INSTRUCCIÓN WHILE

Ejercicio U3_B3_1:

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i<11){
          System.out.println(i);
         i++;
      }
   }
}
```

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i<=10){
         System.out.println(i++);
      }
   }
}
```

y muchas otras combinaciones todas correctísimas !!!!

Ejercicio U3_B3_2:

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i<11){
         System.out.println(i+" es par");
         i+=2;
      }
   }
}
```

o bien usando operador módulo para detectar pares, aunque quizá es más elegante, por sencilla, la primera solución

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i<11){
         if(i%2==0)
            System.out.println(i+" es par");
         i++;
      }
   }
}
```

El código anterior admite muchas variantes todas correctas, por ejemplo usando <= en la condición de bucle

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i<=10){
         if(i%2==0)
            System.out.println(i+" es par");
         i++;
      }
   }
}
```

Ejercicio U3_B3_3:

El programa se queda colgado porque el bucle jamás termina. El bucle no termina por que sólo se incrementa cuando i es par, ya que el incremento de i sólo se hace cuando i%2 es cierto

también sería incorrecto (impresiones mal)

```
class Unidad3{
   public static void main(String[] args) {
      int i=0;
      while(i<=10)
         if(i++%2==0)
            System.out.println(i+" es par");
    }
}
```

ojo con los incrementos!!!!!

Ejercicio U3_B3_4

Imprime las letras minúsculas del abcedario utilizando un bucle while.

```
class Unidad3 {
    public static void main(String[] args) {
       char c='a';
       while(c<='z')
          System.out.println(c++);
     }
}
```

Ejercicio U3_B3_5

Ejemplo bucles anidados con incremento de contadores en la condición del while. Como las variables se incrementan en la condición que está antes del println() nos vemos forzados a empezar con i y j a 0. Luego según utilice pre-incremento o post-incremento tendré que cambiar el límite o no

```
public class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(++i<4){
        System.out.println("iteración bucle exterior número " +i);
        int j=0;
        while(++j<3)
          System.out.println("\titeración bucle interior número " +j);
       }
   }
}
```

```
class Unidad3 {
   public static void main(String[] args) {
      int i=0;
      while(i++<3){
         System.out.println("iteración bucle exterior número " +i);
         int j=0;
         while(j++<2){
            System.out.println("\titeración bucle interior número " +j);

         }
```

```
    }
  }
}
```

¿Cómo es la mejor forma de escribir un bucle?
No lo dudes, escribe siempre la forma que entiendas mejor y con la que te sientas más cómodo. Lo más importante en los pasos iniciales es entender a la perfección cómo funciona un bucle, muy por encima de escribirlo "así o asá". A menudo incrementar dentro de una condición puede ser una complicación innecesaria.

Ejercicio U3_B3_6

```
import java.util.Scanner;
public class Unidad3{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
                  System.out.print("Palabra: ");
                  String palabra=sc.next();
      String listaDePalabras="";
      while (!palabra.equals("fin")) {
         listaDePalabras=listaDePalabras+"   "+ palabra;
                        System.out.print("Palabra: ");
                        palabra=sc.next();
      }
                  System.out.print(listaDePalabras);
   }
}
```

Se puede hacer más compacto haciendo la asignación en la propia condición del while

```
import java.util.Scanner;
class Unidad3{
   public static void main(String[] args) {
      Scanner teclado= new Scanner(System.in);
      String palabra;
      String ListaPalabras="";
      System.out.println("Palabra: ");
      while(!(palabra=teclado.next()).equals("fin")){
         ListaPalabras+="  "+palabra;
         System.out.println("Palabra: ");
      }
      System.out.println("lista de palabras introducidas: "+ListaPalabras);
   }
}
```

Para razonar esto tienes que pensar que la sentencia de asignación en este contexto también es una subexpresión, y como toda expresión se evalúa y devuelve un valor. La expresión variable=valor devuelven el valor asignado es decir, la parte derecha. En nuestro ejemplo el string devuelto por next()

Ejercicio U3_B3_7

```
import java.util.Scanner;
class Unidad3{
  public static void main(String[] misArgumentos){
     System.out.println("teclea palabras separadas por delimitador");
     Scanner sc = new Scanner(System.in);
     int i=0;
     while(sc.hasNext()){
        sc.next();
        i++;
     }
     System.out.println("palabras introducidas: "+i);
  }
}
```