamos frente a la computadora tenemos que indicar, de alguna manera, cuáles son esos modelos.

En la programación orientada a objetos usamos a las clases como la forma de indicar cuáles son los modelos y cuáles son sus características.

Cuando definimos una clase, definimos una plantilla, un modelo, la "idea de" algo, "la abstracción". Con la definición de la clase, plasmamos ese modelo en nuestro programa. En el mundo real, a partir de un modelo podemos obtener las cosas. Asimismo, en la computadora, a partir de la clase obtenemos los objetos.

© Universidad de Palermo. Prohibida la reproducción total o parcial de imágenes y textos.

Supongamos los planos de una casa: cada vez que queremos construir una casa, tomamos el plano, lo seguimos al pie de la letra y obtenemos la casa. En el plano estarán las medidas, especificaciones y demás características de la casa, junto con cosas que la casa puede hacer, como, por ejemplo: disipar calor en verano o, si la pensamos como un proyecto sustentable, puede tener entre sus acciones recolectar luz solar para generar energía. Con las clases ocurre lo mismo: **las clases definen los atributos o características que ese modelo tendrá y también definen las operaciones que llevarán a cabo el comportamiento**. A algunos de esos métodos se los conoce como constructores y se usan específicamente para construir nuevos objetos a partir de la clase. Tal como una casa, ¿verdad?

Los objetos

Las cosas que nos rodean, los objetos del mundo real, comparten dos particularidades indiscutibles: posee valores medibles y realizan acciones. Si obtuvimos objetos a partir de las clases, entonces las características de un objeto serán las que la clase definió.

A diferencia de las clases, los objetos no son abstractos, sino que son algo bien concreto. Entonces, no solo tienen los atributos que el modelo dicta, sino que **los objetos tienen valores en esos atributos. El conjunto de valores de un objeto es lo que se conoce como "estado interno"**. Por ejemplo, dijimos que la casa tenía ciertas características, una de ellas puede ser la temperatura interior. Como queremos algo sustentable, debería mantenerse entre cierto rango. Entonces, una casa igual a la del plano, ubicada en el mismo barrio, pero con orientación NORTE-SUR puede mantener una temperatura de 18 grados. Otra casa, obtenida a partir del mismo plano, pero con orientación ESTE-OESTE, por cómo llega el sol, puede mantener una temperatura de 20 grados. Entonces, los dos objetos casa están teniendo diferentes valores en su atributo "temperatura interior".

En la programación orientada a objetos, un objeto es una pieza de software que relaciona estado interno y comportamiento. La casa, por ejemplo, tiene un estado (orientación, ubicación, color exterior, temperatura interior) y cierto comportamiento (según lo que vimos, un comportamiento posible sería enfriarse y otro, calentarse). **Se dice que un objeto "almacena su estado" en los valores de sus atributos y con sus operaciones "expone su comportamiento" mediante los llamados métodos**. El comportamiento "expuesto" por el objeto se denomina **"interfaz" o "interfaz pública"**. Se llama así porque son las operaciones que otros objetos puede utilizar o ejecutar.

© Universidad de Palermo. Prohibida la reproducción total o parcial de imágenes y textos.

Supongamos que este ejemplo es parte de una simulación: un objeto “sol” podría calentar a un objeto “casa” dado que la casa expone un método “calentar”.

Objetos e instancias

Por definición, se dice que “un objeto es una instancia de una clase”. Esto quiere decir que cada vez que construimos un objeto, tenemos una **instancia nueva**. Es decir, hay una diferencia entre un objeto y una instancia. Sabiendo que por cada objeto creamos una nueva instancia, entonces cada instancia será única e irrepetible. Cada instancia que obtenemos da lugar a un objeto y, como tal, tiene valores en sus atributos. Si puedo obtener dos casas blancas, con orientación norte-sur ubicadas en el mismo barrio, puedo decir que tengo dos casas iguales. Ahora bien, aun cuando podemos tener objetos iguales, es decir, con iguales valores en sus atributos e igual comportamiento, esto no significa que se trate de la misma instancia.

Esto tiene un correlato con la realidad: si dos objetos fueran exactamente lo mismo: ocuparían incluso el mismo espacio físico, si fuera exactamente, estrictamente la misma casa estaría construida una en el mismo lugar de la otra, y eso es imposible. Esto equivale también en el mundo del software y, en este caso, más aún, lo tiene en hardware: si dos objetos fueran exactamente el mismo objeto, estarían en la misma posición de memoria (es como como querer poner una casa donde ya está otra casa) y sabemos que no se pueden almacenar dos valores diferentes en la misma posición de memoria en el mismo momento.

Las instancias introducen el concepto de **“identidad”**. Puede haber objetos iguales, pero nunca podrán ser “idénticos”. **Las instancias son únicas e irrepetibles**. Con cada objeto que se construye hay una nueva instancia, no habrá otra igual y, por lo tanto, nunca será igual a otra instancia que ya haya existido. Por su lado, puedo tener objetos iguales: dos casas blancas, con orientación norte-sur, cuya temperatura interior actual sea de 20 grados y estén en el mismo barrio. Sin embargo, no son la misma casa. Si observamos detenidamente, siempre encontraremos un atributo que distinga una casa de la otra: está claro que no podremos tener la misma casa con el mismo valor en el atributo dirección. Aun así, yo puedo tenerlas, ¿rompe eso el concepto de identidad? No, porque se puede tratar de un error, simplemente el usuario cargó la misma dirección dos veces: el concepto de identidad vs. igualdad de los objetos se da en el momento de la ejecución: nuevamente, hay objetos con valores iguales, pero no significa que sean el mismo objeto. Es decir, dos objetos idénticos tendrán iguales valores en sus atributos, pero dos objetos iguales no

© Universidad de Palermo. Prohibida la reproducción total o parcial de imágenes y textos.

necesariamente serán idénticos. **Si dos objetos son idénticos, entonces son el mismo objeto, porque estamos hablando de la misma instancia**.

Ahora, trata de imaginarte dos objetos idénticos pero que no sean iguales. A la vez, trata de imaginar dos objetos iguales pero que no sean idénticos. Te invitamos a hacer el ejercicio para identificar más fácilmente objetos e instancias en la realidad y en el código.

© Universidad de Palermo. Prohibida la reproducción total o parcial de imágenes y textos.