# 2. Estructura de un programa en Java

**Objetivos**:
a) Describir la estructura del código fuente de una aplicación Java
b) Presentar los conceptos de comentario y de identificador dentro del código fuente de un programa

Este capítulo tiene como objetivo presentar la estructura de un programa mediante un ejemplo sencillo y mostrar dos elementos típicos del código fuente: los comentarios y los identificadores. La estructura de un programa de Java es similar a la de un programa de C/C++. Por su diseño, permite a los programadores de cualquier otro lenguaje leer código en Java sin mucha dificultad. Java emplea siempre la Programación Orientada a Objetos por lo que todo el código se incluye dentro de las *clases*. Aunque ya se explicarán detenidamente más adelante, las clases son combinaciones de datos (constantes y variables) y rutinas (métodos).

## 2.1. La clase principal y el método `main`

Un programa puede construirse empleando varias *clases*. En el caso más simple se utilizará una única clase. Esta clase contiene el programa, rutina o método principal: `main()` y en éste se incluyen las sentencias del programa principal. Estas sentencias se separan entre sí por caracteres de punto y coma.

La estructura de un programa simple en Java es la siguiente:

```
public class ClasePrincipal {
   public static void main(String[] args) {
      sentencia_1;
      sentencia_2;
      // ...
      sentencia_N;
   }
}
```

Como primer ejemplo sencillo de programa escrito en Java se va a utilizar uno que muestra un mensaje por la pantalla del ordenador. Por ejemplo, el programa `Hola.java`:

```
/**
 *  La clase hola construye un programa que
 *  muestra un mensaje en pantalla
 */
public class Hola {
   public static void main(String[] args) {
      System.out.println("Hola, ");
      System.out.println("me llamo Angel");
      System.out.println("Hasta luego");
   }
}
```

Como se ha indicado anteriormente, en un programa de Java todo se organiza dentro de las *clases*. En el ejemplo anterior, `Hola` es el nombre de la clase principal y del archivo que contiene el código fuente. Todos los programas o aplicaciones independientes escritas en Java tienen un método

`main` o principal que, a su vez, contiene un conjunto de sentencias. En Java los conjuntos o bloques de sentencias se indican entre llaves `{ }`. En el caso anterior, el conjunto de sentencias se reduce a tres sentencias, que son llamadas a dos métodos predefinidos en Java (`print` y `println`) que permiten visualizar texto por el dispositivo de salida de datos por defecto (la pantalla).

Por el momento y hasta que se explique con detalle el concepto de *clase*, los ejemplos de programa que se utilizarán constarán de una sóla clase en la que se declara el método `main`. Este método es el punto de arranque de la ejecución de todo programa en Java.

## 2.2. Comentarios

Los comentarios se emplean para facilitar la tarea de los programadores humanos ya que no realizan ningún papel activo en la generación del código. Los comentarios que se pueden introducir en el código fuente de un programa de Java son del estilo de C y C++. Así, el compilador ignora todo lo que se incluya entre la secuencia de caracteres `//` y el final de la línea. Por ejemplo:

```
// Este es un comentario estilo C++, llega al final de la linea
```

La pareja de caracteres `/` hay que escribirla sin dejar ningún espacio en blanco entre ellos. El segundo tipo de comentario es el que se utiliza también en el lenguaje de programación C: el compilador también ignora todo lo que se incluya entre las secuencias de caracteres `/*` y `*/`. Por ejemplo:

```
/* En este otro comentario estilo C, el final
   lo indica la marca */
```

El comentario con `//` es más fácil de teclear si sólo ocupa una línea, pero si ocupa varias, entonces el más sencillo es el de `/* ... */`. Los comentarios pueden incluir cualquier carácter válido en Unicode y no pueden anidarse.

Estos dos formatos de comentario se emplean para los denominados *comentarios de implementación*. Los comentarios de implementación se utilizan en el programa fuente para resaltar código o para aclarar un desarrollo en particular.

Además, en Java existe un tercer tipo de comentario (*doc comments*) que facilita la generación de documentación en formato HTML al emplear algunas herramientas de documentación (por ejemplo, javadoc incluida en el Kit de Desarrollo de Java). Los comentarios para la documentación se emplean para describir la especificación del código, desde una perspectiva independiente cómo se ha implementado y ser leido por desarrolladores que no tengan necesariamente el código fuente a la vista. Debe evitarse emplear los comentarios para dar panorámicas del código y facilitar información que no facilite la propia lectura del código. Ejemplos de este tipo de comentarios:

```
/** Comentario estilo javadoc, se incluye
automaticamente en documentacion HTML */

/**
* Muchos programadores
* suelen incluir un asterisco
* al principio de cada linea del
* comentario para facilitar su lectura
*/
```

Los comentarios deberían contener sólo información relevante para la lectura y comprensión del programa. Todos los programas deben empezar por un comentario que describa el propósito del programa. La discusión de decisiones de diseño que no sean obvias (triviales) es apropiada, pero debe evitarse la duplicidad de información dada por el propio código. Es muy común que los comentarios redundantes se desactualicen. En general, se debe evitar incluir comentarios que puedan quedarse obsoletos conforme el código se desarrolle. Por ejemplo, dentro de un comentario **no** debería incluirse información acerca de cómo se ha construido un archivo fuente determinado o en que directorio se haya el código.

A veces la frecuencia de los comentarios refleja una peor calidad de código. Cuando se sienta la necesidad de añadir un comentario, debe considerarse la reescritura del código para hacerlo más claro.

**Convención para los comentarios al inicio de los programas**

Todos los archivos fuente deberían comenzar con un comentario estilo C que liste el nombre de la clase, información de la versión, fecha y el aviso de *copyright*:

```
/*
 * Nombre de la clase
 *
 * Información de la version
 *
 * Fecha
 *
 * Aviso de Copyright
 */
```

## 2.3. Identificadores

Los identificadores son nombres que se les asignan a variables, métodos, clases… en el código fuente de un programa. Los identificadores sólo existen en el código del programa fuente y no en el programa objeto (resultado de la compilación del programa fuente). Todo nuevo identificador que se emplee en un programa Java debe definirse previamente a su utilización. Las normas para la construcción de un identificador empleando el lenguaje de programación Java son las siguientes:

- Un identificador comienza por una letra, un carácter de subrayado (_) o un carácter de dólar ($). Aunque no se recomienda emplear el carácter $, ya que el compilador suele utilizarlos de forma interna para crear identificadores propios.
- Los siguientes caracteres pueden ser también dígitos. Pero no pueden emplearse espacios en blanco u otros caracteres como el signo de interrogación (?) o el signo del tanto por ciento (%).
- No hay límite máximo de caracteres.
- En los identificadores del código fuente de un programa en Java se distinguen las mayúsculas de las minúsculas. Por ejemplo, `casa`, `CASA` y `Casa` son tres identificadores diferentes.
- Pueden incluir caracteres Unicode, con lo que se pueden emplear secuencias de escape `/uxxxx` para representar estos caracteres.
- No puede emplearse el identificador de una variable o cualquier otro elemento del código fuente del programa para otro ya existente en el mismo bloque. Excepción: variable miembro y local con el mismo identificador.

- Existe una serie de **palabras reservadas** que no pueden emplearse como identificadores por el programador en el código fuente para otros usos. Por ejemplo, la palabra `double` se utiliza para definir un tipo de dato real y la palabra `for` se emplea para construir un tipo determinado de bucle. En la Tabla 2.1 se muestran las palabras reservadas en Java.

Tabla 2.1. Palabras reservadas en Java

| | | | | |
|---|---|---|---|---|
| `abstract` | `do` | `implements` | `protected` | `throw` |
| `boolean` | `double` | `import` | `public` | `throws` |
| `break` | `else` | `instanceof` | `rest` | `transient` |
| `byte` | `extends` | `int` | `return` | `true` |
| `case` | `false` | `interface` | `short` | `try` |
| `catch` | `final` | `long` | `static` | `void` |
| `char` | `finally` | `native` | `strictfp` | `volatile` |
| `class` | `float` | `new` | `super` | `while` |
| `const`* | `for` | `null` | `switch` | |
| `continue` | `goto`* | `package` | `synchronized` | |
| `default` | `if` | `private` | `this` | |

Algunos de estos identificadores reservados no tienen todavía uso en la implementación actual del lenguaje Java. En concreto, los identificadores marcados con un asterisco * no se utilizan actualmente. Por otro lado en la Tabla 2.2 se muestran los nombres de métodos reservados en Java cuyo significado y utilización se explicará más adelante cuando se menciones la clase predefinida `Object`.

Tabla 2.2. Identificadores de métodos reservados en Java

| | | | | |
|---|---|---|---|---|
| `clone` | `equals` | `finalize` | `getClass` | `hashCode` |
| `notify` | `notifyAll` | `toString` | `wait` | |

Aunque la forma de escribir los identificadores en Java no está *normalizada*, a continuación se dan algunas *recomendaciones* para la elección de estos identificadores:

- Es conveniente utilizar identificadores **autoexplicativos** para orientar al usuario o a cualquier otra persona que accede al código fuente de un programa sobre lo que representan, sin necesidad de utilizar los comentarios en el propio código fuente. Por ejemplo, para almacenar el valor del radio de una esfera podría declararse una variable con el identificador `radio`, `radioesfera` o `radiodeunaesfera`.
- Si el identificador se compone de más de una palabra, éstas se agrupan y después de la primera el resto comienza con una letra mayúscula, como por ejemplo: `esVisible` o `radioEsfera`.
- El signo de subrayado se acepta en cualquier identificador pero por convención sólo se emplea para separar palabras en identificadores en constantes (por convención las constantes se escriben completamente en mayúsculas y esto dificulta la separación de los nombres). Por ejemplo, para un valor constante que almacena un valor numérico que representa el número máximo de alumnos con los que trabaja el programa se podría declarar el identificador de la constante `NUMERO_MAXIMO_ALUMNOS`.

### Convenciones para la legibilidad del programa fuente

Aunque la definición exacta de la indentación (espacios vs tabuladores) no se especifica en el lenguaje Java, se recomienda emplear una secuencia de cuatro espacios como unidad de indentación.

Por otro lado se recomienda evitar las lineas de más de 80 caracteres en el código fuente ya que no se manejan correctamente por muchos terminales y herramientas de edición.