ESTRUCTURA DE ITERACIÓN

DEFINICIÓN

- Las **Sentencias de Iteración** o **Ciclos** son estructuras de control que repiten la ejecución de un grupo de instrucciones.
- Una sentencia de iteración es una estructura de control condicional, ya que dentro de la misma se repite la ejecución de una o más instrucciones mientras que una condición específica se cumpla.
- Muchas veces tenemos que repetir un número definido o indefinido de veces un grupo de instrucciones por lo que en estos casos utilizamos este tipo de sentencias.

SENTENCIA FOR

- La sentencia for es útil para los casos en donde se conoce de antemano el número de veces que una o más sentencias han de repetirse.

```
for(contador; final; incremento)
{
    Codigo a Repetir;
}
```

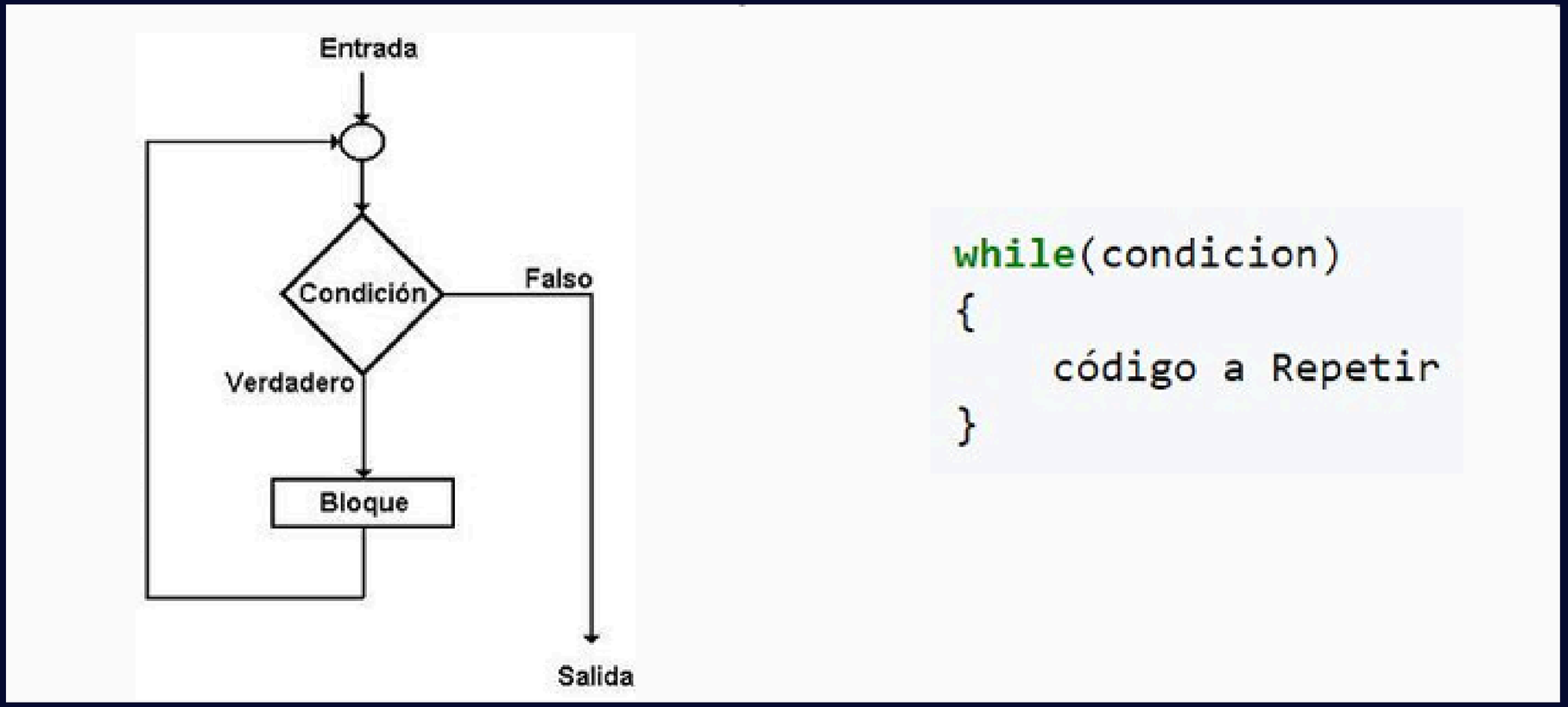
SENTENCIA FOR

- Caracteristicas:
  - **Contador** es una variable numérica.
  - **Final** es la condición que se evalúa para finalizar el ciclo (puede ser independiente del contador)
  - **Incremento** es el valor que se suma o resta del contador
  - El for evalúa la condición de finalización y mientras esta se cumpla continuarán las repeticiones

```
for(contador; final; incremento)
{
    Codigo a Repetir;
}
```

SENTENCIA WHILE

- La sentencia while es útil en aquellos casos en donde no se conoce de antemano el número de veces que una o más sentencias se tienen que repetir.




```
while(condicion)
{
    código a Repetir
}
```

SENTENCIA WHILE

- Características estructura WHILE:
  - **Condición** es la expresión a evaluar.
  - Si la **condición** es **verdadera**, continúa el ciclo de iteración.
  - Si la **condición** inicialmente es falsa, nunca ingresa al ciclo.
  - Evita crear una condición que nunca se vuelva falsa, ya que esto provocara un ciclo de iteracion infinito, provocando que la aplicación falle por alta de memoria

```
while(condicion)
{
    código a Repetir
}
```

SENTENCIA WHILE CORRECTA

```
int i = 1;
while (i <= 5) {
    System.out.println("Número: " + i);
    i++; // la condición se actualiza
}
```

Imprime:

```
Número: 1
Número: 2
Número: 3
Número: 4
Número: 5
```

Flujo de ejecución

1. i vale 1 → condición 1 <= 5 es true → imprime Número: 1 → hace i++ (ahora i=2).
2. Repite hasta que i llegue a 6.
3. Cuando i es 6 la condición 6 <= 5 es false → sale del while y el programa continúa.

SENTENCIA WHILE INCORRECTA

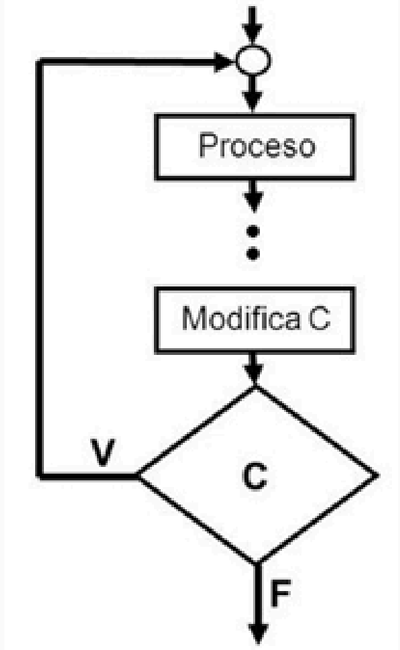
```
int i = 1;
while (i <= 5) {
    System.out.println("Número: " + i);
    // falta i++ → la condición nunca cambia
}
```

Flujo de ejecución

1. Falta de actualización → no hay i++ ni otra instrucción que cambie i. Como i nunca cambia permanece en 1 → la condición i <= 5 seguirá siendo true para siempre.
2. CONSECUENCIA: El programa entra en un bucle infinito: imprimirá Número: 1 repetidamente y nunca terminará por sí mismo. En la práctica consume CPU y la ejecución no continúa más allá de ese while.

SENTENCIA DO

La sentencia do es usada generalmente en cooperación con while para garantizar que una o más instrucciones se ejecuten al menos una vez.




```do
{
    código a Repetir
}
while(condicion)
```

SENTENCIA DO

- Características estructuca DO...While
  - **Condición** es la expresión a evaluar.
  - Si la **condición** es **verdadera,** continúa el ciclo de iteración.
  - Se ejecuta al menos una vez el código a repetir, antes de evaluar la condición.
  - Evitar crear una condición que nunca se vuelva Falsa, ya que esto provocará un ciclo de iteración infinito, provocando que la aplicación falle por falta de memoria.

SENTENCIAS BREAK Y CONTINUE

- En la condición **switch**, la sentencia **break** es utilizada con el propósito de forzar un salto dentro del bloque switch hacia el final del mismo.
- La misma sentencia se puede usar un conjunto con las distintas operaciones de iteración.
- Además existe una nueva sentencia que se puede incluir en los ciclos iterativos: **continue**

SENTENCIA BREAK

- La sentencia break se usa para forzar un salto hacia el final de un ciclo controlado por for o por while.
- La construcción del ciclo while para el caso de la sentencia break es diferente, esto para garantizar que el ciclo no vaya a caer en una iteración infinita.

```
for(contador; final; incremento)
{
   if(condicion) break;
   Codigo a Repetir;
}
```

```
while(condicion)
{
  if(condicion 2) break;
  código a Repetir
}
```

SENTENCIA CONTINUE

- La sentencia continue se usa para ignorar una iteración dentro de un ciclo controlado por for o por while.
- La construcción del ciclo while para el caso de la sentencia continue es diferente, esto para garantizar que el ciclo no vaya a caer en una iteración infinita.

```
for(contador; final; incremento)
{
   if(condicion)continue;
   Codigo a Repetir;
}
```

```
while(condicion)
{
  if(condicion 2) continue;
  código a Repetir
}
```

CONTADOR

- Un contador es una variable cuyo valor se incrementa o decrementa en una cantidad constante cada vez que se produce un determinado suceso o acción.
- Los contadores se utilizan con la finalidad de contar sucesos o acciones internas de un bucle; deben realizar una operación de inicialización y posteriormente las sucesivas de incremento o decremento del mismo.
- La inicialización consiste en asignarle al contador un valor y se situará antes y fuera del bucle.

```
int contador = 0;
while(contador<=10)
{
    contador = contador + 1;
}
```

```
int contador = 0;
while(contador<=10)
{
    contador++;
}
```

ACUMULADORES (TOTALIZADORES)

- Es una variable que suma sobre sí misma un conjunto de valores y de esta manera tener la suma de todos ellos en una sola variable.
- La diferencia entre un contador y un acumulador es que mientras el primero va aumentando constantemente es decir de uno en uno, de dos en dos, de diez en diez, etc, el acumulador va aumentando en una cantidad variable.

```
int total = 0;

total = num + 1;

total = total + num;

total = total + 1;
```