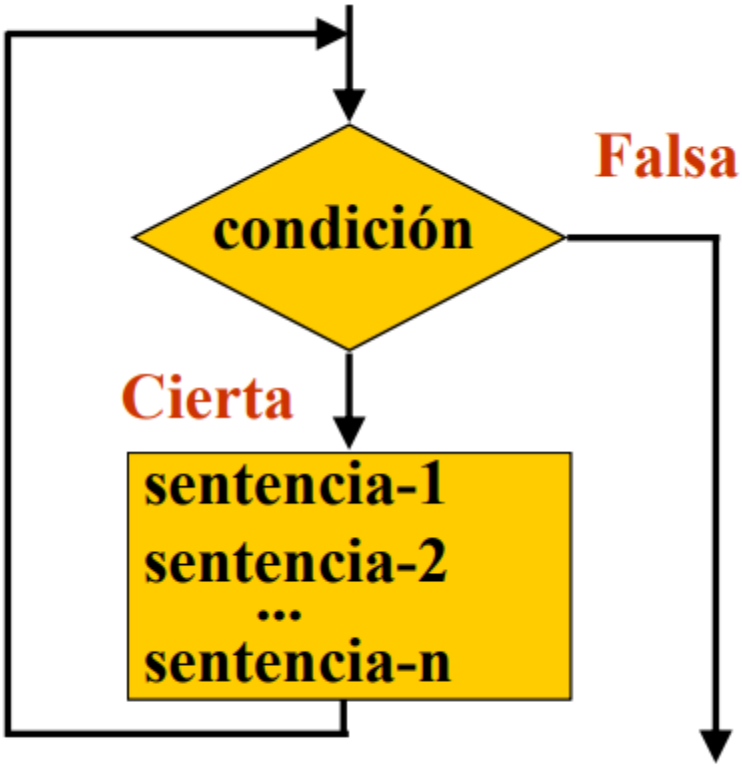
Program Control Statements. Part 3

iterativa while:Ejecuta el bloque de sentencias mientras la condición se evalúe a cierta
Esta estructura de control permite repetir o iterar el [Bloque de Instrucciones] mientras la condición sea verdadera o dicho de otra manera, estas iteraciones terminarán cuando la condición sea falsa y entonces se podrá ejecutar la siguiente instrucción después del while.
señalar que en esta instrucción la condición se evalúa al inicio del ciclo por lo que si la primera vez que se evalúa la condición esta es falsa, el ciclo no llegará a realizarse.



1. **The while Loop**

```
while(condition)
    statement;
```

The general form of the **while** loop is:

```
while(condition)
{
    statement;
}
```

Ejemplo:

```
class WhileDemo {
   public static void main(String args[]) {
      char ch;
      ch = 'a';
      while (ch <= 'z') {
         System.out.print(ch);
         ch++;
       }
    }
}
```

Ejercicios:

i. Programa que lee números mientras la suma de todos ellos no llegue a 100. Muestra al final cuántos números se han introducido .

ii. Programa que lee de manera consecutiva números hasta que se introducen dos números iguales seguidos.

iii. Programa que lee 11 números comprendidos entre el 20 y 40, ambos inclusive. En el caso en el que algún número introducido no se encontrara dentro del rango permitido, se mostrará un mensaje de error y seguirá pidiendo números. El programa finalizará cuando se hayan introducido 11 números válidos y mostrará el número máximo introducido de los 11 válidos, y además nos indicará cuántos números de los introducidos no han sido válidos.

2. **The do-while Loop** (mientras se cumpla la condición se repite el bucle)

La estructura de control do/while es otra sentencia de iteración en la que la condición se evalúa por primera vez una vez que se haya ejecutado el [Bloque de Instrucciones] y tienen la siguiente sintaxis:

```
do {
        statements;
} while(condition);
```

Una vez realizada la evaluación si el resultado es true, se vuelve a ejecutar el [Bloque de Instrucciones], en caso contrario se ejecutan las instrucciones que sigan a continuación saliendo del ciclo

Ejemplo:

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

```
        char ch;
        do {
            System.out.print("Press a key followed by ENTER: ");
            ch=sc.next().charAt(0);
        } while (ch != 'q');
    }
}
```

Ejercicios:

iv. Programa que lea un carácter a través del teclado hasta que éste sea una letra mayúscula.

v. Programa consistente en un juego en el que el Jugador1 introduce un número entre el 1 y el 100 y el Jugador2 tendrá que adivinarlo en un máximo de 10 intentos. Por cada intento del Jugador2, y en el caso de no acertar, se mostrará el mensaje "Te pasaste", o "No llegaste", según corresponda. De la misma manera, y al terminar el programa, se mostrarán los mensajes "Acertaste" o bien "Intentos agotados".

3. **Use break to Exit a Loop**
Dentro de la iteracion en un bucle, de cualquiera de los tipos (while, do-while, for), el uso de esta sentencia rompe la iteracion de dicho bucle.

Por ejemplo

```
for(int j = 0; j<10; j++){
   sentencia 1;
   sentencia 2;
   sentencia 3;
   break;
};
```

Este bucle debería ejecutarse 10 veces, desde j = 0 hasta j = 9, sin embargo la utilización de la sentencia break, rompe la iteración del bucle, de tal manera que tras la primera ejecución el bucle acaba habiéndose ejecutado una sola vez.

```
class BreakDemo1 {
    public static void main(String args[]) {
        int num;
        num = 100;
        for (int i = 0; i < num; i++) {
            if (i * i >= num) {
                break; // terminate loop if i*i >= 100
            }
            System.out.print(i + " ");
        }
        System.out.println("Loop complete.");
```

```
    }
}


import java.util.Scanner;
class BreakDemo2 {
    public static void main(String args[]){
        Scanner sc= new Scanner(System.in);
        char ch;
        System.out.println("Write characters, for finish type q");
        for (;;) {
            ch=sc.next().charAt(0);
            if (ch == 'q') {
                break;
            }
        }
        System.out.println("You pressed q!");
    }
}

class BreakDemo3 {
    public static void main(String args[]) {
        for (int i = 0; i < 3; i++) {
            System.out.println("Outer loop count: " + i);
            System.out.print(" Inner loop count: ");
            int t = 0;
            while (t < 100) {
                if (t == 10) {
                    break; // terminate loop if t is 10
                }
                System.out.print(t + " ");
                t++;
            }
            System.out.println();
        }
        System.out.println("Loops complete.");
    }
}
```

4. **Use continue**
   continue sirve para salir de la iteración actual y saltar directamente a la siguiente iteración, provocando que se ejecute la siguiente iteracion de dicho bucle, ignorando las sentencias posteriores a "continue"
   El caso normal de uso de continue es cuando dentro de un bucle estamos haciendo muchas comprobaciones y en caso de que se cumpla alguna no hace falta seguir comprobando el resto pero no queremos meter unas comprobaciones anidadas dentro de otras para una mayor legibilidad.

La sentencia continue siempre tiene que estar dentro de un bucle porque si no producirá un error de compilación, en cualquier caso no tiene sentido ponerla fuera de un bucle.

```
class ContinueDemo1
{
   public static void main(String args[])
   {
      for (int i = 0; i < 10; i++)
      {
         // Si el número es par
         // omitir y continuar
         if (i%2 == 0)
            continue;

         // Si el número es impar, imprímalo
         System.out.print(i + " ");
      }
   }
}
```

```
class ContinueDemo2 {
   public static void main(String args[]) {
      int i;
      for (i = 0; i <= 100; i++) {
         if ((i % 2) != 0) {
            continue; // iterate
         }
         System.out.println(i);
      }
   }
}
```

Ejercicios de bucles:

vi. Programa que lee cantidades y precios de productos, para al final indicar el total de la factura. Si el importe supera los 1000€ se aplicará un descuento del 5%. El programa finaliza al introducir una cantidad 0.

vii. Programa que genera los N primeros números de la serie de Fibonacci. El valor N deberá ser leído por el teclado. NOTA: Los dos primeros números son 1, y el resto se obtiene sumando los dos anteriores: 1,1,2,3,5,8,13,21…

viii. Programa que pide el número de calificaciones a introducir, y una nota de corte. A continuación se introducirán las calificaciones, para finalmente indicar cuántas calificaciones han igualado o superado la nota de corte.