Clases en Java

La clase es la unidad fundamental de programación en Java.

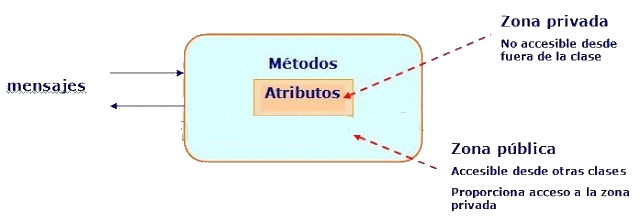
Un programa Java Orientado a Objetos está formado por un conjunto de clases. A partir de esas clases se crearán objetos que interactuarán entre ellos enviándose mensajes para resolver el problema.

Una claserepresenta al conjunto de objetos que comparten una estructura y un comportamiento comunes.

Puede considerarse como una *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.

Las clases incluyen por tanto atributos y métodos. Los atributos definen el estado de cada objeto de esa clase y los métodos su comportamiento.

Los atributos debemos considerarlos como la zona más interna, oculta a los usuarios del objeto. El acceso a esta zona se realizará a través de los métodos.



La sintaxis general para definir una clase en Java es:

[modificadorDeAcceso] **class** *NombreClase* [**extends** *NombreSuperClase*]

 [**implements** *NombreInterface1*, *NombreInterface2*, … ] {

       //atributos de la clase (0 ó más atributos)

       [modificadorDeAcceso]  *tipo* *nombreAtributo*;

       //métodos de la clase (0 ó más métodos)

       [modificadorDeAcceso] *tipoDevuelto nombreMetodo*([*lista parámetros*])

                                                             [throws *listaExcepciones*]{

                        // instrucciones del método

                        [return *valor*;]

       }

}

Todo lo que aparece entre corchetes es opcional, por lo tanto la definición mínima de una clase en Java es:

class *NombreClase*{

}

Como hemos visto antes, el concepto de clase incluye la idea de ocultación de datos, que básicamente consiste en que no se puede acceder a los datos directamente (zona privada), sino que hay que hacerlo a través de los métodos de la clase.

De esta forma se consiguen dos objetivos importantes:

         Que el usuario no tenga acceso directo a la estructura interna de la clase, para no poder generar código basado en la estructura de los datos.

         Si en un momento dado alteramos la estructura de la clase todo el código del usuario no tendrá que ser retocado.

El **modificador de acceso** se utiliza para definir el nivel de ocultación o visibilidad de los miembros de la clase (atributos y métodos) y de la propia clase.

Los modificadores de acceso **ordenados de menor a mayor visibilidad** son:

|  |  |  |
| --- | --- | --- |
| **MODIFICADOR DE ACCESO** | **EFECTO** | **APLICABLE A** |
| **private** | Restringe la visibilidad al **interior de la clase**. Un atributo o método definido como private solo puede ser usado en el interior de su propia clase. | Atributos  Métodos |
| **<Sin modificador>** | Cuando no se especifica un modificador, el elemento adquiere el *acceso por defecto o friendly*. También se le conoce como acceso de package (paquete). Solo puede ser usado por las **clases dentro de su mismo paquete**. | Clases  Atributos  Métodos |
| **protected** | Se emplea en la herencia. El elemento puede ser utilizado por **cualquier clase dentro de su paquete** **y por cualquier subclase** independientemente del paquete donde se encuentre. | Atributos  Métodos |
| **public** | Es el nivel máximo de visibilidad. El elemento es visible desde cualquier clase. | Clases  Atributos  Métodos |

**class**: palabra reservada para crear una clase en Java.

**NombreClase**: nombre de la clase que se define. Si la clase es pública, el nombre del archivo que la contiene debe coincidir con este nombre. Debe describir de forma apropiada la entidad que se quiere representar. Los nombres deben empezar por mayúscula y si está formado por varias palabras, la primera letra de cada palabra irá en mayúsculas.

**extends** *NombreSuperclase*: (opcional) extends es la palabra reservada para indicar la herencia en Java. NombreSuperClase es la clase de la que hereda esta clase. Si no aparece extendsla clase hereda directamente de una clase general del sistema llamada **Object.**

Object es la raíz de toda la jerarquía de clases en Java.

Es la superclase de las que heredan directa o indirectamente todas las clases Java.

Cuando una clase deriva de otra, se llama **superclase**a la clase base de la que deriva la nueva clase (clase derivada o subclase) **La clase derivada hereda todos los atributos y métodos de su superclase**.

**implements** *NombreInterface1*, *NombreInterface2*, ... : (opcional) implements es la palabra reservada para indicar que la clase implementa el o los interfaces que se indican separados por comas.

Un interface es un conjunto de constantes y declaraciones de métodos (lo que en C/C++ equivaldría al prototipo) no su implementación o cuerpo.

Si una clase implementa un interface está obligada a implementar todos los métodos de la interface. Veremos los interface más adelante.

-           En Java solo puede haber una clase pública por archivo de código fuente .java

-           El nombre de la clase pública debe coincidir con el nombre del archivo fuente. Por ejemplo, si el nombre de la clase pública es Persona el archivo será Persona.java

-           En una aplicación habrá una clase principal que será la que contenga el método main. Esta clase deberá haber sido declarada como pública

Junto al modificador de acceso pueden aparecer **otros modificadores**aplicables a clases, atributos y métodos:

|  |  |  |
| --- | --- | --- |
| **MODIFICADOR** | **EFECTO** | **APLICABLE A** |
| **abstract** | Aplicado a una clase, la declara como clase abstracta. No se pueden crear objetos de una clase abstracta. Solo pueden usarse como superclases.  Aplicado a un método, la definición del método se hace en las subclases. | Clases  Métodos |
| **final** | Aplicado a una clase significa que no se puede extender (heredar), es decir que no puede tener subclases.  Aplicado a un método significa que no puede ser sobrescrito en las subclases.  Aplicado a un atributo significa que contiene un valor constante que no se puede modificar | Clases  Atributos  Métodos |
| **static** | Aplicado a un atributo indica que es una variable de clase. Esta variable es única y compartida por todos los objetos de la clase.  Aplicado a un método indica que se puede invocar sin crear ningún objeto de su clase. | Atributos  Métodos |
| **volatile** | Un atributo volatile puede ser modificado por métodos no sincronizados en un entorno multihilo. | Atributos |
| **transient** | Un atributo transient no es parte del estado persistente de las instancias. | Atributos |
| **Sincronizad** | Métodos para entornos multihilo. | Métodos |

Como ejemplo, vamos a definir una clase Persona.

Clase Persona en Java:

public class Persona {

    private String nombre;

    private int edad;

    public void setNombre(String nom) {

        nombre = nom;

    }

    public String getNombre() {

        return nombre;

    }

    public void setEdad(int ed) {

        edad = ed;

    }

    public int getEdad() {

        return edad;

    }

}

La clase Persona es pública y tiene dos atributos, nombre y edad, y cuatro métodos.

Los métodos que aparecen en esta clase se conocen como métodos de acceso ó setters/getters.

Son métodos que solo sirven para asignar y obtener los valores de los atributos individualmente.

En cada clase es conveniente escribir un método set y otro get para cada atributo.

**MIEMBROS DE UNA CLASE: ATRIBUTOS Y MÉTODOS**

**ATRIBUTOS**

Una clase puede tener cero o más atributos.

Sirven para almacenar los datos de los objetos. En el ejemplo anterior almacenan el nombre y la edad de cada objeto Persona.

Se declaran generalmente al principio de la clase.

La declaración es similar a la declaración de una variable local en un método.

La declaración contiene un modificador de acceso de los vistos anteriormente: private, package, protected, public.

Pueden ser variables de tipo primitivo o referencias a objetos.

En la clase Persona se ha declarado edad de tipo primitivo y nombre String. Ambas private y por lo tanto solo accesibles desde los métodos de la propia clase.

private String nombre;

private int edad;

Los atributos toman el valor inicial por defecto:

-               0 para tipos numéricos

-               '\0' para el tipo char

-               null para String y resto de referencias a objetos.

También se les puede asignar un valor inicial en la declaración aunque lo normal es hacerlo en el constructor.

El valor de los atributos en cada momento determina el estado del objeto.

Podemos distinguir dos tipos de atributos o variables:

           **Atributos de instancia**: son todos los atributos no static. Cada objeto de la clase tiene sus propios valores para estas variables, es decir, cada objeto que se crea incluirá su propia copia de los atributos con sus propios valores.

           **Atributos de clase**: son los declarados **static**. También se llaman atributos estáticos. Un atributo de clase no es específico de cada objeto. Solo hay una copia del mismo y su valor es compartido por todos los objetos de la clase. Un atributo de clase existe y puede utilizarse aunque no existan objetos de la clase. Podemos considerarlo como una *variable global* a la que tienen acceso todos los objetos de la clase.

Para acceder a un atributo de clase se escibe:

NombreClase.Atributo;

Por ejemplo, en la clase Persona podemos añadir un atributo contadorPersonas que indique cuantos objetos de la clase se han creado. Sería un atributo de clase ya que no es un valor propio de cada persona:

static int contadorPersonas;

Cada vez que se crea una persona podemos incrementar su valor:

Personas.contadorPersonas++;

Si lo declaramos además como private:

private static int contadorPersonas

solo podremos acceder al atributo a través de un método.

**MÉTODOS**

Una clase puede contener cero o más métodos.

Definen el comportamiento de los objetos de la clase.

A través de los métodos se accede a los datos de la clase.

Desde el punto de vista de la POO **el conjunto de métodos de la clase se corresponden con el conjunto de mensajes a los que los objetos de esa clase pueden responder**.

Al conjunto de métodos de una clase se le llama **interfaz** de la clase.

Es conveniente que todas las clases implementen los métodos de acceso ó setters/getters para cada atributo.

Los métodos pueden clasificarse en:

           **Métodos de instancia**: Son todos los métodos no static. Operan sobre las variables de instancia de los objetos pero también tienen acceso a los atributos estáticos.

La sintaxis de llamada a un método de instancia es:

idObjeto.metodo(parametros); // Llamada típica a un método de instancia

Todas las instancias de una clase comparten la misma implementación para un método de instancia.

Dentro de un método de instancia, el identificador de una variable de instancia hace referencia al atributo de la instancia concreta que hace la llamada al método (suponiendo que el identificador del atributo no ha sido *ocultado*en el método).

           **Métodos de clase**: Son los métodos declarados como **static**. Tienen acceso solo a los atributos estáticos de la clase. No es necesario instanciar un objeto para poder utilizar un método estático.

Para acceder a un método de clase se escribe:

NombreClase.metodo;

Por ejemplo para la clase Fecha podemos escribir un método estático que incremente el contador de personas:

public static void incrementarContador(){

        contadorPersonas++;

}

Para invocar a este método:

Persona.incrementarContador();

La API de Java proporciona muchas clases con métodos estáticos, por ejemplo, los métodos de la clase Math: Math.sqrt(), Math.pow(), etc.