INSTRUCIONES DE CONTROL

Como cualquier otro lenguaje de programación, Java dispone de un juego de instrucciones para controlar el flujo de ejecución de un programa. Tenemos instrucciones alternativas y repetitivas, a continuación estudiaremos cada una de ellas.

1 . Instrucción *if*

La instrucción ***if*** es una sentencia de tipo *alternativa simple* que permite comprobar una condición dentro de un programa. En caso de que la condición se cumpla se ejecutarán un determinado conjunto de instrucciones, mientras que si no se cumple, se podrá optar por ejecutar otro conjunto diferente de instrucciones o por no ejecutar ninguna.

En la figura 1 se muestra el formato de esta instrucción con un ejemplo de utilización.

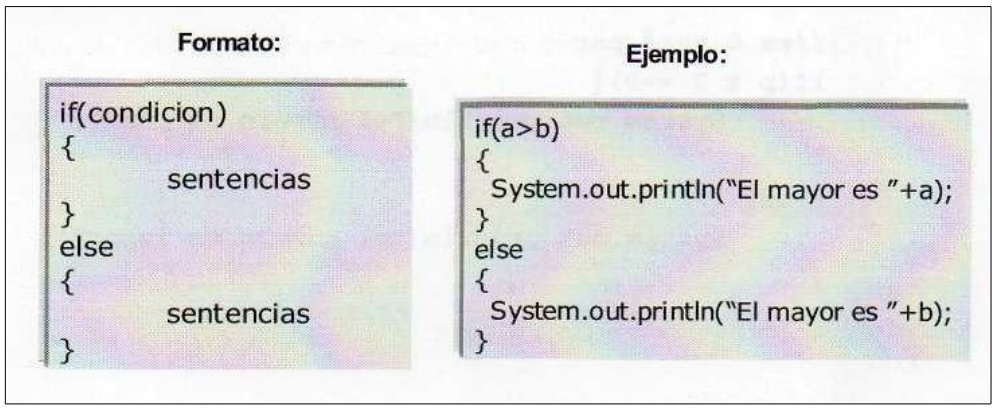

Fig. 1 - Formato y utilización de la instrucción if

A la hora de utilizar esta instrucción hay que tener en cuenta lo siguiente:

- La condición de comprobación puede ser cualquier expresión cuyo resultado sea de tipo boolean *(true* o *false)*, en cualquier otro caso se producirá un error de compilación. El siguiente código representa una utilización incorrecta de *if*.

```
int a = 5;
if(a) //error de compilación
```

- El bloque *else* es opcional. En este caso, si la condición no se cumple el programa continuará su ejecución en la siguiente línea después de la llave de cierre "`}`" del ***if***.

- Cuando el bloque de sentencias, bien de ***if*** o bien de ***else***, está formado únicamente por una instrucción, la utilización de las llaves delimitadoras es opcional. No obstante, para una mayor claridad en el código, **se recomienda su uso en cualquier caso**.

- Las instrucciones ***if*** se pueden anidar.

El siguiente programa utiliza una instrucción ***if*** para indicarnos si el número almacenado en cierta variable es par o impar:

```
public class CompruebaPar {
   public static void main (String [] args) {
      int p =5;   //variable con el número a comprobar
                  //si el resto de la división del
                  //número entre 2 es 0 será par
      if (p % 2 ==0} {
            System. out . println { "el número es par");
      } else {
            System. out .println ( "el número es impar"};
      }
   }
}
```

2 . La instrucción ***switch***

Se trata de una instrucción de tipo alternativa múltiple. Permite ejecutar diferentes bloques de instrucciones en función del resultado de una expresión.

La figura 2 muestra el formato de la instrucción y un ejemplo de utilización.

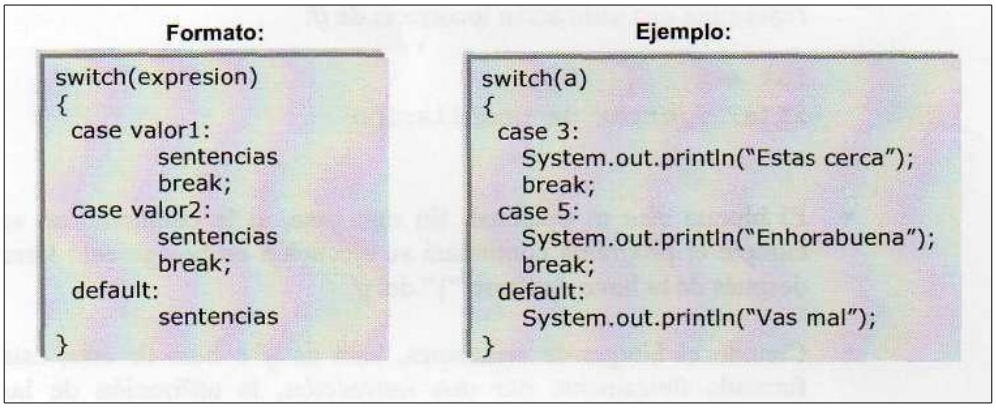

Fig. 2 - Formato y ejemplo de utilización de la instrucción switch

En caso de que el resultado de la expresión coincida con el valor representado por *valor1*, se ejecutarán las sentencias definidas en este bloque, si no coincide se comparará con *valor2*, y así sucesivamente. Si el resultado no coincide con ninguno de los valores indicados en los ***case***, se ejecutará el bloque de instrucciones indicado en ***default***.

Sobre el uso de la instrucción ***switch*** hay que tener en cuenta lo siguiente:

- Un ***switch*** puede contener cualquier número de ***case***, aunque **no puede haber dos case con el mismo valor.**

- La sentencia ***break*** es opcional y se emplea para provocar la finalización del ***switch*** al terminar la ejecución de un ***case***. En caso de que un determinado ***case*** no incluya esta instrucción y se produzca la ejecución de su bloque de sentencias, al salir de dicho bloque, el programa continuará con la ejecución del siguiente ***case***, **independientemente de que el resultado de la expresión coincida o no con el valor indicado en el mismo.** Por ejemplo, el siguiente código:

```
 int h=5;
 switch(h*2){
   case 10:
        System.out.println ("El resultado es 10");
   case 20:
        System.out.println{"El tamaño es demasiado alto");
        break;
 default:
        System.out.println("El resultado no es correcto");
}
```

Imprimirá en pantalla:

El resultado es 10
El tamaño es demasiado alto

- El bloque ***default*** se ejecutará si el resultado de la expresión no coincide con ningún ***case***. Su uso es opcional.

3 . La instrucción *for*

La instrucción repetitiva ***for*** permite ejecutar un conjunto de instrucciones un número determinado de veces. Su formato y ejemplo de utilización se muestran en la figura 3.

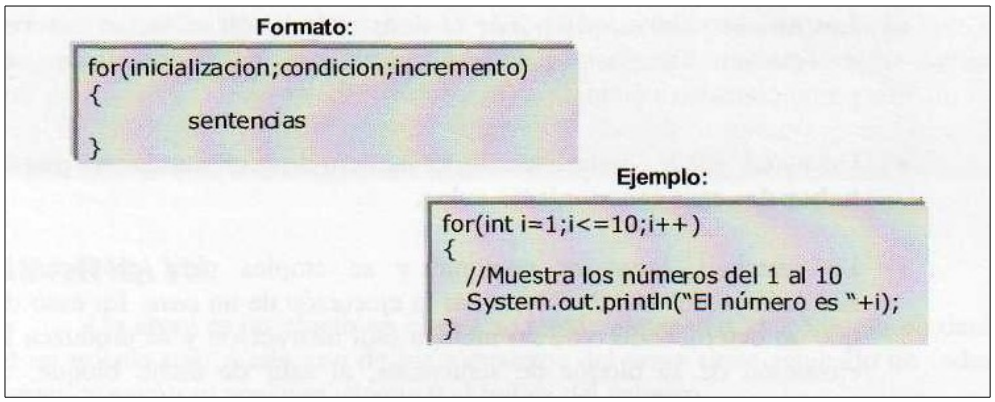

Fig. 3 - Utilización de la instrucción for

La ejecución del bucle ***for*** comienza con la instrucción de inicialización, que, como su nombre indica, suele realizar la inicialización de una variable de control, incluyendo su declaración. A continuación, se comprueba la condición, cuyo resultado debe ser siempre de tipo ***boolean;*** en caso de que el resultado sea *true*, se ejecutarán las instrucciones delimitadas por el bloque de llaves `{}`, después se ejecutará la instrucción de incremento y volverá a comprobarse la condición. En el momento en que la condición sea *false*, las instrucciones del bloque no se ejecutarán, continuando el programa en la siguiente línea al bloque de instrucciones.

Sobre la utilización de la instrucción ***for*** hay que tener en cuenta lo siguiente:

- **Las instrucciones de control del bucle *for*** (inicialización, condición e incremento) son **opcionales.** En cualquier caso, el delimitador de instrucciones "`;`" siempre debe estar presente. Por ejemplo, si se omiten las instrucciones de comparación e incremento, la cabecera del ***for*** quedaría:

  ```
  for(int i=0;;)
  ```

- Si se declara una variable en la instrucción de inicialización, ésta será accesible únicamente desde el interior del ***for***.
- Al igual que sucede con ***if***, las llaves delimitadoras de bloque solamente son obligatorias si el ***for*** está compuesto por más de una instrucción.

El siguiente programa utiliza un bucle ***for*** para realizar el cálculo del factorial de un número almacenado en una variable:

```
    public class Factorial{public static void main (String [] args) {
        long p =5; //variable con el número a calcular
        long r=1; //variable que almacena el resultado
        for(int i=1 ; i<=p; i++) {
            r* = i ;
            }
        System. out . println { "El factorial de "+p +" es "+r) ;
    }
}
```

4 . La instrucción *while*

Permite ejecutar un bloque de instrucciones mientras se cumpla una determinada condición dentro del programa. Los dos posibles formatos que admite esta instrucción y un ejemplo de utilización se muestran en la figura 4.

En ambos casos, el bloque de instrucciones se ejecuta mientras la condición se cumple. En el segundo formato se ejecutan las instrucciones y luego se comprueba la condición, lo que garantiza que el bloque de instrucciones se ejecute por lo menos una vez.

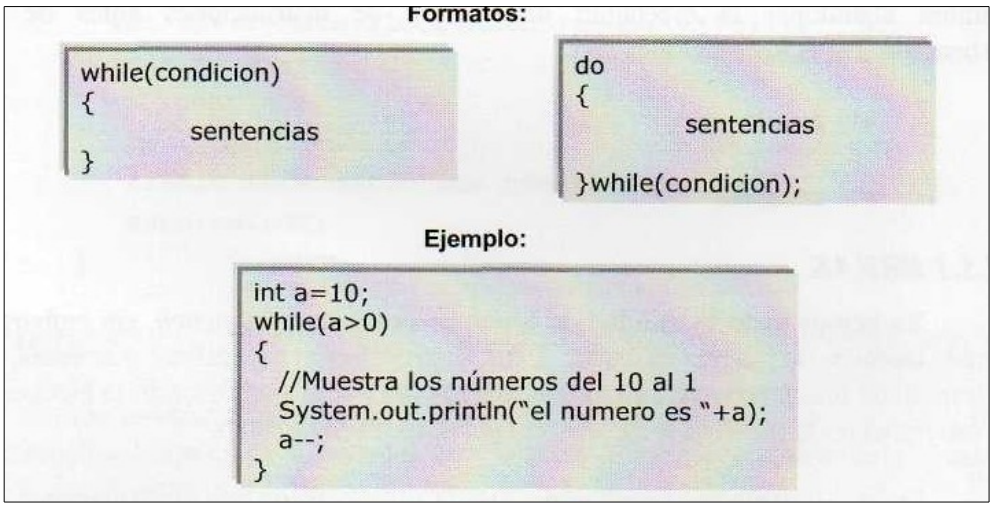

Fig. 4 - Instrucción `while` y formato de utilización

Como en el caso de la instrucción ***for***, la utilización de llaves para delimitar un bloque de instrucciones sólo es obligatoria si éste está formado por más de una instrucción.

El siguiente programa utiliza un ***while*** para calcular la suma de todos los números enteros entre dos dados. En el momento en que la suma parcial llegue a alcanzar o superar el valor 1000 se dejará de realizar la operación:

```
public class Sumador{
      public static void main (String [] args){
            int n1 = 5; //número más pequeño
            int n2 = 15; //número mayor
            int res = n1; //variable que almacena el
                        //resultado
            while(res<1000 && n1<n2){
                  res += ++n1;//incrementa ni y lo acumula
                              //a las sumas parciales
            }
      }
}
```

5 . Salida forzada de un bucle

Las instrucciones repetitivas ***for*** y ***while***, cuentan con dos instrucciones que permiten abandonar la ejecución del bloque de instrucciones antes de su finalización. Estas instrucciones son:

- ***break***
- ***continue***

BREAK

Ya hemos visto la utilidad de ***break*** en una sentencia ***switch***, sin embargo, su uso también se puede extender a las instrucciones repetitivas. En éstas, la utilización de ***break*** provoca una salida forzada del bucle, continuando la ejecución del programa en la primera sentencia situada después del mismo.

En el siguiente ejemplo, se detiene la ejecución de ***for*** al producirse una determinada situación:

```
int numero=6;
boolean acíerto=false;
for (int i=1 ; i<=5 ; i++)
{
   int aleat=Math.floor(Math.random( )*10+1);
   if (aleat == numero)
      {
          Acierto = true ;
          //si acierta el número finaliza la ejecución de
          //la instrucción for
          break ;
      }
}
```

CONTINUE

La instrucción ***continue***, provoca que el bucle detenga la iteración actual y pase, en el caso de ***for***, *a* ejecutar la instrucción de incremento o, en el caso de ***while***, *a* comprobar la condición de entrada.

El siguiente trozo de código suma números impares aleatorios entre 1 y 10, mientras la suma total sea inferior a 1000. La utilización de ***continue*** en este ejemplo impide que los números pares sean sumados:

```
int suma=0;
while (suma<1000) {
     int aleat=Math.floor(Math.random()*10+1);
     if (aleat%2==0) {
       continue;
     }
     //sólo se sumarán los números impares
     suma+=aleat;
}
```