Capítulo 2
El diseño orientado a objetos

1. Enfoque procedural y de descomposición funcional

Antes de enumerar los conceptos básicos de la POO, vamos a revisar el enfoque procedural gracias a un ejemplo concreto de organización de código.

En este libro encontraremos habitualmente los términos «código» y «codificación»; no se trata ni de alarmas ni de funciones de cifrado de contraseña. El código o código fuente es el nombre que se da al contenido que el desarrollador entra durante todo el día en su editor de texto favorito, para posteriormente convertirse (o incluso compilarse) en un flujo de instrucciones ejecutables por el ordenador.

La programación procedural es un paradigma de programación, que considera los diferentes actores de un sistema como objetos prácticamente pasivos que un procedimiento central utilizará para una función determinada.

Tomemos como ejemplo la distribución de agua corriente en nuestras casas e intentemos modelizar este principio en una aplicación muy sencilla. El análisis procedural (igual que el análisis orientado a objetos) muestra una lista de objetos como los siguientes:

– el grifo del fregadero;
– el depósito de agua;
– un detector del nivel de agua con un contador en el depósito;
– la bomba de alimentación que lleva el agua al río.

El código del programa «procedural» consistiría en crear un conjunto de variables que represente a los argumentos de cada componente y, a continuación, escribir un bucle de operación de la gestión central, probando los valores leídos y actuando en función del resultado de las pruebas. Observe que por un lado hay variables y por otro, acciones.

2. La transición hacia el enfoque orientado a objetos

La POO es un paradigma de programación que considera los diferentes actores de un sistema como objetos activos y relacionados. El enfoque orientado a objetos normalmente se parece mucho a la realidad.

En nuestro ejemplo:

– El usuario abre el grifo.
– El grifo libera la presión y el agua fluye desde el depósito hasta el fregadero.
– Como nuestro usuario no es el único en consumir, el detector/flotador del depósito llega a un nivel que pone en marcha la bomba de alimentación del río.
– El usuario cierra el grifo.
– Alimentado por la bomba, el depósito de agua continúa llenándose hasta que el detector/flotador llegue al nivel suficiente que detendrá la bomba.
– Parada de la bomba.

En este enfoque, puede comprobar que los objetos «interactúan»; no existe operación central que defina dinámicamente el funcionamiento de los objetos. Ha habido un análisis funcional que ha conducido a la creación de los diferentes objetos, su realización y su puesta en relación.

El código del programa «objeto» va a seguir esta realidad, proponiendo tantos objetos como se han descrito con anterioridad, pero definiendo entre estos objetos los métodos de intercambio adecuados, que conducirán al funcionamiento esperado.

Observación

Los conceptos objetos son muy próximos a la realidad.

3. Las características de la POO

3.1 El objeto, la clase y la referencia

3.1.1 El objeto

El objeto es el elemento básico de la POO. El objeto es la unión de:

– una lista de variables de estados;

– una lista de comportamientos;

– una identificación.

Las variables de estado cambian a lo largo del ciclo de vida del objeto. Tomemos el caso de un reproductor de música digital. Cuando se compra el aparato, los estados de este objeto podrán ser:

– memoria libre = **100 %**;

– tasa de carga de la batería = **baja**;

– aspecto exterior = **nuevo**.

A medida que se va utilizando, sus estados se modifican. Rápidamente la memoria libre caerá, la tasa de carga de la batería variará y el aspecto exterior va a cambiar en función del cuidado que ponga el usuario durante su utilización.

Los comportamientos del objeto definen lo que se puede hacer con él:

– reproducir la música;
– ir a la pista siguiente;
– ir a la pista anterior;
– subir el sonido;
– etc.

Una parte de los comportamientos del objeto es accesible desde el exterior de este objeto: Botón Play, Botón Stop... y otra parte solo es accesible internamente: lectura de la tarjeta de memoria, decodificación de la música a partir del archivo. Se habla de «encapsulación» para definir el límite entre los comportamientos accesibles desde el exterior y los comportamientos internos.

La identificación de un objeto es una información, separada de la lista de estados, que permite diferenciar este objeto particular del resto (es decir, de otros objetos que son del mismo tipo). La identificación puede ser un número de referencia o una cadena de caracteres única, construida durante la creación del objeto; también puede ser la dirección de memoria donde se almacena el objeto. En realidad, su forma depende totalmente de la problemática asociada.

El objeto programado se puede adjuntar a una entidad real, como para nuestro reproductor digital, pero también se puede adjuntar a una entidad totalmente virtual, como una cuenta de cliente, la entrada de un directorio telefónico, etc. La finalidad del objeto informático es administrar la entidad física o emularla.

Incluso si este no forma parte de su naturaleza básica, el objeto se puede hacer persistente, es decir, que sus estados se pueden guardar en un soporte de memoria de información, de manera permanente. A partir de este almacenamiento, el objeto se podrá reconstruir con sus estados cuando el sistema desee. El soporte típico capaz de tal hazaña es la base de datos.

3.1.2 La clase

La clase es el «molde» a partir del que el objeto se va a crear en memoria. La clase describe los estados y los comportamientos comunes de un mismo tipo. Los valores de estos estados estarán contenidos en los objetos de la clase.

Todas las cuentas corrientes de un mismo banco contienen los mismos argumentos (detalles del titular, saldo, etc.) y todas tienen las mismas funciones (disponer de una cantidad a crédito, a débito, recordar la operación bancaria si es necesario, etc.). Estas definiciones deben estar contenidas en una clase y, cada vez que un cliente abre una nueva cuenta, esta clase servirá de modelo durante la creación del nuevo objeto cuenta.

La pantalla de reproductores de música digital ofrece un mismo modelo en diferentes colores, con tamaños de memoria diferentes y modulables, etc. Cada dispositivo de la pantalla es un objeto que ha sido fabricado a partir de la información de una única clase. Durante la construcción, los atributos del aparato se seleccionan en función de criterios estéticos y comerciales decididos por los responsables de producto, teniendo en cuenta las tendencias de compra de los clientes.

Una clase puede contener muchos atributos. Pueden ser de tipo primitivo –enteros, caracteres, etc.–, así como de tipos más complejos. En efecto, una clase puede contener una o varias clases de otros tipos. Hablamos entonces de composición, incluso de «acoplamiento fuerte». En este caso, cuando se elimina la clase principal, ello implica evidentemente la destrucción de las clases que contiene. Por ejemplo, si una clase hostal contiene una lista de habitaciones, la eliminación del hostal implica la destrucción de sus habitaciones.

Una clase puede hacer «referencia» a otra clase; en este caso, el acoplamiento se llama «débil» y los objetos pueden vivir independientemente. Por ejemplo, su PC está relacionado con su impresora. Si su PC pasa a mejor vida, su impresora funcionará con su nuevo PC. Se habla en este caso de asociación.

Además de sus atributos, la clase contiene también una serie de «comportamientos», es decir, una serie de métodos con sus firmas y sus implementaciones adjuntas. Estos métodos pertenecen a los objetos y se utilizan tal cual.

Declaramos la clase y su contenido en un mismo archivo fuente, gracias a una sintaxis que estudiaremos en detalle. Los desarrolladores C++ apreciarán el hecho de que ya no exista, por un lado, una parte de definiciones del contenido de la clase y, por otro, una parte de implementaciones. En efecto, en Java (como sucede en C#), el archivo de programa (de extensión .java) contiene las definiciones de todos los estados, eventualmente con un valor «de inicio» y las definiciones e implementaciones de todos los comportamientos. Al contrario de lo sucede con C#, una clase Java se debe definir en un único archivo fuente.

3.1.3 La referencia

Los objetos se construyen a partir de la clase, por medio de un proceso llamado instanciación y, por lo tanto, cualquier objeto es una instancia de una clase. Cada instancia empieza en una ubicación de memoria única. La dirección de esta ubicación de memoria, conocida con el nombre de puntero por los desarrolladores C y C++, se convierte en una referencia para los desarrolladores Java y C#.

Cuando el desarrollador necesita un objeto durante una operación, debe realizar dos acciones:

– declarar y asignar un nombre a una variable del tipo de la clase que se ha de utilizar;
– instanciar el objeto y guardar su referencia en esta variable.

Una vez que se realiza esta instanciación, el programa accederá a las propiedades y métodos del objeto, utilizando la variable que contiene su referencia. Cada instancia es única. Por el contrario, varias variables pueden «apuntar» a una misma instancia. Cuando ninguna variable apunta a una instancia dada, el *garbage collector* guarda esta instancia como instancia para eliminar.

3.2 La encapsulación

La encapsulación consiste en crear una especie de caja negra, que contiene de manera interna un mecanismo protegido y, de manera externa, un conjunto de comandos que van a permitir manipularla. Este juego de comandos se hace de tal manera que será imposible alterar el mecanismo protegido en caso de una utilización incorrecta. La caja negra será tan opaca que resultará imposible para el usuario intervenir directamente sobre el mecanismo.