5. Sentencias selectivas o condicionales

Objetivos:

a) Describir el funcionamiento de las sentencias selectivas o condicionales (if-else y switch)
b) Interpretar el resultado de una secuencia de sentencias condicionales combinadas o no
c) Codificar una tarea sencilla convenientemente especificada, utilizando la secuencia y combinación de sentencias condicionales

5.1. Sentencias de control del flujo de un programa

Cuando se escribe un programa, se introduce la secuencia de sentencias dentro de un archivo. Sin sentencias de control del flujo, el intérprete ejecuta las sentencias conforme aparecen en el programa de principio a fin. Las sentencias de control de flujo se emplean en los programas para ejecutar sentencias condicionalmente, repetir un conjunto de sentencias o, en general, cambiar el flujo secuencial de ejecución. Las sentencias selectivas o condicionales se verán en este capítulo y las sentencias repetitivas en el siguiente.

5.2. Sentencia if-else

Es una bifurcación o sentencia condicional de una o dos ramas. La sentencia de control evalúa la condición lógica o booleana. Si esta condición es cierta entonces se ejecuta la sentencia o sentencias (1) que se encuentra a continuacion. En caso contrario, se ejecuta la sentencia (2) que sigue a `else` (si ésta existe). La sentencia puede constar opcionalmente de una o dos ramas con sus correspondientes sentencias.

Sintaxis:

```
if (expresionLogica) {
  sentencia_1;
  }
```

o bien (con dos ramas):

```
if (expresionLogica) {
      sentencia_1;
    }
   else {
      sentencia_2;
    }
```

La *`expresionLogica`* debe ir entre paréntesis. Las llaves sólo son obligatorias si las sentencias (1) ó (2) son compuestas (las llaves sirven para agrupar varias sentencias simples).

La parte `else` y la sentencia posterior entre llaves no son obligatorias. En este caso quedaría una sentencia selectiva con una rama (Figura 5.1).

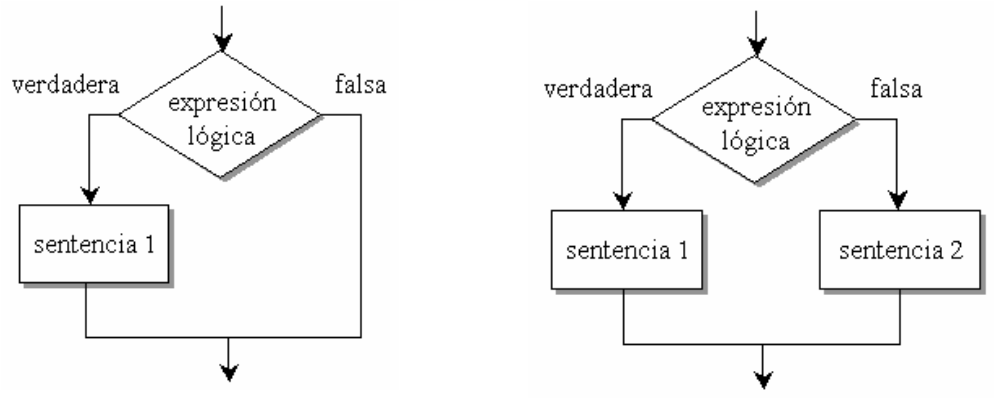

Figura 5.1. Flujograma de la sentencia if. Con una rama (a la izquierda) y con dos ramas (a la derecha)

Un ejemplo muy sencillo que muestra este tipo de sentencia es el siguiente:

```
// Codigo autoexplicativo
if (a>b) {
   System.out.println(”a es mayor que b");
   }
   else {
   System.out.println(”a no es mayor que b");
   }
```

Ejemplo de programa completo:

```
/**
 *  cuadrado: Ejemplo de sentencia if
 *  Muestra el cuadrado de un valor entero introducido como parametro
 *  A. Garcia-Beltran, 16 de marzo de 2004
 */

public class Cuadrado {
   public static void main (String [] args) {
   int valor;
   if (args.length == 0)
      System.out.println("Por favor, introduce un argumento entero");
   else {
      valor = Integer.parseInt(args[0]);
      System.out.println("El cuadrado es " + valor*valor);
      }
   System.out.println("Hasta pronto");
   }
}
```

Ejemplos de ejecución del programa anterior y salidas correspondientes por pantalla:

```
$>java Cuadrado↵
Por favor, introduce un argumento entero
Hasta pronto

$>java Cuadrado 4↵
El cuadrado es 16
Hasta pronto
```

Todo programa o aplicación independiente de Java debe declarar un método principal con la siguiente cabecera:

```
   public static void main (String [] args)
```

Esta declaración indica que al método `main` se le transfiere un vector de cadenas, `args`, (*array* de *strings*). Este vector contiene todos los parámetros o argumentos indicados en la línea de comandos al realizar la ejecución del intérprete de Java seguido del nombre de la clase a ejecutar. El primer elemento de este vector es `args[0]`. El tamaño del vector podría determinarse añadiendo `.length` a su identificador. Como el índice del primer elemento del vector es `0`, si el tamaño del vector es *n*, entonces el último elemento del vector tiene índice *n-1*. En el ejemplo anterior de ejecución del programa, `args[0]` vale `"4"`.

Las sentencias `if–else` pueden ir anidadas unas dentro de otras en el código fuente del programa. Por ejemplo:

```
/**
 *  esPar: Ejemplo de sentencias if anidadas
 *  Indica si el valor entero introducido como parametro es par o no
 *  A. Garcia-Beltran, 29 de octubre de 2004
 */

public class EsPar {
   public static void main (String [] args) {
      int valor;
      if (args.length == 0)
         System.out.println("Por favor, introduce un argumento entero");
      else {
         valor = Integer.parseInt(args[0]);
         if (valor % 2 == 0) {
            System.out.println("El valor " + valor + " es par");
         }
         else {
          System.out.println("El valor " + valor + " es impar");
         }
      }
   System.out.println("Hasta pronto");
   }
}
```

Ejemplos de ejecución del programa anterior y salidas correspondientes por pantalla:

```
$>java EsPar↵
Por favor, introduce un argumento entero
Hasta pronto

$>java EsPar 4↵
El valor 4 es par
Hasta pronto


$>java EsPar 5↵
El valor 5 es impar
Hasta pronto
```

5.3. Sentencia switch

Es una sentencia condicional multiramificada o de selección multiple: dependiendo del valor de una variable o expresion entera permite ejecutar una o varias sentencias de entre muchas. La expresión

puede ser de un tipo ordinal (de tipo entero `byte`, `short` ó `int` o de tipo carácter `char`) pero no puede ser de un tipo real o de un tipo cadena.

Sintaxis:

```
switch (expresion) {
    case valor_1: sentencias_1; break;
    case valor_2: sentencias_2; break;
    ...
    case valor_n: sentencias_n; break;
[default: sentencias_x;]
    }
```

Cada sentencia `case` contiene un único valor distinto del de las demás sentencias `case`. A continuación del valor se introduce la sentencia o sentencias que se ejecutan en el caso de que el valor indicado coincida con el de la variable o expresión selector. Las sentencias que siguen a cada uno de los valores no se engloban entre llaves, pero suelen ir seguidas de un `break`.

Si la expresión no coincide con ningún valor se ejecuta la sentencia que sigue a `default`, aunque esta parte (`default`) no es obligatoria.

Si no existe algún `break`, continua la ejecución de la siguiente opción hasta el siguiente `break` o hasta el final de la sentencia `switch`.

El flujograma de la sentencia `switch` se muestra en la Figura 5.2.

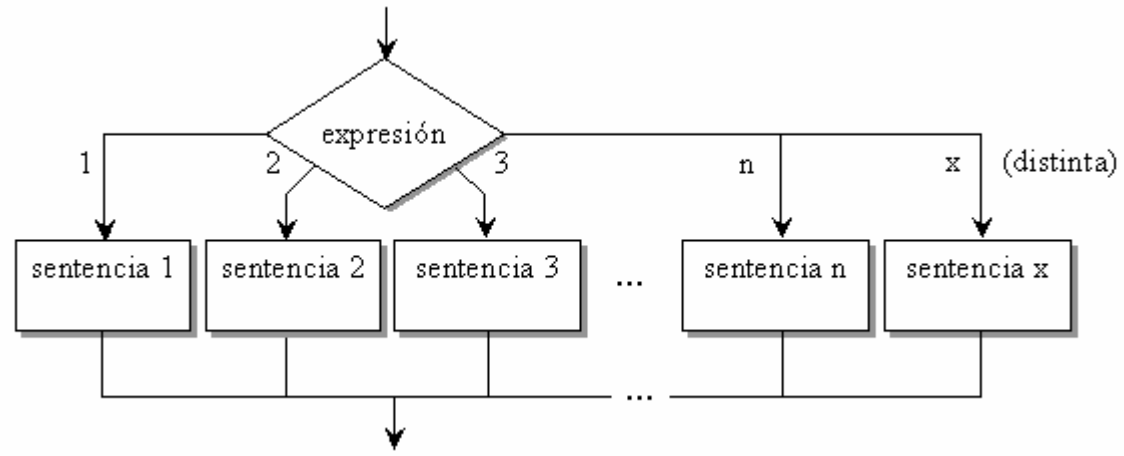

Figura 5.2. Flujograma de la sentencia switch

A continuación se muestra un ejemplo de programa que emplea la sentencia `switch` para visualizar en pantalla mensajes distintos, en función del primer caracter del primer argumento introducido en la línea de comandos de la ejecución:

```
/**
 *  Ejemplo de uso de la sentencia switch
 *  A. Garcia-Beltran - marzo, 2004
 */
public class Caracter {
  public static void main (String [] args) {
```

```
     char c;
     c=args[0].charAt(0);
     switch (c) {
       case 'a': System.out.println("Es la vocal a"); break;
       case 'e': System.out.println("Es la vocal e"); break;
       case 'i': System.out.println("Es la vocal i"); break;
       case 'o': System.out.println("Es la vocal o"); break;
       case 'u': System.out.println("Es la vocal u"); break;
       case 'A': System.out.println("Es la vocal A");
       case 'E':
       case 'I':
       case 'O':
       case 'U': System.out.println("Vocal mayuscula"); break;
       default: System.out.println("No es una vocal"); break;
     }
  System.out.println("Hasta la vista");
  }
}
```

Ejemplos de ejecución del programa anterior y salidas correspondientes por pantalla:

```
$>java Caracter a↵
Es la vocal a
Hasta la vista

$>java Caracter e↵
Es la vocal e
Hasta la vista

$>java Caracter u↵
Es la vocal u
Hasta la vista

$>java Caracter A↵
Es la vocal A
Es una vocal mayuscula
Hasta la vista

$>java Caracter E↵
Es una vocal mayuscula
Hasta la vista

$>java Caracter B↵
No es una vocal
Hasta la vista
```