12. Tipos de atributos

Objetivos:

a) Profundizar en el concepto de atributo de una clase e indicar los tipos de atributos en Java
b) Interpretar el código fuente de una aplicación Java donde aparecen distintos tipos de atributos
c) Construir una aplicación Java sencilla, convenientemente especificada, que emplee clases con diferentes tipos de atributos.

Los atributos, también llamados datos o variables miembro son porciones de información que un objeto posee o conoce de sí mismo. Una clase puede tener cualquier número de atributos o no tener ninguno. Se declaran con un identificador y el tipo de dato correspondiente. Además los atributos y tienen asociado un modificador que define su visibilidad según se muestra en la Tabla 12.1.

Tabla 12.1. Modificadores y visibilidad correspondiente

Modificador	Visibilidad
`public`	Pública (+)
`protectec`	Protegida / en la herencia(#)
`private`	Privada(-)
`package`	De paquete (~)

En este capítulo se presentan tres tipos de atributos: las variables *de instancia*, las variables *de clase* y las variables *finales* o constantes.

12.1. Variables de instancia

Cuando se declara el atributo o variable miembro `euros` en la clase `Precio` de la siguiente forma:

```
public class Precio {
    // Declaracion de atributos o variables miembro
    public double euros;
    // Declaracion de metodos . . .
}
```

se está declarando el atributo `euros` como una *variable de instancia*. En consecuencia, cada vez que se crea una instancia de la clase `Precio`, se reserva espacio en memoria para una variable de instancia `euros`. Por ejemplo, el código:

```
// Creacion de dos instancias de la clase precio
Precio p = new Precio();
p.pone(56.8);
Precio q = new Precio();
q.pone(75.6);
```

genera dos instancias de la clase `Precio` como muestra la Figura 9.2. En este caso, cada una de las dos instancias, `p` y `q`, de la clase `Precio` tiene una variable de instancia `euros` propia. Las

respectivas llamadas al método `pone` para cada instancia (`p.pone(56.8)` y `q.pone(75.6)`), permiten asignar un valor a las variables de instancia correspondientes.

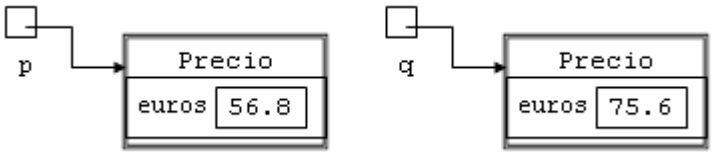

Figura 12.1. Representación gráfica del espacio de la memoria utilizado por cada instancia de la clase `Precio`

Otro ejemplo. En la declaración de la clase `Fecha` se incluyen tres atributos (`dia`, `mes` y `anho`) que son variables de instancia:

```
public class Fecha {
    // Declaracion de atributos o variables miembro
    private int dia;
    private int mes;
    private int anho;
    // Declaracion de metodos . . .
}
```

Con el siguiente código se crean dos instancias de la clase `Fecha`:

```
// Creacion de dos instancias de la clase Fecha
Fecha a = new Fecha();
Fecha b = new Fecha();
```

Cada una de las instancias de la clase `Fecha` reserva espacio en memoria para cada una de las variables de instancia como muestra la Figura 12.2.

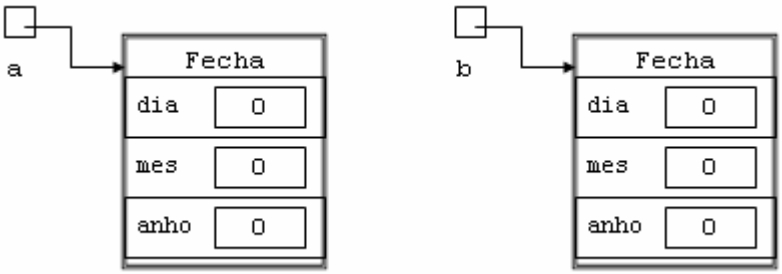

Figura 12.2. Representación gráfica del espacio de la memoria utilizado por cada instancia de la clase `Fecha`

Las variables de instancia pueden declararse como `public` o como `private` y pueden pertenecer a cualquiera de los tipos de datos primitivos de Java o bien, a otra clase existente en Java o declarada por el usuario. En principio, la única limitación para el número de variables de instancia que puede declarar una clase es el espacio libre disponible en la memoria del sistema que ejecute el programa.

12.2. Variables de clase (`static`)

Las *variables de clase* son atributos diferentes de las *variables de instancia*. Las variables de clase implican una sola zona de memoria reservada para todas las instancias de la clase, y no una copia por objeto, como sucede con las variables de instancia. Para diferenciarlas de éstas en el código fuente de Java, las variables de clase se distinguen con el modificador `static` en la

declaración del atributo correspondiente. Por defecto (si no se indica la palabra `static`), el atributo declarado se considera variable de instancia.

Durante la ejecución del programa, el sistema reserva un único espacio en memoria para cada variable *estáticas* o de clase independientemente del número de instancias creadas de una clase. Esta reserva se produce la primera vez que encuentra dicha clase en el código, de forma que todas las instancias pertenecientes a una clase comparten la misma variable de clase. A diferencias de las variables globales fuera de la POO, las variables de clase garantizan la encapsulación.

Las variables de clase sirven para almacenar características comunes (constantes) a todos los objetos (número de ruedas de una bicicleta) o para almacenar características que dependen de todos los objetos (número total de billetes de lotería). Por ejemplo, la clase `CuentaBancaria` tiene una variable de instancia, `saldo`, y una variable de clase, `totalCuentas`.

```
public class CuentaBancaria {
    // Atributos o variables miembro
    public double saldo;                    // Variable de instancia
    public static int totalCuentas=0;       // Variable de clase
    // Declaraciones de metodos
    ...
    }
```

La creación de varias instancias de la clase `CuentaBancaria` no conlleva la existencia de varias variables `totalCuentas`. Durante la ejecución de un programa que utilice la clase `CuentaBancaria` sólo existirá una variable de clase `totalCuentas`, independientemente del número de instancias de la clase `CuentaBancaria` que se generen (Figura 12.3). Es más, no es necesario siquiera que exista una instancia de la clase, para que lo haga la variable de clase. De hecho, se inicializan a `false`, cero o `null` (dependiendo de su tipo) antes de que se genere una instancia de la clase.

```
// Creacion de dos instancias de la clase CuentaBancaria
CuentaBancaria c1 = new CuentaBancaria();
CuentaBancaria c2 = new CuentaBancaria();
```

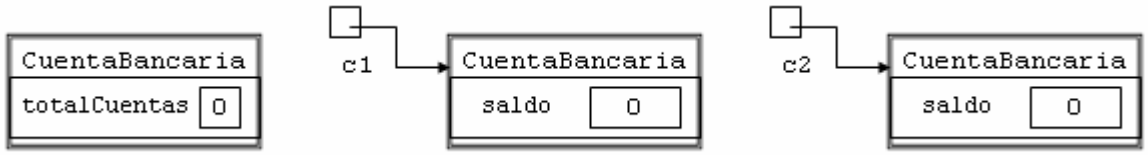

Figura 12.3. Representación gráfica del espacio de la memoria utilizado por la variable de clase `totalCuentas` y por cada instancia de la clase `CuentaBancaria`

Las variables de clase se emplean cuando sólo es necesaria **una copia** por clase que, además, esté accesible por todas las instancias de la clase a la que pertenece. En este caso, al ser un atributo `public` y `static`, puede accederse directamente a la variable de clase (`totalCuentas`) a través de una instancia (`c1` o `c2`) o de la clase en sí (`CuentaBancaria`). Un atributo estático puede ser accedido desde cualquier instancia de la clase, ya que es miembro de la propia clase.

```
public class PruebaCuentaBancaria {
    public static void main (String [] args) {
        CuentaBancaria c1 = new CuentaBancaria();
        c1.totalCuentas++;
```

```
        System.out.println("Total cuentas: " + c1.totalCuentas);
        CuentaBancaria c2 = new CuentaBancaria();
        c2.totalCuentas++;
        System.out.println("Total cuentas: " + c2.totalCuentas);
        // Acceso a traves de la clase:
        CuentaBancaria.totalCuentas++;
        System.out.println("Total cuentas: " + CuentaBancaria.totalCuentas);
        // Resto de sentencias . . .
    }
}
```

La ejecución del código anterior origina la siguiente salida por pantalla:

```
$>java PruebaCuentaBancaria↵

Total cuentas: 1
Total cuentas: 2
Total cuentas: 3
```

Para operar con variables de clase también podemos implementar métodos en la clase. Por ejemplo el siguiente método `inctotalCuentas` incrementa en una unidad el valor de la variable de clase `totalCuentas`.

```
public static void inctotalCuentas() {
    totalCuentas++;
    }
```

Al ser declarado el método `inctotalCuentas` como un método `public` y `static`, puede llamarse mediante cualquiera de las instancias (`c1` o `c2`) o de la clase en sí (`CuentaBancaria`).

Las variables de clase pueden declararse como `public` o como `private` y pueden pertenecer a cualquiera de los tipos de datos primitivos de Java o bien, a otra clase existente en Java o declarada por el usuario. En principio, la única limitación para el número de variables de clase que puede declarar una clase es el espacio libre disponible por el sistema que ejecute el programa.

12.3. Constantes o variables finales (`final`)

Una clase puede contener atributos de valor constante o *variables finales*. Este tipo de atributo se indica con la palabra reservada `final`. Las variables finales se suelen declarar además como variables de clase (`static final`) por razones de ahorro de memoria ya que, al no modificar su valor sólo suele ser necesaria una copia en memoria por clase (y no una por instancia). Por ejemplo, en la clase `Circulo`:

```
public class Circulo {
    // Atributos o variables miembro
    private static final double PI = 3.141592;       // Constante de clase
    private double radio;
    // Declaracion de metodos ...
    }
```

La palabra reservada `final` indica que el atributo debe comportarse como una constante, es decir, no puede ser modificada una vez declarada e inicializada. Por otro lado, se puede separar la declaración de la inicialización de la variable final, realizándose ésta más tarde. En este caso, el valor

asignado a la variable final puede hacerse en función de otros datos o de llamadas a métodos, con lo que el valor de la *constante* no tiene porqué ser el mismo para diferentes ejecuciones del programa.