3. Programación orientada a objetos.

Desarrollo de Aplicaciones Multiplataforma. Programación.

Contenido

1. Objetos.

- Entender que es un objeto es la clave para entender cualquier lenguaje orientado a objetos.
- Un objeto del mundo real es cualquier cosa que vemos a nuestro alrededor. Digamos que para leer este tema lo hacemos a través del monitor y un ordenador, ambos son objetos, al igual que nuestro teléfono, un árbol o un coche.
- Analicemos un poco más a un objeto del mundo real, como el ordenador. No necesitamos ser expertos en hardware para saber que un ordenador está compuesto internamente por varios componentes: la placa, el chip del procesador, un disco duro, una tarjeta gráfica, y otras partes más. El trabajo en conjunto de todos estos componentes hace trabajar a un ordenador.
- Internamente, cada uno de estos componentes puede ser sumamente complicado, pero nosotros no necesitamos saber cómo trabajan cada uno de estos componentes. Cada componente es una unidad autónoma, y todo lo que necesitamos saber es cómo interactúan entre sí, saber por ejemplo si el procesador y las memorias son compatibles con la placa, o conocer dónde se coloca la tarjeta gráfica. Cuando conocemos como interaccionan los componentes entre sí, podremos armar fácilmente el ordenador.

1. Objetos.

- ¿Qué tiene que ver esto con la programación? La programación orientada a objetos trabaja de esta manera. Todo el programa está construido en base a diferentes componentes (Objetos), cada uno tiene un rol específico en el programa y todos los componentes pueden comunicarse entre ellos de formas predefinidas.
- Podemos pensar que todo objeto del mundo real tiene 3 componentes: nombre, características y comportamiento.
- Por ejemplo, a lo que damos nombre de automóvil tiene características (marca, modelo, color, velocidad máxima, etc.) y comportamiento (frenar, acelerar, retroceder, llenar combustible, cambiar llantas, etc.).
- Los Objetos de Software, al igual que los objetos del mundo real, también tienen nombre, características y comportamientos. Un objeto de software tiene un identificador, mantiene sus características en una o más "variables" llamadas atributos, e implementa su comportamiento en funciones o subrutinas llamadas métodos.

1. Objetos.

- Imaginemos que tenemos un Ford Focus color azul que alcanza una velocidad de 260 km/h. Si pasamos ese objeto del mundo real al mundo del software, tendremos un objeto Automóvil con sus características predeterminadas:
- Marca = Ford; Modelo = Focus; Color = Azul; Velocidad Máxima = 260 km/h
- Cuando a las características del objeto le ponemos valores decimos que el objeto tiene estados. Las variables almacenan los estados de un objeto en un determinado momento.
- El método describe los mecanismos que se encargan de realizar sus tareas; y oculta al usuario las tareas complejas que realiza. Por ejemplo, el pedal del acelerador de un automóvil oculta al conductor los complejos mecanismos para hacer que el automóvil vaya más rápido. Esto permite que las personas con poco o nada de conocimiento acerca de cómo funcionan los motores puedan conducir un automóvil con facilidad.

1. Objetos.

Definición teórica: un objeto es una unidad de código compuesto de atributos y métodos relacionados.

En resumen:

- Un objeto en un sistema posee: una identidad, un estado y un comportamiento.
- La identidad es la propiedad que determina que cada objeto es único aunque tenga el mismo estado. No existen dos objetos iguales.
- El estado son los valores concretos que tiene un objeto en cada uno de sus atributos. Marca las condiciones de existencia del objeto dentro del programa. Lógicamente este estado puede cambiar. Un coche puede estar parado, en marcha, estropeado, funcionando, sin gasolina, etc. El estado lo marca el valor que tengan las propiedades del objeto.
- El comportamiento determina cómo responde el objeto ante peticiones de otros objetos. Por ejemplo un objeto conductor puede lanzar el mensaje arrancar a un coche. El comportamiento determina qué es lo que hará el objeto (es decir, qué ocurre cuando se arranca el coche). El comportamiento está relacionado por tanto con los métodos.

2. Clases.

- Las clases son las plantillas para hacer objetos. Una clase sirve para definir una serie de objetos con propiedades (atributos), comportamientos (operaciones o métodos), y semántica comunes. Hay que pensar en una clase como un molde.
- Supongamos que queremos conducir el automóvil del ejemplo anterior. Para hacer que aumente su velocidad, debe presionar el pedal del acelerador. ¿Qué debe ocurrir antes de que podamos hacer esto? Bueno, antes de poder conducir un automóvil, alguien tiene que diseñarlo.
- Los dibujos de ingeniería incluyen por ejemplo, el diseño del pedal del acelerador para que el automóvil aumente su velocidad. A partir de los dibujos de ingeniería se pueden fabricar muchos automóviles del mismo modelo. Los dibujos de ingeniería del automóvil serían un símil de CLASE, y los automóviles que sacamos a partir de los dibujos serían un símil de OBJETOS.
- Desafortunadamente, no se pueden conducir los dibujos de ingeniería de un automóvil. Antes de poder conducirlo, éste debe construirse a partir de los dibujos de ingeniería que lo describen.
- Así como no podemos conducir un dibujo de ingeniería de un automóvil, tampoco podemos “conducir” una clase. De la misma forma que alguien tiene que construir un automóvil a partir de sus dibujos de ingeniería para poder conducirlo, también debemos construir un objeto de una clase para poder hacer que un programa realice las tareas que la clase le describe cómo realizar.

2. Clases.

En resumen:

- Una clase representa al conjunto de objetos que comparten una estructura y un comportamiento comunes. Una clase es una combinación específica de atributos y métodos y puede considerarse un tipo de dato de cualquier tipo no primitivo.
- Así, una clase es una especie de plantilla o prototipo de objetos: define los atributos que componen ese tipo de objetos y los métodos que pueden emplearse para trabajar con esos objetos.
- Aunque, por otro lado, una clase también puede estar compuesta por métodos estáticos que no necesitan de objetos (como las clases construidas en los temas anteriores que contienen un método estático main).

2. Clases.

Antes de poder utilizar un objeto se debe definir la clase a la que pertenece, esa definición incluye:

- El nombre o identificador de clase. Debe empezar con letra mayúscula y seguirle letras minúsculas (si el nombre se compone de varias palabras, la inicial de cada palabra se deja en mayúscula).

- Sus atributos. Es decir, los datos miembros de esa clase. A los atributos también se les llama propiedades y campos. Los atributos son valores que poseerá cada objeto de la clase y por lo tanto marcarán el estado de los mismos. Se declaran con un tipo de dato correspondiente y un nombre identificador.

- Sus métodos. Las funciones miembro de la clase. Son las acciones (u operaciones) que puede realizar la clase. Sin duda es lo más complicado de programar y delimitar. Las clases se comunican entre sí invocando a métodos (a esto se le llama enviar mensajes).

2. Clases.

- El formato general para crear una clase en Java es:

```
[acceso] class NombreDeClase {
[acceso] [static] tipo atributo1;
[acceso] [static] tipo atributo2;
[acceso] [static] tipo atributo3;
...
[acceso] [static] tipo nombreMétodo1([listaDeArgumentos]) {
...código del método...
}
[acceso] [static] tipo nombreMétodo2([listaDeArgumentos]) {
...código del método...
}
...
}
```

2. Clases.

- La palabra opcional static sirve para hacer que el método o la propiedad a la que precede se pueda utilizar de manera genérica, los métodos o propiedades así definidos se llaman atributos de clase y métodos de clase respectivamente. Su uso se verá más adelante.

- En Java cada clase debe ocupar un archivo, que además debe de tener el mismo nombre. Hasta ahora identificábamos archivo con programa; bien, en realidad no es así. En un archivo se declara y define una clase, que tiene que tener exactamente el mismo nombre que el archivo.

- El método main, no se declara en cada archivo. La mayoría de clases no tienen métodos main, sólo disponen de él las clases que se pueden ejecutar (en cada proyecto real hay una).

3. Clases.

- Una instancia es un elemento tangible (ocupa memoria durante la ejecución del programa) generado a partir de una definición de clase. Todos los objetos empleados en un programa han de pertenecer a una clase determinada.
- Un objeto, como variable que es, tiene un espacio de memoria asignado con el fin de guardar los datos que contiene la clase a que pertenece.
- La instanciación de un objeto consiste, precisamente, en poder asignarle dicha memoria que le hace falta para poder guardar datos.

Para poder realizar esto, hay que llevar a cabo estos pasos:

- Declarar objeto: ClaseDeObjeto nombreObjeto;
- Crear objeto (instanciar): nombreObjeto = new ClaseDeObjeto();
- Para crear un objeto se tiene que haber definido previamente la clase a la que pertenece el objeto. Para ello primero necesitamos declarar una referencia al objeto lo cual se hace igual que al declarar una variable, es decir se indica la clase de objeto que es y se indica su nombre, es decir la sintaxis es: Clase objeto;

3. Clases.

Ejemplo:

- Automovil miAutomovil;

Esta instrucción declara miAutomovil, que será una referencia al tipo Automovil.

La creación del objeto se hace con el identificador new: objeto= new Clase(); Ejemplo:

- miAutomovil= new Automovil();
- En un solo paso: Clase objeto= new Clase();
- En un solo paso: Automovil miAutomovil =new Automovil();

4. Métodos.

- Un método es una llamada a una operación sobre un determinado objeto. Al realizar esta llamada (también se le llama enviar un mensaje), el control del programa pasa a ese método y lo mantendrá hasta que el método finalice.
- Hay métodos que devuelven un resultado (gracias a la palabra return). Cuando se define este tipo de método hay que indicar el tipo de dato que devuelve.
- Si el método no devuelve ningún resultado se define en su cabecera como tipo void.
- Dentro de cada método se pueden utilizar las estructuras de programación que hemos visto hasta ahora en el curso como: la estructura for, if, los operadores, las variables, etc.
- Los métodos pueden admitir en su definición una serie de valores. A estos valores se les denomina parámetros o argumentos. Los parámetros pueden tener un tipo básico (char, int, boolean, etc.) o bien ser un objeto.

4. Métodos.

- Cuando una clase ya tiene definidos sus métodos, es posible invocarlos utilizando los objetos definidos de esa clase. En esa invocación, se deben indicar los parámetros (o argumentos) que cada método requiere para poder realizar su labor.
- Usar métodos de una clase: nombreObjeto.nombre_método();
- Las llamadas a los métodos para realizar las distintas acciones se llevan a cabo separando los identificadores de la referencia y del método correspondiente con el operador punto.

Ejemplos:

- miAutomovil.getVelocidad();
- miAutomovil.acelera(50);

4. Métodos.

Para los métodos hay que tener en cuenta lo siguiente:

- 1. Sus especificadores de alcance o visibilidad. Si el alcance es private, el método sólo se podrá utilizar dentro de otro método en la misma clase; si es public podrá ser invocado desde cualquier clase; si es protected desde la propia clase y sus descendientes y si no lleva especificador, desde clases que estén en el mismo paquete.
- 2. El tipo de datos o de objeto que devuelve. Si el resultado del método es un número entero, o un booleano, o un String o un objeto de una clase determinada, etc. Si el método no devuelve valor alguno se debe indicar como tipo el valor void.
- 3. El identificador del método. El identificador de un método debe empezar por una letra en minúscula, y si se enlaza más de una palabra, se unirán por guiones bajos .
- 4. Los parámetros. Los métodos pueden necesitar datos para realizar su tarea. Dichos parámetros en realidad son una lista de variables u objetos y los tipos o clases de los mismos. La existencia de esas variables u objetos está ligada a la del propio método; es decir, cuando el método finaliza, los parámetros se eliminan.
- 5. El cuerpo del método. Es decir el código que permite al método realizar su tarea. Es lo más complicado. Dentro de ese código se pueden declarar variables, objetos y utilizar cualquier conjunto de instrucciones de Java, así como invocar a métodos de otras clases y objetos (si disponemos de visibilidad para ello). El valor resultado del método si lo hay se realiza mediante la instrucción return.

4. Métodos.

Ejemplo:

- Supongamos la clase Vehículo con atributos ruedas y velocidad, y los métodos: acelerar, frenar y obtenerVelocidad.
- Esto podemos representarlo gráficamente y en código, de la forma:

```
public class Vehiculo {
        public int ruedas;
        private double velocidad=0;
        public Vehiculo(){
        }
        public void acelerar(double cantidad) {
                velocidad += cantidad;
        }
        public void frenar(double cantidad) {
                velocidad -= cantidad;
        }
        public double obtenerVelocidad(){
                return velocidad;
        }
}
```

4. Métodos.

CLASE PRINCIPAL

```
package Ejemplo;

public class Principal {

    public static void main(String args[]){

        Vehiculo miCoche = new Vehiculo();

        miCoche.acelerar(12);

        miCoche.frenar(5);

        System.out.println(miCoche.obtenerVelocidad());

    }

}
```