PROGRAMACIÓN UD-3

BUCLES

ÍNDICE

[INTRODUCCIÓN 3](#_Toc117448048)

[BUCLES CONTROLADOS POR CONDICIÓN 3](#_Toc117448049)

[A. WHILE 3](#_Toc117448050)

[1. SINTAXIS 3](#_Toc117448051)

[2. COMPORTAMIENTO 3](#_Toc117448052)

[B. DO WHILE 4](#_Toc117448053)

[1. SINTAXIS 4](#_Toc117448054)

[2. COMPORTAMIENTO 4](#_Toc117448055)

[BUCLES CONTROLADOS POR CONTADOR 4](#_Toc117448056)

[A. FOR 4](#_Toc117448057)

[1. SINTAXIS 5](#_Toc117448058)

[2. COMPORTAMIENTO 5](#_Toc117448059)

[SALIDAS ANTICIPADAS 5](#_Toc117448060)

[BUCLES ANIDADOS 6](#_Toc117448061)

[A. BUCLES INDEPENDIENTES 6](#_Toc117448062)

[B. BUCLES DEPENDIENTES 6](#_Toc117448063)

# INTRODUCCIÓN

Los **bucles** son estructuras que contienen bloques de instrucciones las cuales se ejecutan varias veces, cada una de estas veces recibe el nombre de **iteración**.

Su utilización simplifica la estructura del código evitando la aparición de duplicidad de código.

Cuando haya un fragmento de código que haya que repetir varias veces lo mejor es hacerlo a través de un bucle. En Java diferenciamos 3 bucles: 2 bucles controlados por condición (**bucle *while*** y **bucle *do* *while***) y 1 bucle controlado por contador (**bucle *for***).

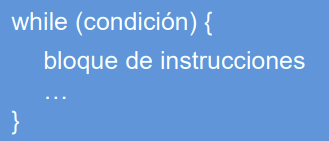
# BUCLES CONTROLADOS POR CONDICIÓN

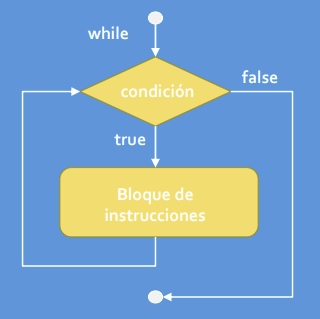
El control del número de iteraciones se lleva a cabo mediante una condición. Mientras la condición sea cierta (true), el bucle realizará una nueva iteración.

## *WHILE*

Su comportamiento depende de la evaluación de una condición. El bucle ***while*** decide si realizar una nueva iteración basándose en el valor de una condición.

1. SINTAXIS



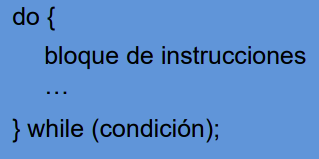
1. COMPORTAMIENTO
2. Se evalúa la condición.
3. Si esta condición da como respuesta un:
   1. TRUE, se accede al bucle. Se ejecutan las instrucciones y se vuelve al punto 1 para reevaluar si debe realizarse una nueva iteración.
   2. FALSE, no se accede al bucle. Se salta a la siguiente instrucción del programa (si la hubiera).

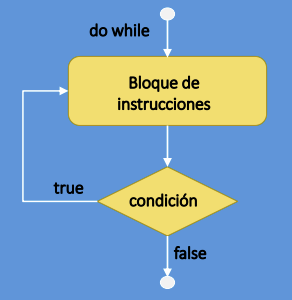
Por tanto, un bucle while puede repetirse un número indeterminado de veces incluidas 0 veces si la primera vez que se evalúe la condición esta resulta en un FALSE. Así mismo puede repetirse un número infinito de veces si la condición siempre resulta en un TRUE, esto es lo que se conoce como **bucle infinito**.

## DO WHILE

La diferencia principal con el *while* es que el bucle ***do while*** siempre se realizará una vez ya que la **evaluación** no se realiza antes de la entrada al bucle sino **al final**.

1. SINTAXIS



1. COMPORTAMIENTO
2. Se accede al bucle y se ejecutan las instrucciones vez.
3. Se evalúa la condición:
   1. TRUE. Se vuelve al punto 1.
   2. FALSE. Se termina el bucle y se pasa a la siguiente instrucción del programa (si la hubiera).

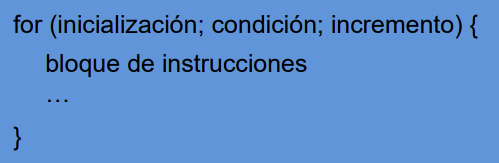
Por tanto, en este caso el bucle se podrá ejecutar un número indeterminado de veces, pero el mínimo siempre será 1.

# BUCLES CONTROLADOS POR CONTADOR

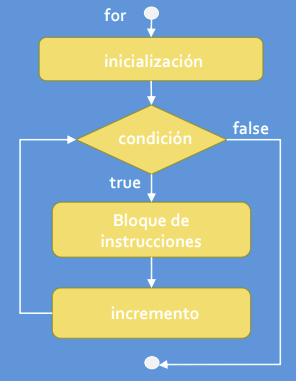
## FOR

Este bucle permite controlar el número de iteraciones mediante una variable (contador).

1. SINTAXIS



* Inicialización: Lista de instrucciones, separadas por comas, donde generalmente se inicializan las variables que van a controlar el bucle. Se ejecutan una sola vez antes de la primera iteración (por ejemplo, “i = 0;”).
* Condición: Booleano que controla las iteraciones del bucle. Se evalúa antes de cada iteración, mientras lance un TRUE como resultado, el bucle se repetirá (por ejemplo, “i >= 10;”).
* Incremento: Lista de instrucciones, separadas por comas, donde se suelen modificar las variables que controlan la condición. Se ejecutan al final de cada iteración (por ejemplo, “i++”).

1. COMPORTAMIENTO
2. Se ejecuta la inicialización.
3. Se evalúa la condición:
   1. Si es TRUE, se ejecuta el bucle (se pasa al punto 3).
   2. Si es FALSE, se salta el bucle y se pasa a la siguiente instrucción del programa si la hubiera.
4. Se ejecutan las instrucciones del bucle y, al final, se produce el incremento.
5. Se vuelve a comprobar la condición (punto 2).

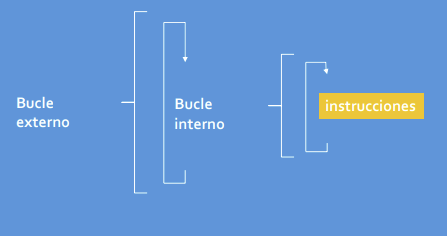
En este caso el bucle se repetirá un número determinado de condiciones porque se configura a través de la inicialización y el incremento de las variables que controlan el bucle.

# SALIDAS ANTICIPADAS

Dependido de la lógica a implementar en un programa puede ser necesario terminar un bucle antes de tiempo y no esperar a que termine por condición (hasta que realice todas las iteraciones). Disponemos de:

* ***Break***: Interrumpe la ejecución del bucle.
* ***Continue***: Detiene la iteración actual y continua con la siguiente.

# BUCLES ANIDADOS

La **anidación** de bucles es muy frecuente, consiste en incluir un bucle dentro de otro:

Al hacerlo se multiplica el número de veces que se ejecuta el bloque de instrucciones de los bucles internos. Los bucles anidados pueden estar relacionados con las condiciones de los bucles externos (**bucles dependientes**) o no estarlo (**bucles independientes**).

## BUCLES INDEPENDIENTES

Son las anidaciones resultantes de bucles que no dependen unos de otros para determinar el número de iteraciones.

Facilitan el procesado de tablas multidimensionales utilizando cada nivel de anidación para manejar el índice de cada dimensión.

Sin embargo, el uso descuidado de bucles anidados puede convertir el algoritmo en ineficiente al aumentar demasiado el número de instrucciones a realizar.

## BUCLES DEPENDIENTES

Son las anidaciones resultantes de bucles que sí depende unos de otros para determinar el número de iteraciones.