



Recibe una cálida:

¡Bienvenida!

Te estábamos esperando 😊 

» El entorno Java para la programación

Plan formativo: Desarrollo de Aplicaciones Full Stack Java Trainee V2.0

HOJA DE RUTA

¿Cuáles **skill** conforman el programa?



REPASO CLASE ANTERIOR

En la clase anterior trabajamos :

- ✓ Sentencias condicionales *If*
- ✓ Sentencia condicional *Switch*

LEARNING PATHWAY

4.3

Start! 🏁

El Entorno Java para
la programación

Bucles en Java

Iterando

Sentencias repetitivas

OBJETIVOS DE APRENDIZAJE

¿Qué aprenderemos?



Implementar sentencias repetitivas para controlar el flujo de un programa Java.





Rompehielo

Compartamos experiencias:

Las tareas repetitivas son constantes en la programación, pero también en la vida real. De ésta manera ahorramos pasos de trabajo y simplificamos nuestro día a día.



Consigna: Respondan en el chat o levantando la mano

1. ¿En qué momentos del día a día piensan que están realizando tareas en bucle?
2. ¿Sería más difícil realizar las mismas rutinas si no fuera de manera repetitiva?
3. ¿Cuál es la repetición más habitual entre quienes compartieron? ¿Hay coincidencias?



› Sentencias Repetitivas



Sentencias repetitivas



¿Qué son las sentencias repetitivas?:

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan **bucles**, y se denomina iteración al hecho de repetir la ejecución de una secuencia de acciones.



Todo bucle se encuentra asociado a una condición, que es la que va a determinar cuándo y cuántas veces se repite.



› Bucle while



While



Sintaxis en Java

Esta estructura de control **repite un conjunto de instrucciones mientras una condición se cumpla (true)**, en cuanto la condición no se cumple el bucle deja de ejecutarse. Es por ello que dentro del bloque deberán existir sentencias que modifiquen la evaluación de la expresión, ya que de no hacerse se podría entrar en un bucle infinito.



```
while (<condición>) {  
<sentencias>;  
}
```

En el caso de que la condición se evalúe por primera vez como false, el ciclo no será ejecutado ni una vez.



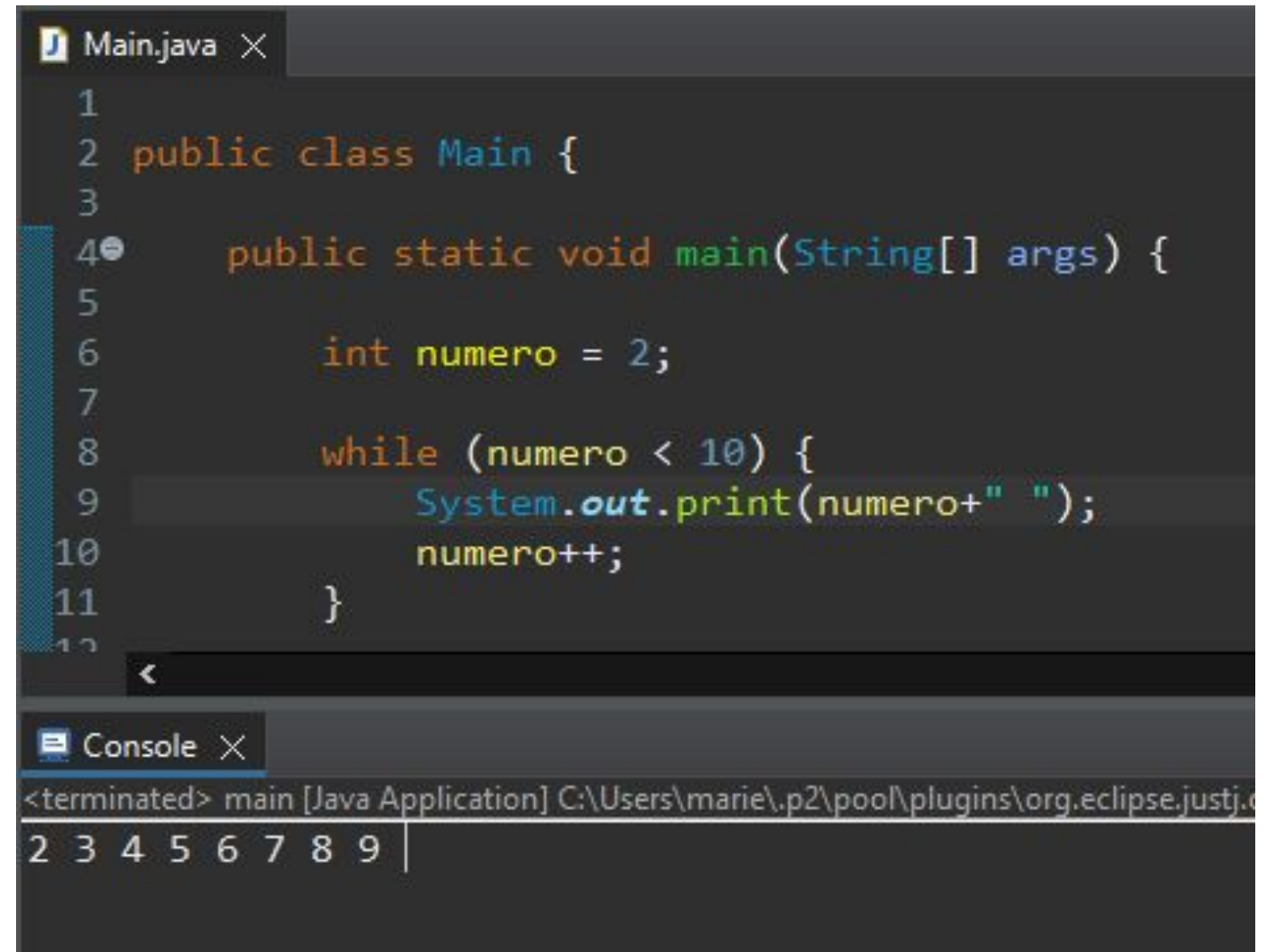
While

Iteración:

En este ejemplo utilizamos la variable **número** como **iterador**. Es por esto que es necesario **incrementar** esta variable (línea 10) para que se cumpla en algún momento la **condición de finalización** del bucle.

Podemos ver en la salida por consola como se imprimen todos los números menores a diez.

Truco: para escribir en una misma línea, se utiliza la función **print** (sin “\n”) del System.out.



```
Main.java X
1
2 public class Main {
3
4     public static void main(String[] args) {
5
6         int numero = 2;
7
8         while (numero < 10) {
9             System.out.print(numero+" ");
10            numero++;
11        }
12    }
13}

Console X
<terminated> main [Java Application] C:\Users\marie\.p2\pool\plugins\org.eclipse.justj.c
2 3 4 5 6 7 8 9 |
```

LIVE CODING

Ejemplo en vivo

while:

Vamos a ver un ejemplo de bucle while dónde la variable iterador será modificada por el usuario.

1. *Desarrollar un bucle while que lea números ingresados por el usuario, hasta que se ingrese 0 (cero).*

 **Tiempo: 15 minutos**

› Bucle do-while



Do-While

Sintaxis en Java

En el caso de la estructura repetitiva do-while el funcionamiento es casi el mismo que el de while. Pero con una diferencia, **primero se ejecuta el bloque de sentencias y luego se evalúa la expresión**. Al igual que en el while, en el bloque de sentencias deberemos modificar la condición para poder salir del bucle.



```
do{  
    <sentencias>;  
}while (<condición>);
```

Por lo tanto, **siempre se ejecutará al menos una vez**.



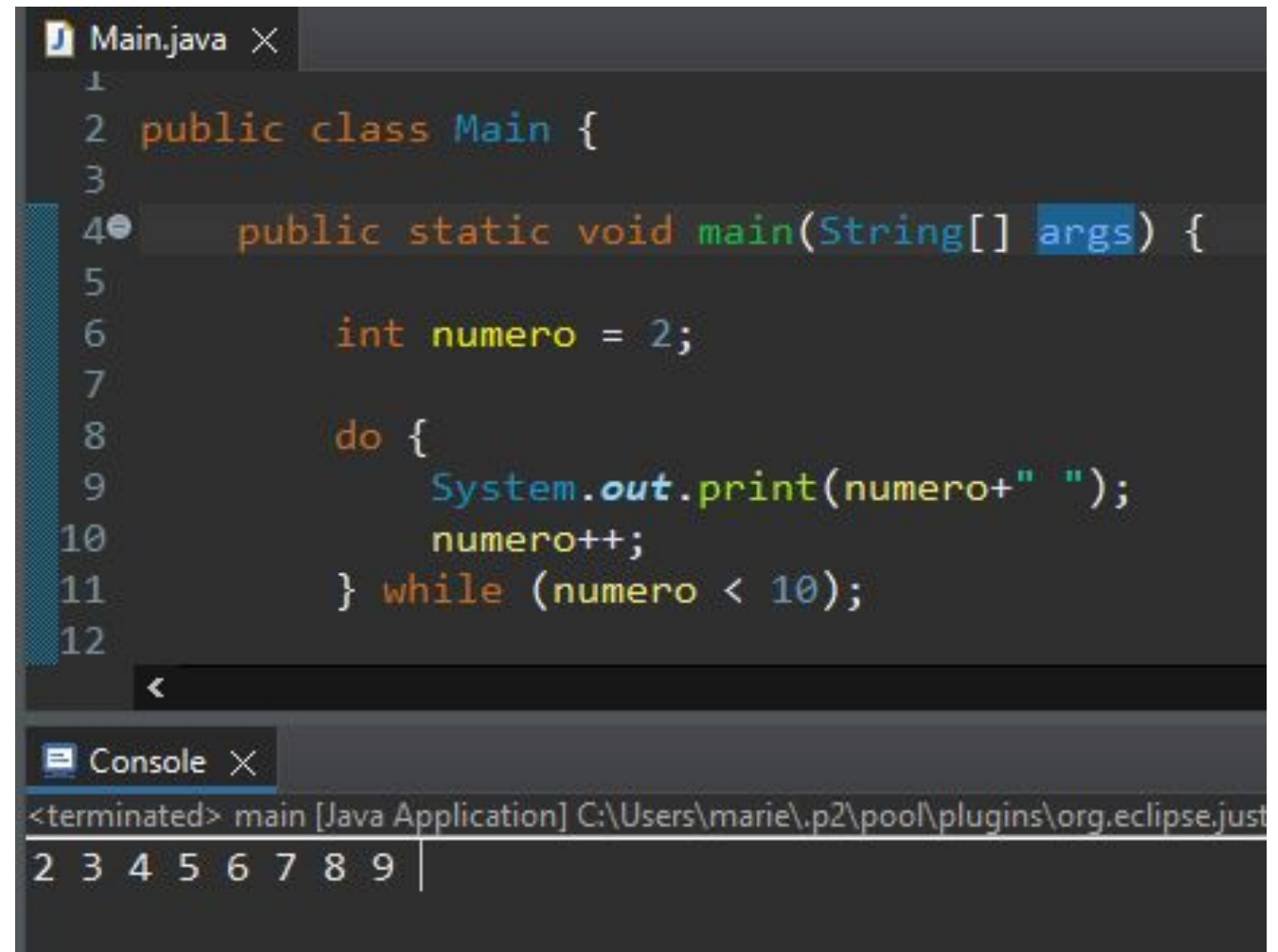


Do-while

Iteración:

En este ejemplo utilizamos nuevamente la variable **número** como **iterador**. Es por esto que es necesario **incrementar** esta variable (línea 10) para que se cumpla en algún momento la **condición de finalización** del bucle.

Podemos ver en la salida por consola como se imprimen todos los números menores a diez.



```
Main.java ×
1
2 public class Main {
3
4 public static void main(String[] args) {
5
6     int numero = 2;
7
8     do {
9         System.out.print(numero+" ");
10        numero++;
11    } while (numero < 10);
12
Console ×
<terminated> main [Java Application] C:\Users\marie\.p2\pool\plugins\org.eclipse.just
2 3 4 5 6 7 8 9 |
```


LIVE CODING

Ejemplo en vivo

do - while:

Vamos a ver un ejemplo de bucle do-while dónde la variable iterador será modificada por una operación aritmética.

- 1. Desarrollar un bucle do-while en el cual se ingrese un valor límite positivo, y a continuación solicite números al usuario hasta que la suma de los números introducidos supere el límite inicial.*

 **Tiempo: 15 minutos**



› Bucle for

For



Sintaxis en Java

La estructura for proporciona una forma compacta de recorrer un rango de valores cuando la cantidad de veces que se debe iterar un bloque de código es conocida.

```
for (<inicialización>; <terminación>; <incremento>) {  
    <sentencias>;  
}
```



La expresión **<inicialización>** inicializa el bucle y se ejecuta una sola vez al comienzo de la ejecución del bucle. El bucle termina cuando al evaluar la expresión **<terminación>** el resultado que se obtiene es **false**.

La expresión **<incremento>** se invoca después de cada iteración que realiza el bucle; esta expresión incrementa o decrementa un valor **hasta que se cumpla la condición de <terminación>** del bucle.

For

Iteración:

En este ejemplo podemos ver que **la variable de incremento ya está definida por el bucle** (línea 9, `i++`), por lo tanto usamos el iterador **i** para imprimir los números mientras se cumpla la condición.

El primer número que se imprime es el **cero**, porque es el número de **inicialización de i** (línea 9, `i = 0`).



```
Main.java X
1 import java.util.Iterator;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         int numero = 10;
8
9         for (int i = 0; i < numero; i++) {
10             System.out.print(i+" ");
11         }
12     }
13 }

Console X
<terminated> main [Java Application] C:\Users\marie\.p2\pool\plugins\org.eclipse.jst
0 1 2 3 4 5 6 7 8 9 |
```

LIVE CODING

Ejemplo en vivo

for:

Vamos a ver un ejemplo de bucle for dónde la variable iterador ya se encuentra previamente definida. El bucle debe dar 10 vueltas.

- 1. Desarrollar un bucle for en el cual se muestre por pantalla la tabla del 5. Se debe mostrar hasta 5x20.*

 **Tiempo: 15 minutos**



Bucles



¿Cuál usar?:

Como regla general podemos afirmar que se utilizará el bucle **for** cuando se conozca de antemano **el número exacto de veces que debe repetirse** un determinado bloque de instrucciones.



Se utilizará el bucle **do-while** cuando no se conoce exactamente el número de veces que se ejecutará el bucle, pero **se sabe que por lo menos se ha de ejecutar una vez**.

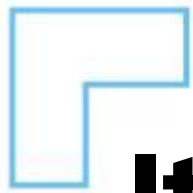
Se utilizará el bucle **while** cuando es posible **que no deba ejecutarse ninguna vez**.





Ejercicio N° 1

Iterando



Iterando

La importancia de los bucles: 🙌

Los bucles son estructuras fundamentales en programación. Permiten repetir un bloque de código de forma controlada. De esta manera se puede ejecutar tareas iterativas de manera simple y con pocas líneas de código. **Las estructuras repetitivas son herramientas de uso constante en programación.** Practiquemos un poco...

Consigna: 📝

Escribir un programa que sume los números del 1 al 10 utilizando.

- 1- bucle while.
- 2- bucle do-while
- 3- bucle for

Tiempo 🕒: 20 minutos

Paso a paso: ⚙️

Recuerda definir los iteradores en los bucles correspondientes. Dejamos un ejemplo de bucle while:

```
while(i <= 10) {  
    suma = suma + i;  
    i++;  
}
```


○

¿Alguna consulta?

+



RESUMEN

¿Qué logramos en esta clase?

- ✓ *Aprender a aplicar bucles de repetición*
- ✓ *Reconocer los bucles while, do-while y for*
- ✓ *Comprender su sintaxis e implementación*



#WorkingTime

Continuemos ejercitando

¡Antes de cerrar la clase! Te invitamos a: 🙌 🙌 🙌

1. Repasar nuevamente la grabación de esta clase
2. Revisar el material compartido en la plataforma de Moodle (lo que se vio en clase y algún ejercicio adicional)
 - a. Material 1 (Foro)
 - a. *Lectura Módulo 4, Lección 3: páginas 9 - 10*
3. Traer al próximo encuentro, todas tus dudas y consultas para verlas antes de iniciar nuevo tema.

¡Muchas Gracias!

Nos vemos en la próxima clase 🙌



Momento: ✚

Time-out!

🕒 5 min.

