

Tema 6 – S1

Contenidos:

1. Introducción a la iteración
2. La iteración como estrategia de diseño: fases y elementos
 - Estrategias para descubrir la estructura iterativa del problema
 - Mecánica y Semántica del bucle `while`
 - Ejemplos y ejercicios



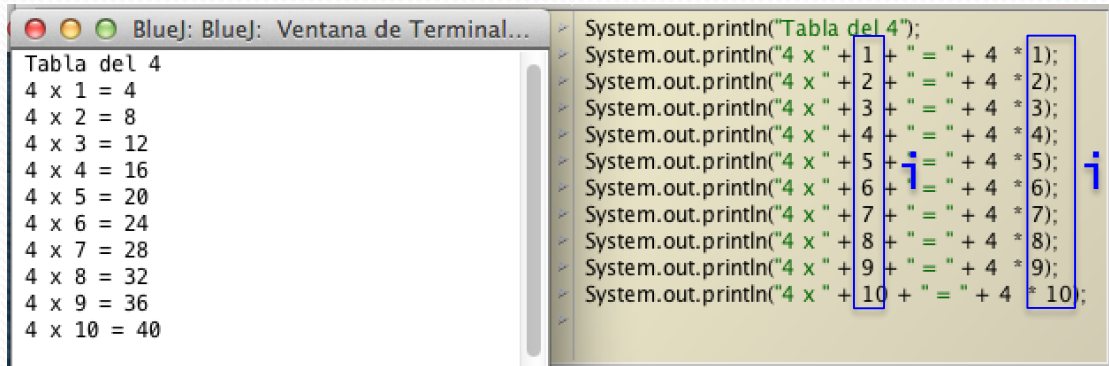
BlueJ: ejemplos S1 – Tema 6

- **Descarga** (desde mi carpeta Tema 6 de PoliformaT) y **descomprime** el proyecto *BlueJ ejemplos S1 – Tema 6*
- **Abre el proyecto** (clic en el icono de BlueJ) y prepárate para usarlo

La necesidad de repetir instrucciones y de la instrucción “repetir”

Observa el resultado que aparece en el *Terminal de BlueJ* cuando se ejecuta el código del *Code Pad* para los ejemplos que siguen. Luego, responde (por escrito) las cuestiones que se plantean para cada ejemplo

1. Mostrar por pantalla la tabla de multiplicar del 4

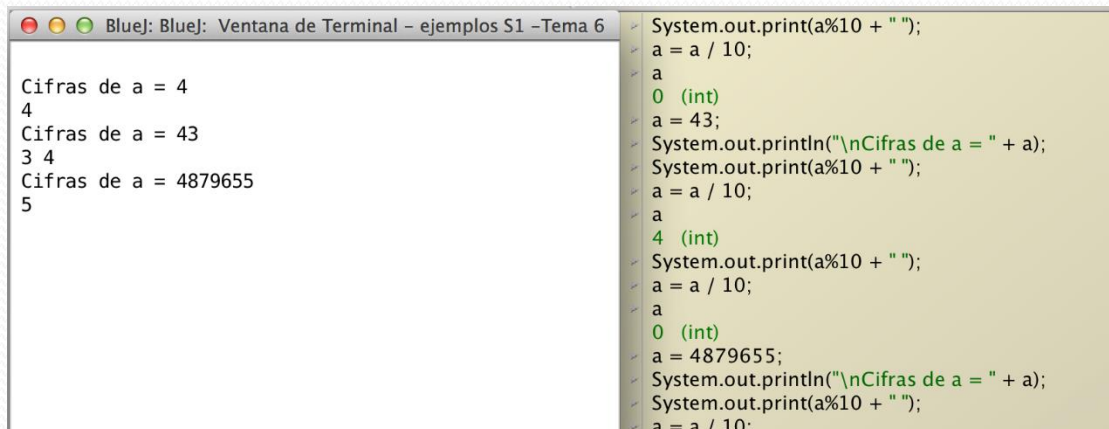


The screenshot shows a BlueJ terminal window titled "BlueJ: BlueJ: Ventana de Terminal...". The left pane displays the output of a Java program: a multiplication table for the number 4, listing products from 4 x 1 to 4 x 10. The right pane shows the Java code used to generate this output, which consists of a series of `System.out.println` statements. Each statement concatenates the string "Tabla del 4", the number 4, the multiplication sign, the number 4, the variable `i`, the equals sign, the number 4, the multiplication sign, and the variable `i`. The variable `i` is highlighted with a blue box in both the code and the output, indicating its role in the repetition.

```
System.out.println("Tabla del 4");
System.out.println("4 x " + 1 + " = " + 4 * 1);
System.out.println("4 x " + 2 + " = " + 4 * 2);
System.out.println("4 x " + 3 + " = " + 4 * 3);
System.out.println("4 x " + 4 + " = " + 4 * 4);
System.out.println("4 x " + 5 + " = " + 4 * 5);
System.out.println("4 x " + 6 + " = " + 4 * 6);
System.out.println("4 x " + 7 + " = " + 4 * 7);
System.out.println("4 x " + 8 + " = " + 4 * 8);
System.out.println("4 x " + 9 + " = " + 4 * 9);
System.out.println("4 x " + 10 + " = " + 4 * 10);
```

$\forall i: 1 \leq i \leq 10: \{ \text{system.out.println}("4 \times " + i + " = " + 4 * i); \}$

2. Mostrar por pantalla las cifras de `a`, un int positivo dado



The screenshot shows a BlueJ terminal window titled "BlueJ: BlueJ: Ventana de Terminal - ejemplos S1 - Tema 6". The left pane displays the output of a Java program: the digits of the number 4, 43, and 4879655, each on a new line. The right pane shows the Java code used to generate this output. The code uses a loop to repeatedly print the last digit of a number `a` (using `a % 10`) and then divide `a` by 10 (using `a = a / 10`) until `a` becomes 0. The variable `a` is highlighted with a blue box in both the code and the output, indicating its role in the repetition.

```
System.out.print(a%10 + " ");
a = a / 10;
a
0 (int)
a = 43;
System.out.println("\nCifras de a = " + a);
System.out.print(a%10 + " ");
a = a / 10;
a
4 (int)
System.out.print(a%10 + " ");
a = a / 10;
a
0 (int)
a = 4879655;
System.out.println("\nCifras de a = " + a);
System.out.print(a%10 + " ");
a = a / 10;
```

$\forall a: a \neq 0: \{ \text{system.out.println}(a \% 10); a = a / 10; \}$

Cuestiones:

- ¿Qué instrucción se repite?
- ¿Cuántas veces?
- ¿Sería fácil escribir el código para mostrar la tabla del 4 hasta el 4 x 200? ¿Y la del 896?

Para cada valor de `a`, ...

- ¿Qué instrucciones se repiten?
- ¿Cuántas veces?
- ¿Variaría alguna de tus respuestas anteriores si `a` se leyera de teclado?

Introducción: algoritmo general para la iteración

```
inicializar_variables_Iteración // variables contador y acumulador (del resultado)
repetir mientras (condición) {
    instrucciones_Iteración; // actualizar variables contador y acumulador
}
```

```
INICIO // variables contador y acumulador (del resultado)
repetir mientras ( GUARDA ) {
    CUERPO // actualizar variables contador y acumulador
}
```

Usar una iteración/bucle **añade 2 tareas de validación** al programador ...

1. Demostrar que el nº de repeticiones que realiza es finito (**Terminación**)
2. Demostrar que el nº de repeticiones que realiza es el menor posible (**Eficiencia**)

Bucles Java: traducciones del algoritmo general

Según el problema, el programador Java puede elegir usar ...

<pre>INICIO while (GUARDA) { CUERPO }</pre>	<pre>INICIO do { CUERPO } while (GUARDA);</pre>
---	---

<pre>INICIO acumulador for (INICIO contador GUARDA actualizar contador) { actualizar acumulador }</pre>

La iteración como estrategia de diseño

Fases

1. Descubrir la *estructura iterativa* del problema, o **Estrategia Iterativa**



2. Determinar los elementos (*cuerpo, inicio y guarda*) de una iteración **CORRECTA**



3. Traducir la *iteración* diseñada a *bucle Java*



4. Comprobar la *corrección y terminación* del bucle diseñado

La iteración como estrategia de diseño

¿Cómo aprender a partir de ejemplos la instrucción a repetir?

(a) ¿Cuál es el (valor del) siguiente término?

Observa la solución para la primera serie, por si te ayuda a resolver las siguientes

1, 1, 2, 6, 24, ...	120
1, 2, 4, 8, 16, 32, ...	64
0, 1, 3, 6, 10, 15, ...	21

(b) ¿Qué relación general existe entre un término n y el anterior a él?

Observa la solución para la primera serie, por si te ayuda a resolver las siguientes

1, 1, 2, 6, 24, ... 120	si $n > 0$, $n * (n - 1)!$; si $n = 0$, $(0)! = 1$
1, 2, 4, 8, 16, 32, ...	si $n > 0$, $2 * 2^{n-1}$; si $n = 0$, $2^0 = 1$
0, 1, 3, 6, 10, 15, ...	si $n > 0$, $n + \sum_{i=0 \dots n-1} i$; si $n=0$, $\sum_{i=0} i = 0$

(c) ¿De qué función matemática hablamos?

Observa la solución para la primera serie, por si te ayuda a resolver las siguientes

1, 1, 2, 6, 24, ... 120	$n!$ (<u>factorial de n</u>)
1, 2, 4, 8, 16, 32, ...	2^n
0, 1, 3, 6, 10, 15, ...	$\sum_{i=0 \dots n-1} i$

¿Qué estrategia o tipo de razonamiento has usado para descubrirlo?

¿Se parece a la que empleaste para resolver el puzle de Multiplicación “a la Rusa”

La iteración como estrategia de diseño - Ejemplo 1 (I)

Diseña un método que devuelva el producto de a y b, enteros no negativos, **SIN** usar el operador *

Fase 1a: expresa en Java el enunciado del problema, nombrando sus elementos y condiciones

// **PRECONDICIÓN:** $a \geq 0$ AND $b \geq 0$

```
public static int productoSinUsarX(int a, int b) {
```

```
    int res;
```

```
    /* COMPLETAR: traducir a Java la estrategia iterativa */
```

```
    return res;
```

Repetir a veces $res = res + b$

Fase 1b: descubre “la” estrategia (iterativa) que resuelve el problema...

- Usa la definición matemática de producto: $res = a * b = \sum_{i=1}^{a} b = \underbrace{b + b + \dots + b}_{a \text{ veces}}$
- Alternativamente, “**apréndela a partir de ejemplos**”:

Supón $b = 8$ y... $a = 1 \rightarrow res_{a=1} = 8$ ($1 * 8$) = b (sumado b 1 vez)

$a = 2 \rightarrow res_{a=2} = 16$ ($2 * 8$) = $8 + 8 = res_{a=1} + b$ (sumado b 2 veces)

$a = 3 \rightarrow res_{a=3} = 24$ ($3 * 8$) = $8 + 8 + 8 = res_{a=2} + b$ (sumado b 3 veces)

• • •

○ Cuando $a = i$, ¿podrías formular ya una **HIPÓTESIS** de cuánto vale res?

$a = i \rightarrow res_{a=i} = res_{a=i-1} + b$ (sumado b a veces)

acumulador

contador

La iteración como estrategia de diseño - Ejemplo 1 (II)

Diseña un método que devuelva el producto de a y b, enteros no negativos, **SIN** usar el operador *

Fase 2-3: determinar **cuerpo**, **inicio** y **guarda** **CORRECTOS** de la iteración-bucle **while**

// **PRECONDICIÓN:** $a \geq 0$ AND $b \geq 0$

```
public static int productoSinUsarX(int a, int b) {
```

```
    int res =  ; int i =  ;
```

```
    while (   ) {
```

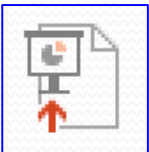
```
         
    }
```

```
    return res;
}
```

Estrategia: Repetir a veces $res = res + b$

IMPORTANTE

- Unimos las fases 2 y 3 porque el bucle **while** y algoritmo iterativo general son **equivalentes**
- **Fijada la estrategia, en** la fase 2-3 hay que resolver un puzzle con la piezas que ves; para que sea **CORRECTO** es imprescindible colocarlas en el **orden adecuado**



Tienes el código en el programa TestSPProductoSinUsarX del **proyecto BlueJ ejemplos S1 – Tema 6**

La iteración como estrategia de diseño - Ejemplo 1 (III)

Diseña un método que devuelva el producto de a y b, enteros no negativos, **SIN** usar el operador *

Fase 4: comprobar la corrección y terminación del bucle diseñado

```
// PRECONDICIÓN: a >= 0 AND b >= 0
public static int productoSinUsarX(int a, int b) {
    int res = 0; int i = 0;    // TRAS repetición nº 0: sumADO 0 veces b
    while (i != a) {
        res = res + b;
        i++;                  // TRAS repetición nº i: sumADO i veces b
    }
    // Fin bucle: ¿i = a AND res = a * b? // TRAS repetición nº a: sumADO a veces b
    return res;
}
```

Por construcción y si se cumple la precondition, el bucle diseñado...

- ✓ **Es correcto**, i.e. hace lo que debe, ni más ni menos
- ✓ **Termina**: i es un número entero, como a, y se incrementa una unidad en cada repetición → No puede tardar más que un tiempo finito en alcanzar su valor máximo a, momento en el que dejará de cumplirse la guarda y el bucle terminará
- ✓ **Es eficiente**: por definición, para multiplicar a por b, se deben realizar siempre a sumas de b, el mismo nº de veces que se ejecuta la instrucción `res = res + b`. Nota que la guarda se ejecuta una vez más, para alcanzar la condición de parada

La iteración como estrategia de diseño

Cuestiones propuestas sobre el Ejemplo 1

1. Dado que $1 * b = b$, independientemente del valor de b , se podría haber pensado en la siguiente estrategia iterativa: **Repetir $a - 1$ veces $res = res + b$** . Diseña el bucle `while` **correcto** que corresponde a esta estrategia En package *sesion1* de *ejercicios* – Tema 6
2. Los siguientes bucles se podrían haber propuesto como alternativas al que se ha diseñado. **El primero es correcto y el segundo NO termina** (bucle infinito). Razona por qué; además, en el caso del primero, indica qué representa el contador i

```
int res = 0, i = a;
while (i != 0) {
    res = res + b;
    i--;
}
```

```
int res = 0, i = 0;
while (i >= 0) {
    res = res + b;
    i++;
}
```

La iteración como estrategia de diseño - Ejemplo 2 (I)

Diseña un método que devuelva la suma de las cifras de a , entero no negativo, **SIN** usar `Math.Log10`

Fase 1a: expresa en Java el enunciado del problema, nombrando sus elementos y condiciones

// **PRECONDICIÓN:** $a \geq 0$

```
public static int sumarCifras(int a) {  
    int res;  
    /* COMPLETAR: traducir a Java la estrategia iterativa */  
    return res;  
}
```

Repetir mientras ($a \neq 0$) : $res = res + a \% 10$; $a = a / 10$;

Fase 1b: descubre “la” estrategia (iterativa) que resuelve el problema...

- A un nivel de abstracción alto, aún con detalles por perfilar, el algoritmo sería...

```
int res = 0;  
repetir mientras (quedanCifrasEn(a)) {  
    int cifraDeA = obtenerCifraDe(a);  
    res = res + cifraDeA;  
    a = quitarCifraDe(a, cifraDeA);  
}
```

acumulador

contador

- Para reducir el nivel de abstracción **la clave** es saber cómo obtener, una tras otra, cada cifra de a . Pero esto NO es difícil porque **sabemos cómo dividir un n° en 2 partes**: su última cifra (n° de unidades) y el resto del n° (el n° sin su última cifra):

`obtenerCifraDe(a)` es la instrucción $a \% 10$;

`quitarCifraDe(a, cifraDeA)` es la instrucción $a / 10$;

`quedanCifrasEn(a)` es la expresión booleana $a \neq 0$

La iteración como estrategia de diseño - Ejemplo 2 (II)

Diseña un método que devuelva la suma de las cifras de a , entero no negativo, **SIN** usar `Math.log10`

Fase 2-3: determinar **cuerpo**, **inicio** y **guarda** **CORRECTOS** de la iteración-bucle `while`

// **PRECONDICIÓN:** $a \geq 0$

```
public static int sumarCifras(int a) {  
    int res = ; int i = ;  
    while (  ) {  
          
    }  
  
    return res;  
}
```

Estrategia - Repetir mientras ($a \neq 0$) { $res = res + a \% 10$; $a = a / 10$; }



RECUERDA QUE fijada la estrategia, en esta fase hay que resolver un puzzle con la piezas que ves; para que sea **CORRECTO** es imprescindible colocarlas en el **orden adecuado**

Tienes el código en el programa `TestSumarCifras` del **proyecto BlueJ ejemplos S1 – Tema 6**

La iteración como estrategia de diseño - Mecánica del while



BlueJ: ejemplos S1 -Tema 6

- **Sitúa**, como en la imagen, un punto de ruptura en la línea 32 del main de TestSumarCifras

Luego, **traza** la ejecución del método `sumarCifras` con ayuda del depurador de BlueJ y **observa** cómo se modifican las variables `a` (contador) y `res` en cada repetición (*Step*), hasta que termine su ejecución. Usa **435**, por ejemplo, como valor de `a`

- **Repite** el proceso anterior usando como argumento un valor negativo, por ejemplo **-435** ... **¿Qué sucede?**

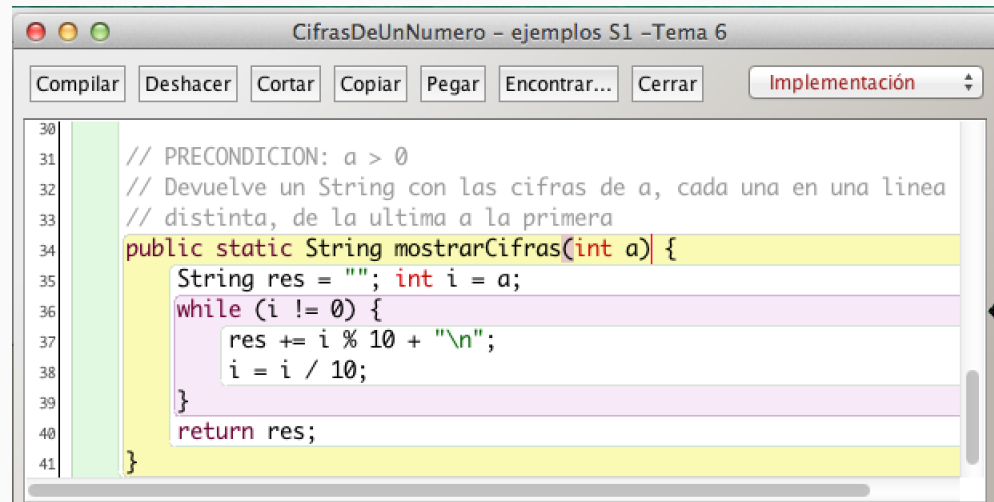
La iteración como estrategia de diseño

Cuestión sobre el Ejemplo 2



BlueJ: ejemplos S1 -Tema 6

- **Edita** el programa CifrasDeUnNumero del proyecto y **observa** el código de su método mostrarCifras, que devuelve el nº de cifras de a, un nº entero positivo



```
30
31 // PRECONDICION: a > 0
32 // Devuelve un String con las cifras de a, cada una en una linea
33 // distinta, de la ultima a la primera
34 public static String mostrarCifras(int a) {
35     String res = ""; int i = a;
36     while (i != 0) {
37         res += i % 10 + "\n";
38         i = i / 10;
39     }
40     return res;
41 }
```

- **Modificando** donde creas necesario la estrategia del ejemplo 2, **escribe** la estrategia seguida para obtener el bucle de mostrarCifras

Luego, en base a la estrategia que has escrito, **indica** por qué el cuerpo, inicio y guarda del bucle de mostrarCifras solo pueden ser los que son

La iteración como estrategia de diseño

Ejercicios propuestos

- **Nº 1 TRANSPARENCIAS:** modificando donde creas necesario la estrategia del Ejemplo 2 (sumar las cifras de a , un nº entero no negativo), diseña un método que devuelva el nº cifras de a En package *sesion1* del proyecto BlueJ *ejercicios – Tema 6*
- Del “**BloC de la Sesión 1**”, disponible en mi carpeta “Tema 6” de Recursos de la PoliformaT...
 - a. Intenta descubrir la estrategia iterativa a seguir para resolver los ejercicios de las páginas 7 y 8 (del *Examen PoliformaT “Actividad Tema 6: sintaxis y semántica del bucle while, ejercicios 2 y 4*)
 - b. Intenta resolver los puzle del ejercicio de la página 9 (una versión del nº 2 del capítulo 8 del libro de la asignatura) También en *sesion1* de *ejercicios – Tema 6*