## VARIABLES AUXILIARES DE UN PROGRAMA.

Son objetos que utiliza un programa y por la función que realizan dentro del mismo toman un nombre especial, modelizando su funcionamiento debido a su frecuente utilización. En estas variables se almacenan datos, por ejemplo, **la suma de una serie de importes**, o **la cuenta del número de alumnos que hay en clase**, o **la nota máxima y mínima de un grupo de alumnos**, etc.

##### Contadores.

Un contador es un objeto que se utiliza para **contar cualquier evento que pueda ocurrir dentro de un programa.** En general **suelen contar de forma natural desde 0 y de 1 en 1,** aunque se pueden realizar otros tipos de cuenta necesarios en algunos procesos.

Se utilizan realizando sobre ellos dos operaciones básicas:

* **Inicialización**: Todo contador se inicializa a 0 si realiza una cuenta natural, o a otro valor si se desea realizar otro tipo de cuenta.

int C1 = 0;

int CP = 2;

* **Contabilización o incremento:** Cada vez que aparece el evento a contar se ha de incrementar el contador en 1 si se realiza cuenta natural o en otro valor si se realiza otro tipo de cuenta.

C1 = C1 + 1;

CP = CP + 2;

##### Acumuladores.

Son objetos que se utilizan en un programa para **acumular elementos sucesivos con una misma operación**. En general se utilizan para calcular sumas y productos, sin descartar otros posibles tipos de acumulación.

Para utilizarlos se realizarán sobre ellos dos operaciones básicas:

* **Inicialización**: Todo acumulador se inicializa a 0 si realiza sumas y a 1 si realiza productos.

double SUMA = 0; int PRODUC = 1;

* **Acumulación**: Cuando se hace presente en la memoria el elemento a acumular por la realización de una lectura o un cálculo, se efectúa la acumulación del mismo por medio de la asignación:

SUMA = SUMA + CANTIDAD; PRODUC = PRODUC \* NUMERO;

##### Máximos y Mínimos.

Un **máximo es una variable que guarda el máximo valor de un conjunto de valores**. El máximo se inicializa a un valor muy pequeño para que tengamos la seguridad de que el primer valor que se almacene en esta variable sea el mayor.

Un **mínimo es similar al máximo solo que coge el valor más pequeño de una serie de valores**. El mínimo se inicializa a un valor máximo.

int maximo = -9999; int minimo = 9999

Ejemplo:

**if**(nota > maximo ) { maximo = nota;

}

**if**(nota < minimo ) { minimo = nota;

}

##### Switches. Banderas. Flags

Son variables que se utilizan en un programa y sólo pueden tomar dos valores (true y false, 0 y 1) realizando la función de transmitir información de un punto a otro dentro del programa. Se utilizan iniciándolos con un valor y en los puntos en que corresponda se cambian al valor contrario.

int sw = 0;

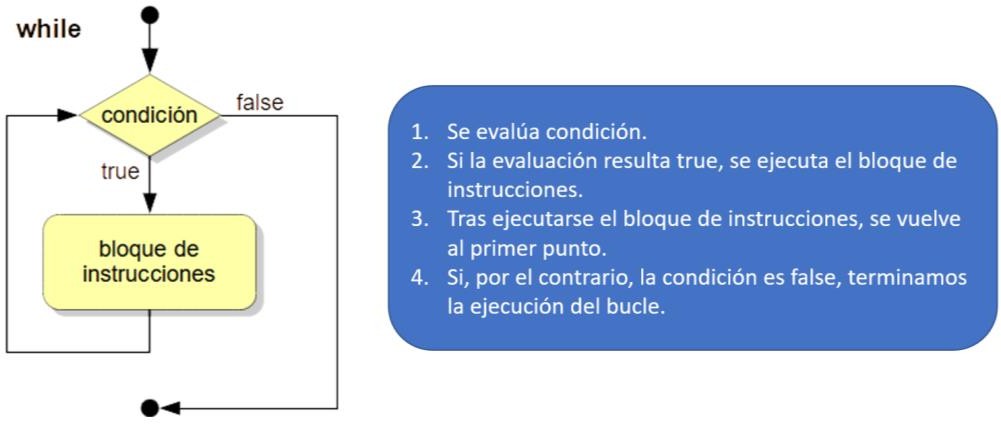
if(sw == 0) {

//acciones sw = 1;

}

Por ejemplo, se pueden usar para pintar una cabecera solamente una vez dentro de un bucle. Si el switch es igual 0 se pinta la cabecera y se cambia el valor del switch para que no vuelva a pasar por ahí la ejecución y solo se pinte una vez la cabecera.

## WHILE



**while** (condición) {

bloque de instrucciones

. . . . . .

}

#### Ejemplos:

**int** i = 1; //valor inicial

**while**(i <= 3 ) {

System.***out***.println(i); i++;

}

###### Hacer la Traza de ejecución:

|  |  |
| --- | --- |
| I | Salida en pantalla |
| 1 |  |

**int** suma = 0;

**int** c = 1;

**while** (c <= 5) {

suma += c; c++;

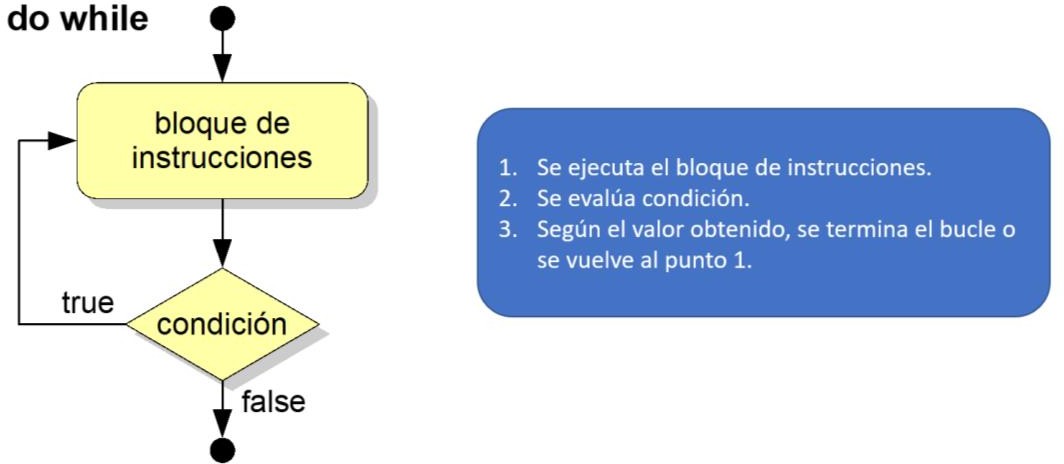
}

System.***out***.println("Valor de c: " + c); System.***out***.println("Valor de suma: " + suma);

**Hacer la Traza de ejecución:**

|  |  |  |
| --- | --- | --- |
| Suma | c | Salida en pantalla |
| 0 | 1 |  |

## DO-WHILE

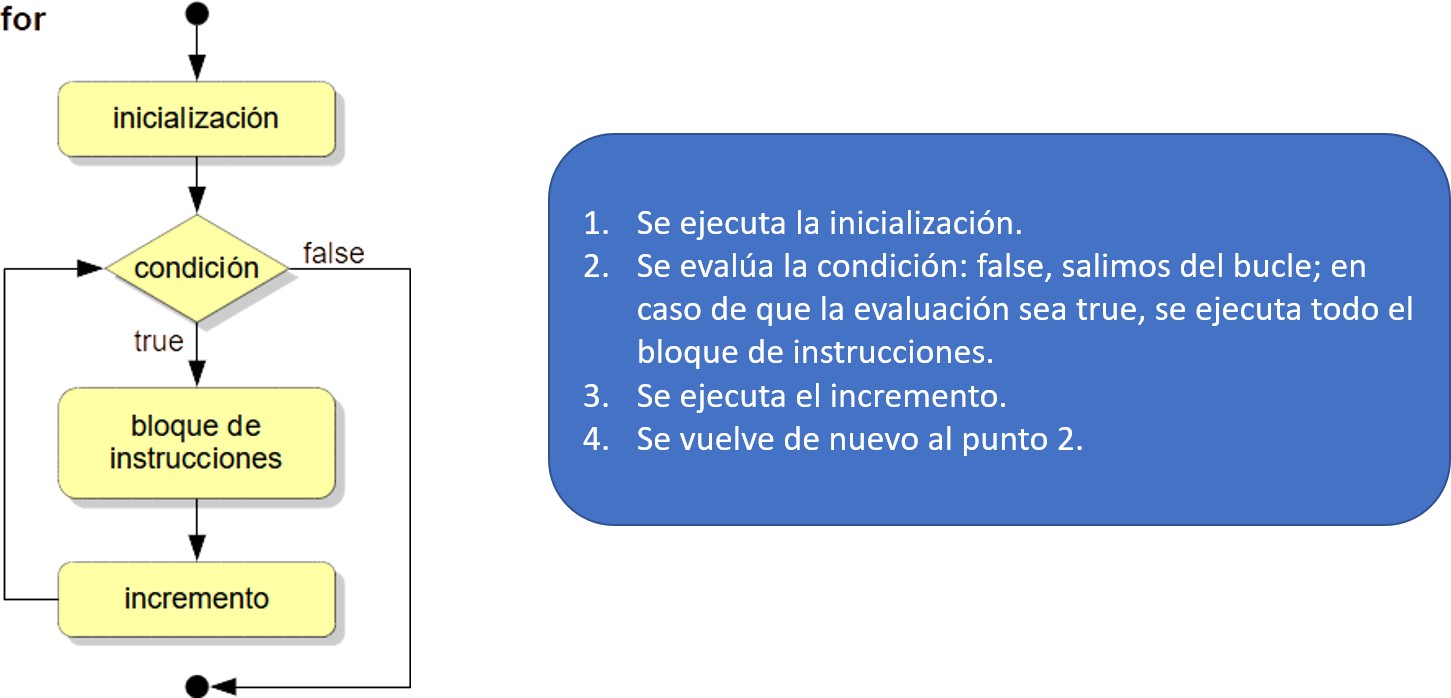


**do** {

### bloque de instrucciones

} **while** (condición);

## FOR



**for** (Inicialización; Condición; Incremento) {

### bloque de instrucciones

*. . . . . ..*

};

**Ejemplos:**

**for** (**int** n = 0; n < 10; n++) { System.***out***.print(n + “ “);

}

*Visualiza: 0 1 2 3 4 5 6 7 8 9*

Partes del bucle for

* **Inicialización** de la variable de control. La inicialización puede tener varias expresiones separadas por comas. También se pueden declarar variables en esta zona.
* **Condición**: condición que se debe de cumplir para que se realice el bucle y se ejecuten las sentencias. Normalmente es una comprobación del valor de la variable de control en una expresión condicional.
* **Incremento:** de la variable de control o de las variables que controlan el bucle. Puede tener varias expresiones separadas por comas.

## BREAK Y CONTINUE

**break**: interrumpe completamente la ejecución del bucle.

**continue**: detiene la iteración actual y continua la siguiente.

# continue:

* + Se emplea sólo en bucles
  + Al ejecutarse la iteración en la que se encuentra, esta finaliza y se inicia la siguiente iteración.

public class TablaProducto1 {

public static void main (String [] args) { int valor;

valor = 8;

System.out.println("Tabla de multiplicar del numero " + valor); for (int i=1; i<=10; i++) {

###### if (i==5) continue;

System.out.println(valor + " \* " + i + " = " + valor\*i );

}

}

}

Muestra la siguiente salida, en la que **no aparece el 5**, ya que si la i es 5 no se ejecutan las sentencias que hay debajo y comienza la siguiente iteración:

Tabla de multiplicar del numero 8

8 \* 1 = 8

8 \* 2 = 16

8 \* 3 = 24

8 \* 10 = 80

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | \* | 4 | = | 32 |
| 8 | \* | 6 | = | 48 |
| 8 | \* | 7 | = | 56 |
| 8 | \* | 8 | = | 64 |
| 8 | \* | 9 | = | 72 |

# break:

* + Se usa en sentencias **switch** o en bucles
  + Deja el ámbito de la sentencia en la que se encuentra y pasa a la siguiente sentencia. Es decir, si se está ejecutando dentro de un bucle interrumpe completamente la ejecución del bucle.

###### public class TablaProducto2 {

public static void main (String [] args) { int valor;

valor = 8;

System.out.println("Tabla de multiplicar del numero " + valor); for (int i=1; i<=10; i++) {

###### if (i==5) break;

System.out.println(valor + " \* " + i + " = " + valor\*i );

}

}

}

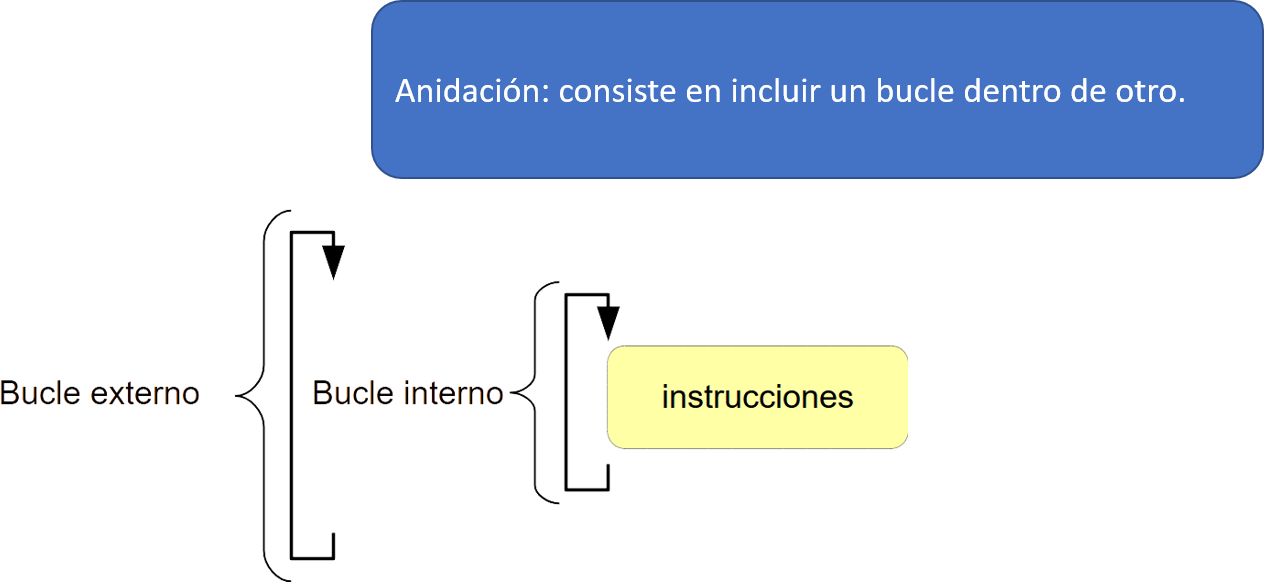
Muestra la siguiente salida, en la que no aparecen las multiplicaciones a partir del 5, ya que cuando la i es 5 el bucle for finaliza:

Tabla de multiplicar del numero 8

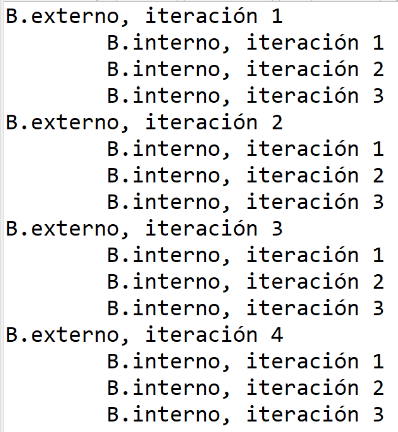
8 \* 1 = 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 8 | \* | 2 | = | 16 |
| 8 | \* | 3 | = | 24 |
| 8 | \* | 4 | = | 32 |

## BUCLES ANIDADOS



#### Bucles independientes

LOS BUCLES ANIDADOS NO DEPENDEN UNOS DE OTROS PARA DETERMINAR EL NÚMERO DE ITERACIONES. EN ESTA CASO EL PRIMER BUCLE HACE 4 ITERACIONES Y EL SEGUNDO 3.

//bucle independiente

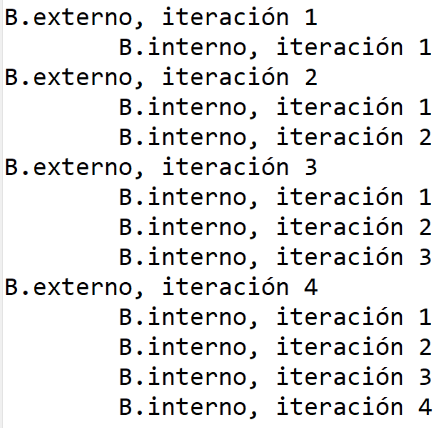
**for** (**int** i = 1; i <= 4; i++) { System.***out***.println("B.externo, iteración " + i); **for** (**int** j = 1; j <= 3; j++) {

System.***out***.println("\tB.interno, iteración " + j);

}

}

#### Bucles dependientes

EL NÚMERO DE ITERACIONES DE UN BUCLE INTERNO DEPENDE DE LOS BUCLES EXTERIORES, DE SUS VARIABLES DE CONTROL. EN ESTE CASO EL PRIMER BUCLE HACE 4 ITERACIONES Y EL NÚMERO DE ITERACIONES DEL SEGUNDO BUCLE DEPENDERÁ DE LA VARIABLE DE CONTROL DEL PRIMER BUCLE.

//bucle dependiente

**for** (**int** i = 1; i <= 4; i++) { System.***out***.println("B.externo, iteración " + i); **for** (**int** j = 1; j <= i; j++) {

System.***out***.println("\tB.interno, iteración " + j);

}

}