

### 3. Estructuras de selección

Existen tres tipos básicos de instrucciones de bifurcación. En general, cabe distinguir entre selección simple, selección doble y/o selección múltiple, dependiendo del número de opciones que necesitemos.

#### Estructuras condicionales

En Java, las estructuras condicionales se implementan mediante estos tres tipos:

Selección simple: `if (condicion) { ... }`

Selección doble: `if (condicion) { ... } else { ... }`  
`operador (condicion)? { ... } : { ... }`

Selección múltiple: `if (condicion1) { ... } else { ... }` encadenados  
`(if (condicion) { ... } else if (condición) { ... } else)`  
`switch (condicion) case { ... }`

La condición debe ser una expresión booleana, es decir, debe dar como resultado un valor booleano (true o false) tras evaluar la expresión que contenga, la cual puede ser simple o compleja, y añadiendo más operadores lógicos dentro de la misma.

#### 3.1. Selección simple

Se evalúa la condición, que será una expresión booleana. Si esta se cumple, entonces se ejecuta una determinada acción o un grupo de acciones. En caso contrario, se salta dicho grupo de acciones sin realizarlas, y se prosigue con el siguiente bloque de código que aparezca a continuación.

```
if (expresión_booleana) {
    instrucción 1;
    instrucción 2;
    // Tantas instrucciones como necesitemos ...
    instrucción N;
}
```

Si el bloque de instrucciones contiene una sola instrucción, no es necesario escribir las llaves {}. No obstante, para evitar confusiones se recomienda escribir las llaves siempre, por si en un futuro se amplía ese bloque de código con más instrucciones.

**Ejemplo:** veamos un programa que pide la edad al usuario y determina si es mayor de edad.

```
import java.util.Scanner;
public class Edades {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        int edad;
        System.out.println("Introduzca la edad: ");
        edad = teclado.nextInt();
        // Comprobar si es mayor de edad, también valdría la condición
        de si edad > 17.
        if (edad >= 18){
            System.out.println("Eres mayor de edad.");
        }
    }
}
```

3.2. Selección doble

Se evalúa la condición con su expresión booleana y, si esta se cumple, se ejecuta una determinada instrucción o el bloque de instrucciones del primer bloque de código. Pero si no se cumple, entonces se ejecuta otra instrucción o el bloque de instrucciones alternativos del segundo bloque.

```
if (expresión booleana) {
    instrucciones del bloque 1;
    // Tantas instrucciones como necesitemos
}
else {
    instrucciones del bloque 2 o alternativo;
    // Tantas instrucciones como necesitemos
}
```

Solo se ejecutará uno de los dos bloques de código, nunca se ejecutarán ambos.

El operador condicional ?:

Se puede utilizar en sustitución de la sentencia de control if-else. Se trata de un operador condicional ternario, pues está formado por los caracteres “?” y “:” con tres operandos.

Su sintaxis es: **expresión1 ? expresión2 : expresión3**

Si expresión1 —que representa la condición— es cierta, entonces se evalúa expresión2 y este será el valor de la expresión condicional. Por otro lado, si expresión1 es falsa, se evalúa expresión3 y este será el valor de la expresión condicional.

**Ejemplo:** programas equivalentes que piden por teclado un número y devuelven el mensaje de si es par o impar.

Estructura if ... else ...	Operador condicional ?
<pre>import java.util.Scanner; public class ParImpar1 {     public static void main(String[] args) {         Scanner teclado = new Scanner(System.in);         int num;          System.out.println("Introduzca numero: ");         num = teclado.nextInt();          //Mostrar si es número par o impar         if ((num%2) == 0){             System.out.println("PAR");         }         else{             System.out.println("IMPAR");         }     } }</pre>	<pre>import java.util.Scanner; public class ParImpar2 {     public static void main(String[] args) {         Scanner teclado = new Scanner(System.in);         int num;          System.out.println("Introduzca numero: ");         num = teclado.nextInt();          //Mostrar si es número par o impar         System.out.println((num%2) == 0 ? "PAR" : "IMPAR");     } }</pre>

Tabla 3. 1. Ejemplo de la if ... else... y del operador condicional ?

Como puedes ver, el operador condicional ternario simplifica el código, pues nos permite hacer lo mismo con menos líneas de código.

### 3.3. Selección múltiple: sentencias if (condición) { ... } else { ... } anidadas

Se obtiene anidando sentencias if (condición) { ... } else { ... }. Esto permite construir estructuras de selección más complejas con más de una alternativa. Al estar anidadas, vamos descartando opciones incorrectas hasta llegar a la que sí sea cierta para nuestro objetivo.

```
if (expresion_booleana1) {  
    instrucciones;  
}  
else if (expresion_booleana2) {  
    instrucciones;  
}  
else {  
    instrucciones;  
}
```

De este modo, cada else se corresponde con el if más próximo que no haya sido emparejado.

Una vez que se ejecuta un bloque de instrucciones, la ejecución continúa en la siguiente instrucción que aparezca después de las sentencias if (condición) { ... } else { ... } anidadas.



#### IMPORTANTE

Descarga el **ejemplo 1** de la Unidad con un programa que muestra un saludo según la hora indicada.

#### A. Selección múltiple. Instrucción switch

Se utiliza para seleccionar una de entre múltiples alternativas. Equivale a las sentencias if (condición) { ... } else { ... } anidadas, pero da como resultado un bloque de código más claro y breve. La forma general de la instrucción switch en Java es la siguiente:

```
switch (expresión) {  
    case valor 1:  
        instrucciones;  
        break;  
    case valor 2:  
        instrucciones;  
        break;  
    // Tantos casos con su valor como haga falta evaluar...  
    // Y por último el caso por defecto que admite todos los que no se  
    // hayan evaluado antes.  
    default:  
        instrucciones para el caso por defecto;  
}
```

La instrucción switch se puede usar con datos de tipo byte, short, char e int. También con tipos enumerados y con las clases envolventes Character, Byte, Short e Integer. A partir de Java 7, también pueden usarse con datos de tipo String en un switch.

