

Bucles en Java

Ciclo while

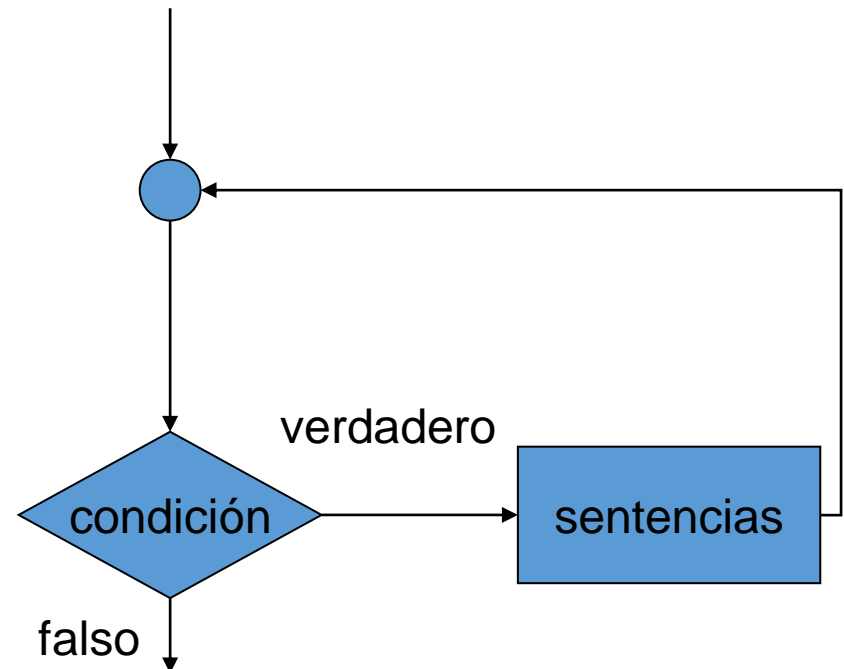
La sentencia while permite repetir un bloque de instrucciones.

La sintaxis del ciclo while es:

```
while (condición)  
    sentencia o bloque;
```

Si la condición se cumple se ejecutan las sentencias del bloque y se regresa el flujo de control a evaluar nuevamente la condición. El proceso se repite hasta que la condición sea falsa.

El ciclo puede ejecutarse 0 veces si la condición no se cumple al entraren él.



while

```
inicializacion;  
while (condicion) {  
    ...  
    step;  
}
```

```
int i = 0;  
while (i <= 10) {  
    System.out.println(i) ;  
    i++;  
}
```

```
1 package guia3;
2 public class EstructuraRepetitiva03
3 {
4     public static void main(String[] args)
5     {
6         long factorial = 1;
7         int n = 7;
8         int i = 1;
9         while(i < n + 1)
10        {
11            factorial = factorial * i;
12            i++;
13        }
14        System.out.println("Factorial de " + n + " es " + factorial);
15    }
16 }
```

Ciclos controlados por centinela

Si no se conoce de antemano el número de datos, se utiliza un valor de entrada especial como una bandera o centinela para terminar la entrada de datos.

En el caso del promedio los números positivos ingresados. el centinela puede ser -1.

Algoritmo

1. Solicitar un nuevo valor para promediar
2. Mientras valor diferente de -1 hacer
3. Acumular suma e incrementar contador
4. Solicitar nuevo valor para promediar
5. Fin ciclo
- 6 si contador > 0
7. Calcular promedio e imprimir

Ciclos anidados

Muchos algoritmos se definen mediante ciclos dentro de otros ciclos.

Es posible incluir dentro de un ciclo cualquier estructura de control incluyendo otro ciclo.

Ejemplo

Programa para imprimir un rectángulo de asteriscos (*) leyendo el tamaño del lado y la altura con un máximo de 20.


Ejemplo:

Si se teclea 10 8, imprimirá

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Este problema puede resolverse fácilmente haciendo un ciclo dentro de otro. El algoritmo es


1. Leer el valor de n y m
2. Si $n > 1$ y $n < 20$ y $m > 1$ y $m < 20$
3. $r = 0$
4. Mientras $r < m$ hacer
5. $c = 0$
6. Mientras $c < n$ hacer
7. Imprimir “*”
8. Incrementar c
9. Fin ciclo
10. Cambiar de línea
11. Incrementar r
12. Fin ciclo

3  [-] import javax.swing.JOptionPane;

4
5 public class clase52 {

6 [-] public static void main(String[] args) {

7 String numero;

8  int i=0, j=0;

9  while (j<8)

10 { i=0;

11 System.out.print("\n");

12 while (i<6)

13 {

14 System.out.print(" * ");

15 i++;

16 }

17 j++;

18 }

19 }

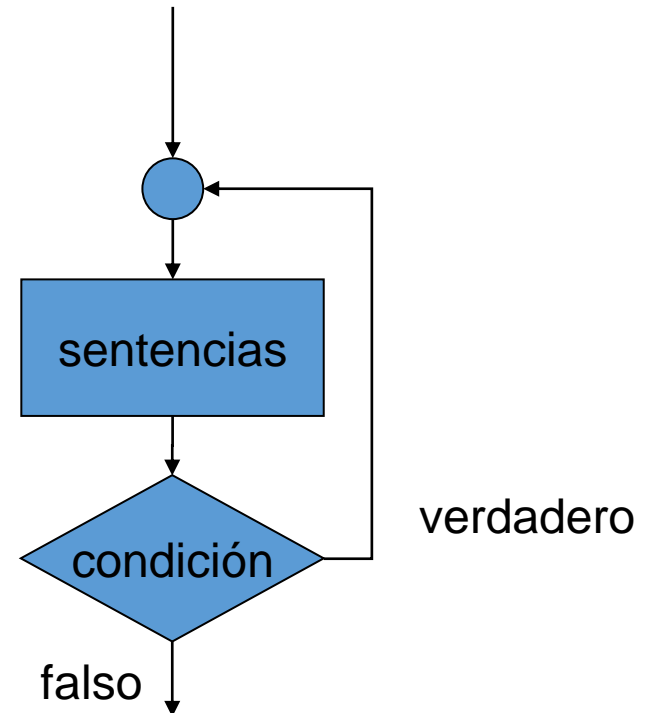
20 }

Ciclo do - while

El ciclo do-while es similar al ciclo while excepto que la prueba se realiza al final del ciclo, esto fuerza a que se ejecute por lo menos una vez.

Sintaxis

```
do{  
    sentencias;  
}while(condición);
```



Lectura con validación

El ciclo do-while puede usarse para validar la entrada de datos.

Algoritmo de validación:

1. Hacer
2. Leer datos
3. Mientras datos inválidos

do

inicializacion;

do {

...

step;

} while (condición) ;

int i = 0;

do {

System.out.println(i) ;

i++;

} while (i <= 10) ;

Estructura Repetitiva *for*

```
1  package guia3;
2  public class EstructuraRepetitiva01
3  {
4      public static void main(String[] args)
5      {
6          long factorial = 1;
7          int i = 1;
8          int n = 10;
9          do{
10             factorial = factorial * i;
11             i++;
12         }while(i < n + 1);
13         System.out.println("Factorial de " + n + " es " + factorial);
14     }
15 }
```

La sentencia for permite definir fácilmente ciclos controlados por contador.

El formato general de la estructura for es:

```
for(expresion1; expresion2; expresion3)  
    instrucción;
```

Esta es equivalente a la siguiente
sentencia while:

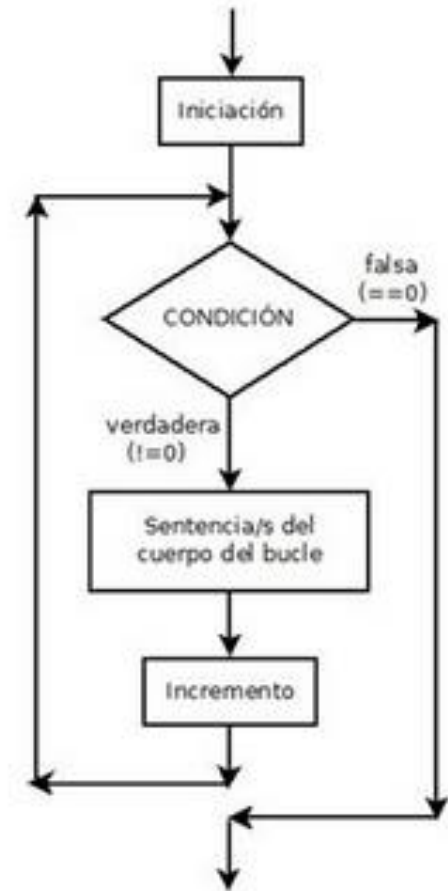
```
expresion1;  
while(expresion2) {  
    instrucción;  
    expresion3;  
}
```

expresion1 = sentencia de iniciación

expresion2 = condición de terminación

expresion3 = sentencia de incremento

Ciclo for



Inicialización

- `for(int i = 10, j = 20; ;);` //Podemos usar varias variables del mismo tipo
- `for(double salary = 3455.78F; ;);`
- `for(int i = 10, double d1 = 20.5; ;);` /* No usar variables de distinto tipo */
- `int i = 100;`
 `for(i++; ;);` // OK
- `for(System.out.println("Hello"); ;);` // OK
- `int num = 100;`
 `for(System.out.println("Hello"), num++; ;);`
- `int i = 10;`
 `for (int i = 0; ;);` //No se permite re-declarar una variable
- `int i = 10; i = 500;`
 `for (i = 0; ;);` // Si se permite re-inicializar una variable

CONDICIONES

a) modifica la variable de control de 1 a 100 en incrementos de 1.

```
for(i = 1; i <= 100; i++)
```

b) modifica la variable de control de 100 a 1 en decrementos de 1.

```
for(i = 100; i >= 1; i--)
```

c) modifica la variable de control de 7 a 77 en incrementos de 7.

```
for(i = 7; i <= 77; i += 7)
```

d) modifica la variable de control de 20 a 2 en decrementos de -2.

```
for(i = 20; i >= 2; i -= 2)
```

e) modifica la variable de control de 2 a 20 en incrementos de 3.

```
for(i = 2; i <= 20; i += 3)
```

f) modifica la variable de control de 99 a 0 en decrementos de -11.

```
for(i = 99; i >= 0; i -= 11)
```


Incremento

- `for(int num = 1; num <= 10; System.out.println(num), num++);`
- `for(int num = 1; num <= 10; System.out.println(num++));`
- `for(int num = 1; num <= 10; System.out.println(++num));`

Bucles FOR infinito y el uso de BREAK

- Bucle infinito

```
for( ; ; ) { /* Bucle infinito */  
}
```

```
for( ; ; );  
/* Bucle infinito */
```

- La sentencia BREAK, permite detener un bucle infinito.

```
• for(int num = 1; ; num++) {  
    System.out.println(num);  
    if (num == 10) {  
        break;                // Salimos del bucle  
    }  
}
```

Sentencia BREAK sin ETIQUETA

- BREAK Sin Etiqueta, nos permite salir de un bloque de comandos:

- ```
for(int i = 1; i <= 3; i++) {
 for(int j = 1; j <= 3; j++) {
 System.out.print (i + "" + j);
 if (i == j) {
 break; // Comando Break sin etiqueta
 }
 System.out.print("\t");
 }
 System.out.println();
}
```

# Sentencia BREAK con ETIQUETA

- Podemos usar la sentencia BREAK con Etiqueta o Sin Etiqueta:

- AVISO: *// esta es una etiqueta permitida en JAVA*

```
for(int i = 1; i <= 3; i++) {
 for(int j = 1; j <= 3; j++) {
 System.out.print(i + "" + j);
 if (i == j) {
 break AVISO; // Este break nos lleva al fin de AVISO
 }
 System.out.print("\t");
 }
 System.out.println();
} // Aqui termina la etiqueta AVISO
```

# Errores en el uso de BREAK

- lab1:

```
{
 int i = 10;
 if (i == 10)
 break lab1;
}
```

lab2:

```
{
 int i = 10;
 if (i == 10)
 break lab1;
}
```

# Sentencia CONTINUE en un bucle FOR

- Se puede utilizar dentro de bucles FOR para saltar todos los comandos siguientes y ejecutar el Incremento.

- ```
for (int i = 1; i < 10; i += 2) {  
    System.out.println(i);  
}
```

- ```
for (int i = 1; i < 10; i++) {
 if (i%2==0)
 continue;
 System.out.println(i);
}
```

# Sentencia CONTINUE con Etiqueta en un bucle FOR

- AVISO2: // Etiqueta valida en JAVA

```
for(int i = 1; i <= 3; i++) {
 for(int j = 1; j <= 3; j++) {
 System.out.print(i + "" + j);
 System.out.print("\t");
 if (i == j) {
 System.out.println();
 continue AVISO2; // salida de los dos bucles
 }
 }
}
} // FIN de la etiqueta AVISO2
```

# Sentencia CONTINUE en un bucle WHILE

- Se puede utilizar dentro de bucles WHILE para saltar todos los comandos siguientes y volver a evaluar la Condicion.

- ```
int i = 1;
while (i < 10) {
    if (i%2==0) {
        i++;
        continue;
    }
    System.out.println(i);
    i++;
}
```


EJERCICIO

Escriba un programa para encontrar todos los divisores comunes de tres números.

Ejemplo:

Teclee un número: 16,8,10

Los divisores comunes son: 1,2

ejercicio

Un número es perfecto si es igual a la suma de sus divisores, por ejemplo 6 es perfecto porque $6 = 1 + 2 + 3$. Escriba un programa para encontrar todos los números perfectos entre 1 y 10000.

ejercicio

Escriba un programa que despliegue la siguiente tabla de multiplicar. Asegúrese que las columnas estén bien alineadas.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
...									
10	20	30	40	50	60	70	80	90	100