5.A. Programación orientada a objetos.

Sitio: ALONSO DE AVELLANEDA
Curso: Entornos de desarrollo
Libro: 5.A. Programación orientada a objetos.
Imprimido por: Iván Jiménez Utiel
Día: martes, 17 de marzo de 2020, 01:12

Tabla de contenidos

1. Introducción.

Crear software es un proceso cuyo objetivo es dar solución a problemas utilizando una herramienta informática. Como en cualquier otra disciplina en la que se obtenga un producto final de cierta complejidad, es preciso realizar un análisis e identificar los resultados que pretendemos conseguir antes de iniciar el desarrollo.

El enfoque estructurado.

Sin embargo, cómo se hace es algo que ha ido evolucionando con el tiempo, en un principio se tomaba el problema de partida y se iba sometiendo a un proceso de división en sub-problemas más pequeños reiteradas veces, hasta que se llegaba a problemas elementales que se podían resolver utilizando una función. Luego las funciones se hilaban y entretejían hasta formar una solución global al problema de partida. Era pues, un proceso centrado en los procedimientos, se codificaban mediante funciones que actuaban sobre estructuras de datos, por eso a este tipo de programación se le llama programación estructurada.

Enfoque orientado a objetos.

La orientación a objetos ha roto con esta forma de hacer las cosas. Con este nuevo **paradigma** el proceso se centra en simular los elementos de la realidad asociada al problema de la forma más cercana posible. La abstracción que permite representar estos elementos se denomina **objeto**, y tiene las siguientes características:

- Está formado por un conjunto de **atributos**, que son los datos que le caracterizan y
- Un conjunto de **operaciones** que definen su comportamiento. Las operaciones asociadas a un objeto actúan sobre sus atributos para modificar su **estado**. Cuando se indica a un objeto que ejecute una operación determinada se dice que se le pasa un **mensaje**.

Las aplicaciones orientadas a objetos están formadas por un conjunto de objetos que interaccionan enviándose mensajes para producir resultados. Los objetos similares se abstraen en **clases**, se dice que un objeto es una instancia de una clase.

Cuando se ejecuta un programa orientado a objetos ocurren tres sucesos:

- Primero, los objetos se crean a medida que se necesitan.
- Segundo, los mensajes se mueven de un objeto a otro (o del usuario a un objeto) a medida que el programa procesa información o responde a la entrada del usuario.
- Tercero, cuando los objetos ya no se necesitan, se borran y se libera la memoria.

2. Conceptos de la orientación a objetos.

Como hemos visto, la orientación a objetos trata de resolver un problema simulando los elementos que participan en su resolución y basa su desarrollo en los siguientes conceptos:

- **Abstracción.** Permite capturar las características y comportamientos similares de un conjunto de objetos con el objetivo de darles una descripción formal. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a definir un conjunto de clases que permitan modelar la realidad, o el problema que se quiere atacar.
- **Encapsulación**. Organiza los datos y métodos de una clase, evitando el acceso a datos por cualquier otro medio distinto a los definidos. El estado de los objetos sólo debería poder ser modificado desde métodos de la propia clase.
- **Modularidad**. Propiedad que permite subdividir una aplicación en partes más pequeñas (llamadas módulos), cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes. En orientación a objetos es algo consustancial, ya que los objetos se pueden considerar los módulos más básicos del sistema.
- **Principio de ocultación.** La implementación de una clase sólo es conocida por los responsables de su desarrollo. Gracias a la ocultación, ésta podrá ser modificada para mejorar su algoritmo de implementación sin tener repercusión en el resto del programa.
- **Polimorfismo.** Consiste en reunir bajo el mismo nombre comportamientos diferentes. El modo en el que actuará un objeto en cada momento depende del mensaje que se le envíe al solicitar la ejecución de sus métodos.
- **Herencia**. Relación que se establece entre objetos en los que unos utilizan las propiedades y comportamientos de otros formando una jerarquía. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.
- **Recolección de basura.** Técnica por la cual el programa se encarga de destruir automáticamente los objetos que hayan quedado sin referenciar, y por tanto desvincular su memoria asociada. Característica disponible en algunos lenguajes orientados a objetos, no en todos.

3. Ventajas de la orientación a objetos.

El **paradigma orientado a objetos** tiene las siguientes **ventajas** con respecto a otros:

- Permite desarrollar software en mucho menos tiempo, con menos coste. Al ser completamente modular facilita la creación de **código reusable**.
- Se consigue **aumentar la calidad de los sistemas**.
- El software orientado a objetos es **más fácil de modificar y mantener** porque se basa en criterios de modularidad y encapsulación, en el que el sistema se descompone en objetos con unas responsabilidades claramente especificadas e independientes del resto.
- La tecnología de objetos facilita la adaptación al entorno y el cambio haciendo **aplicaciones escalables**.

4. Clases, atributos, métodos.

Los objetos de un sistema se abstraen, en función de sus características comunes, en **clases**. Una clase está formada por un conjunto de procedimientos y datos que resumen características similares de un conjunto de objetos. **La clase tiene dos propósitos: definir abstracciones y favorecer la modularidad**.

Una clase se describe por su **nombre** y sus **miembros**:

- **Nombre**. Identifica cada clase en el programa.
- **Atributos.** Características asociadas a una clase. Pueden verse como una relación entre una clase y todos los posibles valores que puede tomar cada atributo. El conjunto de valores de los atributos de un objeto definen su **estado**, permitiendo diferenciar unos de otros. Se definen por su nombre y su tipo y puede ser simples o compuestos (objetos de otra clase).
- **Protocolo.** Operaciones que manipulan el estado de los objetos. Los **métodos** determinan como actúan los objetos cuando reciben un **mensaje**, es decir, cuando se requiere que el objeto realice una acción descrita en un método se le envía un mensaje. El conjunto de mensajes a los cuales puede responder un objeto es conocido como protocolo del objeto.

Por ejemplo, si consideramos un objeto *icono* en una aplicación gráfica; tendrá como atributos el tamaño, o la imagen que muestra; y su protocolo puede constar de mensajes producidos al pulsar el botón sobre él.

Se denomina **miembros** al conjunto de atributos y métodos de una clase .

5. Visibilidad.

En el proceso de diseño e implementación de las clases hay que diferenciar los interfaces de acceso de su protocolo y los detalles de implementación. Entendemos estos términos como:

- **Interfaz**: captura la visión externa de una clase para el resto de participantes en el programa, el comportamiento que tendrán los objetos de esa clase al recibir eventos y los parámetros que tiene accesibles.
- **Implementación**: responde al desarrollo de los algoritmos que hacen que los objetos tengan un determinado comportamiento, se trata de aspectos intenos de la clase, nada relevantes para otros objetos del programa.

Existen distintos niveles de ocultación que se implementan en lo que se denomina **visibilidad**. Es una característica que define el **tipo de acceso que se permite a atributos y métodos** y que podemos establecer como:

- **Público**: se pueden acceder desde cualquier clase y cualquier parte del programa.
- **Privado**: sólo se pueden acceder desde operaciones de la clase.
- **Protegido**: sólo se pueden acceder desde operaciones de la clase o de clases derivadas en cualquier nivel.

Como norma general a la hora de definir la visibilidad tendremos en cuenta que:

- El estado debe ser privado. Los atributos de una clase se deben modificar mediante métodos de la clase creados a tal efecto.
- Las operaciones que definen la funcionalidad de la clase deben ser públicas.
- Las operaciones que ayudan a implementar parte de la funcionalidad deben ser privadas (si no se utilizan desde clases derivadas) o protegidas (si se utilizan desde clases derivadas).

6. Objetos. Instanciación.

Cada vez que se construye un **objeto** en un programa informático a partir de una clase se crea lo que se conoce como **instancia** de esa clase (objeto). Cada instancia puede realizar un trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema.

Un objeto se define por:

- **Su estado**: definido como el conjunto de valores concretos que tienen sus parámetros.
- **Su comportamiento**: definido por los métodos públicos de su clase.
- **Su tiempo de vida**: intervalo de tiempo a lo largo del programa en el que el objeto existe. Comienza con su creación a través del mecanismo de instanciación y finaliza cuando el objeto se destruye.

La **encapsulación** y el **ocultamiento** aseguran que los datos de un objeto están ocultos, con lo que no se pueden modificar accidentalmente por funciones externas al objeto.

Existe un caso particular de clase, llamada **clase abstracta,** que por sus características, no puede ser instanciada. Se suelen usar para definir métodos genéricos relacionados con el sistema que no serán traducidos a objetos concretos, o para definir las interfaces de métodos, cuya implementación se postpone a futuras clases derivadas.