13. Constructores

Objetivos:
a) Introducir el concepto de constructor de una clase en Java
b) Interpretar el código fuente de una aplicación Java donde aparecen declaraciones y llamadas a constructores
c) Construir una aplicación Java sencilla, convenientemente especificada, que emplee clases en los que se declaren explícitamente constructores

Aunque en un principio pueda parecer lo contrario, un *constructor* **no** es en realidad un método estrictamente hablando. Un constructor es un elemento de una clase cuyo identificador coincide con el de la clase correspondiente y que tiene por objetivo obligar a y controlar cómo se inicializa una instancia de una determinada clase, ya que el lenguaje Java no permite que las variables miembro de una nueva instancia queden sin inicializar. Además, a diferencia de los métodos, los constructores sólo se emplean cuando se quiere crear una nueva instancia.

13.1. Constructores

Por defecto toda clase tiene un constructor sin parámetros cuyo identificador coincide con el de la clase y que, al ejecutarse, inicializa el valor de cada atributo de la nueva instancia: los atributos de tipo primitivo se inicializan a `0` o `false`, mientras que los atributos de tipo *objeto* (referencia) se inicializan a `null`.

En el ejemplo de la clase `PruebaPrecio`, que utiliza una instancia de la clase `Precio`, la llamada al constructor se produce en la sentencia `p = new Precio();`. Mientras que la ejecución de `new` genera una nueva instancia y devuelve su dirección de memoria, la ejecución del constructor `Precio()` inicializa los valores de los atributos (Figura 13.1).

```
public class PruebaPrecio {
    public static void main (String [] args ) {
        Precio p;              // Crea una referencia de la clase Precio
        p = new Precio();      // Crea el objeto de la clase Precio y realiza
                               // una llamada al metodo constructor
    //  Resto del codigo ...
    }
}
```

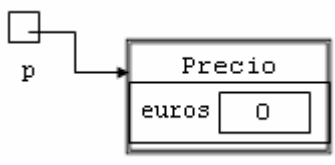

Figura 13.1. Resultado de la ejecución del constructor: inicialización de los atributos del nuevo objeto

13.2. Declaración de un constructor

La declaración de un constructor diferente del constructor por defecto, obliga a que se le asigne el mismo identificador que la clase y que no se indique de forma explícita un tipo de valor de retorno. La existencia o no de parámetros es opcional. Por otro lado, la *sobrecarga* permite que puedan declararse varios constructores (con el mismo identificador que el de la clase), siempre y

cuando tengan un tipo y/o número de parámetros distinto. Por ejemplo, para la clase `Fecha` se declaran dos constructores, el primero sin parámetros (por lo tanto se redefine el constructor por defecto) y el segundo con tres parámetros:

```
/**
 *  Declaracion de la clase Fecha
 *  @author A. Garcia-Beltran
 *  Ultima revision: noviembre, 2008
 */

public class Fecha {

   // Atributos o variables miembro
   private int dia;
   private int mes;
   private int anho;
  /**
   * Constructor 1
   * Asigna los valores 1, 1 y 2000 a los atributos
   * dia, mes y anho respectivamente
   */
    public Fecha() {
        dia = 1;
        mes = 1;
        anho = 2000;
    }
  /**
   * Constructor 2
   * @param ndia el dia del mes a almacenar
   * @param nmes el mes del anho a almacenar
   * @param nanho el anho a almacenar
   */
    public Fecha(int ndia, int nmes, int nanho) {
        dia = ndia;
        mes = nmes;
        anho = nanho;
    }
    public String toString() {
        return dia + "/" + mes + "/" + anho;
    }
}
```

La *sobrecarga* permite que puedan declararse varios constructores (con el mismo identificador que el de la clase), siempre y cuando tengan un tipo y/o número de parámetros distinto. Para probar el código anterior, se construye el siguiente programa:

```
/**
 *  Ejemplo de uso de la clase Fecha
 *  A. Garcia-Beltran - febrero, 2009
 */
public class PruebaFecha {
    public static void main (String [] args) {
        Fecha origen = new Fecha();
        Fecha actual = new Fecha(16,2,2009);
        System.out.println("Primera fecha: " + origen.toString());
        System.out.println("Segunda fecha: " + actual.toString());
    }
}
```

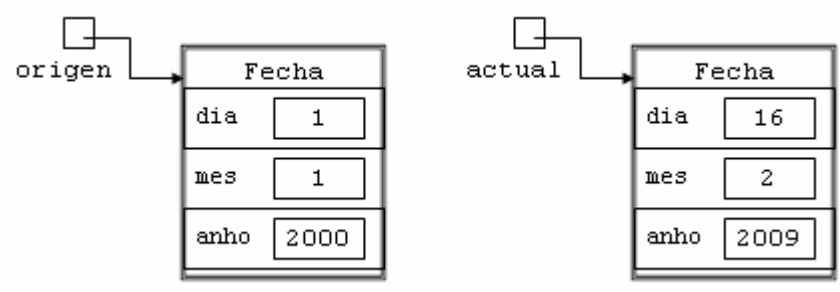

Figura 13.2. Resultado de la ejecución de los respectivos constructores para las nuevas instancias referenciadas por `origen` y `actual`

El código anterior puede compilarse y ejecutarse, mostrando la siguiente salida por pantalla:

```
$>javac PruebaFecha.java↵

Primera fecha: 1/1/2000
Segunda fecha: 16/2/2009
```

Nota: una vez construido un constructor ya no se puede emplear el constructor por defecto sin parámetros. Si se desea trabajar con él, es necesario declararlo explícitamente.

13.3. Más sobre la declaración y uso de varios constructores

Un constructor sólo puede ser llamado por otros constructores o por métodos de clase (`static`). En el siguiente código se muestra un ejemplo de cómo se declaran dos constructores `CuentaBancaria`: el primero no tiene parámetros y hace una llamada al segundo constructor, que tiene un parámetro numérico real.

```
/**
 *  Declaracion de la clase CuentaBancaria
 *  Ejemplo de declaracion de variables
 *  metodos estaticos y uso de this
 *  @author A. Garcia-Beltran
 *  Ultima revision: abril, 2005
 */
public class CuentaBancaria {
    // Atributos o variables miembro
    private double saldo;
    public static int totalCuentas=0;

    // Metodos
    public CuentaBancaria( ) {
        this(0.0);          // Llamada al constructor que tiene un parametro
    }
    public CuentaBancaria( double ingreso ) {
        saldo = ingreso;
        incCuentas();
    }
    public double saldo() {
        return saldo;
    }
    public static void incCuentas () {
        totalCuentas++;
    }
    public void transferencia( CuentaBancaria origen ) {
        saldo += origen.saldo;
        origen.saldo=0;
```

```
    }
}
```

La sentencia **`this`**`(0.0);` en el primer constructor realiza la llamada el segundo constructor. Ejemplo de programa que emplea la clase `CuentaBancaria`:

```
/**
 *  Ejemplo de uso de la clase CuentaBancaria
 *  @author A. Garcia-Beltran
 *  Ultima revision: abril, 2004
 */
public class PruebaCuentaBancaria {
    public static void main (String [] args) {
        System.out.println("Total cuentas: " + CuentaBancaria.totalCuentas);
        CuentaBancaria c1;
        c1 = new CuentaBancaria();
        System.out.println("Nueva cuenta con: " + c1.saldo() + " euros");
        System.out.println("Total cuentas: " + CuentaBancaria.totalCuentas);
        CuentaBancaria c2;
        c2 = new CuentaBancaria(20.0);
        System.out.println("Nueva cuenta con: " + c2.saldo() + " euros");
        System.out.println("Total cuentas: " + CuentaBancaria.totalCuentas);
    }
}
```

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

```
$>java PruebaCuentaBancaria↵

Total cuentas: 0
Nueva cuenta con: 0.0 euros
Total cuentas: 1
Nueva cuenta con: 20.0 euros
Total cuentas: 2
```