BREAK Y CONTINUE

Esta instrucción ya la utilizamos en el switch, se usaba para forzar la salida del switch. La instrucción *break* **dentro de un bucle** nos sirve para forzar la salida del bucle.

En el ejemplo de abajo, el uso del break es innecesario ya que bastaría con escribir la condición del bucle como j<=2, pero al margen de esto, ilustra de forma simple el comportamiento del break y damos el ejemplo por bueno. El break fuerza a que el bucle interno se finalice en la mitad de la iteración 3, no llegandose a imprimir el mensaje.

```
class Unidad3 {
    public static void main(String[] args) {
        for(int i=1; i<=3;i++){
            System.out.println("Bucle externo, COMIENZA iteración:"+i);
            for(int j=1;j<=5;j++){
                if(j==3)
                    break;
                System.out.println("\tBucle interno en iteración:"+j);
            }
            System.out.println("Bucle externo, FINALIZA iteración:"+i +"\n");
        }
    }
}
```

El break es muy utilizado en la programación moderna ya que en muchas situaciones simplifica mucho la escritura de bucles y por tanto, aumenta mucho la legibilidad del código. El break no era un recurso disponible en los lenguajes que representan a la "programación estructurada pura" como PASCAL. ,se introdujo posteriormente como una relajación conveniente de los principios de programación estructurada pura. Evidentemente, el uso del break resultará apropiado si se usa para mejorar el código no para complicarlo y empeorarlo.

INSTRUCCIÓN CONTINUE

El *break* se utiliza para salir de un bucle, y el *continue* para salir de una iteración de un bucle. En la práctica hay menos situaciones de uso para *continue* que para break pero no deja de ser un recurso interesante.

Ejemplo: Sustituye en el ejemplo anterior el break por continue y observa como en el bucle interno se fuerza la salida de la iteración 3, pero por el contrario, no se sale del bucle y la iteraciones 4 y 5 se ejecutan con normalidad.

Ejercicio U3_B6_1:

Escribe un bucle que utilizando break detecte el índice del primer carácter blanco. Si no hay espacio en blanco indica "no existe ningún espacio".

```
class Unidad3 {
   public static void main(String[] args) {
      String s = "1110 001";
      int i=0;
      while (.... ) {
         …
      }
      if (i == s.length()) {
         System.out.println("no existe ningún espacio");
      } else {
         System.out.println("Indice buscado : " + i);
      }

   }
}
```

Escribe también una versión sin break

Ejercicio U3_B6_2: Escribe el código equivalente sin break

```
class Unidad3 {

   public static void main(String[] args) {
      int num = 12345;
      int cifras = 0;
      while (num > 0) {
         num /= 10;
         cifras++;
         if (cifras == 5) {
            break;
         }
      }
      if (cifras == 5) {
         System.out.println("El número tiene 5 o más dígitos");
      } else {
         System.out.println("El número tiene menos de 5  dígitos");
      }
   }
}
```

Ejercicio U3_B6_3: Escribe el código equivalente sin break. El programa comprueba si hay una letra 't', o varias en cuyo caso imprime "Sí que hay t"

```
class Unidad3 {
```

```
    public static void main(String[] args) {
        String  frase = "que frase tan bonita";
        for( int i = 0 ; i < frase.length() ; i++ ){
          char caracter=frase.charAt(i);
          if( caracter == 't' ){
             System.out.println("Si hay t");
             break;
          }
        }
    }
}
```

ETIQUETAS.

El break/continue fuerza la salida del switch o del bucle que lo contiene, es decir, del bloque en que se encuentra. Si hay una estructura de bucles anidados y con un break en un bucle interno quiero salir "más afuera" tengo que usar etiquetas. Una etiqueta se coloca antes de la sentencia a etiquetar seguida de ':'

Por ejemplo si en el primer ejemplo ahora quiero salir de ambos bucles al encontrar el break

```
class Unidad3 {
    public static void main(String[] args) {
        mibloque://etiqueta a la siguiente instrucción, es decir, el primer  for
        for(int i=1; i<=3;i++){
            System.out.println("Bucle externo, COMIENZA iteración:"+i);
            for(int j=1;j<=5;j++){
                if(j==3)
                    break mibloque; //indica que hay que salir de la instrucción etiquetada
                System.out.println("\tBucle interno en iteración:"+j);
            }
            System.out.println("Bucle externo, FINALIZA iteración:"+i +"\n");
        }

    }
}
```

Por cierto: este tipo de etiquetas, no tienen nada que ver con las famosas etiquetas que se usaban en combinación con la *instrucción goto* en lenguajes de programación ancestrales que tantos problemas causaron en la historia de la programación.

Ejercicio U3_B6_4: Escribe el ejemplo anterior sin utilizar etiquetas.

ABUSOS DEL BREAK

Se debe usar el break para mejorar la legibilidad del código, no para empeoala. El siguiente ejemplo.

```
class Unidad3{
   public static void main(String[] args) {
         int suma = 0 ;
         for(int i=5;i<11;i++){
            suma+=i;
         }
         System.out.println("Suma : " + suma);
   }
}
```

Pierde legibilidad de la siguiente manera

```
class Unidad3{
   public static void main(String[] args) {
         int suma = 0 ;
         int i;
         for(i=5;;i++){
            if(i>10)
               break;
             suma+=i;
         }
         System.out.println("Suma : " + suma);
   }
}
```

En la cabecera del bucle debe ir siempre la condición "más "general de salida del bucle y debemos usar el break sólo para salidas que consideremos una excepción (aunque siempre es relativo esto de general y excepcional).

Observa que la condición de la cabecera y del break tienen la lógica opuesta en el sentido que en la cabecera decimos cuando entramos en el cuerpo y en el break cuando salimos, en el ejemplo anterior:

- Entramos cuando i<11
- Salimos cuando i>10.

Muchas veces nos es más fácil expresar cuando salimos que cuando continuamos pero debemos pensar en la legibilidad por terceros de nuestro código y por tanto no debemos

omitir la condición de la cabecera del bucle ya que es una importantísima información para dar legibilidad a nuestro código.

Hacer siempre un bucle infinito y salir con un break se puede llegar a convertirse en un vicio. Por ejemplo, sumar n números

```
import java.util.Scanner;

class Unidad3{
   public static void main(String[] args) {
      Scanner sc= new Scanner(System.in);
      int numero=sc.nextInt();
      int suma=0;
      while(numero!=0){
         suma=suma+numero;
         numero=sc.nextInt();
      }
      System.out.println(suma);
   }
}
```

Un viciado del break lo escribiría

```
import java.util.Scanner;

class Unidad3{
   public static void main(String[] args) {
      Scanner sc= new Scanner(System.in);
      int numero=sc.nextInt();
      int suma=0;
      while(true){
         if(numero==0)
            break;
         suma=suma+numero;
         numero=sc.nextInt();
      }
      System.out.println(suma);
   }
}
```

Si se ve un while(true) un programador lo primero que piensa es que el funcionamiento general del programa es que debe estar indefinidamente ejecutándose como un servidor

o similar. Pero aquí no existe tal indefinición de terminación, se sabe de antemano que se va acabar cuando se introduce el número 0.