ESTRUCTURAS REPETITIVAS - BUCLES

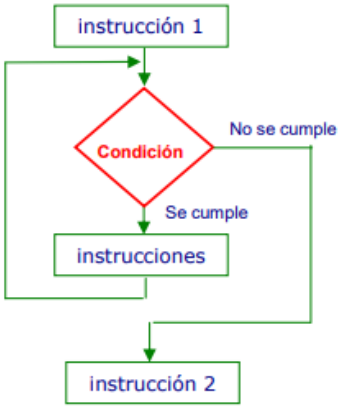
- Permiten ejecutar de forma **repetida** un bloque de código formado por una o más instrucciones.
- Las instrucciones se repiten mientras que se cumpla una determinada condición, llamada **condición de salida** del bucle.

En **Java,** las estructuras repetitivas disponible son:

- **while**
- **do .. while**
- **for**
    - Variante **for-each** para recorrer **Arrays**
    - Objeto **Iterator** para recorrer **Colecciones**.

BUCLE WHILE

- Las instrucciones se repiten mientras la condición de entrada al bucle sea cierta.
- En un bucle **while** la condición **se comprueba al principio, antes de la 1ª iteración** por lo que las instrucciones contenidas en el bloque de código dentro del bucle se **ejecutan 0 ó más veces**.



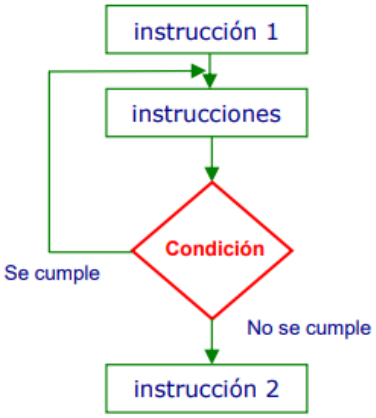

```
while (condición){
    instrucciones;
}
```

La ejecución de un bucle **while** sigue los siguientes pasos:

1. Se evalúa la condición.
2. Si el resultado es false las instrucciones no se ejecutan y se produce la salida del bucle.
3. Si el resultado es true se ejecutan las instrucciones y se vuelve al paso 1

BUCLE DO – WHILE

- Las instrucciones se ejecutan mientras la condición sea cierta.
- La condición **se comprueba al final** del bucle, por lo que el bloque de instrucciones se ejecutará **al menos una vez**.




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

La ejecución de un bucle **do - while** sigue los siguientes pasos:

1. Se ejecutan las instrucciones a partir de la instrucción **do {**
2. Se evalúa la condición.
3. Si la condición no se cumple la ejecución sale del bucle.
4. Si la condición se cumple volvemos al paso 1.

DIFERENCIAS entre **while** y **do .. while**:

- Bucle **while** se ejecuta **0** o más veces.
- Bucle **do .. while** se ejecuta **1** o más veces.

BUCLE FOR

Un **for** hace que una instrucción o bloque de instrucciones se repitan un **número determinado de veces mientras se cumpla la condición**.

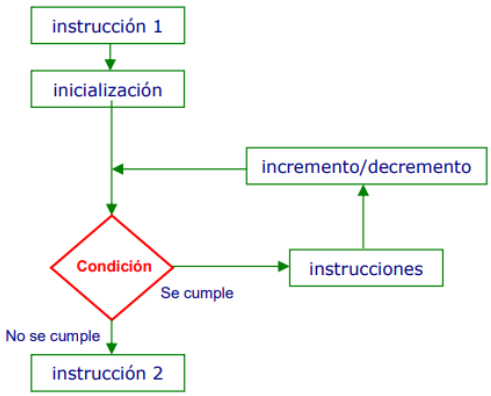
Los bucles **for** son los más adecuados cuando se conoce el número de veces que se van a repetir las instrucciones.

for(inicialización; condición; incremento/decremento){
instrucciones;
}

zona de condición

```
for(inicialización; condición; incremento/decremento){
```

zona de inicialización — zona de incremento ó decremento



Las tres zonas son **opcionales**. Si en alguna ocasión no fuese necesario escribir alguna de ellas se pueden dejar en blanco, pero los **";"** deben aparecer siempre.

Inicialización - es la parte en la que la variable o variables de control del bucle toman su valor inicial. Puede haber una o más instrucciones en la zona de inicialización. Si hay varias instrucciones deben estar separadas por comas. La inicialización **se realiza solo una vez**.

Condición - es una expresión booleana que determina si la sentencia o bloque de sentencias se ejecutan o no. Las instrucciones contenidas dentro del bucle *for* se ejecutan **mientras que la condición sea cierta**.

Incremento/decremento - es una expresión que modifica la variable o variables de control del bucle. En esta zona puede haber más de una expresión para modificar las variables. Si hay varias expresiones deben estar separadas por comas.

- Al Igual que el bucle *while*, **un bucle for se puede ejecutar 0 ó más veces**.

Ejemplo: Programa que muestra por pantalla los números del 1 al 10.

```
public class Ejemplo1While {
   public static void main(String[] args) {
      int i = 1;
      while (i<=10) {
         System.out.print(i + " ");
         i++;
      }
   }
}

public class Ejemplo1DoWhile {
   public static void main(String[] args) {
      int i = 1;
      do {
         System.out.print(i + " ");
         i++;
      } while (i<=10);
   }
}

public class Ejemplo1For {
   public static void main(String[] args) {
      for (int i = 1; i <= 10; i++) {
         System.out.print(i + " ");
      }
   }
}
```

Ejemplo Inverso: Programa que muestra por pantalla los números del 10 al 1.

```
public class Ejemplo2While {
   public static void main(String[] args) {
      int i = 10;
      while (i>=1) {
         System.out.print(i + " ");
         i--;
      }
   }
}

public class Ejemplo2DoWhile {
   public static void main(String[] args) {
      int i = 10;
      do {
         System.out.print(i + " ");
         i--;
      } while (i>=1);
   }
}

public class Ejemplo2For {
   public static void main(String[] args) {
      for (int i = 10; i>=1; i--) {
         System.out.print(i + " ");
      }
   }
}
```

Ejemplo: Programa que muestra por pantalla los números de 1 al N (introducido por teclado).

```
import java.util.Scanner;
public class Ejemplo3While {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Teclea un número: ");
      int n = sc.nextInt();
      int i = 1;
      while (i <= n) {
         System.out.print(i + " ");
         i++;
      }
   }
}
```

```
import java.util.Scanner;
public class Ejemplo3DoWhile {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Teclea un número: ");
      int n = sc.nextInt();
      int i = 1;
      do {
         System.out.print(i + " ");
         i++;
      } while (i <= n);
   }
}
```

```
import java.util.Scanner;
public class Ejemplo3For {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Teclea un número: ");
      int n = sc.nextInt();
      for (int i = 1; i <= n; i++) {
         System.out.print(i + " ");
      }
   }
}
```

Ejemplo: Programa que muestra por pantalla los números de 1 al N (introducido por teclado). En primer lugar en una fila los impares y justo debajo, en otra fila, los pares.

```
import java.util.Scanner;

public class Ejemplo4While {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Teclea un número: ");
      int n = sc.nextInt();
```

WHILE	DO WHILE	FOR

```
    int i = 1;
    while (i <= n) {
       System.out.print(i + " ");
       i+=2;
    }
    System.out.println();
    i=2;
    while (i <= n) {
       System.out.print(i + " ");
       i+=2;
    }
  }
}
```

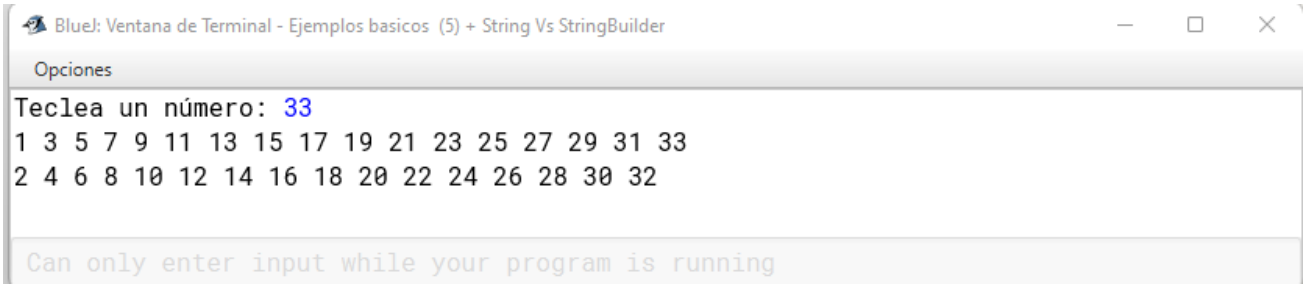
```
int i = 1;
do {
    System.out.print(i + " ");
    i+=2;
}while (i <= n);
System.out.println();
i=2;
do {
    System.out.print(i + " ");
    i+=2;
}while (i <= n);
```

```
for (int i = 1; i <= n; i+=2) {
    System.out.print(i + " ");
}

System.out.println();

for (int i = 2; i <= n; i+=2) {
    System.out.print(i + " ");
}
```

Ejemplo de ejecución:



Ejemplo: programa que pide introducir números enteros por teclado y los va sumando.

Este proceso se repetirá mientras no se introduzca un número negativo, que puede ser incluso el primer número tecleado, en cuyo caso la suma sería 0. Al terminar el programa muestra la suma de todos los números introducidos. **Es un claro ejemplo de bucle while,** pues no sabemos a priori cuántas veces se repetirán las instrucciones, e incluso puede darse el caso de que si el primer número introducido es negativo, el bucle no se ejecute ni una sola vez. Ello nos obliga a realizar la lectura del primer número fuera del bucle.

```
import java.util.Scanner;
public class Ejemplo3While {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int num;
      int suma = 0;
      System.out.print("Introduzca un número positivo (negativo para finalizar): ");
      num = sc.nextInt();

      while (num >= 0){
           suma = suma + num;  //suma +=num
           System.out.print("Introduzca un número positivo (negativo para finalizar): ");
           num = sc.nextInt();
      }

      System.out.println("La suma de los números introducidos es: " + suma );
   }
}
```

BUCLES ANIDADOS

Se pueden anidar bucles, tanto del mismo tipo como de distinto tipo. Es frecuente hacerlo para trabajar con estructuras de datos de más de una dimensión, como las matrices. Es importante que los bucles internos estén completamente embebidos en los bucles externos, sin solapamientos. De esa forma para cada "vuelta" o iteración del bucle externo, el bucle interno realizara todas las "vueltas" o iteraciones que le correspondan

```
for (inicialización; condición; incremento/decremento){
    for (inicialización; condición; incremento/decremento){

    }
}

while (condición){
    while (condición){

    }
}

do {
    do {

    } while(condición);
} while(condición);
```

Ejemplo: programa que dibuja un rectángulo formado por asteriscos. El número de filas y columnas del rectángulo se pide por teclado. El número de filas y de columnas debe ser > 1.

```
import java.util.Scanner;
public class Ejemplo1BuclesAnidados {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int filas, columnas;

      do{
        System.out.print("Teclea número de filas de \"*\" : ");
        filas = sc.nextInt();
      }while(filas < 2);

      do{
        System.out.print("Teclea número de columnas de \"*\" : ");
        columnas = sc.nextInt();
      }while(columnas < 2);

      for(int i = 1; i <= filas; i++){
         for(int j = 1; j <= columnas; j++){
            System.out.print(" * ");
         }
         System.out.println();
      }
   }
}
```

- En este programa se han anidado dos bucles **for** para mostrar el rectángulo de asteriscos.
- El bucle **for** externo es el que corresponde a las filas. El **for** interno corresponde a las columnas.
- Para cada una de las filas se ejecuta completamente el **for** de las columnas.
- Al final de cada fila se escribe un salto de línea para que la siguiente fila comience a mostrarse en la línea siguiente.