UD02.2.Estructuras de Control. Bucles Apuntes

DAM1-Programación 2024-25

- Estructuras de control en Java

Estructura Repetitiva o Iterativa

Esta estructura permite repetir la ejecución de una instrucción o bloque de instrucciones en función del valor de una expresión lógica que ha de evaluarse. Estas estructuras también se conocen como **bucles** o ciclos, y las instrucciones que se reptiten se conocen como **cuerpo del bucle**. Cada ejecución o repetición del bucle también se denomina **iteración**. La expresión lógica que determina si se ejecuta o no el bucle también se denomina **condición de salida**.

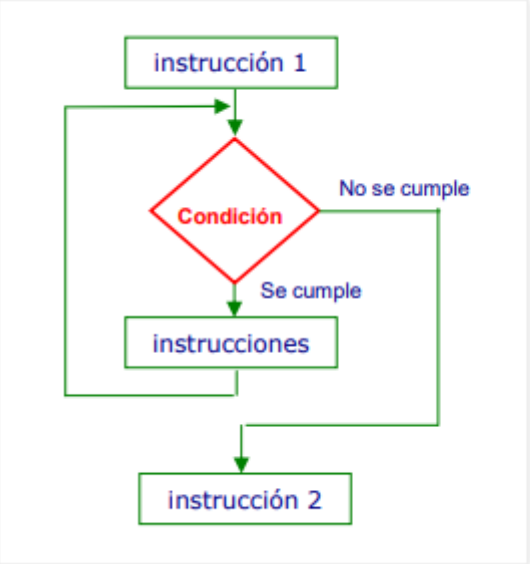
Java implementa 3 tipos de bucles básicos:

- while
- do .. while
- for

Los bucles de los dos primeros tipos se denominan también **bucles controlados por condición** y los del tercero **bucles controlado por contador**.

Un error habitual en bucles consiste en no implementar adecuadamente la condición de salida de modo que el bucle se repita para siempre, lo que se conoce como **bucle infinito**.

Bucle while



Sintaxis:

```
instrucción 1;
while (condición){ //inicio while
        instrucciones;
}  //fin while
instrucción 2;
```

- Las instrucciones se ejecutan **mientras** la condición sea cierta.
- La condición se evalúa al inicio.
- El cuerpo del bucle se ejecuta cero o más veces.

Veamos un ejemplo de un bucle `while` que nunca llega a ejecutarse:

```
int cuentaAtras = -8; //valor negativo
   while (cuentaAtras >= 0) {
   ...
}
```

En este caso, independientemente del bloque de instrucciones asociado a la estructura `while`, no se llega a entrar nunca en el bucle, debido a que la condición no se cumple ni siquiera la primera vez. Se realizan cero iteraciones. En cambio,

```
int cuentaAtras = 10;
while (cuentaAtras >= 0) {
   System.out.println(cuentaAtras);
}
```

Dentro del bloque de instrucciones no hay nada que modifique la variable `cuentaAtras`, lo que hace que la condición permanezca idéntica, evaluándose siempre `true` y haciendo que el bucle sea infinito.

Actividad propuesta 3.1

Diseña una aplicación que muestre la edad máxima y mínima de un grupo de alumnos. El usuario introducirá las edades y terminará escribiendo un –1.

Técnica de lectura adelantada o anticipada.

Se utiliza cuando no sabemos a priori cuántas veces se repetirán las instrucciones. lectura. Por ejemplo, cuando leemos números de teclado hasta que el usuario introduzca un número negativo.

Ejemplo: Programa que sume números hasta introducir uno negativo, que no se sumará.

```
int num;
int suma = 0;  // acumulador números

// Lectura adelantada
System.out.print("Número ( < 0 para finalizar): ");
num = sc.nextInt();

while (num >= 0){ //inicio del bucle while
        suma += num;

        //lectura del siguiente número
        System.out.print("Número ( < 0 para finalizar): ");
        num = sc.nextInt();
}
```

Bucle controlado por contador

En ocasiones utilizamos una variable, denominada contador o **variable de control**, para controlar el número de repeticiones de un bucle.

Ejemplo: Un bucle que se repite **n** veces

```
int n = 10
int contador = 0;

while (contador < n){
        System.out.print(contador);
        contador++;
}

// Imprime los números de 0 a 9
```

Resuelve los siguientes ejercicios sobre bucles en un nuevo paquete:

`ud2.ejerciciosbucles`

E0301. Diseñar un programa que muestre, para cada número introducido por teclado, si es par, si es positivo y su cuadrado. El proceso se repetirá hasta que el número introducido sea 0.

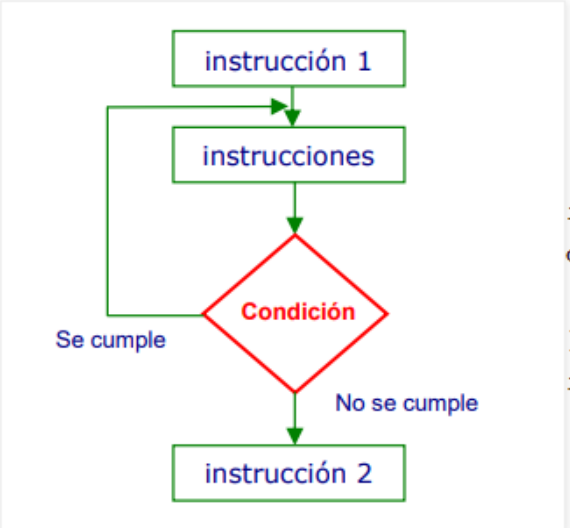
E0302. Implementar una aplicación para calcular datos estadísticos de las edades de los alumnos de un centro educativo. Se introducirán datos hasta que uno de ellos sea negativo, y se mostrará: la **suma** de todas las edades introducidas, la **media**, el **número** de alumnos y **cuántos son mayores de edad**.

E0303. Codificar el juego “el número secreto”, que consiste en acertar un número entre 1 y 100 (generado aleatoriamente). Para ello se introduce por teclado una serie de números, para los que se indica: “mayor” o “menor”, según sea mayor o menor con respecto al número secreto. El proceso termina cuando el usuario acierta o cuando se rinde (introduciendo un -1).

Amplía el programa para que muestre el número secreto cuando el usuario se rinda y el número de intentos cuando haya acertado.

E0304. Un centro de investigación de la flora urbana necesita una aplicación que muestre cuál es el árbol más alto. Para ello introducirá por teclado la altura (en centímetros) de cada árbol (terminando la introducción de datos cuando se utilice el -1 como altura). Los árboles se identifican mediante etiquetas con números únicos consecutivos, comenzando en 0. Diseñar una aplicación que, al terminar la introducción de datos, muestre el número y la altura del árbol más alto.

Bucle do-while



Sintaxis do .. while:

```
instrucción1;
do{  //inicio do .. while
    instrucciones;
}while(condición);  //fin do .. while
instrucción2;
```

- Las instrucciones se ejecutan **mientras** la condición sea cierta.
- La condición se evalúa al final.
- El cuerpo del bucle se ejecuta al menos una vez, es decir, una o más veces.

El bucle do-while es el único cuya sintaxis en Java termina en punto y coma (;).

E0305. Desarrollar un juego que ayude a mejorar el cálculo mental de la suma. El jugador tendrá que introducir la solución de la suma de dos números aleatorios comprendidos entre 1 y 100. Mientras la solución sea correcta, el juego continuará. En caso contrario, el programa terminará y mostrará el número de operaciones realizadas correctamente.

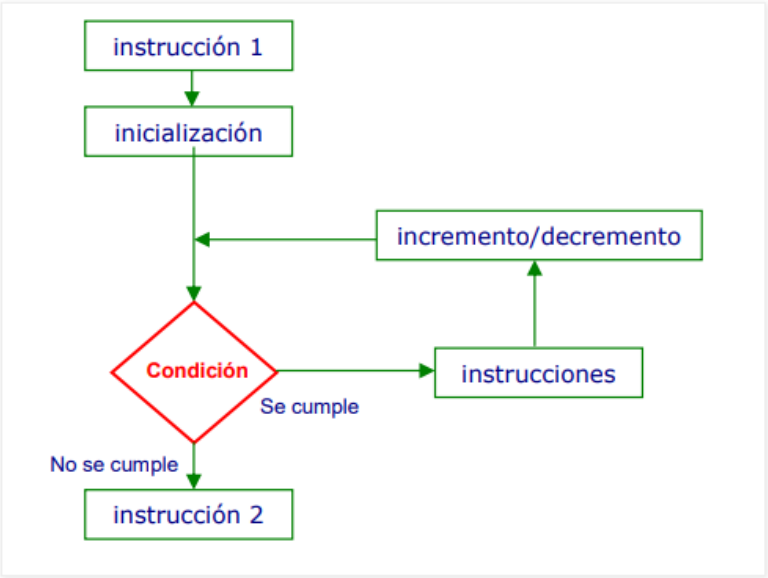
Amplía el programa para que muestre el número de aciertos al terminar.

Variante. Permite al usuario un número limitado de fallos.

La principal diferencia entre el bucle `while` y el `do-while` es que éste último se ejecuta, al menos, una vez. Además, en cuanto a la sintaxis, es el único que termina en punto y coma (`;`).

Bucle for

Este bucle está especialmente diseñado para resolver problemas de bucles controlados por contador.




```
instrucción1;

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

     instrucciones;

}   //fin for

instrucción2;
```

- **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. Se evalúa por primera vez después de ejecutar la inicialización y antes de la primera ejecución del cuerpo del bucle, que podría no llegar a ejecutarse.
- **Incremento/decremento** es una expresión que modifica la variable o variables de control del bucle. Se ejecuta al final de cada iteración, después del cuerpo del bucle. En esta zona puede haber más de una expresión para modificar las variables. Si hay varias expresiones deben estar separadas por comas.

Las tres zonas son opcionales. Si en alguna ocasión no fuese necesario escribir alguna de estas zonas se pueden dejar en blanco, pero los punto y coma deben aparecer.

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

Veamos un ejemplo donde solo se usa la variable `i` para controlar el bucle:

```
for (int i = 1; i <= 2; i++) {
   System.out.println("La i vale " + i);
}
```

Argot técnico

En este caso, la variable `i`, además de inicializarse, también se declara en la zona de inicialización. Esto significa que `i` solo puede usarse dentro de la estructura `for`.

E0306. Escribir una aplicación para aprender a contar, que pedirá un número n y mostrará todos los números del 1 al n.

E0307. Escribir todos los múltiplos de 7 menores que 100.

E0308. Pedir diez números enteros por teclado y mostrar la media.

Amplía codificando el número de números como una constante.

E0309. Implementar una aplicación que pida al usuario un número comprendido entre 1 y 10. Hay que mostrar la tabla de multiplicar de dicho número, asegurándose de que el número introducido se encuentra en el rango establecido.

E0310. Diseñar un programa que muestre la suma de los 10 primeros números impares.

E0311. Pedir un número y calcular su factorial. Por ejemplo el factorial de 5 se denota 5! y es igual a 5 x 4 x 3 x 2 x 1 = 120.

E0312. Pedir 5 calificaciones de alumnos y decir al final si hay algún suspenso.

E0313. Dadas 6 notas, escribir la cantidad de alumnos aprobados, condicionados (nota igual a 4) y suspensos.